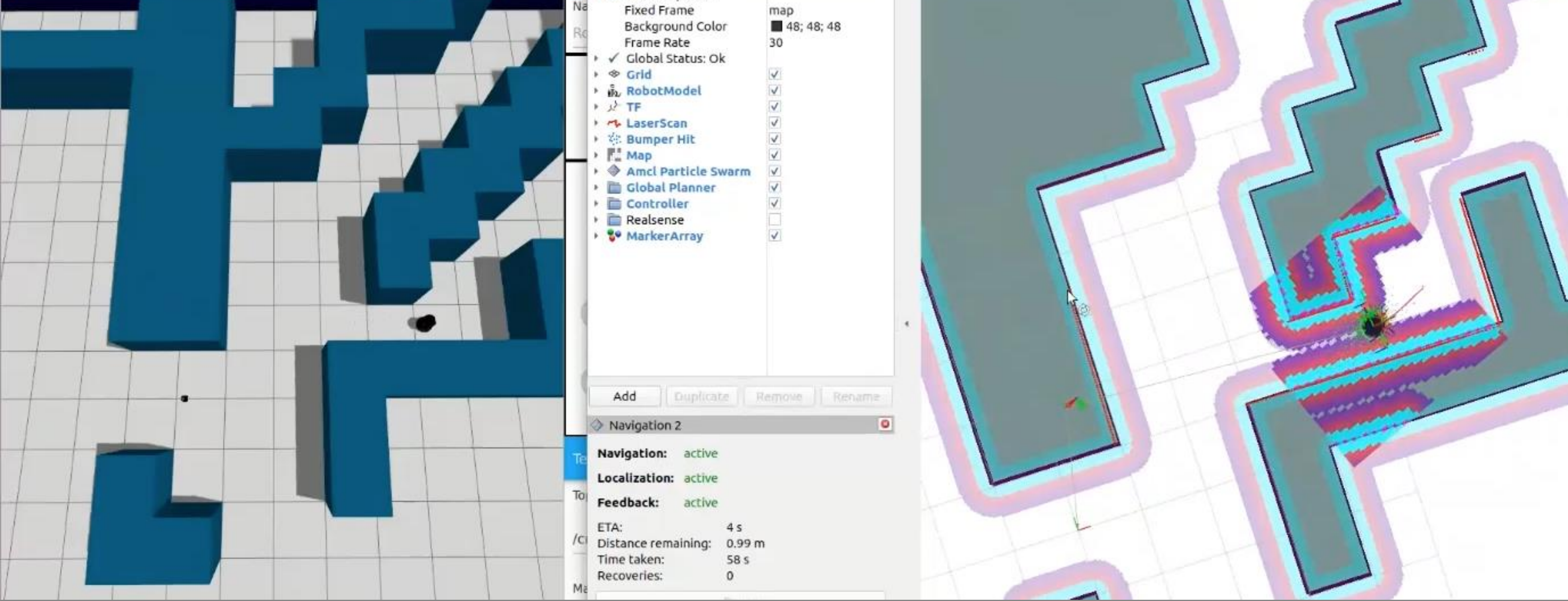


Robotics & XR

5 ECTS
Ilkka Jormanainen and Samuel Yigzaw
{firstname.lastname}@uef.fi

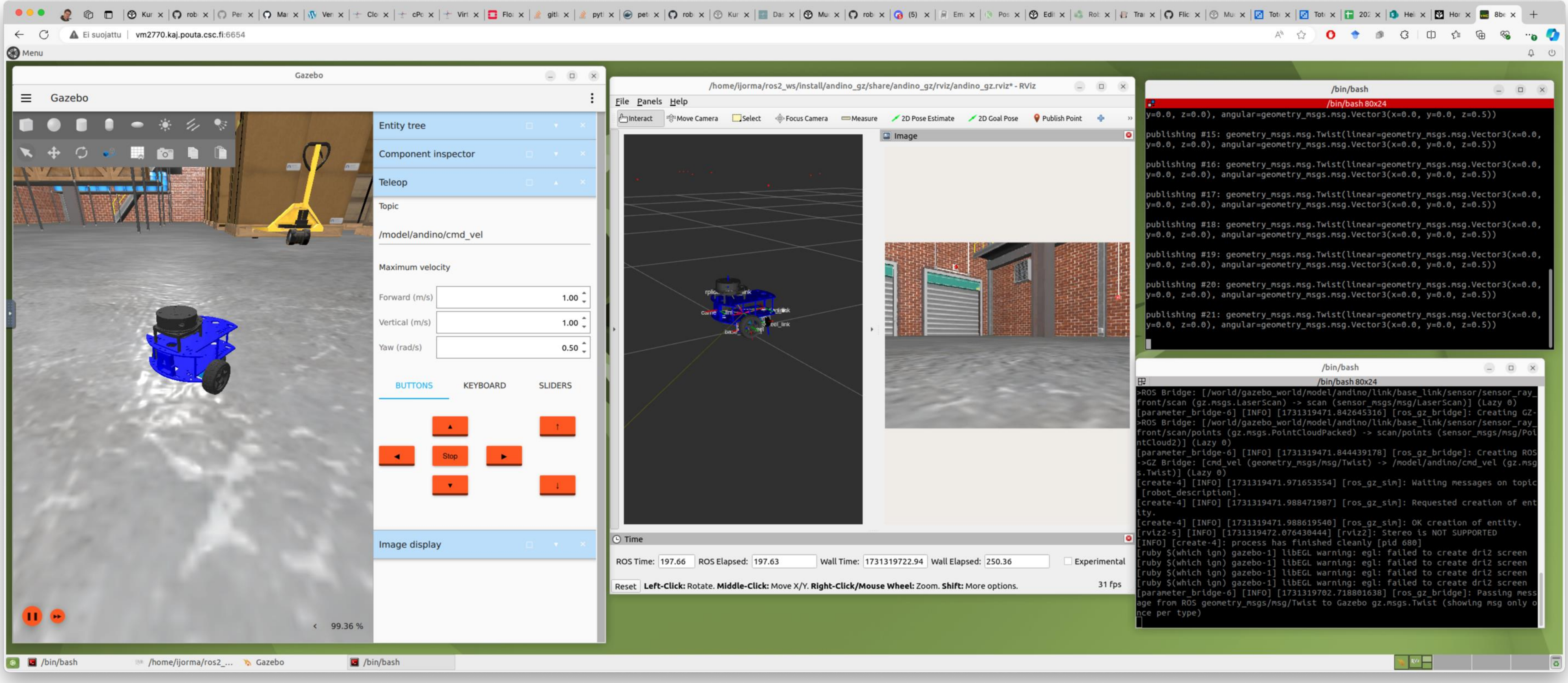
Robot control theory
November 11, 2024

Course contents

- Theme 1: The role of robotics in society and an introduction to ethical issues in robotics
- Theme 2: Different types of robots and their application to real-world problems
- **Theme 3: Robot control theory**
- Theme 4: Navigation
- Theme 5: Robotics & AI
- Theme 6: Extended Reality applications in robotics

Some recap from the last week

- ROS 2 as framework
- Topics, nodes, actions, services
- Issues encountered
  - Docker / ROS installation and configuration
  - Running simulation is GPU intensive
  - Varying platforms
- Give us feedback: https://shorturl.at/SDaop (link is also in Moodle)
- A (partial) solution: remotely accessible sandbox
- Demo


vm2770.kaj.pouta.csc.fi:6654
Ei suojattu
Menu
Gazebo
Entity tree
Component inspector
Teleop
Topic
/model/andino/cmd_vel
Maximum velocity
Forward (m/s) 1.00
Vertical (m/s) 1.00
Yaw (rad/s) 0.50
BUTTONS
KEYBOARD
SLIDERS
Stop
Image display
99.36 %
/home/ijorma/ros2_ws/install/andino_gz/share/andino_gz/rviz/andino_gz.rviz* - RViz
File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Goal Pose
Publish Point
Image
Time
ROS Time: 197.66
ROS Elapsed: 197.63
Wall Time: 1731319722.94
Wall Elapsed: 250.36
Experimental
Reset
Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options.
31 fps
/bin/bash
/bin/bash 80x24
publishing #15: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #16: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #17: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #18: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #19: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #20: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
publishing #21: geometry_msgs.msg.Twist(linear=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.0), angular=geometry_msgs.msg.Vector3(x=0.0, y=0.0, z=0.5))
[create-4] [INFO] [1731319471.971653554] [ros_gz_sim]: Waiting messages on topic [robot_description].
[create-4] [INFO] [1731319471.988471987] [ros_gz_sim]: Requested creation of entity.
[create-4] [INFO] [1731319471.988619540] [ros_gz_sim]: OK creation of entity.
[rviz2-5] [INFO] [1731319472.076430444] [rviz2]: Stereo is NOT SUPPORTED
[INFO] [create-4]: process has finished cleanly [pid 680]
[ruby $(which ign) gazebo-1] libEGL warning: egl: failed to create dri2 screen
/bin/bash
/home/ijorma/ros2_...
Gazebo

