Lecture 5

2025

INFO 802

Master Advanced Mechatronics

Luc Marechal

UNIVERSITÉ SAVOIE MONT BLANC

POLYTECH ANNECY-CHAMBÉRY

ROS

Turtlesim
Turtle to target

Assignement

turtle_to_target (Python)

Assignement

turtle_to_target (Python)

https://github.com/LucMarechal/ROS_Lectures/tree/master/Assignement_Lecture5

Download the files: *spawn_turtle.py*
turtle_to_target.py

Place them in the appropriate folders inside the *turtlesim_tutorials_pkg* package

(create the package if it does not exist yet)

- *spawn_turtle.py* : node that randomly spawn the target (i.e turtle2) in the turtlesim window
- *turtle_to_target.py* : the node that makes the turtle1 move to the turtle2

Assignement

turtle_to_target (Python)

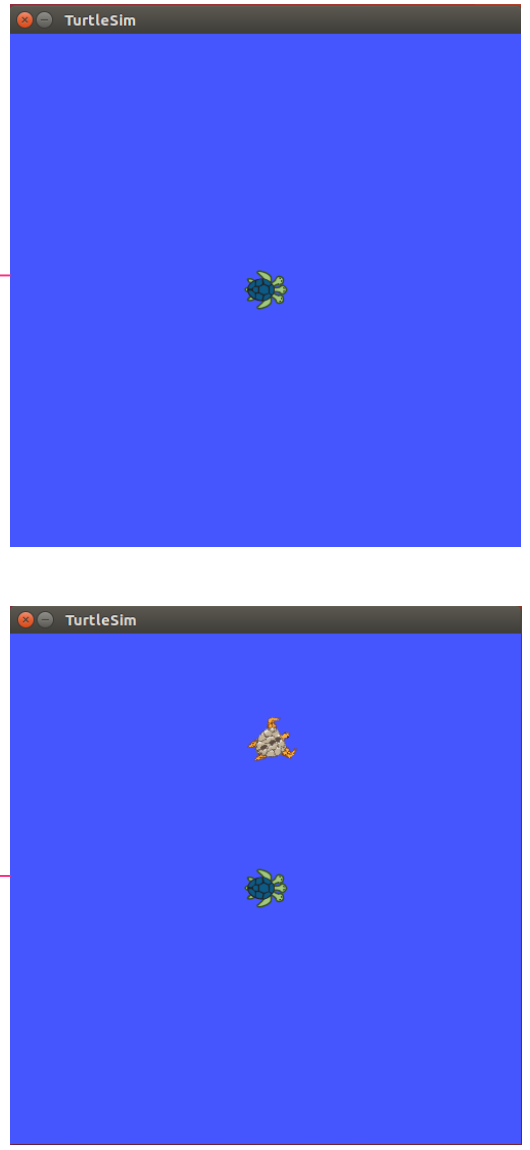
Run the **turtlesim_node** and the **spawn_turtle** node

Find which topics are running on the ROS system

```
luc@USMB:~$ roscore
```

```
luc@USMB:~$ rosrun turtlesim turtlesim_node
```

```
luc@USMB:~$ rosrun turtlesim_tutorials_pkg spawn_turtle.py
```



Assignement

turtle_to_target (Python)

At least, answer the following questions:

- What are the Pose and Twist message like?
- Which library are these messages coming from ?
- What should we import in the python header file to use Pose and Twist objects?

- How can we get the Pose of the turtle?
- How can we get the Pose of the target?

- How can we send velocity command to the turtle?
- What should we then import in the python header file ?

- Explain what the `spawn_turtle` node is doing

Assignement

turtle_to_target (Python)

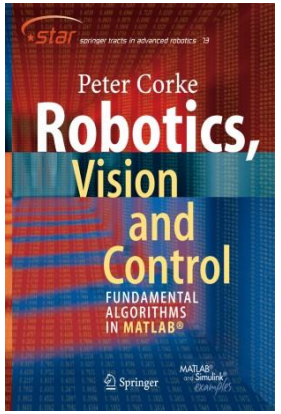
- Create a node called *turtle_to_target* that automatically move the turtle1 to the turtle2. For that you will use linear velocity and angular velocity to control the turtle1.

- The forward velocity is defined as a constant gain of multiplied by the distance between the target turtle2 and the turtle1. This means that the forward velocity is higher the further you are away from the target, and goes to zero as you approach the target.

- The angular velocity is calculated similarly with a gain multiplied by the difference in angle between the line that is directly connecting the turtle and the goal position, and the angular pose of the robot itself (check the meaning of the atan2 in the steering angle computation). This causes the robot to adjust its own theta to eventually move in a straight line to the target.

