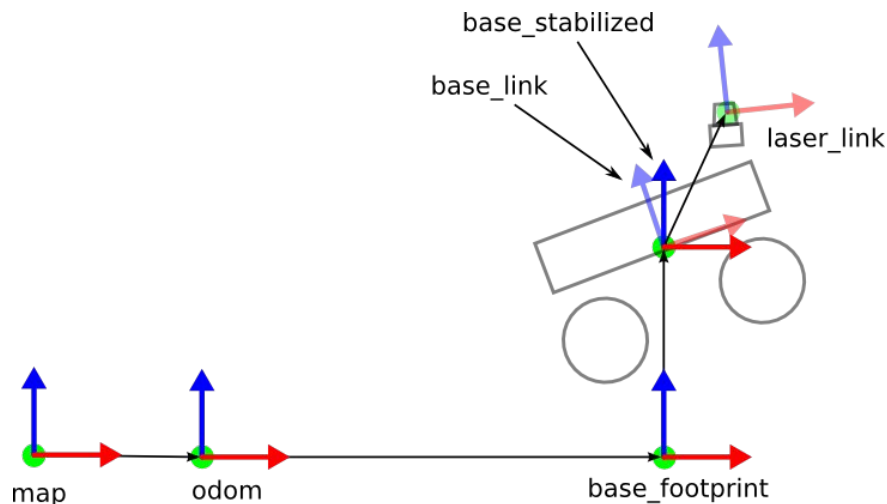


Robot control theory

The ROS perspective



UNIVERSITY OF
EASTERN FINLAND

**Henki
Robotics**

George-Cosmin Porusniuc
Co-Founder & Robotics Consultant @ Henki Robotics

During this lecture...

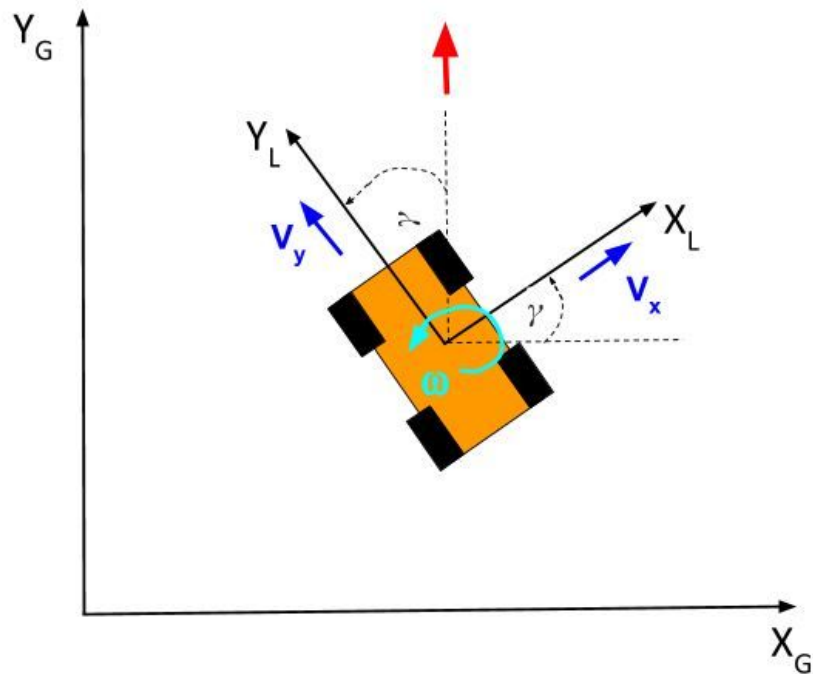
1. TFs & (coordinate) frames
 - a. What are TFs? Why do we need them?
 - b. What are (coordinate) frames?
 - c. How do they relate with TFs
 - d. The most common coordinate frames
 - e. The TF2 ROS 2 library
 - f. The TF tree
 - g. Some short demos
2. Odometry
 - a. What is it? Why do we need it?
 - b. Sources of Odometry?
 - c. Real-world challenges. How do we get good odometry?
 - d. Sensor fusion



