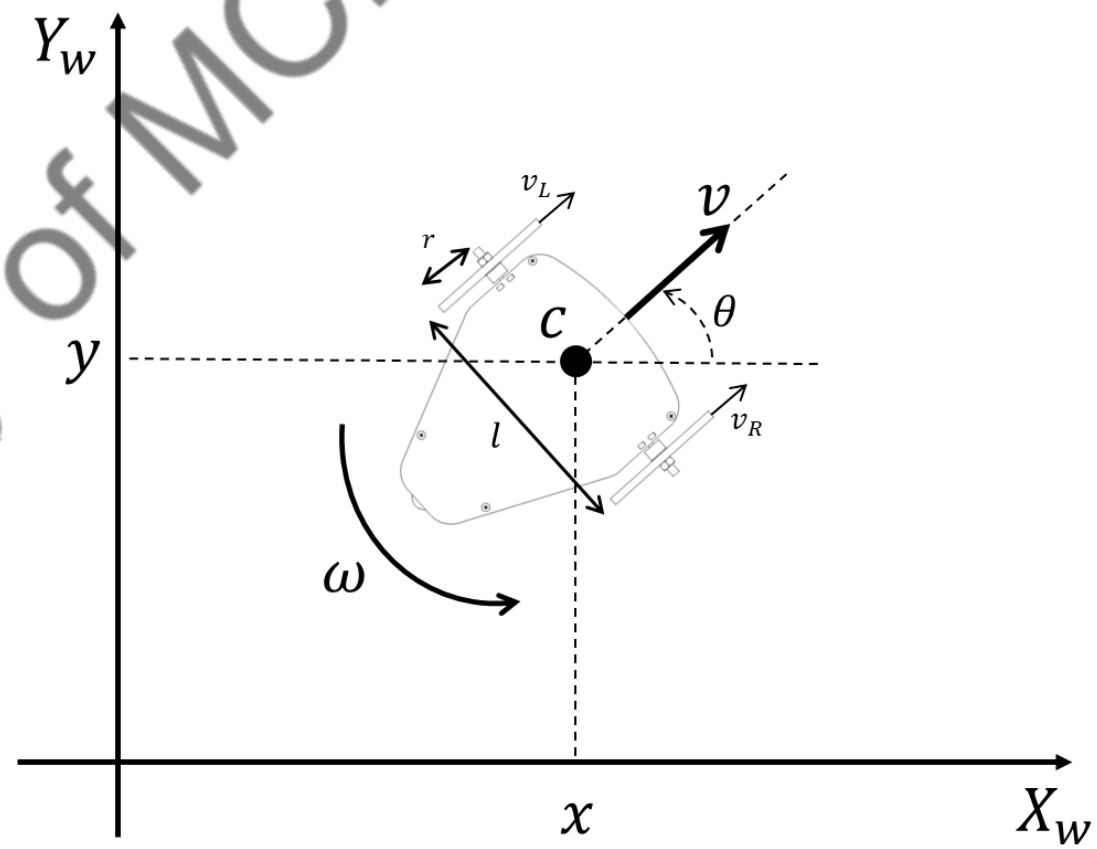
Localisation

Dead Reckoning

{Learn, Create, Innovate};

Mobile Robot Localisation

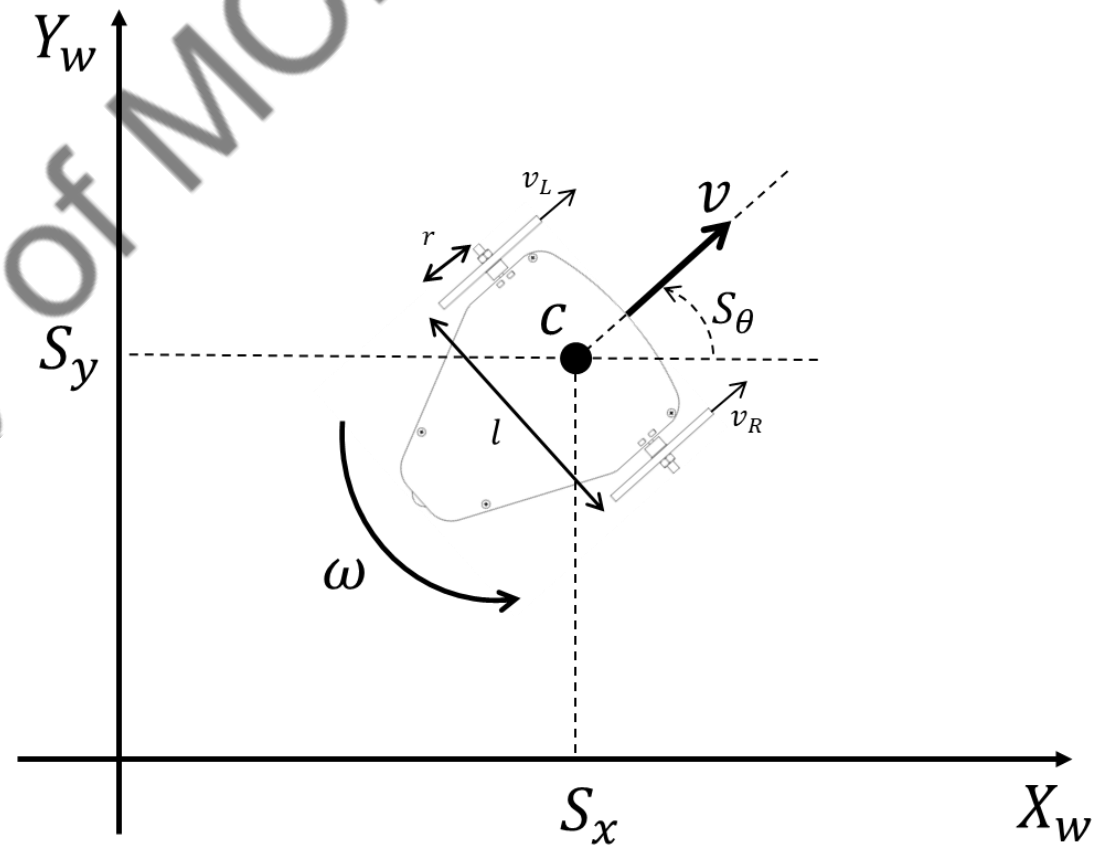
- One of the basic functions of mobile robots is to traverse from a certain position to another one in the environment.
- To accomplish this task the mobile robot needs to know its **pose** in the environment at any time to determine whether it has reached its destination; such process is called **localisation**.
- There are many techniques to approach the localisation problem and they differ in the type of sensors used and how the uncertainty is addressed.



Mobile Robot Localisation

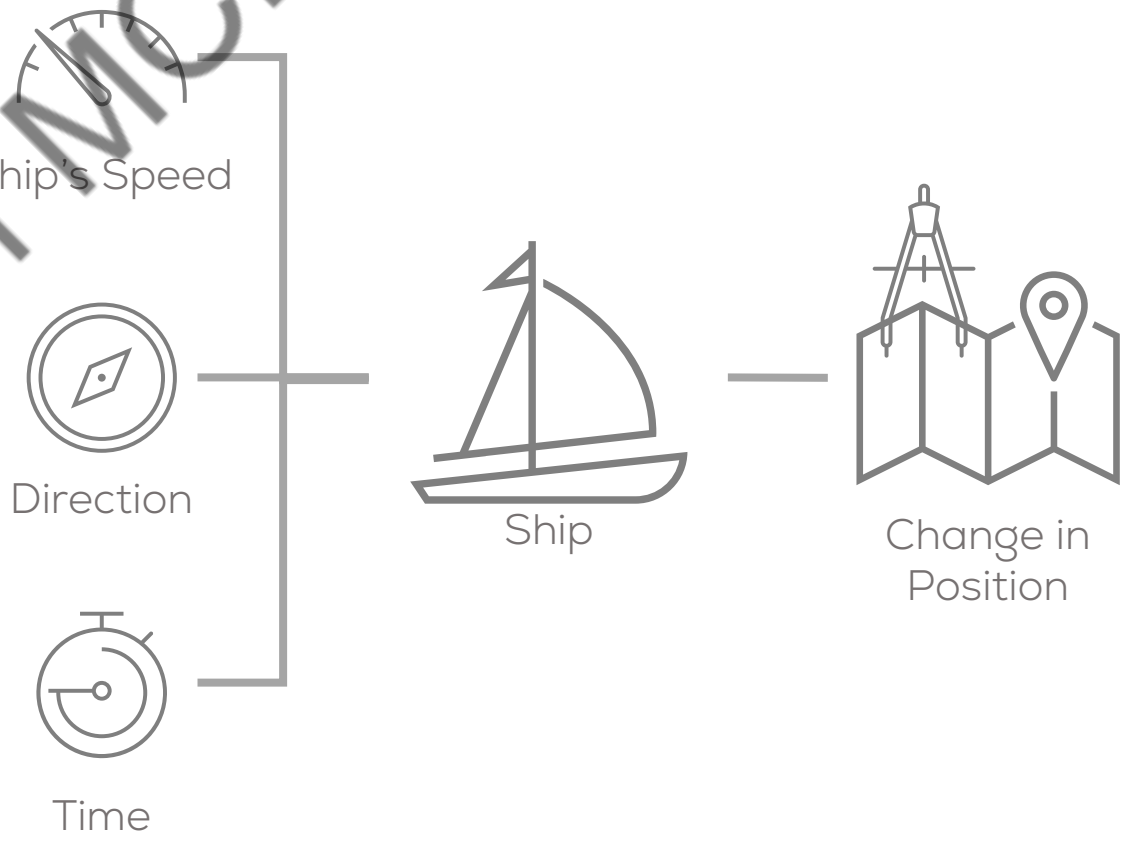
- In general, in the mobile robotic community, the state of a robot is denoted by "$\boldsymbol{s}$", namely the *pose* or *posture*, for this case it consists of the robot position and orientation with respect to a frame of reference (world frame).