Theme 3: Robot control theory

- *Week 46 (November 11 – 17)*
- Theory of robot control
- Basics of matrix computation and linear algebra
- Odometry and sensor fusion, applicable filters

Core concepts

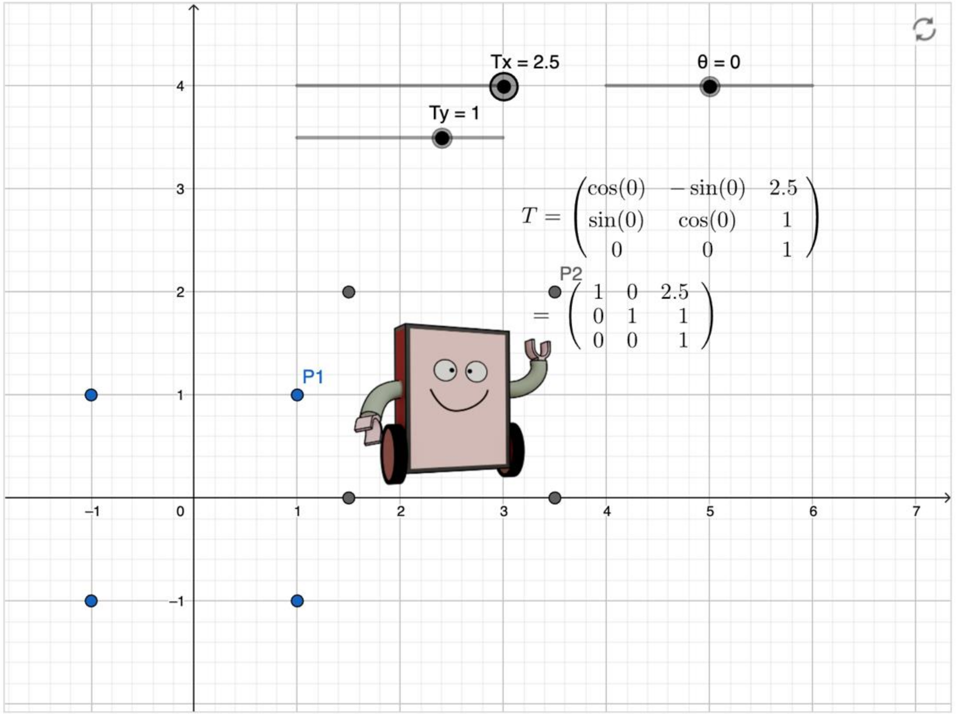
- **Transformations (tf)**
- Frames
- Odometry
- Mapping
- Localization
- Navigation

Transformations (tf)

- Provide information on how a robot could move in *n*-dimensional space
- A mobile robot typically moves in 2D space and is able to rotate (3 DOF)
- Translation: Changing position in Cartesian 2D coordinate system
- Rotation: Rotating the object in the space
- Scaling, shearing, mirroring (manipulating size and shape of the object)
- Useful not only in robotics, but also in computer graphics and game design

Transformations

- Transformation matrix – rotate and translate



https://articulatedrobotics.xyz/tutorials/coordinate-transforms/transformation-matrices

Breaking down the matrix

- Representing a point in 2D / 3D Cartesian space

$$\mathbf{p}_{2\mathrm{D}} = \begin{bmatrix} x \\ y \end{bmatrix}, \mathbf{p}_{3\mathrm{D}} = \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$$\mathbf{p}_k = \begin{bmatrix} x_k \\ y_k \end{bmatrix}$$

