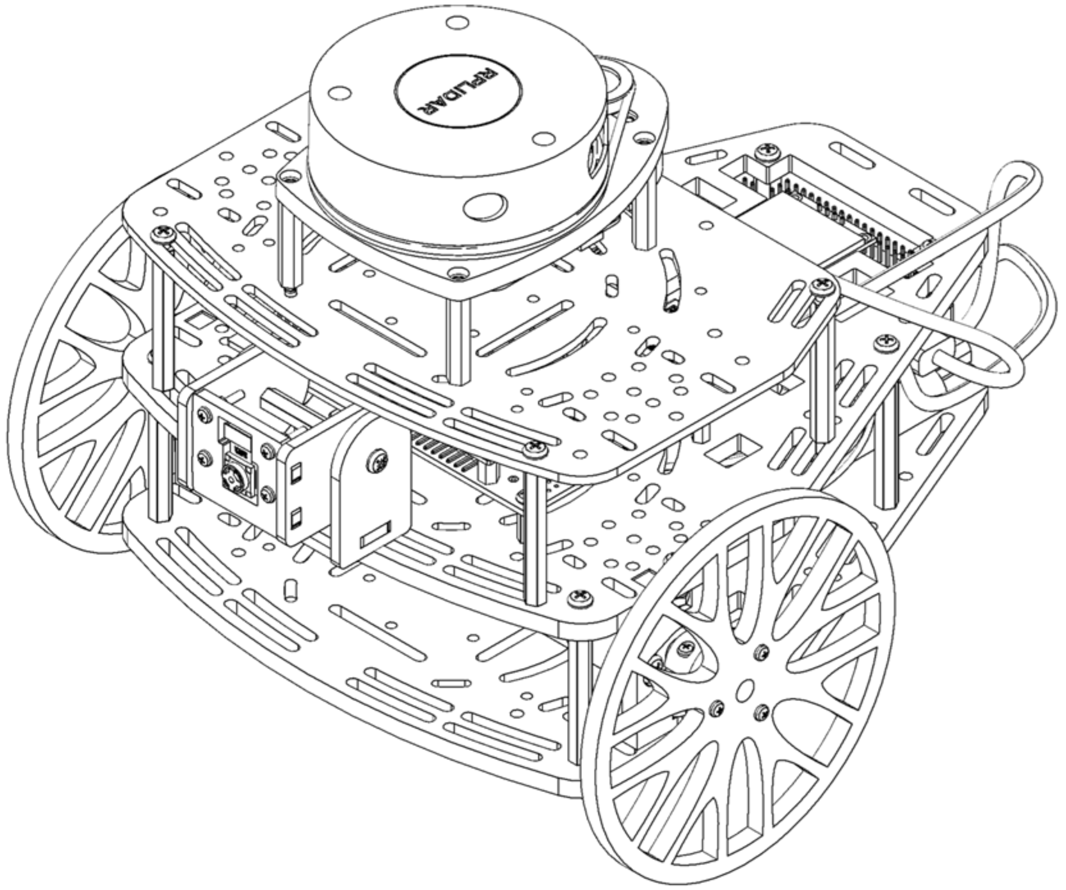
Challenges

Mini challenge 4

{Learn, Create, Innovate};