wTF are TFs 🤔

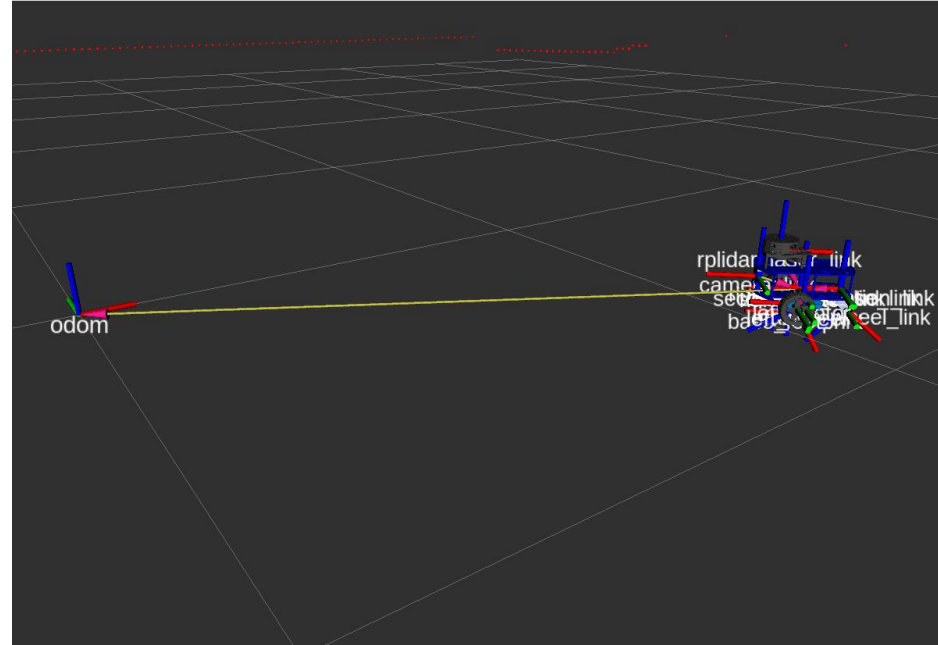
TFs - what are TFs? Why do we need them?

- **transformations** = a mathematical concept that tell us how a robot (or part of it) moves in space
- **TFs** = the ROS concept that describes “transforms” from a robotics, real-world perspective
- **TFs** describe the spatial relationships between different parts in a robot system (chassis, actuators, sensors), and other important landmarks in the world (map, origin)
- **TFs** can be looked at as the combination of **coordinate frames** and the vectorial **transforms** (rotations & translations) that convert positions from one frame to another
- **TFs** are essential for understanding how each part of our robot moves in relation to others, and how our robot moves in relation to the outside world.
- **TFs** can be **static** or **dynamic**
- They sit at the core of applications like:
 - Localization and Mapping
 - Path Planning and Navigation
 - Arm manipulation



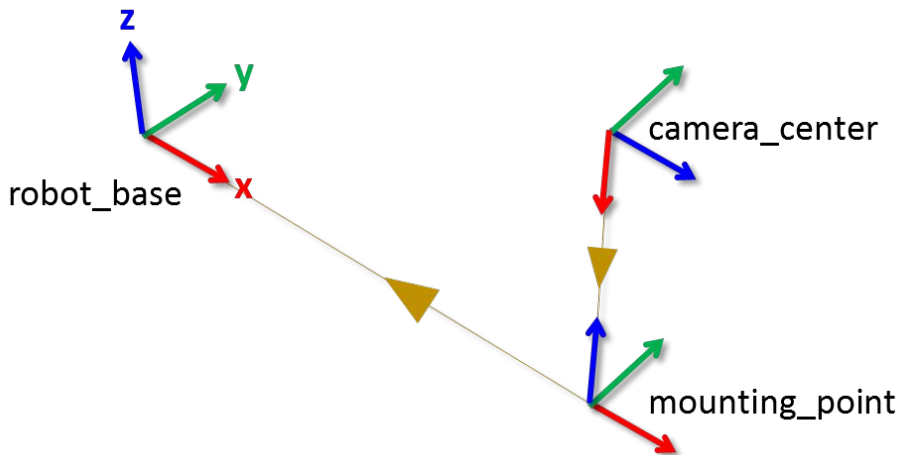
Frames - What are they? Why coordinate frames?