$$\mathbf{s} = [s_x \quad s_y \quad s_\theta]^T$$



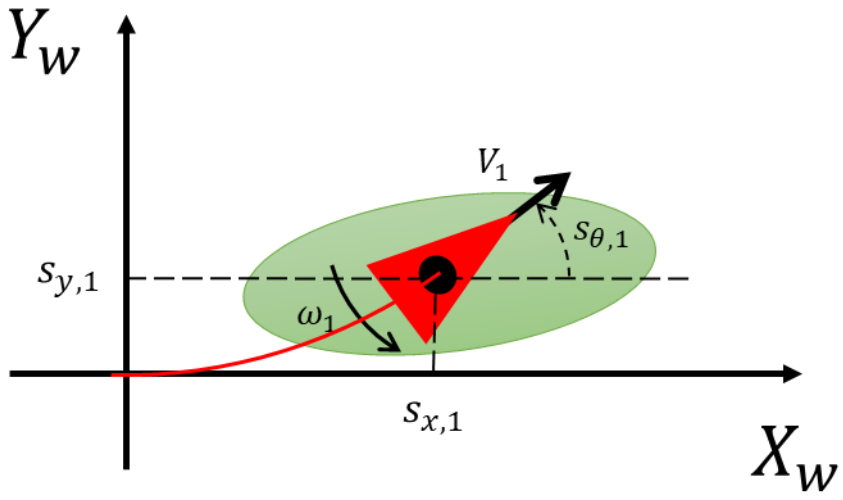
Dead Reckoning

- Dead reckoning (DR) is the typical standalone technique that old sailors used to determine the current position of their ship.
- DR incrementally integrates the distance travelled and the direction of travel relative to a known start location.
- A ship's direction was determined by a magnetic compass, and the distance travelled was computed by the time of travel and the vehicle's speed.
- Posing the technical challenge of building mechanical clocks that work with high accuracy in an environment of rough motions and changing climates.



Dead Reckoning

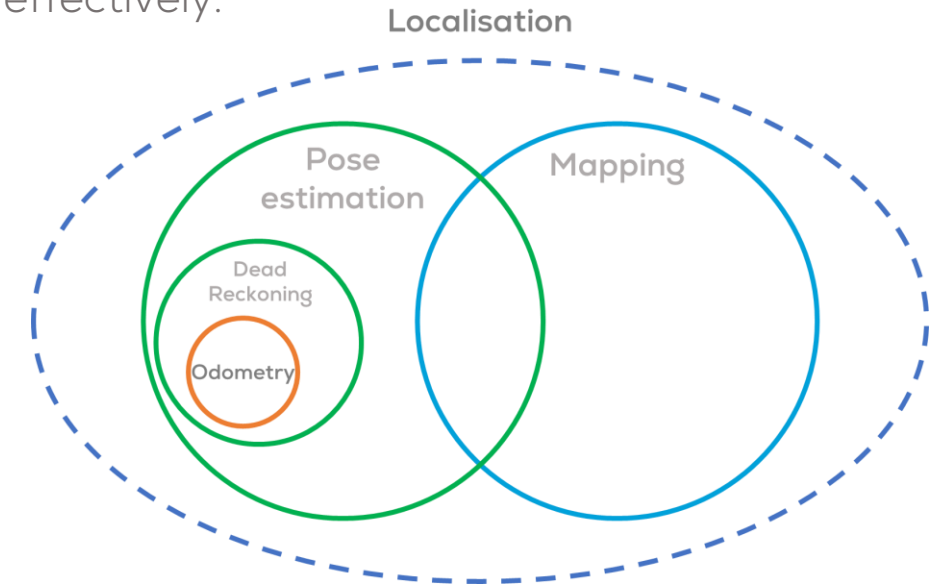
- In navigation, **dead reckoning** is the process of calculating the current position of a moving object by using a previously determined position, and incorporating estimates of speed, heading (or direction or course), and elapsed time.
- This technique uses the internal kinematics of the robot to localise it in the environment.
- The robot may also employ some proprioceptive or exteroceptive sensors such as: encoders, cameras or IMUs, to provide a better estimate of the pose change over time.
- However, such technique suffers from the unbounded growth of uncertainty about the robot pose over time due to the numerical integration and accumulation of error.



Odometry

- The word odometry is composed of the Greek words odos (meaning "route") and metron (meaning "measure").
- **Odometry** is the use of data from motion sensors (encoders for wheeled-based robots) to estimate **changes in position over time**.
- Odometry is a type of dead reckoning localisation, based on estimating the distance travelled.
- It is used in robotics by some legged or wheeled robots to estimate their position relative to a starting location.

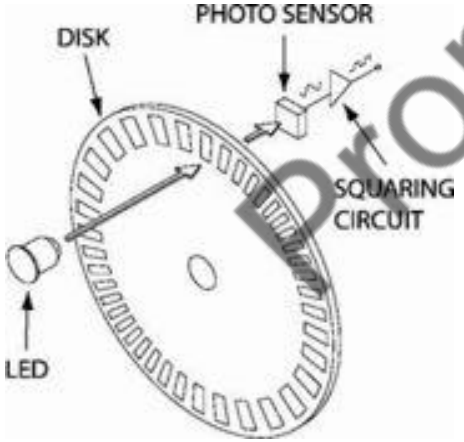
- This method, presents the same errors of other dead reckoning methods.
- Rapid and accurate data collection, instrument calibration, and processing are required in most cases for odometry to be used effectively.

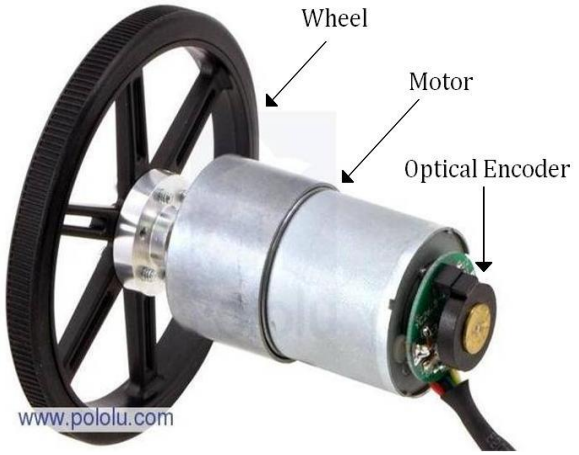


Proprioceptive sensors
Rotary Encoders

- Rotary optical encoders, are proprioceptive sensors, used to determine the angular position of the shaft they are attached to.
- VERY IMPORTANT!!! Encoders in mobile robots are considered proprioceptive sensors because they only acquire information about the robot itself, not the structure of the environment.

Essentially, it is a mechanical light chopper that produces a certain number of pulses for each shaft revolution.





Proprioceptive sensors
Rotary Encoders

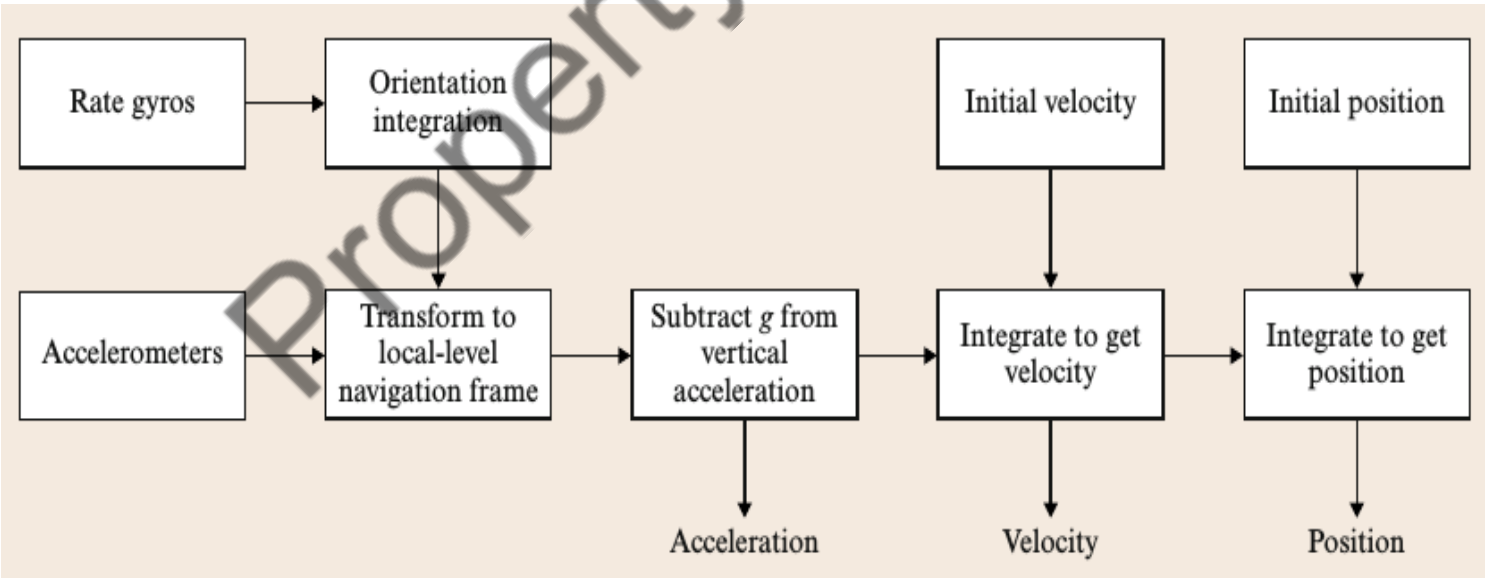
- When an encoder is attached to the axle of each wheel in a differential-drive robot, it is possible to convert the number of pulses into useful information, such as the distance travelled by each wheel. By measuring the wheel diameter, the distance travelled by each wheel is calculated as follows:

$$distance = \frac{counts}{CPR}(\pi d)$$

- CPR – counts per revolution
- d – wheel diameter

Proprioceptive sensors IMUs

Another type of proprioceptive sensing device is the *inertial measurement unit* (IMU). An IMU is an electronic sensor that maintains a 6-degree-of-freedom (DOF) estimate of the relative pose of a mobile vehicle, this is, position in *x*, *y* and *z*, and orientation roll, pitch and yaw. Such task is achieved using a set of gyroscopes and accelerometers. IMUs are a common navigational component of aircrafts and ships, particularly for their capability to determine changes in the yaw, pitch and roll.



Motion-based Localisation (Dead Reckoning)

Kinematic model for a differential robot model

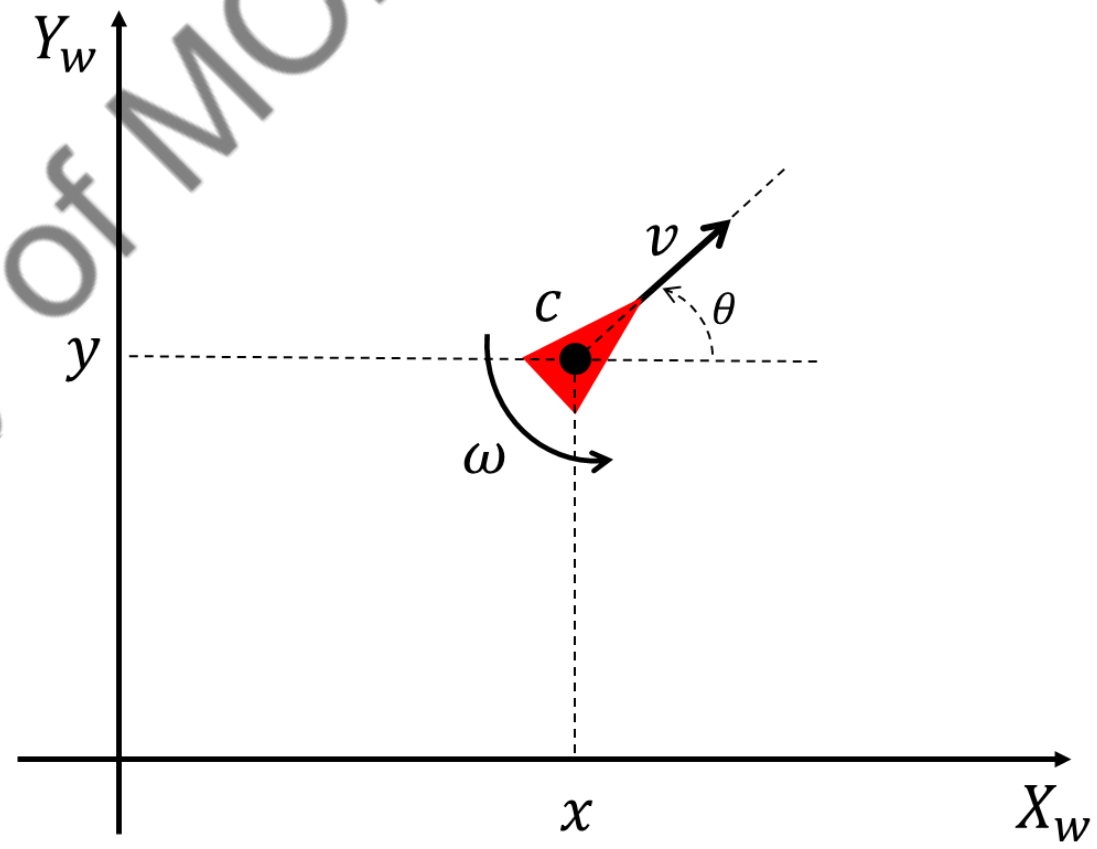
$$\frac{d}{dt}\begin{bmatrix} s_x \\ s_y \\ s_\theta \end{bmatrix} = \begin{bmatrix} \cos(s_\theta) & 0 \\ \sin(s_\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}$$

The robot pose

$$\mathbf{s}_k = [s_x \quad s_y \quad s_\theta]^T$$

The robot inputs

$$\mathbf{u}_k = [v \quad \omega]^T$$



Motion-based Localisation (Dead Reckoning)

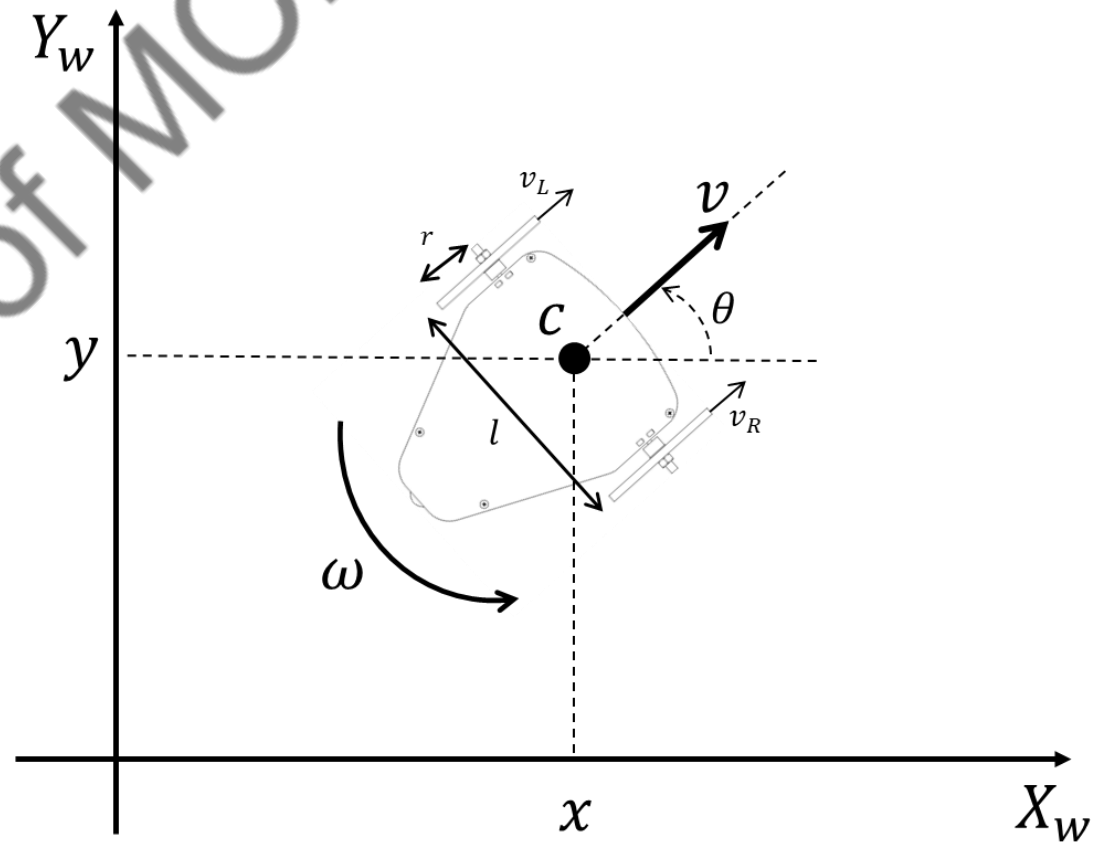
- If Δt is the sampling time, then it is possible to compute the incremental linear and angular displacements, Δd and $\Delta\theta$, as follows:

$$\Delta d = v \cdot \Delta t \qquad \Delta\theta = \omega \cdot \Delta t$$

$$\begin{bmatrix}\Delta d \\ \Delta\theta\end{bmatrix} = \begin{bmatrix}1/2 & 1/2 \\ 1/l & -1/l\end{bmatrix}\begin{bmatrix}\Delta d_r \\ \Delta d_l\end{bmatrix}$$