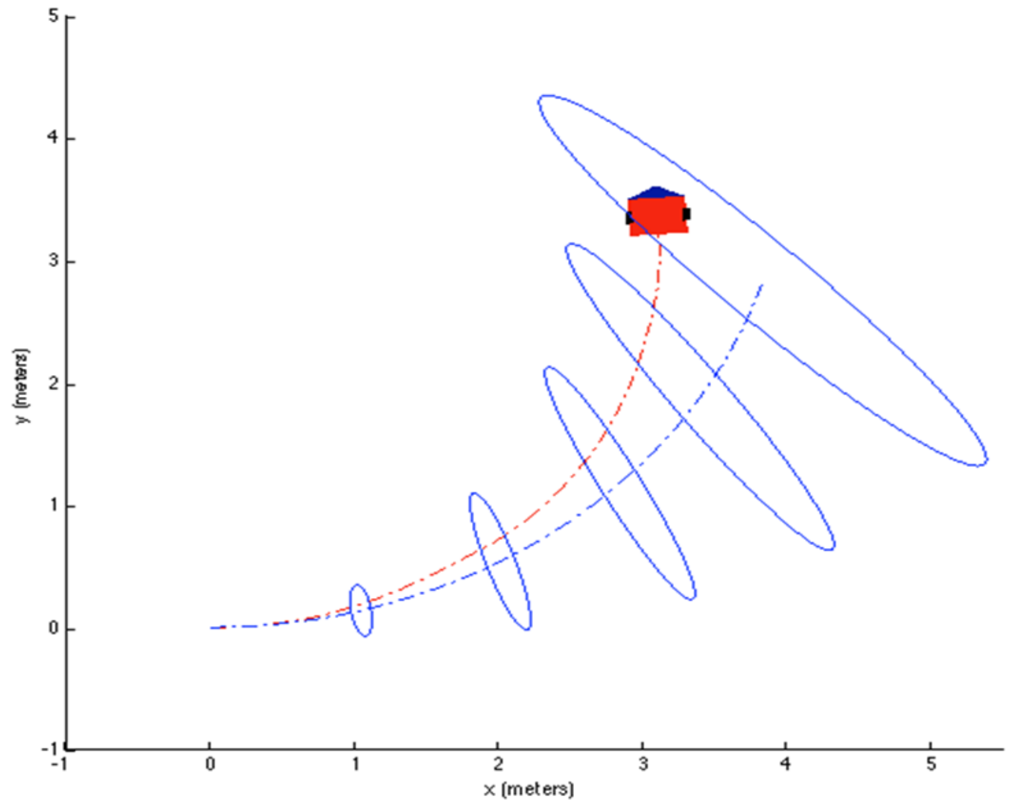
Mini challenge 4

- This challenge is intended for the student to review the concepts introduced this week.
- This challenge aims to show the students how the noise and other perturbations affect robotic platforms.
- This challenge will be divided into different sections.



Mini challenge 4

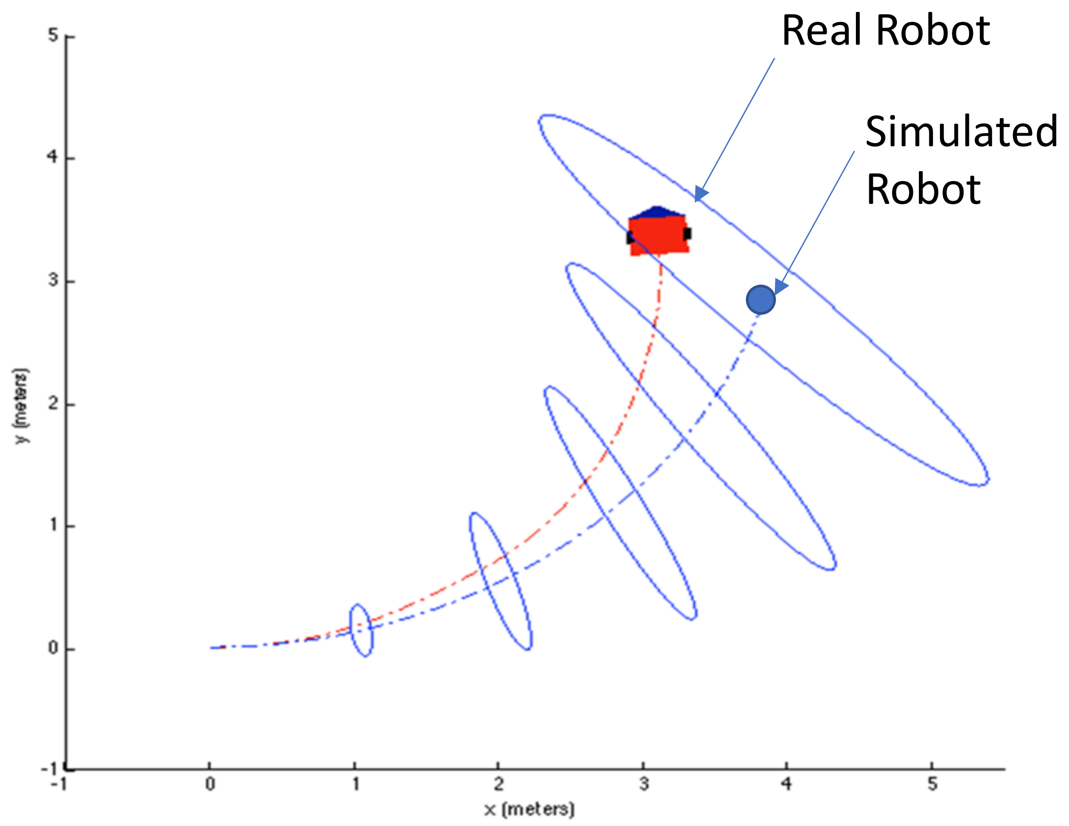
- This challenge will use the simulator developed for Mini-challenge 2 to observe the behaviour of the robot.
- The student must be able to plot the confidence ellipsoid for the simulated Puzzlebot.
- The student must define some tests to estimate and analyse the position uncertainty and calibrate the error constants k_r and k_l



