1. Modern Robotics

(diap 2) Nowadays, there are increasingly more robotic applications for mass public. Beyond the classic applications in industrial environments and vehicle assembly, robots are used, for example, in the packaging of food or the management of warehouses.

This growing pushfulness of robotic technology implies to train professionals in this sector that even take the current borders further and help create new robotic applications that serve people.

1. Robotic Software

(diap 3) One of the factors that allow robots to deploy intelligence and deal with real situations offering similar robustness to the human is its software, its programming. Given the complexity that is usually involved in such software, it tends more and more to be distributed in several layers (drivers, middleware, applications) that operate together to provide intelligence to the robot. The support to link these modules and get them to interoperate is what ROS offers us as middleware, so that we can build our robotic applications on it, including the use of simulators, real-time operation and even the use of heterogeneous hardware.

(diap 4) We present a ROS-based framework whose objective is to teach students the basics of robotics, focusing on the algorithm rather than on the middleware, which will remain hidden. This framework is JdeRobot-Academy, that relies on ROS-Kinetic and Gazebo 7 simulator.

1. Teaching Robotics

(diap 5) Currently training in robotics appears in secondary school and is mainly carried out in the university, with bachelor's degrees and specific postgraduates. This learning environment aims to extend its use in this last phase of people's learning, in university courses, university careers and postgraduate programs mainly. It is ideal for robotics courses in computer science or in introductory courses in robotics. It has been successfully used in some degrees and master's degrees at the Rey Juan Carlos University of Madrid, as well as in some introductory courses in robotics and drones at this university. It has also been used in the first two editions of the championship of robot programming “PROGRAM-A-ROBOT CHALLENGE”.

1. What is JdeRobot-Academy

(diap 6) The teaching environment consists of a collection of independent practices (FIGURE) that pose a specific robotic problem (typically autonomous behavior) and the student has to program the intelligence of the robot that solves it. Attractive practices have been chosen, according to applications that are recently arriving to society: autonomous cars, robotized vacuum cleaners, drones, etc. The idea of ​​this choice is that the student can relate them to applications of real life, in which it is easier and more motivating to see the direct real utility.

Each practice has a design based on three components:

* First, in the lower layer is the robot that is wanted to perform some task in a certain environment, either simulated or real.
* Second, in the intermediate layer there are the respective drivers that give software access to the sensors and actuators of the robot.
* And third, the upper layer is the academic application that analyzes sensory data and makes decisions to action, planning if necessary.

The most important part of this design infrastructure is perhaps the academic application, which will contain everything necessary to support each exercise, and that will hide all the complexity to the students so that they only have to worry about the logic that controls the robot. Programming robots is not easy, so this division in hidden auxiliary functionality and student’s code avoids exposing them to all the complexity at the same time.

1. Lower Layer: Robot

(diap 7) One of the first design decisions was not to focus the practices in a concrete robot, but choose varied platforms: drones, robot TurtleBot with wheels, cars, humanoids ... with the idea of ​​being able to design practices that cover different aspects of robotics without the limitation of a particular robot. It is available the real hardware of some of these robots, and there is also a simulated version of all of them. Simulated robots allow the student not only to make mistakes in the code without serious consequences, but also allow fair comparisons between different codes and the development of automatic code evaluators.

(diap 8) The academic application will connect indistinctly to the real or simulated robot.

1. Intermediate Layer: Drivers and middleware

(diap 9) ROS will act as middleware, taking care of providing the robots' drivers, supporting the distributed software of the applications and providing several other utilities. It provides what is necessary to connect the academic applications with the sensors and actuators of the robot, which will be the same for the real or simulated robot. The academic component will have certain configuration files to specify the interfaces of the robot and its characteristics, allowing the abstraction of the hardware.

1. Upper Layer: Academic Application

(diap 10) For each practice an academic application is created that contains one part already programmed and another part that must be filled by the student. The prepared part is specific, solves auxiliary tasks and includes a single template file for the students to enter their code there. Python is used as the programming language of the robot logic and also of the infrastructure, and the environment encourages the use of standard libraries in the field such as OpenCV or PCL.

In addition to hosting the student code, the academic application offers:

* A graphical interface for debugging purposes, accessible through a programming interface (GUI API) that provides access to debugging tools.
* Provides a simple and local programming interface for accessing sensors and actuators through Python methods.

1. Distribution and installation

(diap 11) The presented environment is free software, downloadable from GitHub, and it is open to collaborations, extensions, modifications and use in whole or in part. Binary packages have been prepared for Ubuntu and for Debian that already include the drivers, the simulator and the academic application of each of the practices elaborated. It can also be used on MS-Windows and MacOS platforms through Docker containers. The environment is simple to use, there are recipes execution of each of the practices.

1. Practices

(diap 12) Now we are going to detail some of the existing practices in the teaching environment. Currently there are 19 practices already developed that address various problems of robotics, control, automation and artificial vision.

1. *Car Junction*

The goal of this practice is to implement the logic of a navigation algorithm for an automated vehicle. The vehicle will reach a road junction and it must Stop at the T joint, where there is a stop sign, wait until there are no cars at the intersection and and join the perpendicular road once it is clear.

To do this, the simulated car model has three cameras (left, center and right) and an odometric sensor, in addition to the motors necessary for movement that are handled with an interface of intermediate level that accepts orders of advance or turn speed.