- Where v and ω can be estimated using the readings from the encoders as follows:

$$\begin{bmatrix}v \\ \omega\end{bmatrix} = \begin{bmatrix}\frac{r}{2} & \frac{r}{2} \\ \frac{r}{l} & -\frac{r}{l}\end{bmatrix}\begin{bmatrix}\omega_R \\ \omega_L\end{bmatrix}$$

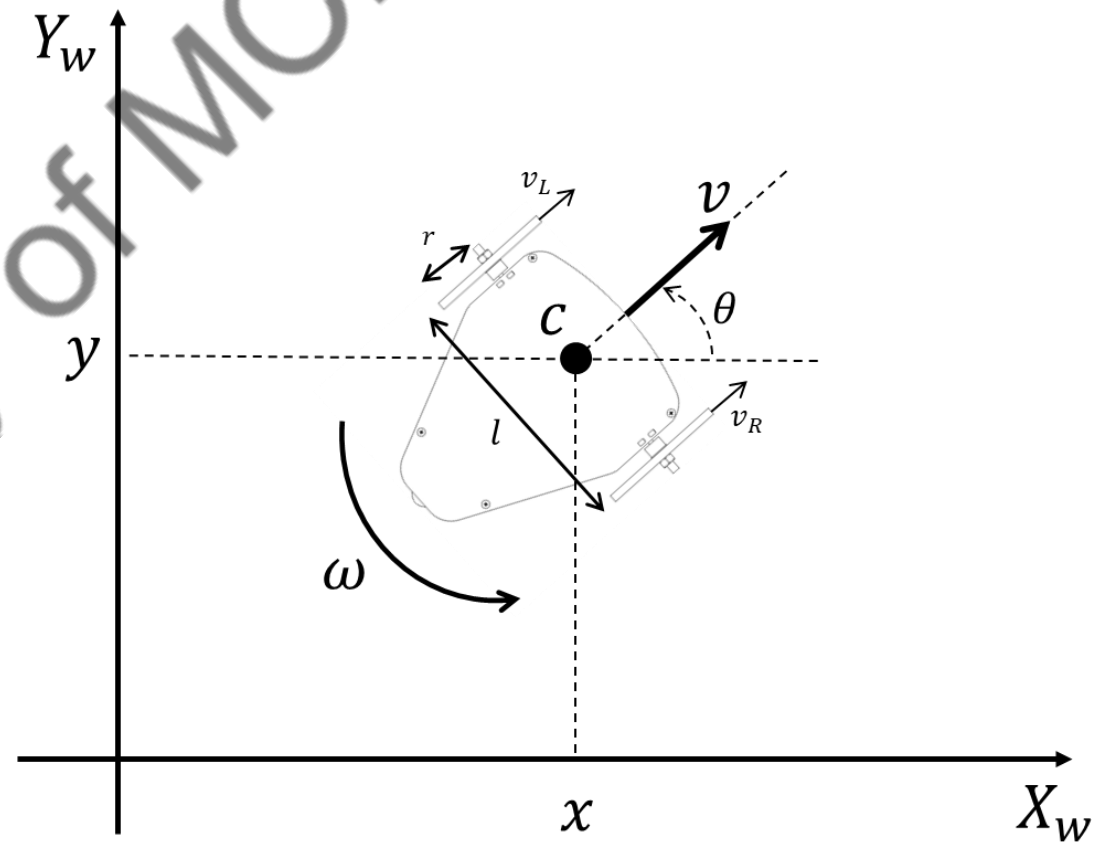


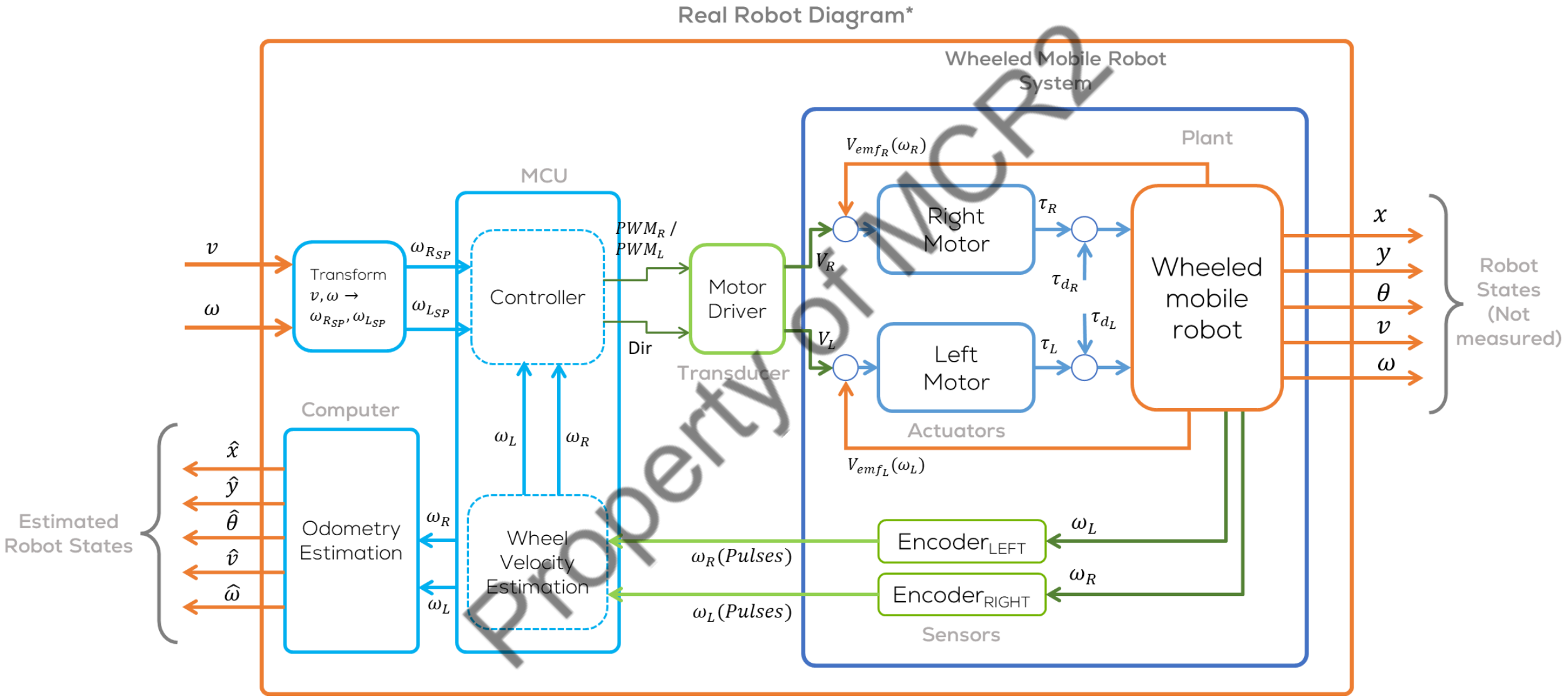
Motion-based Localisation (Dead Reckoning)

To compute the robot's pose at any given time step, the kinematic model must be numerically integrated.

$$\begin{bmatrix} s_{x,k} \\ s_{y,k} \\ s_{\theta,k} \end{bmatrix} = \begin{bmatrix} s_{x,k-1} \\ s_{y,k-1} \\ s_{\theta,k-1} \end{bmatrix} + \begin{bmatrix} \Delta d \cos(s_{\theta,k-1}) \\ \Delta d \sin(s_{\theta,k-1}) \\ \Delta\theta \end{bmatrix}$$

This approximation follows the **Markov assumption**, in which the current robot pose depends only on the previous pose and the input velocities.





Motion-based Localisation (Dead Reckoning)

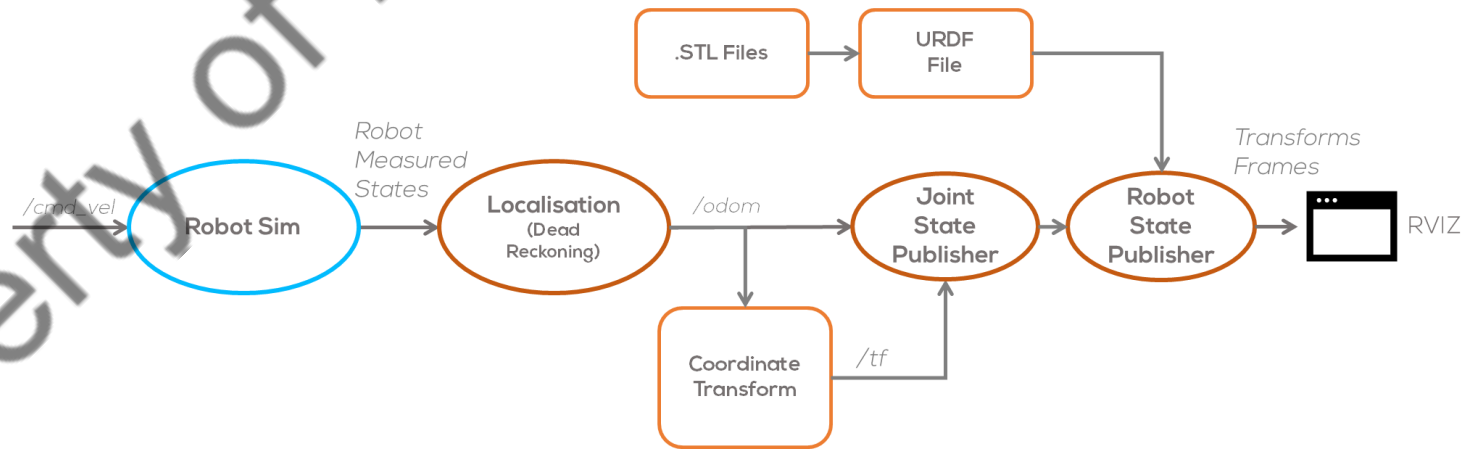
The pose estimation of a mobile robot is **always associated with some uncertainty** with respect to its state parameters.