Mini challenge 4

Task 1:

- Plot the covariance ellipsoid for the robot's pose using the uncertainty propagation model and the different tests to analyse uncertainty.
- For this part of the challenge, the student must complete the 3x3 pose covariance matrix of the "odometry" message in the localisation node, shown in the previous challenge.



Odometry Message

Odom Message:

- **Odometry():**

```
odometry.header.stamp = rospy.Time.now()              #time stamp
odometry.header.frame_id = "world"                     #parent frame (joint)
odometry.child_frame_id = "base_link"                  #child frame
odometry.pose.pose.position.x = 0.0                    #position of the robot "x" w.r.t "parent frame"
odometry.pose.pose.position.y = 0.0                    # position of the robot "x" w.r.t "parent frame"
odometry.pose.pose.position.z = (Wheel Radius)         #position of the robot "x" w.r.t "parent frame"
odometry.pose.pose.orientation.x = 0.0                 #Orientation quaternion "x" w.r.t "parent frame"
odometry.pose.pose.orientation.y = 0.0                 #Orientation quaternion "y" w.r.t "parent frame"
odometry.pose.pose.orientation.z = 0.0                 #Orientation quaternion "z" w.r.t "parent frame"
odometry.pose.pose.orientation.w = 0.0                 #Orientation quaternion "w" w.r.t "parent frame"
odometry.pose.covariance = [0]*36                      #Pose Covariance 6x6 matrix (empty for now)
odometry.twist.twist.linear.x = 0.0                    #Linear velocity "x"
odometry.twist.twist.linear.y = 0.0                    #Linear velocity "y"
odometry.twist.twist.linear.z = 0.0                    #Linear velocity "z"
odometry.twist.twist.angular.x = 0.0                   #Angular velocity around x axis (roll)
odometry.twist.twist.angular.y = 0.0                   #Angular velocity around x axis (pitch)
odometry.twist.twist.angular.z = 0.0                   #Angular velocity around x axis (yaw)
odometry.twist.covariance = [0]*36                     #Velocity Covariance 6x6 matrix (empty for now)
```