Linear transformation and identity matrix

- $f(\mathbf{p})=\mathbf{Ap}$
  - where A is an *n x n* matrix, *n* is number of dimensions in **p**
- If the function is something else than two matrices multiplied, it is non-linear
- When multiplying a point matrix with identity matrix, we get the same result

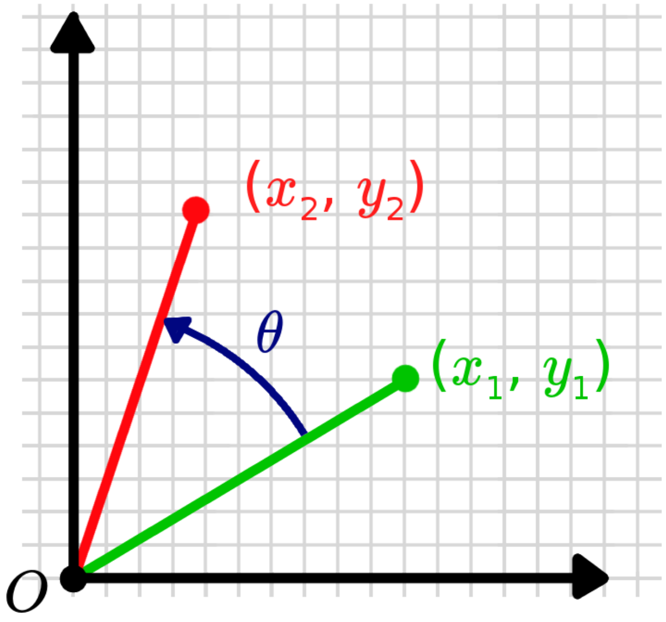
$$[a],\ \begin{bmatrix} a & b \\ c & d \end{bmatrix},\ \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

$$\mathbf{p}_2 = \mathbf{A}\mathbf{p}_1$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} ax_1 + by_1 \\ cx_1 + dy_1 \end{bmatrix}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

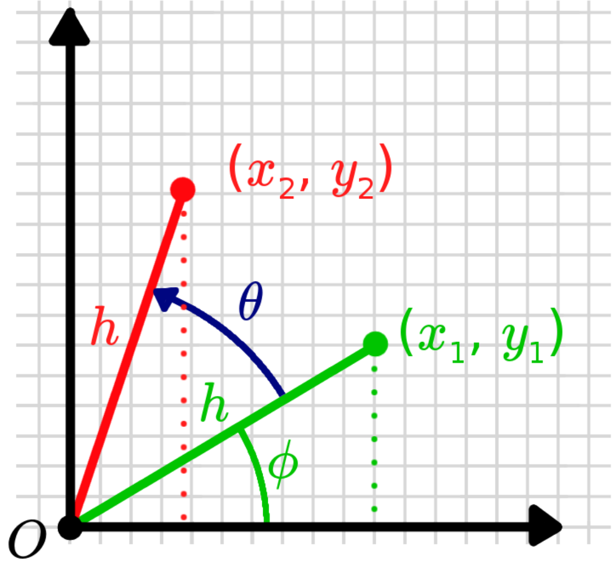
Rotation – defining transformation matrix



$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} ax_1 + by_1 \\ cx_1 + dy_1 \end{bmatrix}$$

Rotation – defining transformation matrix

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} ax_1 + by_1 \\ cx_1 + dy_1 \end{bmatrix}$$