From a geometric point of view, the error in differential-drive robots is classified into three groups:

- **Range error:** it is associated with the computation of Δd over time.
- **Turn error:** it is associated with the computation of $\Delta \theta$ over time.
- **Drift error:** it is associated with the difference between the angular speed of the wheels and it affects the error in the angular rotation of the robot.

Localisation ROS

- In ROS, localisation is typically done in a separate node.
- The concept of localisation states that it should be an action independent of the robot can be performed by an internal or an external computing unit.
- This provides more flexibility and independence when developing robotics so that the user can define the internal parameters of the node and reuse it for other robots and not only for a specific type of robot platform.

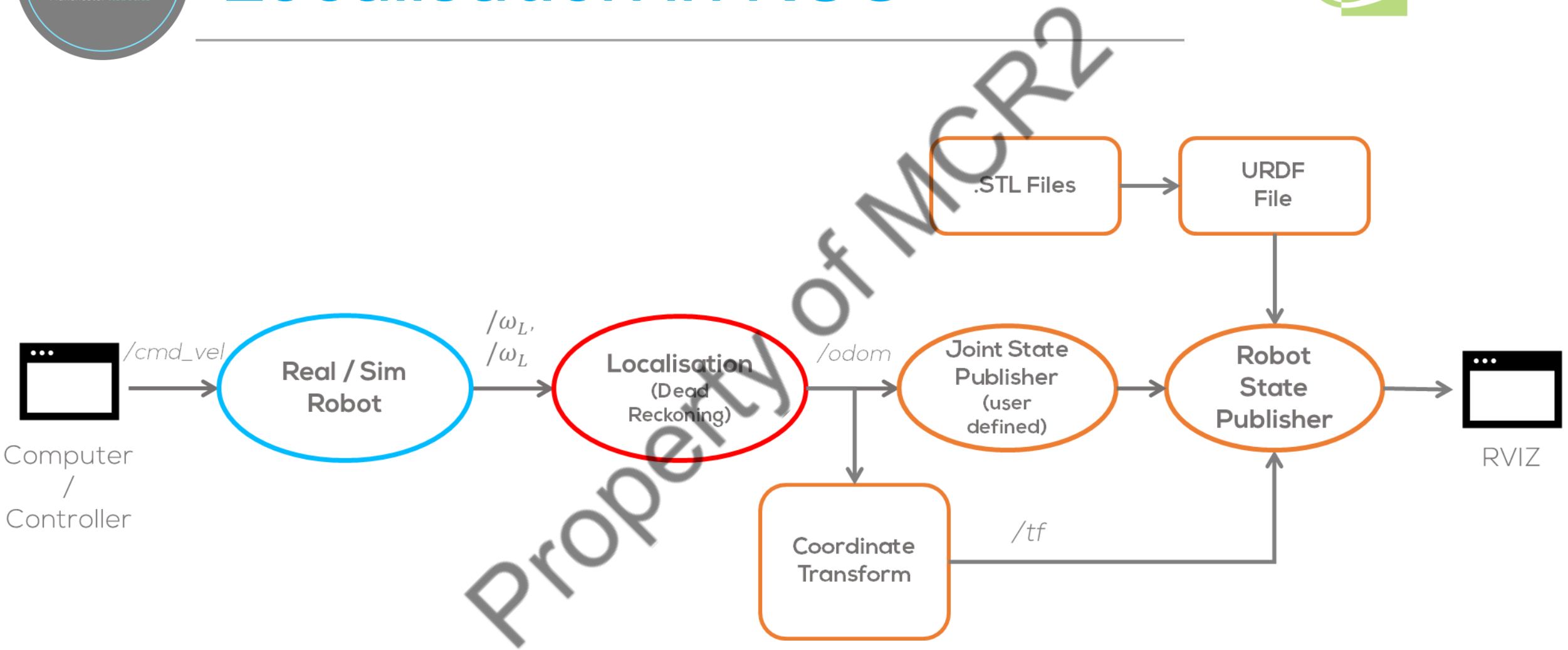


Localisation ROS

- In ROS, the odometry of a robot is published using an Odometry message.
- The message is found in "*nav_msgs/Odometry*"
- Typically, the odometry message uses the topic "*odom*".
- The pose in this message corresponds to the estimated position of the robot in the odometric frame along with an optional covariance for the certainty of that pose estimate.
- The twist in this message corresponds to the robot's velocity in the child frame, normally the coordinate frame of the mobile base, along with an optional covariance for the certainty of that velocity estimate.

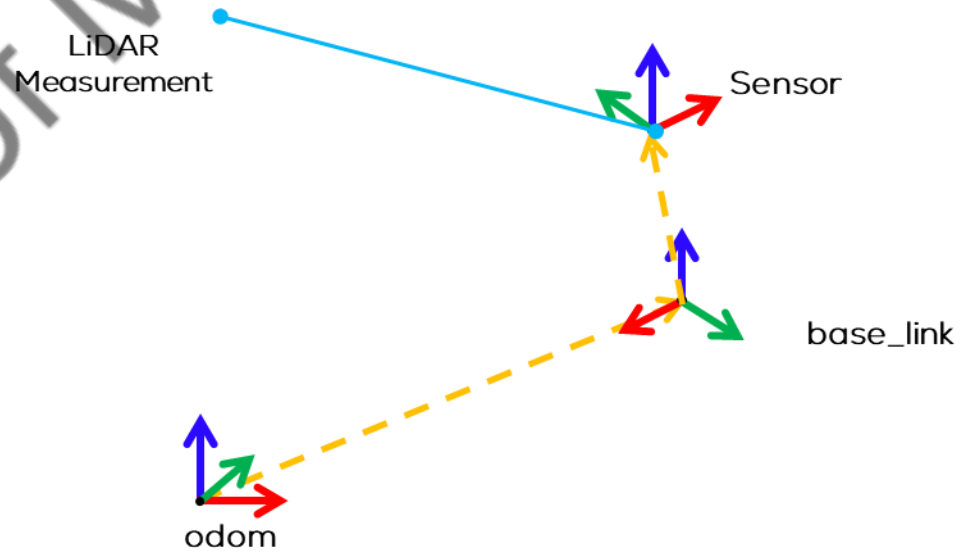
```
Odometry Message:

# This represents an estimate of a position and velocity in
free space.
# The pose in this message should be specified in the
coordinate frame given by header.frame_id.
# The twist in this message should be specified in the
coordinate frame given by the child_frame_id

Header header
string child_frame_id
geometry_msgs/PoseWithCovariance pose
geometry_msgs/TwistWithCovariance twist
```