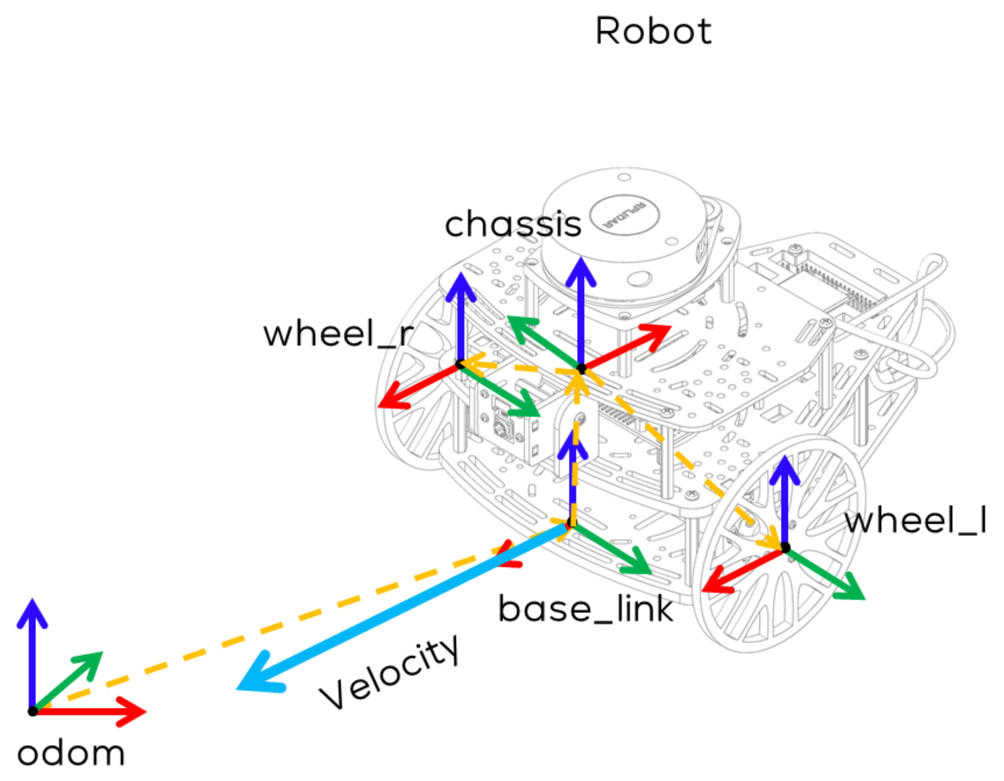
- The odometry message is specially used for publishing the estimated pose, velocity and covariance of a robot.
- This message is read by RVIZ and plotted automatically.
- ROS automatically relates the transformation tree with the header and child frames IDs to plot the results in RVIZ alongside the covariance ellipsoids (if implemented).

Odometry Message

- According to the definition of the "Odometry.msg" from the library "nav_msgs", the position and orientation of the robot must be defined with respect to the parent frame. As an example, in the side figure the parent frame would be "odom"

```
# Position of the robot (x,y,z) and Orientation of the robot
# w.r.t "frame" (parent_frame)

odometry.pose.pose.position.x = 0
odometry.pose.pose.position.y = 0
odometry.pose.pose.position.z = 0
odometry.pose.pose.orientation.x = 0
odometry.pose.pose.orientation.y = 0
odometry.pose.pose.orientation.z = 0
odometry.pose.pose.orientation.w = 0.0
```