The solution to this exercise is to implement a perception and control algorithm. In its perceptive part, this practice gives rise to color filters and ad-hoc processes that extract some measure of whether there are road lines or stop signs in the image, if the robot is curved or straight, if it is deviated to the right or left and to what extent. It is about programming an image segmentation algorithm that is able to detect different elements: lines, other cars and stop signs, and correct the position and speed of the robot according to the segmented. The graphic interface of this practice includes visualization spaces of the segmented images that can be established from the student's code.

The main evaluation criterion is that the maneuver is performed correctly and without collisions. It can also be taken into account the speed of execution of the maneuver and even the deviation committed when joining the new lane.

1. *Global Navigation*

The objective of this exercise is that a self-driving car can navigate from one point to another of a city which the map is provided. This requires programming a route planning algorithm and a piloting one based on position. The simulation part of this practice includes the autonomous car, a taxi, in a simplified city, with its streets and buildings.

The taxi is equipped with a GPS sensor that delivers at any moment an estimate of its position within the city. As for actuators, it has the already mentioned intermediate interface of locomotion based on forward speed and speed of Turn. The utility car model has been created for Gazebo with its realistic appearance and its embedded sensors, homologous to those used in real autonomous cars like those of Google or Tesla.

In this practice the student has to program the GPP global navigation algorithm to plan its way from the current position to one marked by the user in the graphical interface. The GUI of this academic application shows the 2D map of the city, the ideal calculated route and also allows you to see the value of the field generated by the student's code, which is a product of this technique. Once the route is calculated, the code includes in addition a piloting algorithm that takes into account the instantaneous position and the generated field to command speeds to the robot so that it navigates towards the destination.

This exercise will be evaluated by measuring the execution time, including the field calculation and navigation.

1. *Follow Line*

The objective of this exercise is that a mobile Turtle-Bot follow the red line in a circuit in the possible time possible. The student must program the algorithm that extracts the information necessary from the pixels of a camera on board the robot and order its motors the proper movement. This practice has both real hardware and simulated versions, where there is a real circuit with a white line and a simulated circuit with a red line to follow respectively.

The TurtleBot has two drive wheels with differential traction, to which commands are ordered through the intermediate movement interface: forward speed and speed of turn. It is equipped with a camera, that of the on-board laptop in the real robot.

In its perceptive part, this practice gives rise to color filters and ad-hoc processes that extract some measure of whether there is a line in the image, if the robot is in a curve or in a straight line, and if it is deviated to the right or left. A typical solution is to process four or five lines of each frame and filter by color only in those lines, measuring the deviation from the situation in which the robot is centered on the line. In its part of performance, this practice illustrates very well the usefulness of a PID control. You can try to adjust the values ​​of its constants Kp, Kd and Ki. It looks also the utility of programming an imaging-based control, with certain intermediate processing adhoc, instead of one-dimensional sensory signals. The graphical interface of this academic application facilitates the visualization of the result of the processing done on the image.

The simplest evaluation criterion is the time per turn, as long as the robot does not get too far out of the line. Other criteria are to measure the error in the trajectory with respect to the ideal (thus penalizing the oscillations) or see if it loses many times the line and if it recovers it. Both are measured too indirectly with the time per round.

1. *Vacuum Cleaner with SLAM*

The intention of this exercise is to implement the logic of a navigation algorithm for an autonomous vacuum with autolocation. The main objective will be to cover the largest area of ​​known house, which a map has been provided from, so that the robot can clean as much dirt as possible.

The interfaces that this robot has include a laser sensor, an odometer and a bumper sensor, in addition to the motors controlled through the intermediate motion interface. The model created for the simulation is based on Roomba from iRobot.

The solution of the exercise begins with the planning of the route, for which it is necessary to create a grid on the image to carry out a zigzag route, checking at any time if the adjoining cells belong to an obstacle, if it has already been gone through them, or if they are areas that are still to be cleaned. To do this, the analysis of the image and the value of each pixel of the cells in the reference map will be used, and with the data obtained a path will be drawn to the next free cell in the grid, through which orders will be sent to the motors depending on whether the maneuver executed is a turn, if it is an advance or if it is a return path when a critical point is reached. The graphical interface will show the map of the house, and over it the path followed by the robot.

This practice has an automatic evaluator that connects with the simulator to establish an objective note according to the performance of the algorithm. The evaluator connects to the interfaces of the robot and controls the surface of the house that has been cleaned successfully, in addition to having a countdown with the time limit of the practice. When this time ends, it gives the student a note based on the covered area.

1. Results of use

Among students of degrees, postgraduates and university courses, the platform has been used by more than 120 students. The surveys between them showed a very satisfactory opinion of the environment and of the practices, with 80% of equal or above evaluations of 8 points out of 10. Also revealed that the evaluation of the installation is positive but must be simplified a bit more.

1. Future lines of work

JdeRobot-Academy developers are working on a web version of the teaching environment that will allow students to access the practices from any computer with network availability. There is already a first functional version of the web with one of the practices of the environment, which uses Jupyter Notebooks to replace the academic application, with all the necessary infrastructure.

They are also working on the incorporation of MAVROS nodes to drone practices, to extend the functionality that these robots offer.

Finally, the migration of the learning platform infrastructure is being planned to work with ROS-Melodic and Gazebo 9.

.