A Comprehensive Approach to Teaching Mobile Robotics

Submitted: 10th February 2017; accepted: 4th August 2017

Łukasz Chechliński, Daniel Koguciuk

DOI: 10.14313/JAMRIS_3-2017/23

Abstract:

Mobile robotics can be an interesting subject for many students in a variety of engineering science fields. It builds a bridge between the pure theoretical digital world and the real, open environment. Several research results show that learning mobile robotics gives not only the ability to control certain types of robots but also develop many science-related attributes, technical and social skills. On the other hand, programming mobile robots is hard, and without a good guide, students are likely to lose their inspiration. For this purpose, we decided to develop a set of four exercises showing mobile robotics in the accessible and comprehensive way. The tasks were prepared for two types of wheeled robots: first equipped with a webcam, and second with sonar range finders. Both robots run using the ROS framework, as we find it the most popular robotics tool. The exercises are also designed considering the limited budget of educational institutions. Finally, guides for the tasks described in this paper have been shared on-line with the robotics community.

Keywords: *teaching, mobile robotics*

1. Introduction

The latest research show that the educational aspect of mobile robotics can be more important than teaching constructors of autonomous mobile robots, because it can bring direct benefits to a wider range of students. This resulted in many publications and special issues in high-indexed journals. In this article, we present a modern set of exercises designed to provide engineering students a rapid and engaging introduction to the programming of mobile robots, as well as to aid in the development of skills in problem solving, artificial intelligence, computer vision and general programming.

In [3] and [9] the large scale research shows that students of any age can benefit from educational robotics, improving their technical and social skills, such as computational thinking, general programming, computer science, scientific investigation, self-efficiency, problem solving, goal setting skills and team-work attributes. Influence of amateur robotics competitions (i.e. competitions for pupils and students constructing low-budged robots for such tasks as line follower, mini-sumo or micro mouse) is presented in [13]. Also [6] shows that team work and solving well-defined robotics problems can result in the evolution of engineering skills. Several researches were made on developing low-cost, printable educational robots [2] [14] [11]. An interesting usage of such robots is presented in [8], in which low-cost wheeled mobile robots together with a simulator builds a multi-robot formation control platform. Hardware-oriented robotics student projects are described in [16], [7] and [4].

The topic of educational robotics is also present in the *Journal of Automation, Mobile Robotics & Intelligent Systems*. Impact of mobile robotics into modern engineering skills is presented in [5]. More social and generally-technical aspects are presented in [10], where common learning of pre-school children, pupils and grandparents is presented. Article [12] suggest, that the ROS is the best tool for learning robotics. As far as we know, no better tool is available for free, because the ROS gives not only a piece of useful code, but also a large community support.

Our approach has several similarities to these mentioned above. Developed exercises are based on ROS;, we also focus on the idea of the general technical and social skills improvement. To our best knowledge, this article describes some novels: education in fields of robotics and computer vision is merged, presented tasks are easily applicable to some popular kinds of robots (the lack of well-described exercises was noticeable previously).

We think that journals such as this one are the best place for discussing engineering education topics. On the one hand, an international character allows worldwide idea sharing, and on the other hand popularity in the region facilitate sharing educational materials directly in a students' native language. Also, aspects of the engineering education management and financing can be discussed.

Students after a third year of robotics studies are the main target for our course. Background in general programming skills and computer vision is necessary.

In our opinion, effective teaching of robotics is hard, but the benefits are worth it. This article presents the first step of our methodology. On each step, the lecturer is active mainly during the lecture, while during the laboratories he keeps on asking the questions, preventing students from time-wasting mistakes and directly answering only the difficult questions. All steps are enumerated below:

1) control robot inputs and outputs, programming simple behavioural agents (15 h of lectures and 20 h of laboratories),

2) enrich agents with Artificial Intelligence in well-

defined environments (20 h of lectures and 10 h of laboratories),

3) develop a software for regular autonomous mobile robots (20h of lectures and 10 h of laboratories)

As we can see only the last step is dedicated only for the future mobile robots' developers, while first two steps focus on developing general skills. During the first step students are more strictly guided to get knowledge, abilities and habits necessary to program robots, as well as other devices equipped with sensors and actuators. The second step should keep attention on teaching AI ideas, while giving space for unassisted solving problems introduced in the first step. Finally, the third step should be project-oriented, during it student's teams with the theoretical knowledge from lectures and practical skills and habits from previous steps independently solves the given problem of mobile robotics.

Our methodology is motivated by the idea of giving the student rather the skeletal frame of knowledge, skills and habits than a set of abilities. Many students can never in their future carrier program a mobile robot, but it is quite possible that they will solve some problem connecting a hardware and Artificial Intelligence. Debugging of programs cooperating with a hardware is more difficult than pure software debugging and mistakes can cause in high costs. On the other hand, we cannot teach skills and habits without any detailed example of hardware. We think that mobile robot is a good choice for it, because it is at once complex, well described and universal.