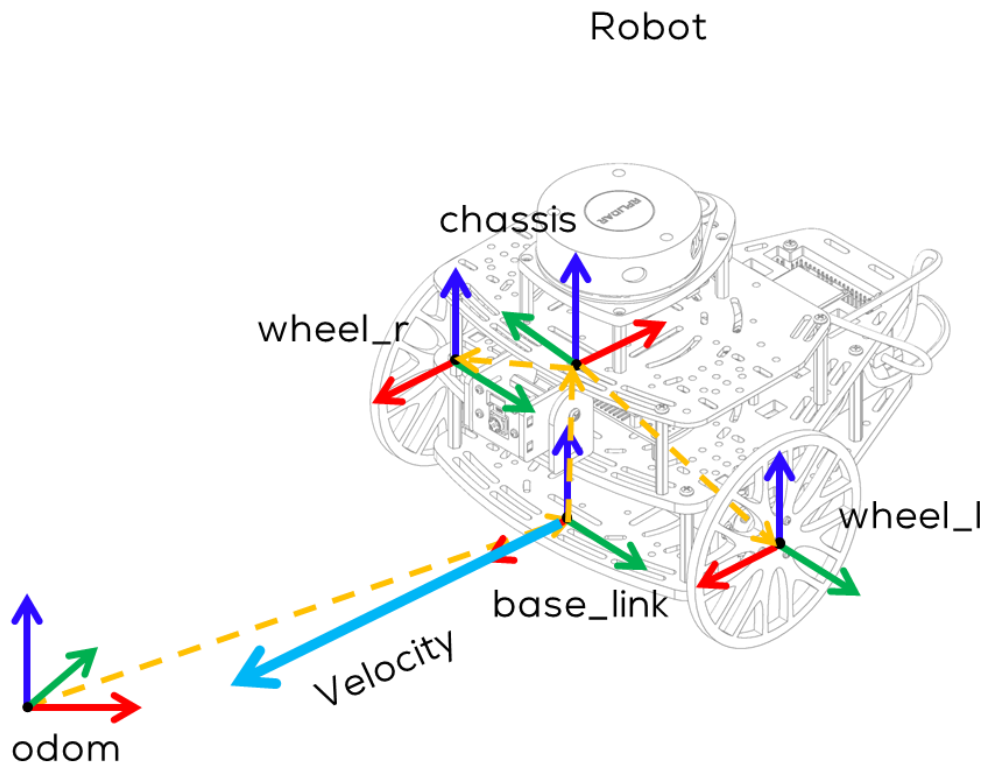


Odometry Message

- According to the definition of the "Odometry.msg" from the library "nav_msgs", the velocity of the robot must be defined with respect to the child frame. For the example shown in the figure this must be "base_link"

```
# Velocity, linear and angular, of the robot (ẋ,ẏ,ż,φ̇,ψ̇,θ̇) and
# w.r.t "child_frame"

odometry.twist.twist.linear.x = 0.0
odometry.twist.twist.linear.y = 0.0
odometry.twist.twist.linear.z = 0.0
odometry.twist.twist.angular.x = 0.0
odometry.twist.twist.angular.y = 0.0
odometry.twist.twist.angular.z = 0.0
```



Odometry Message

- The odometry messages contains two different covariance matrices for the position and for the speed.
- The pose covariance matrix is a 6x6 matrix, for a 6DOF robot. The order of the parameters is as follows

```
(x, y, z, rotation about X axis, rotation about Y axis, rot
ation about Z axis)
```

- In other words (using the class convention):

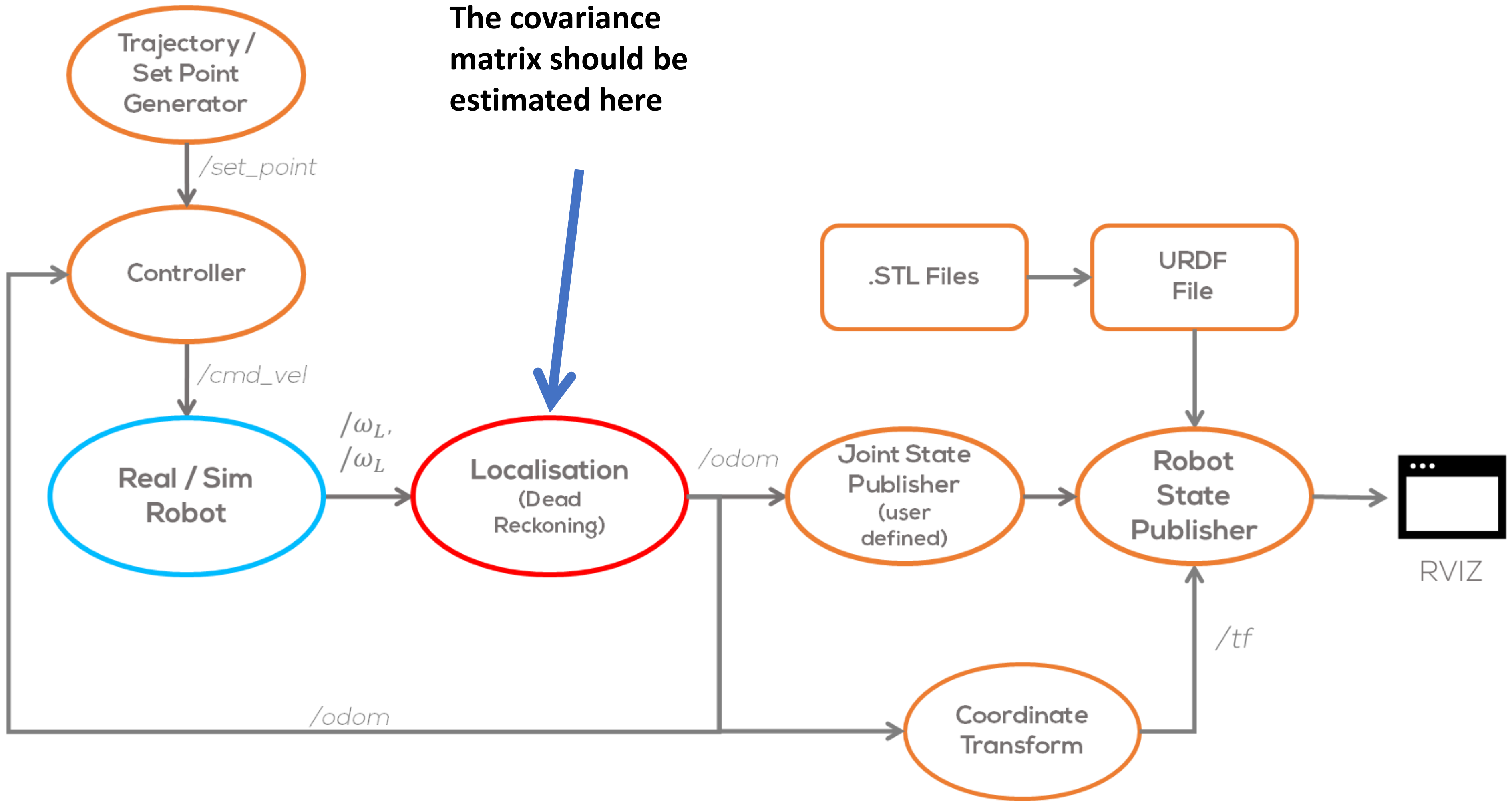
$$[x \quad y \quad z \quad \varphi \quad \psi \quad \theta]$$