Assignement

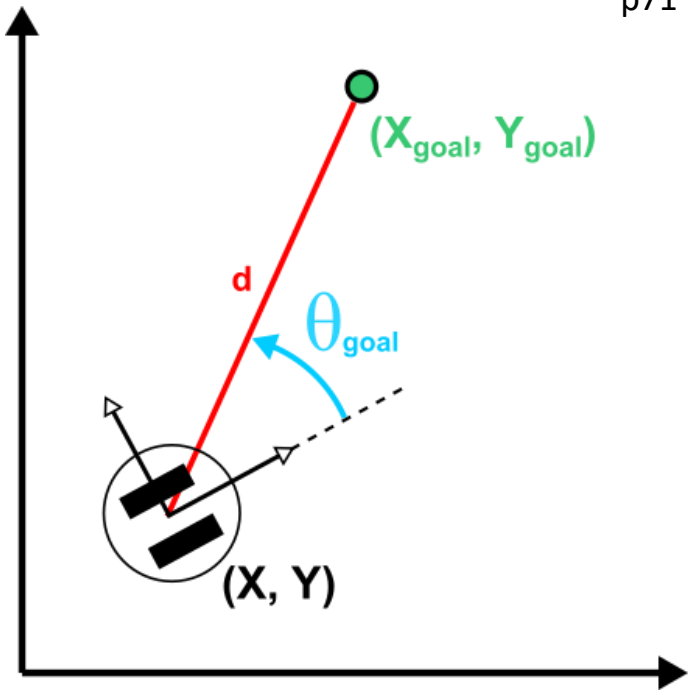
Moving to a Point (x,y) in the 2D plane



p71

- Euclidean distance: $distance = \sqrt{(x_{goal} - x)^2 + (y_{goal} - y)^2}$
- Orientation: $\theta_{goal} = \text{atan2}\frac{(y_{goal} - y)}{(x_{goal} - x)}$
- Proportional Controller: Velocity $v = K_v \times distance$

  Steering angle $\gamma = K_h \times \theta_{goal}$



The robot's velocity is proportional to its distance to the goal

Assignement

turtle_to_target (Python)

- Declare the following global variables

- Create a subscriber to get the x and y coordinates of the target (target = turtle2).

  X_target and Y_target

- Create a subscriber to get the x and y coordinates of your turtle (your turtle = turtle1).

Assignement

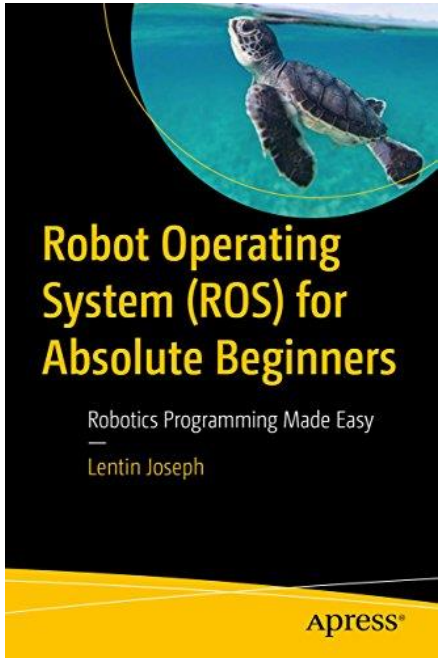
turtle_to_target (Python)

Create a launch file named *turtlesim_target.launch* so it launches :

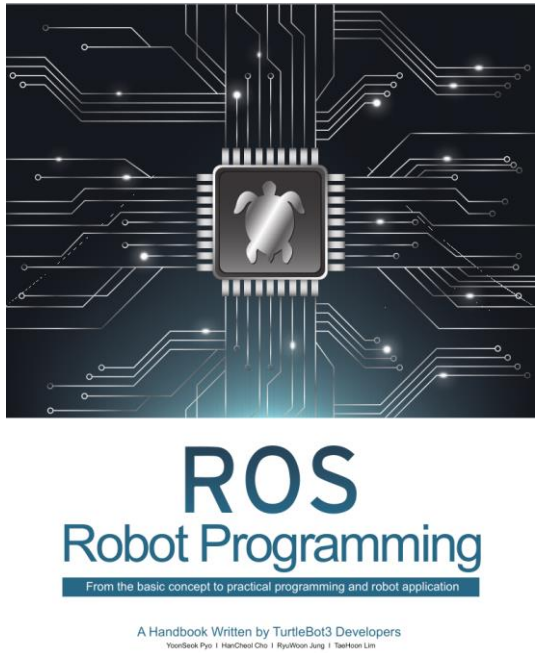
- *turtlesim_node*
- *spawn_turtle.py*
- *turtlesim_target.py*

Further References

- ROS Turtlesim tutorials
  - wiki.ros.org/turtlesim/Tutorials/



Chapter 5



Chapter 10

- ROS Cheat Sheet
  - https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/
  - https://kapeli.com/cheat_sheets/ROS.docset/