- They describe reference points in 2D or 3D space - that is why they are widely known as **coordinate frames**
- They sit in the most important locations in your robot and in the environment.
- They are at the center of the robot body, at the center of sensors (camera, lidar, etc), at joints between the robot body and wheels/legs.
- The most important environment coordinate frames are the “Map” and “Odometry” frames



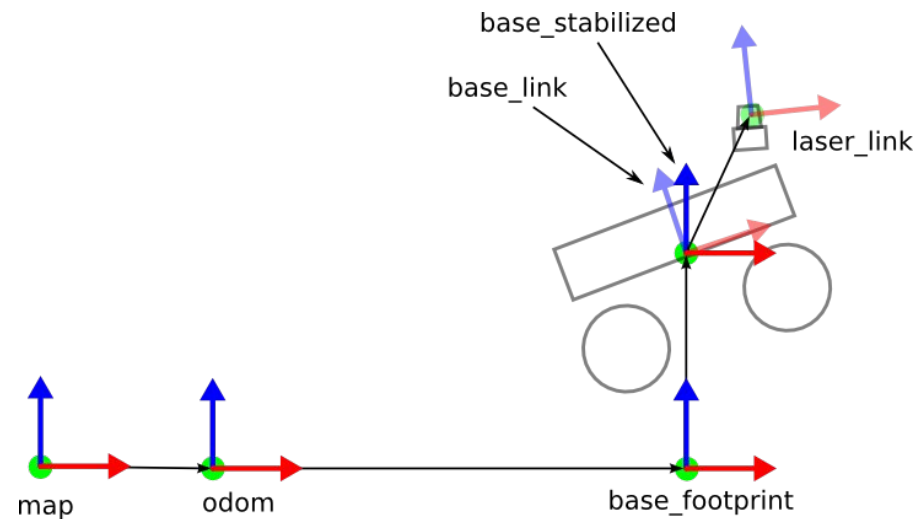
Frames & TFs. How do they relate?

- Frames are the coordinate origins that sit at the center of our robot, components, and landmarks in the environment.
- Transforms tell us how these frames move in relation to each other.
- Frames + Transforms = TFs
- The combination between the 2 allow you to convert positions and orientations from one frame to another, and basically keep track of how each part of your robot moves in relation to the other parts of the robot.



Frames - The most common coordinate frames

- **map**
 - Provides a global reference point for the robot's environment. Typically presents the robot's coordinates on a 2D map.
- **odom**
 - Represents the robot's position based on its odometry data. It tracks the robot's movement
- **base_footprint**
 - 2D projection of the robot's footprint on the ground. Used for planning and movement calculations without considering the robot's height.
- **base_link**
 - Represents the robot's main body and is used as a reference for other components, such as sensors and arms.
- **laser_link**
 - Denotes the position of a laser sensor on the robot. Essential for interpreting the data collected by the laser. Provides a reference for where the sensor is located in relation to other frames.



The TF2 ROS 2 library

- The main component for handling transforms in ROS 2 is the [tf2 library](#). It provides:
 - Coordinate Frames: Each sensor or part of a robot has its own coordinate frame (e.g., the robot's base, sensors, end effectors).
 - Transformations: These include translations (movement along axes) and rotations (changes in orientation) between frames.
 - Both **static** and **dynamic** transforms
- We can easily visualize the TFs generated with the TF2 library using RViz

ROS2-TF2 library

Learn to create Frames and Transformations in ROS2 using TF2-library.

COORDINATE FRAMES

World Frame, Turtle 1, Turtle 2


TYPES OF TRANSFORMS

Dynamic Transform, Static Transform

@Easy_Peasy_Robotics

The TF tree

- Once the robot has lots of moving parts, keeping track of all the relationships between the frames and all the transformations between each frame becomes tough.
- When multiple frames are chained together, we can look at the relationship between each frame as a parent-child relationship.
- Chaining these relationships together, we can put them in a tree-like structure
- The TF tree allows us to easily visualize this tree structure and all the relationships between the frames on our robot and the environment frames