- Therefore, the pose covariance matrix can be defined as follows:

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} & \sigma_{x\varphi} & \sigma_{x\psi} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} & \sigma_{y\varphi} & \sigma_{y\psi} & \sigma_{y\theta} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} & \sigma_{z\varphi} & \sigma_{z\psi} & \sigma_{z\theta} \\ \sigma_{\varphi x} & \sigma_{\varphi y} & \sigma_{\varphi z} & \sigma_{\varphi\varphi} & \sigma_{\varphi\psi} & \sigma_{\varphi\theta} \\ \sigma_{\psi x} & \sigma_{\psi y} & \sigma_{\psi z} & \sigma_{\psi\varphi} & \sigma_{\psi\psi} & \sigma_{\psi\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta z} & \sigma_{\theta\varphi} & \sigma_{\theta\psi} & \sigma_{\theta\theta} \end{bmatrix}$$

- For this challenge the velocity covariance matrix will be zero.

The covariance matrix should be estimated here



Mini challenge 4

Tips:

The robot pose $\mathbf{s}_k$ is given by a random variable

$$\mathbf{s}_k \sim \mathcal{N}(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Where, the mean vector $\boldsymbol{\mu}_k$ is the best estimate of the pose, and $\boldsymbol{\Sigma}_k$ is the covariance matrix.

The pose $\boldsymbol{\mu}_k$ is the one given by your dead reckoning localisation node.

The covariance $\boldsymbol{\Sigma}_k$ can be approximated by:

$$\boldsymbol{\Sigma}_k = \mathbf{H}_k \boldsymbol{\Sigma}_{k-1} \mathbf{H}_k^{\mathrm{T}} + \mathbf{Q}_k$$

Where, $\boldsymbol{\Sigma}_k$is a 3x3 covariance matrix.

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix}$$

- θ represents rotation around z-axis (yaw)

Mini challenge 4

$\mathbf{H}_k$ is a 3x3 Linear model Jacobian of the robot.

$$\mathbf{H}_k = \begin{bmatrix} 1 & 0 & -\Delta t \cdot v_k \cdot \sin(\mu_{\theta,k-1}) \\ 0 & 1 & \Delta t \cdot v_k \cdot \cos(\mu_{\theta,k-1}) \\ 0 & 0 & 1 \end{bmatrix}$$