Localisation in ROS



Localisation ROS

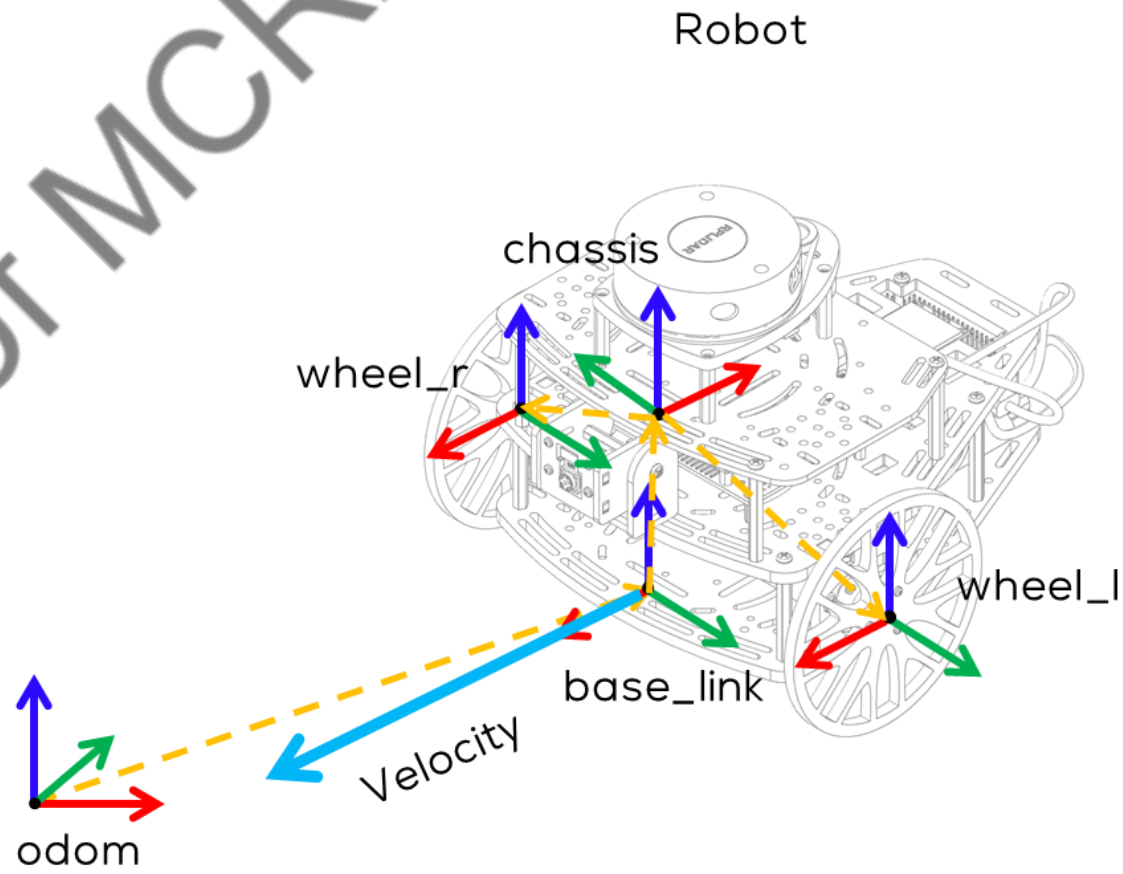
- In ROS, information provided by sensors or other nodes (odometry) requires a coordinate frame to be published into.
- A simple example is the localisation.
- Localisation data (position and velocity) requires different coordinate frames to be published.
  - The position of the robot is always relative to a fixed coordinate frame
  - The velocity belongs to the coordinate frame of the robot itself.

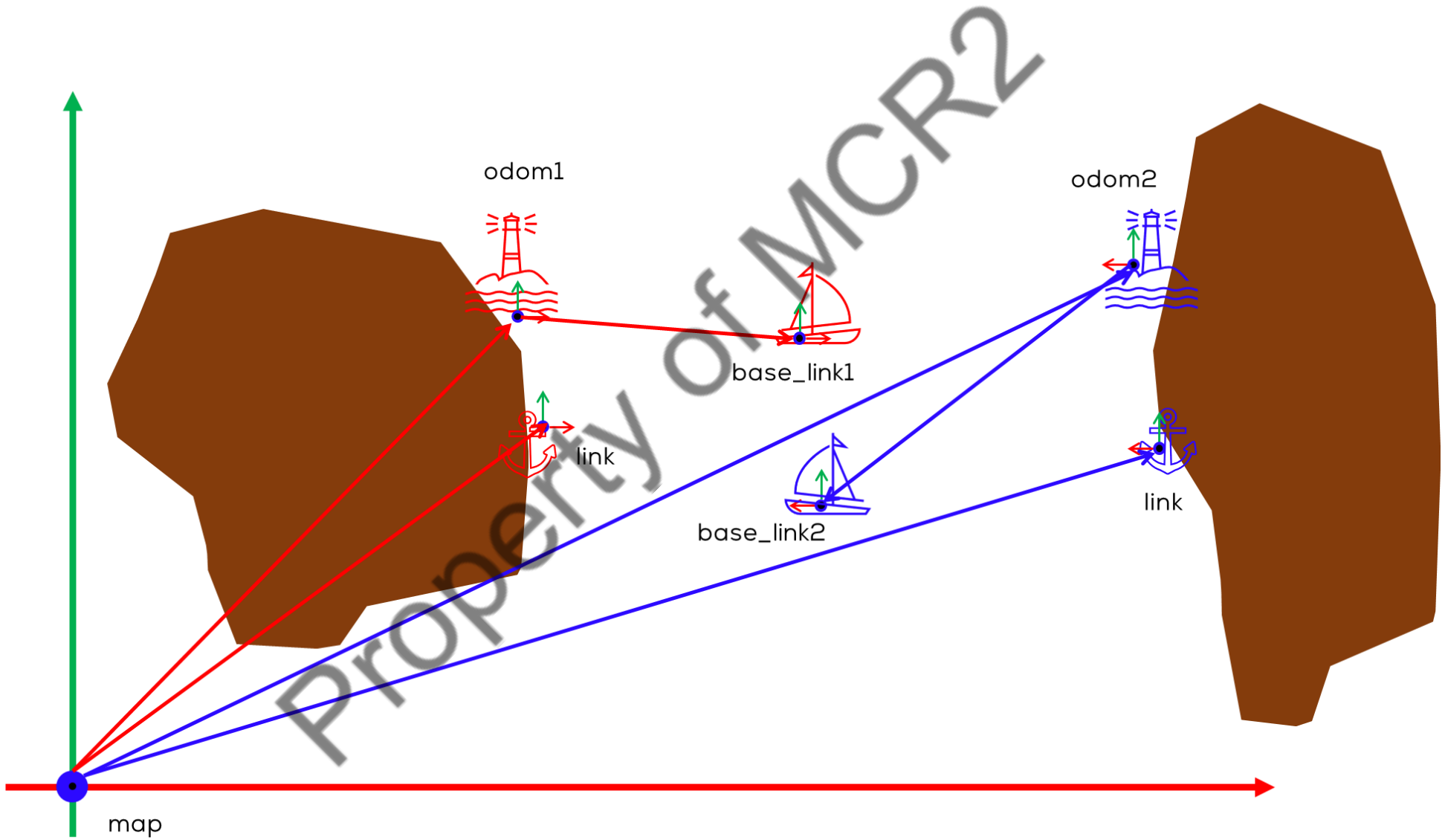


Coordinate frames mobile platforms

Motivation

- ROS2 follows a standard convention for defining coordinate frames, ensuring compatibility across sensors, localization, mapping, and control.
- Driver, model, and library developers need this convention for coordinate frames to improve integration and reuse of software components.
- Shared conventions for coordinate frames provide a specification for developers creating drivers and models for mobile bases.
- Similarly, developers creating libraries and applications can more easily use their software with various mobile bases compatible with this specification.





Coordinate frames mobile platforms

Coordinate Frames

In ROS2, mobile platforms follow a right-handed coordinate system that aligns with the REP-103 (Standard Units of Measure and Coordinate Conventions) and REP-105 (Coordinate Frames for Mobile Platforms) standards.

World/Earth

- Frame designed to allow the interaction of multiple robots in different map frames.
- If the application only needs one map the earth coordinate frame is not expected to be present.

Map

- World Fixed Coordinate frame.
- Z-axis pointing upwards.
- The pose of a mobile platform, relative to the map frame, should not significantly drift over time.
- Not continuous frame (pose of a mobile platform can change in discrete jumps)
    - In a typical setup, a localization component constantly re-computes the robot pose in the map frame based on sensor observations, therefore eliminating drift, but causing discrete jumps when new sensor information arrives.

Coordinate frames mobile platforms

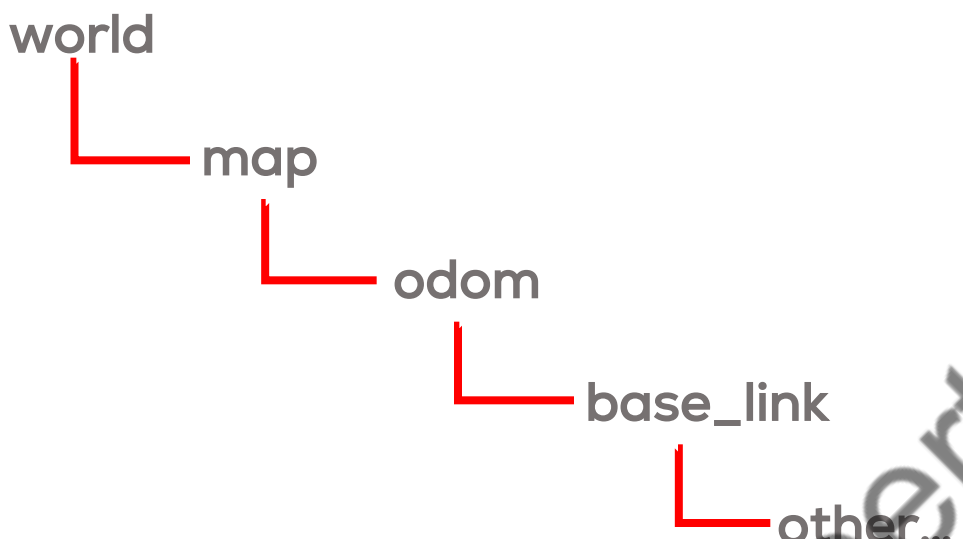
odom

- The coordinate frame called odom is a world-fixed frame.
- The pose of a mobile platform in the odom frame can drift over time, without any bounds.
- Odom frame useless as a long-term global reference.
- However, the pose of a robot in the odom frame is guaranteed to be continuous, without discrete jumps.
- In a typical setup the odom frame is computed based on an odometry source, such as wheel odometry, visual odometry, etc.