▼  TF

▶ ✓ Status: Ok

Show Names

Show Axes

Show Arrows

Marker Scale 0.2

Update Interval 0

Frame Timeout 15

▶ Frames

▼ Tree

▼ odom

▼ base_link

camera_link

caster_base_link

left_motor

right_motor

▶ second_base_link

base_footprint

▶ caster_rotation_link

left_wheel

right_wheel

TFs & Frames practical demos



https://github.com/henki-robotics/robotics_essentials_ros2/tree/main/1-ros_2_introduction

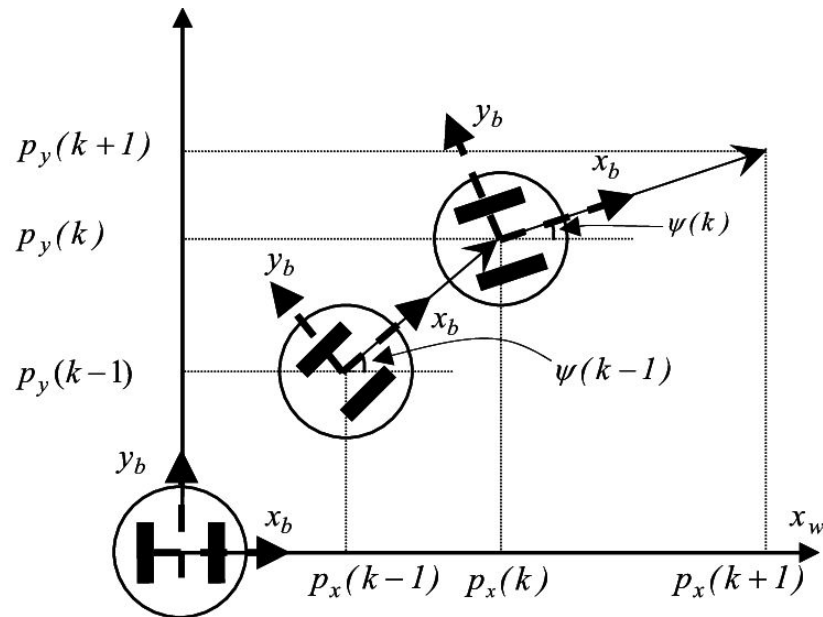
<https://docs.ros.org/en/humble/Tutorials/Intermediate/Tf2/Introduction-To-Tf2.html>

Odometry

Odometry - What? Why? Pose estimation

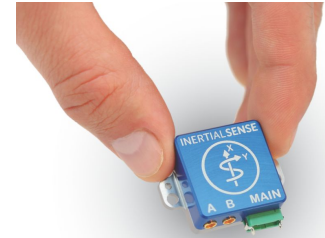
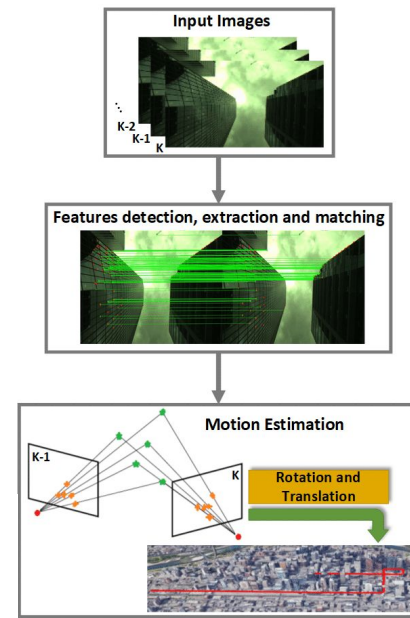
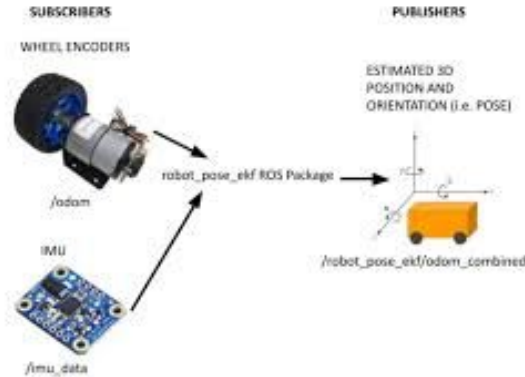
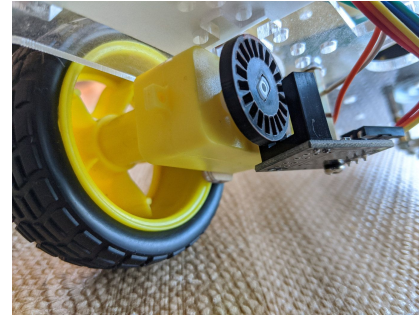
- A technique used to estimate a robot's position and orientation (**pose**) over time
- The **pose** is estimated using data from the robot's sensors
- It helps keep track of the robot's location as it moves through the environment
- It is a fundamental part of a robot's **navigation** system
- typically used for **dead reckoning**, where the robot calculates its position incrementally, step-by-step, based on past movements.