- The matrix Q_k is the nondeterministic error matrix, given by

$$Q_k = \nabla_{\omega_k} \cdot \Sigma_{\Delta,k} \cdot \nabla_{\omega_k}^T$$

where,

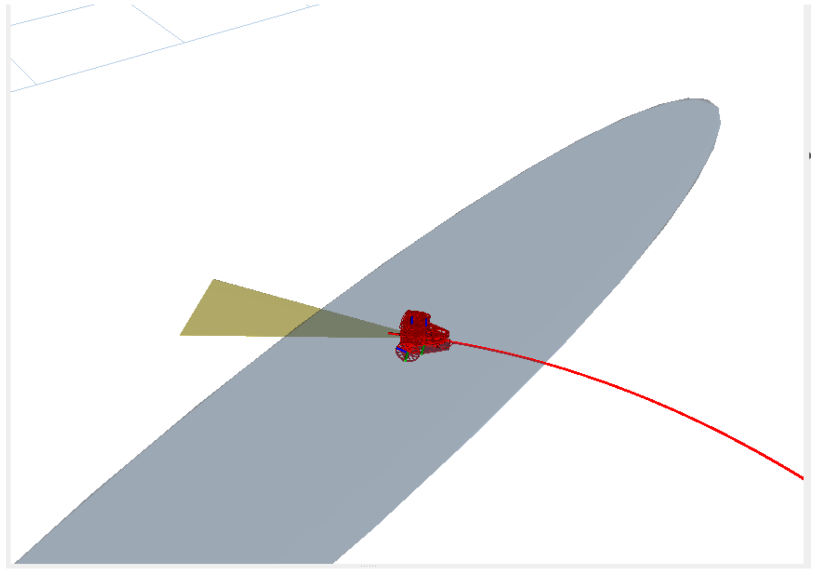
$$\Sigma_{\Delta,k} = \begin{bmatrix} k_r|\omega_{r,k}| & 0 \\ 0 & k_l|\omega_{l,k}| \end{bmatrix}$$

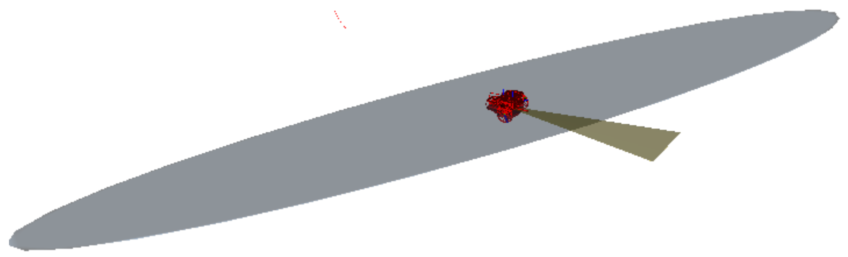
The values of k_r and k_l must be tuned according to some metric or test (Task 2).

$$\nabla_{\omega_k} = \frac{1}{2} r\Delta t \begin{bmatrix} \cos(s_{\theta,k-1}) & \cos(s_{\theta,k-1}) \\ \sin(s_{\theta,k-1}) & \sin(s_{\theta,k-1}) \\ \frac{2}{l} & -\frac{2}{l} \end{bmatrix}$$

Mini challenge 4

- The student must accommodate the matrix Σ_k into the covariance matrix of the odometry message.
- RVIZ automatically plots the covariance ellipsoid for the pose of the robot, given that the correct message and transformation are used (odometry message).
- As per the previous task, the student must define the transformation to be used.
- The students must define the required launch files for this activity.
- The simulation must be tested under different conditions, i.e., different speeds.
- The students must define a correct sampling time for the simulation .
- The students must solve the differential equations using numerical methods.
- The usage of any library is strictly **forbidden**.





