

Robotics Project 2015-2016 (v 0.9)

Matteo Matteucci – Gianluca Bardaro

“Il Kobra non è un serpente ...”

Donatella Rettore – 1981

Introduction

This year project is divided in steps; each of them is worth some points out of the 5/32 points available for the final mark.

Some preliminary important notes:

- We have decided to follow a sort of “homework approach” so you should do the homework to better understand some key concepts from the course ... indeed a few question comes from your colleague observations so blame them!
- We have not decided yet how much each part is worth, we will decide depending on the overall distribution of results in the class to harmonize the overall score.
- Some data might be missing, some data might be useless, do not hesitate to write us by email, for instance, to ask the weight of the robot and learn that it is 14Kg.

Part 1 – Gazebo model for the Kobra robot

You have to create a model of an existing robot called Kobra

<http://www.nuzoo.it/it/prodotti-robot/kobra-videosorveglianza-mobile>

It is a mobile robot moved by a skidsteer system based on two paired wheels (we refer to model with the wheels not the one with the trucks on the website). The minimum required element for the model are:

1. Body of the robot with the correct shape and size.

2. Four wheels, each one actuated by its own motor. Wheels on each side are synchronized and must move together. This is necessary in order to simulate a skidsteer. (Other solution with the same behavior are accepted)
3. Straight vertical arm with a camera on top.
4. Pan tilt camera, it has to be placed at the end of a chain of two 1dof joints placed on the vertical arm.
5. Hokuyo laser on the front of the robot.

All the details of the geometrical details (i.e. shape of the wheels, position of the sensor, etc.) of the robot are available on the robot website, and you can derive useful information from the video as well!

The missing info we have not been able to derive from the robot datasheet and the robot web page is the size of the wheels which we have measured being 4.5cm thick and have a 14.5cm diameter (wheelbase 18cm, track 49.5cm). The two links of the pan-tilt are obtained through dynamixel motors, but you just need to model reasonable joints with 3cm links, not the real servos.



Part 2 – Gazebo plugin and ROS setup

First of all you need to implement any kind of Gazebo plugin required to the correct operation of each element of the robot. Here the minimum requirement:

1. Odometry and control:
 - a. Convert from the speed of the wheels to the linear velocity x and angular velocity ω .
 - b. Integrate the velocity of the robot to estimate a relative displacement after each velocity reading.

- c. Convert from commands sent to the robot using linear velocity x and angular velocity ω to the wheel speed.
 - d. Manage the four motors to achieve a proper skidsteer movement.
2. Camera control: manage the two joints controlling the camera to simulate a pan-tilt movement in terms of pan and tilt.
3. Laser: if necessary, manage the laser accordingly using a plugin.

Then you have to prepare the interface between Gazebo and the ROS architecture. It is strongly suggested to do this alongside the development of the plugins. Do your own choices for the name of the topics and the types of messages, but remember to follow standards and existing interfaces.

1. Odometry and control:
 - a. `/robot/odometry`: publish the current speed of the robot and the relative displacement at the simulation frequency.
 - b. `/robot/cmd_velocity`: subscribe to this topic to control the robot.
2. Sensors:
 - a. `/robot/image`: publish images coming from the camera.
 - b. `/robot/camera_info`: publish a topic with the characteristics of the camera.
 - c. `/robot/laser_scan`: publish the scans from the laser.
 - d. `/robot/ptz`: subscribe to a custom message defined as `ptz.msg` containing the pan and tilt commands as floats (radians) plus a third float representing the zoom (not used)
3. `tf`:
 - a. Publish a low frequency transform of the real pose of the robot taken from the simulator, available only for testing purpose `/robot_gt`.
 - b. Publish the static transform for each sensor with respect to the `/base_link` frame.
 - c. Publish the transform for each joint controlling the pan-tilt camera with respect to the proper frame.

Part 3 – Odometry ROS node

In this part, you have to develop a ROS node to manage the odometry measurements and integrate them to provide an high frequency absolute position estimate of the

robot using tf from the /odom frame (initially the /base_link frame coincide with the /odom frame).

Three versions of the integration algorithm needs to be implemented and should be selectable through a ROS parameter from a launch file:

1. Simple integration using Euler method (Euler)
2. Runge-Kutta integration (Runge-Kutta)
3. Exact integration (Exact)

You can find the details of the three methods in these slides

http://www.dis.uniroma1.it/~oriolo/amr/slides/OdometricLocalization_Slides.pdf

Note: in the next days a bag file will be provided with commands from a joypad (sensor_msgs/Joy.msg) to control the robot around and test the camera, in the meanwhile, for your testing and debugging, you can use rostopic or the turtle_teleop node.

Part 4 – Navigation and planning

Available soon ... stay tuned!