base_link

- The coordinate frame called base_link is rigidly attached to the mobile robot base.
- For every hardware platform there will be a different place on the base that provides an obvious point of reference

Transformation Hierarchy

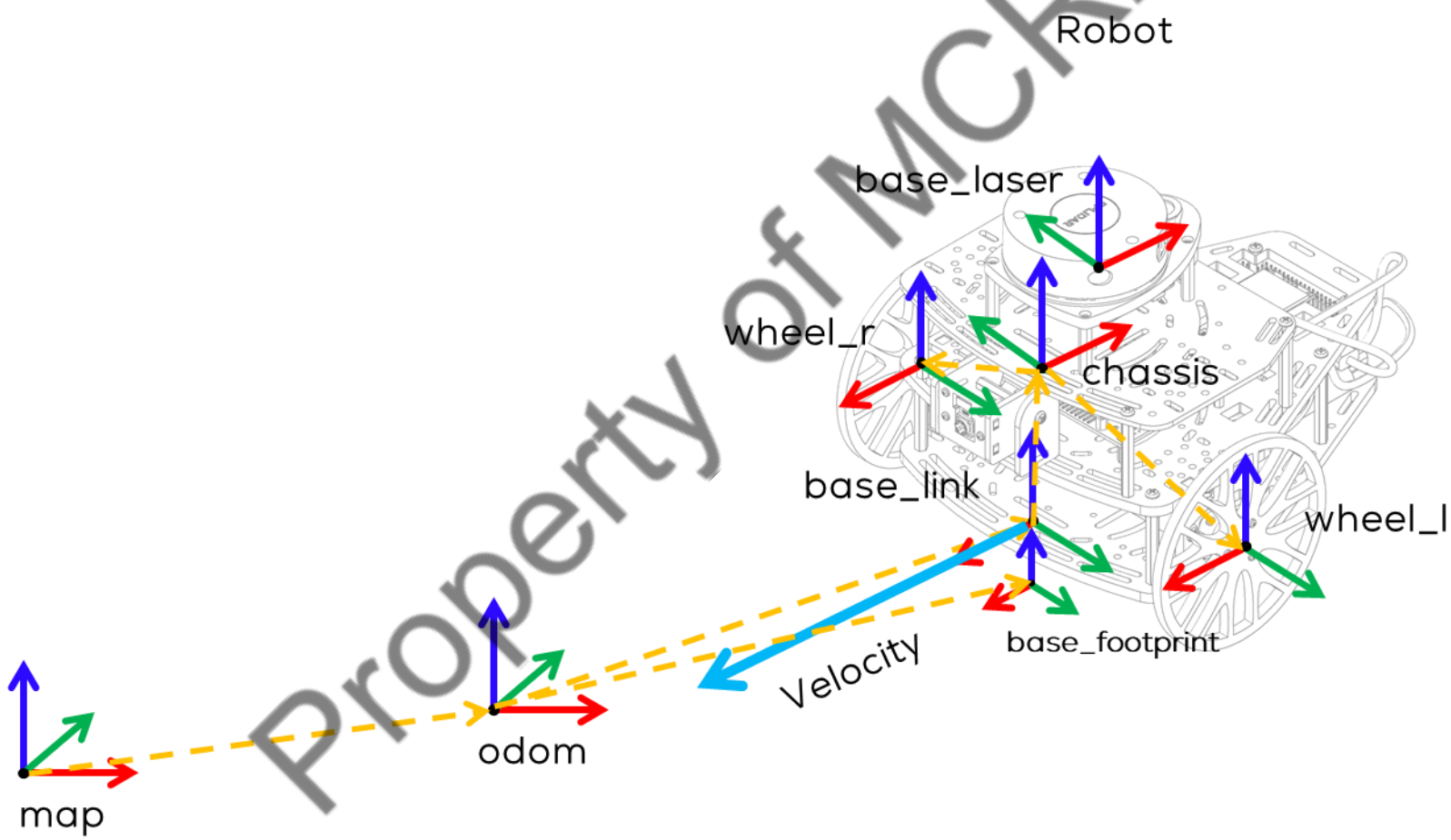


Extra frames

- The basic topology should stay the same, however it is fine to insert additional links in the graph which may provide additional functionality.
- Examples:

  base_footprint: Same as base_link but aligned with the ground.
- **base_laser**: LiDAR sensor frame (mounted on the robot).
- **camera_link**: Camera frame (if available).

Coordinate frames mobile platforms

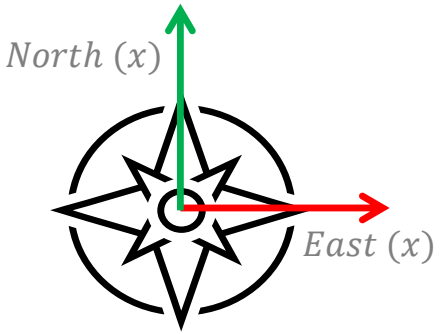


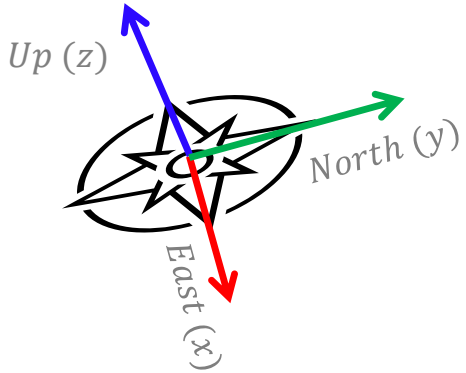
Localisation in ROS

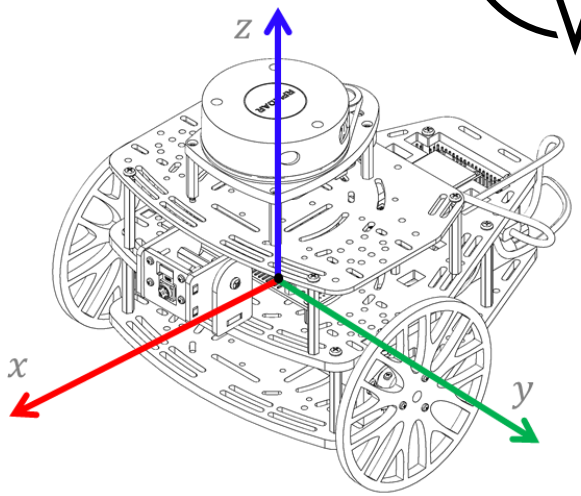
According to ROS best practices, the frame conventions are as follows

1. **Chirality**
    - All systems are right-handed. This means they comply with the right-hand rule.
2. **Axis Orientation**
    - In relation to a body the standard is:
        - x forward
        - y left
        - z up
- For short-range Cartesian representations of geographic locations, use the east-north up (ENU) convention:
    - X east
    - Y north
    - Z up

To avoid precision problems with large float32 values, it is recommended to choose a nearby origin, such as your system's starting position.







Localisation in ROS

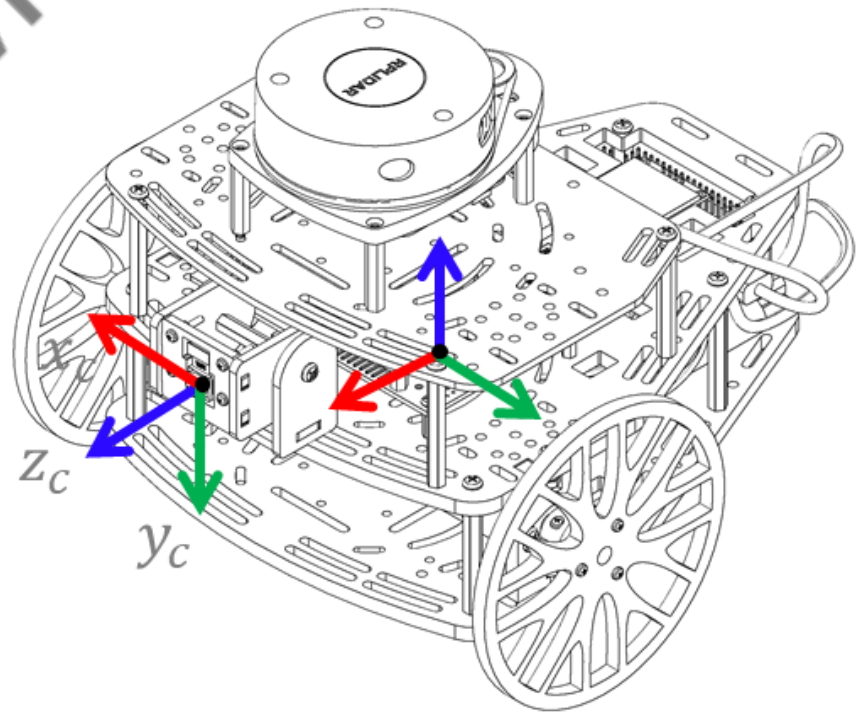
3. Suffix Frames

In the case of cameras, there is often a second frame defined with a "_optical" suffix. This uses a slightly different convention:

- z forward
- x right
- y down

For outdoor systems where it is desirable to work under the northeast down (NED) convention, define an appropriately transformed secondary frame with the "_ned" suffix:

- X north
- Y east
- Z down

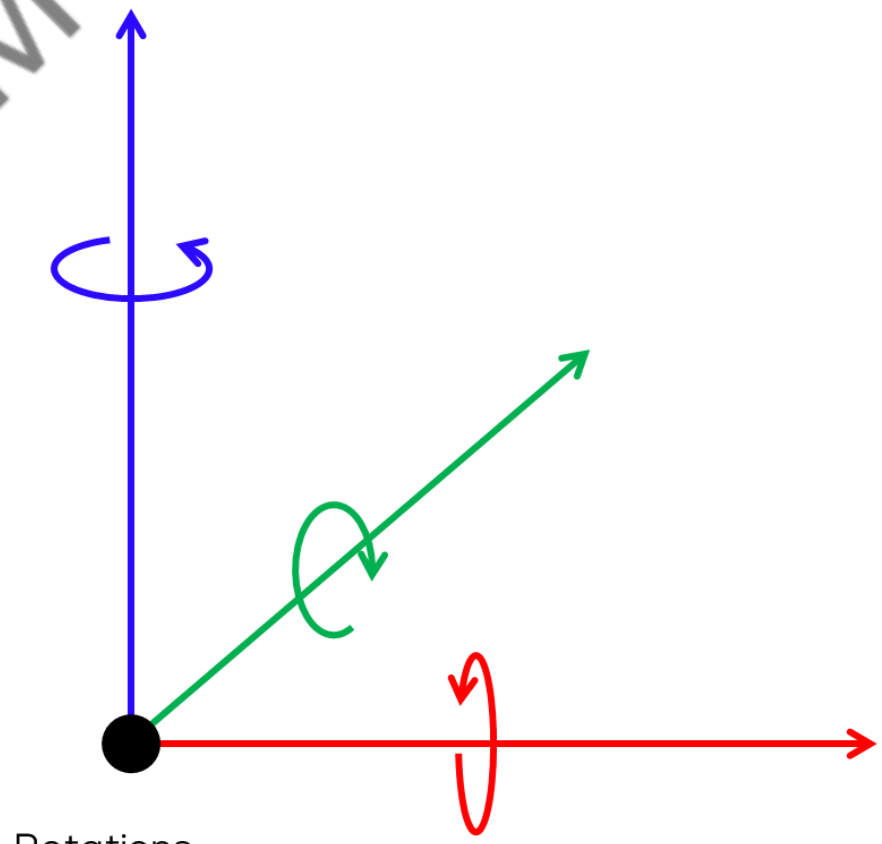


Localisation in ROS

4. Rotation Representation

There are many ways to represent rotations. The preferred order is listed below, along with rationale.

- Quaternion
  - •Compact representation
  - •No singularities
- rotation matrix
  - •No singularities
- fixed axis roll, pitch, yaw about X, Y, Z axes respectively
  - •No ambiguity on order
  - •Used for angular velocities
- Euler angles yaw, pitch, and roll about Z, Y, X axes respectively
  - •Euler angles are generally discouraged due to having 24 'valid' conventions with different domains using different conventions by default.



Rotations

Localisation in ROS (steps)

Localisation Node

1. Develop a simple node called "localisation" that subscribes to the velocity topics of the simulated/real robot.
2. Use this information to create an Odometry Message and publish the message in an *odom* topic.
3. Verify that the results are correct using the "*rqt_plot*" or the *"rqt_multiplot"*

Coordinate transform node (can be inside the joint state publisher)

1. Define a static transformation between the frames *"odom"* and *"map"* (can be set in the launch file)
2. Make a dynamic transform between the *"odom"* frame and the *"base_footprint*" and/or *"base_link"* frame using the odometry information.

URDF File

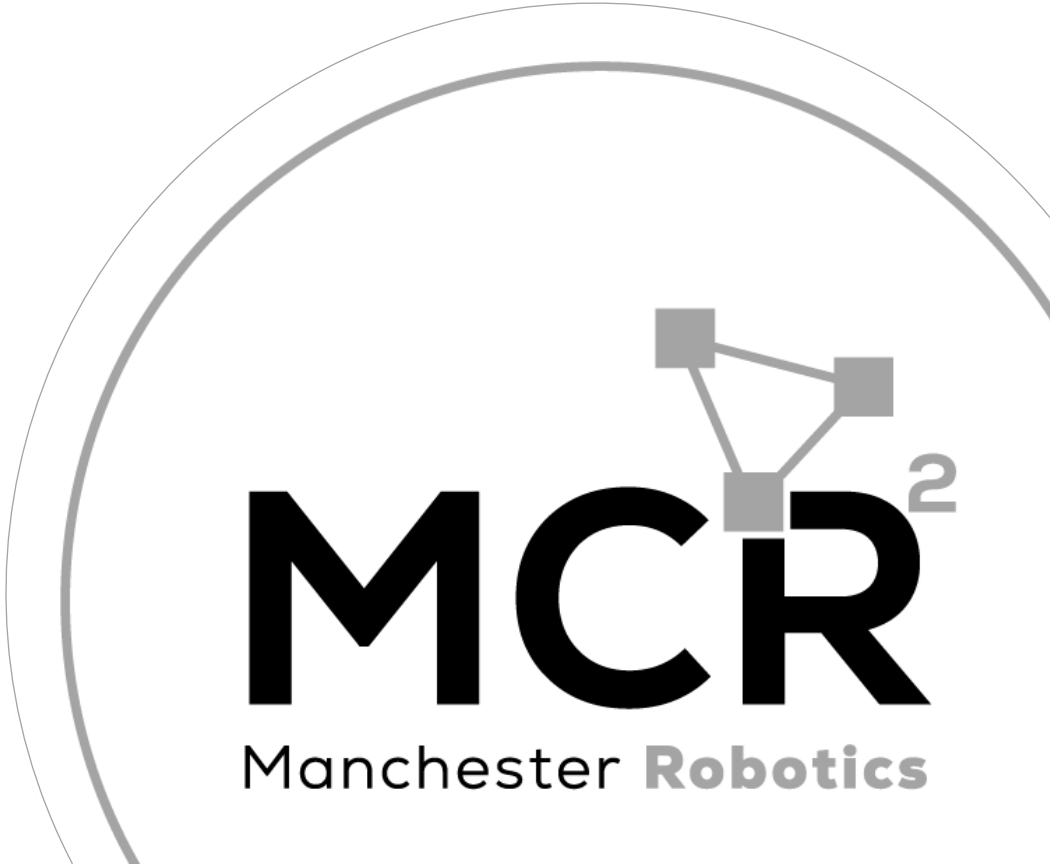
1. Define a "*base_footprint*" frame corresponding to the position in 2D of the robot.
2. Make a static joint for the "*base_footprint*" or "*base_link*" and *"chassis*".
3. Describe the rest of the robot links and joints.
4. Use the robot State publisher to read the URDF file and publish the transformations for your robot.

Joint state publisher

1. Develop a node to publish to the joints of your robots; use the "*sensor_messages/JointState*" message.
2. Transform the values of joints given by the odometry transformed.

Thank you

{Learn, Create, Innovate};



T&C

Terms and conditions

{Learn, Create, Innovate};

MCR^2

Manchester Robotics

Terms and conditions

- *THE PIECES, IMAGES, VIDEOS, DOCUMENTATION, ETC. SHOWN HERE ARE FOR INFORMATIVE PURPOSES ONLY. THE DESIGN IS PROPRIETARY AND CONFIDENTIAL TO MANCHESTER ROBOTICS LTD. (MCR2). THE INFORMATION, CODE, SIMULATORS, DRAWINGS, VIDEOS PRESENTATIONS ETC. CONTAINED IN THIS PRESENTATION IS THE SOLE PROPERTY OF MANCHESTER ROBOTICS LTD. ANY REPRODUCTION, RESELL, REDISTRIBUTION OR USAGE IN PART OR AS A WHOLE WITHOUT THE WRITTEN PERMISSION OF MANCHESTER ROBOTICS LTD. IS STRICTLY PROHIBITED.*
- *THIS PRESENTATION MAY CONTAIN LINKS TO OTHER WEBSITES OR CONTENT BELONGING TO OR ORIGINATING FROM THIRD PARTIES OR LINKS TO WEBSITES AND FEATURES IN BANNERS OR OTHER ADVERTISING. SUCH EXTERNAL LINKS ARE NOT INVESTIGATED, MONITORED, OR CHECKED FOR ACCURACY, ADEQUACY, VALIDITY, RELIABILITY, AVAILABILITY OR COMPLETENESS BY US.*
- *WE DO NOT WARRANT, ENDORSE, GUARANTEE, OR ASSUME RESPONSIBILITY FOR THE ACCURACY OR RELIABILITY OF ANY INFORMATION OFFERED BY THIRD-PARTY WEBSITES LINKED THROUGH THE SITE OR ANY WEBSITE OR FEATURE LINKED IN ANY BANNER OR OTHER ADVERTISING.*