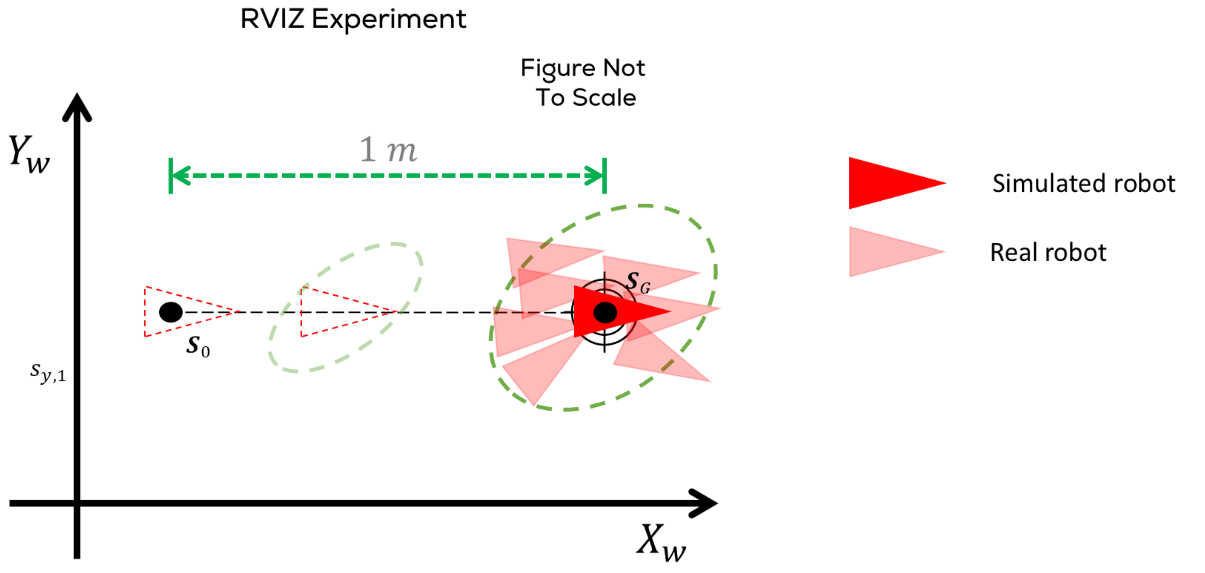
Mini challenge 4

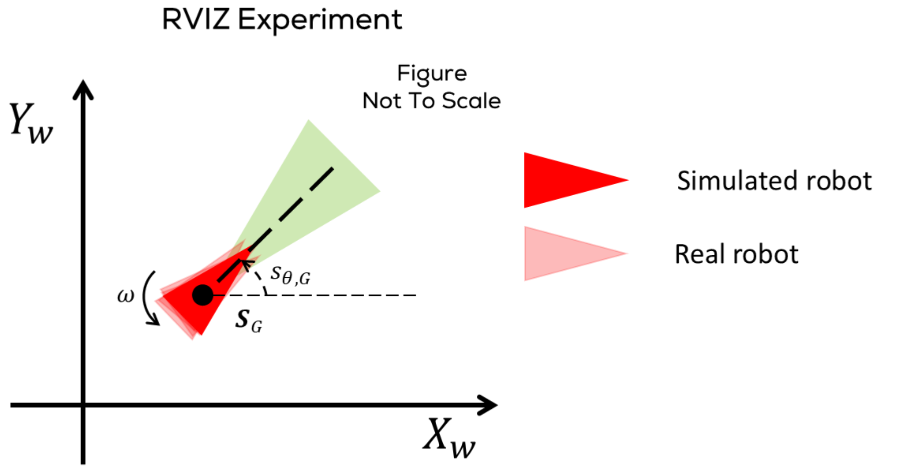
Task 2:

- Tune the parameters k_r and k_l of the matrix:

$$\Sigma_{\Delta,k} = \begin{bmatrix} k_r|\omega_{r,k}| & 0 \\ 0 & k_l|\omega_{l,k}| \end{bmatrix}$$

- Using any of the methodologies viewed in class.
- Prove that the values are well tunned by making an experiment with the real robot.
- As per the previous task, the students must define the required packages and files for this activity.





Rules

- This is challenge **not** a class. The students are encouraged to research, improve tune explain their algorithms by themselves.
- MCR2(Manchester Robotics) Reserves the right to answer a question if it is determined that the questions contains partially or totally an answer.
- The students are welcomed to ask only about the theoretical aspect of the classed.
- No remote control or any other form of human interaction with the simulator or ROS is allowed (except at the start when launching the files).
- It is **forbidden** to use any other internet libraires with the exception of standard libraires or NumPy.
- If in doubt about libraires please ask any teaching assistant.
- Improvements to the algorithms are encouraged and may be used as long as the students provide the reasons and a detailed explanation on the improvements.
- All the students must be respectful towards each other and abide by the previously defined rules.
- Manchester robotics reserves the right to provide any form of grading. Grading and grading methodology are done by the professor in charge of the unit.