Pose Estimation: Odometry provides an ongoing estimate of the robot's pose (position and orientation) in a given coordinate frame, usually relative to the robot's starting point or a known reference frame.



Odometry - Types of odometry

- **Wheel Odometry:** Uses data from wheel encoders, which measure wheel rotations. By calculating how much each wheel has rotated, the robot can estimate how far it has moved.
- **Visual Odometry:** Uses cameras or depth sensors to estimate movement by analyzing changes in images as the robot moves.
- **Inertial Odometry:** Utilizes inertial measurement units (IMUs) that provide data on acceleration and rotational velocity to estimate position changes.
- **Sensor Fusion Odometry:** Combines multiple sources (e.g., wheel encoders, IMUs, cameras) for more accurate and reliable odometry.



Odometry - Real-world challenges. How to improve?

- **Wheel Slippage:** On slippery or uneven surfaces, wheel-based odometry may be inaccurate due to slippage.
- **Accumulated Error:** Small errors from each movement accumulate over time, leading to significant inaccuracy over long distances.
- **Non-uniform Terrain:** Rough or uneven terrain affects the robot's movement accuracy, impacting the reliability of odometry.

Solutions:

To mitigate these limitations, odometry is often combined with **sensor fusion methods**, where data from additional sensors like cameras, LIDAR, and GPS are integrated using algorithms such as the **Kalman filter** or **Extended Kalman filter (EKF)** to improve position estimates and reduce drift.

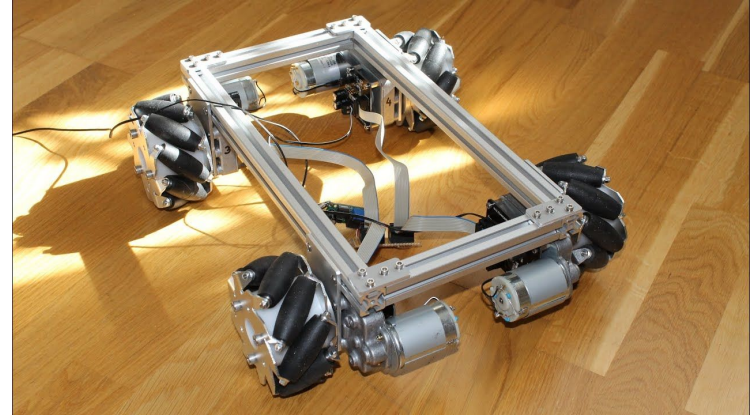
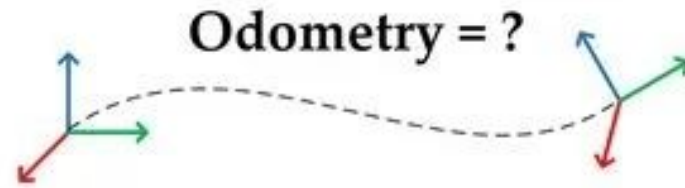


Fig. 1: Proposed Six-Wheeled Multi Terrain Robot (SWMTR)

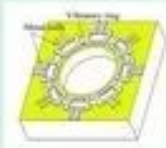
Odometry - Sensor fusion



Sensors

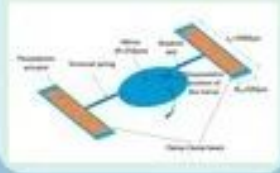
IMU

Polymeric IMU



LiDAR

Polymeric LiDAR



Radar

Camera

Sensor Fusion

Filter-based

Probability theory-based

Evidential reasoning-based

Random finite set-based

Optimization-based

Thank you! 🙌