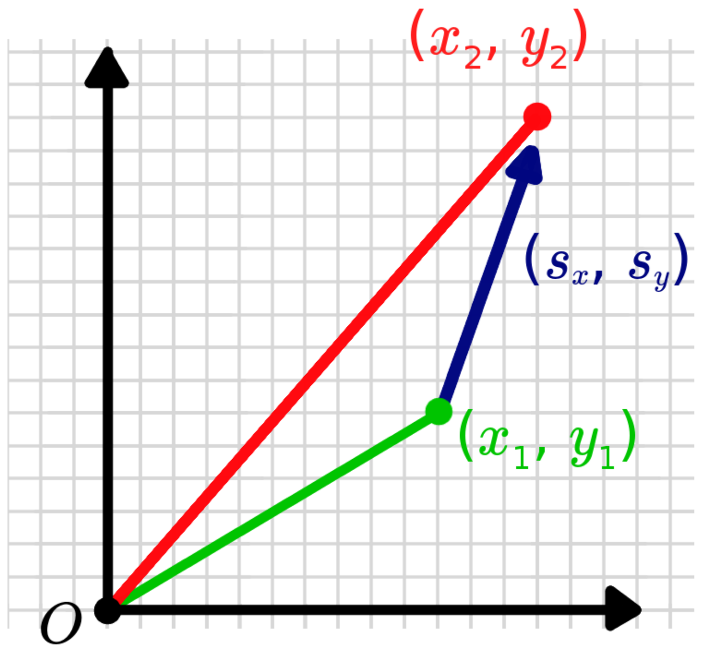


$$\begin{bmatrix} x_1 \\ y_1 \end{bmatrix} = \begin{bmatrix} h\cos(\phi) \\ h\sin(\phi) \end{bmatrix}$$

$$\begin{aligned} \begin{bmatrix} x_2 \\ y_2 \end{bmatrix} &= \begin{bmatrix} h\cos(\phi+\theta) \\ h\sin(\phi+\theta) \end{bmatrix} \\ &= \begin{bmatrix} h\cos(\phi)\cos(\theta) - h\sin(\phi)\sin(\theta) \\ h\sin(\phi)\cos(\theta) + h\cos(\phi)\sin(\theta) \end{bmatrix} \\ &= \begin{bmatrix} x_1\cos(\theta) - y_1\sin(\theta) \\ x_1\sin(\theta) + y_1\cos(\theta) \end{bmatrix} \end{aligned}$$

$$\begin{bmatrix} x_2 \\ y_2 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \end{bmatrix}$$

Translation – moving in coordinates



- Non-linear function $f(\mathbf{p})=\mathbf{p}+\mathbf{b}$
- Use of homogenous coordinates help to derive translation matrix (linear function) – affine transformation

$$\bar{\mathbf{p}}_2 = \mathbf{A}\bar{\mathbf{p}}_1 = \bar{\mathbf{p}}_1 + \mathbf{s}$$

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} ? & ? & ? \\ ? & ? & ? \\ ? & ? & ? \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + s_x \\ y_1 + s_y \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & 0 & s_x \\ 0 & 1 & s_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} x_1 + s_x \\ y_1 + s_y \\ 1 \end{bmatrix}$$

Transformation matrices

- First, we need to augment rotation matrix to homogenous coordinates
- Integrating with translation matrix

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix}$$

$$\mathbf{T} = \bar{\mathbf{B}}\bar{\mathbf{R}} = \begin{bmatrix} 1 & 0 & s_x \\ 0 & 1 & s_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & s_x \\ \sin(\theta) & \cos(\theta) & s_y \\ 0 & 0 & 1 \end{bmatrix}$$

Core concepts

- Transformations (tf)
- **Frames**
- Odometry
- Mapping
- Localization
- Navigation

Frame

- A reference point in 2D or 3D space that helps to understand and describe position and orientation of an object
- Every object in a robotics application has its own frame
  - Coordinate system to measure object’s poosition and orientation
- Frames are either fixed (e.g. map) or mobile (e.g. robot’s base frame)
- Frames hierarchy
  - Robot’s frame attached to its base -> frame moves when the robot moves
  - Robot’s accessories are mounted to this frame -> they also move
- ROS2: tf2 library keeps track on frames and how they move in the space
- Transformation tree: relations between the frames (e.g. map->robot)

Core concepts

- Transformations (tf)
- Frames
- **Odometry**
- Mapping
- Localization
- Navigation

Odometry

- Robot tracks its position based on previous positions and its movements (translations and rotations in space)
- Multiple sensor inputs (e.g. wheel encoders, IMU, gyroscope)
- Accumulating movements in the space
- Relying only one type of sensor (for example wheel encoder) is usually unreliable (**why?**)
- How to improve odometry?

Sensor fusion

- Combining data from various sources
  - Reducing uncertainty and increasing robustness of the system
- Camera, LIDAR, GPS, IMU, encoders, …
- Kalman filters (and other algorithms) and/or sensor data weighting to combine data efficiently
- If one sensor fails, the others continue providing information
- Noise reduction and increased precision
- Consider autonomous cars: **What sensors might be included in fusion?**