In presented exercises we use two types of robots, both running under the ROS. The programming language chosen for exercises is C++, but it can be switched to Python. The robots should in general meet following requirements:

- type A is a mobile robot equipped with a webcam,
- type B is a mobile robot equipped with at least 4 sonar range finders and a quite accurate odometry system.

For each robot we prepared two exercises, e.g.:

- Exercise A1 – Visual Line-follower,
- Exercise A2 – Visual Sumo,
- Exercise B1 – Sonar-based Corridor-follower,
- Exercise B2 – Sonar based Micro-mouse.

We had in mind that splitting exercises into two blocks will result in a more flexible scheduling. It does not matter if a student will start from A1 or B1, or if after exercising A1 will he exercise B1 or A2. It is only important to exercise A2 after A1 and B2 after B1.

The rest of this paper is organized as follows. In the section 2 robots used in our exercises are briefly described. Sections 3.1, 3.2, 3.3, 3.4 describes successive exercises, students' opinion poll is presented in the section 4 while conclusions are presented in the section 5.

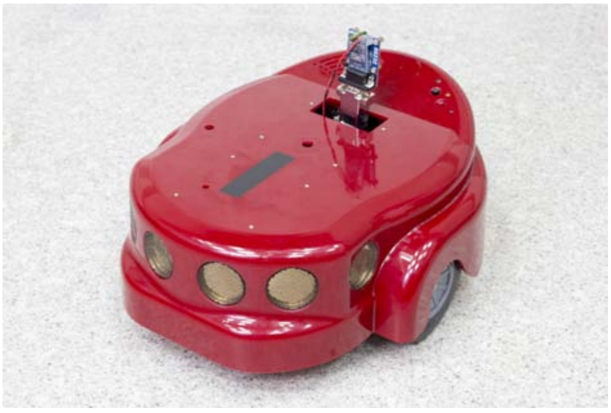


Fig. 1. The Amigobot robot upgraded with a RS232 (the default interface of the robot) – Bluetooth adapter

2. Robots

Preparing a mobile platform for a flexible and convenient operation is a key matter. In our lab, we have two Pioneer (3-AT and 3-DX models) and two Amigobot robots, both from Adept MobileRobots. Originally all three types are just bare slave devices – they must be connected from a host computer with the RS-232 interface, which sends commands and receives a sensor data. Basically, there are three options of the system architecture:

1) the robot is constantly connected to a PC via a cable – this solution does not need any hardware modifications, and one can start programming straight away. The main drawback is a dependency of the host computer and a necessity of a stable communication between the robot and the host,

2) the robot is constantly connected to a PC via the radio – here we consider that the robot communitaces with the host PC via some radio interface such as Bluetooth or Wi-Fi. This solution is more mobile compared to the previous one, but it is still limited to the range of the user interface,

3) the robot has an embedded PC on the board – this is the most time-consuming solution to build, but it is also the most flexible one. The operational area is not limited here with the range of any interface and the robot can also perform autonomous actions independently. Another advantage is a possibility to connect any other sensing device directly to the ePC and build a complex, mobile and independent platform.

2.1. Amigobots

Considering the fact of limited free space in the Amigobot chassis, upgrading the robot with an embedded computer was rejected from the very beginning. We also do not want to run the robot on a leash so the only reasonable option here is a wireless interface. We assume Amigobot robots will operate only in our laboratory so the Bluetooth technology seems to be enough for us.

The chosen Bluetooth module is HC-05v2 interconnected with a serial – RS232 adapter to complete communication compatibility. The Amigobot ro-

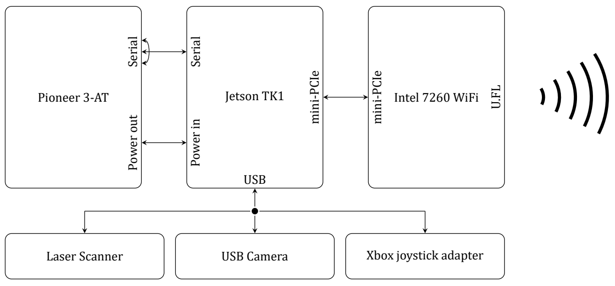


Fig. 2. Pioneer robot hardware configuration

bot with the Bluetooth adapter is shown in the Figure 1.

2.2. Pioneer Robots

Both Pioneer robots (3-AT and 3-DX shown in the Figure 3) have enough space to put an embedded PC inside. In the near future, we plan to launch some parallel processing on-board and that is why we choose the Jetson TK1 development board with 192 CUDA cores from NVIDIA.

The final configuration of both Pioneer robots is shown in the Figure 2. The Jetson board is powered with 12 V and connected with RS-232 interface to the robot. We also added the Intel 7260 Wi-Fi card in the mini-PCIe slot to connect the robot with the outer world. We also equipped both robots with the Hokuyo URG-04LX-UG01 laser scanner, an USB camera and the Xbox joystick adapter for remote control.

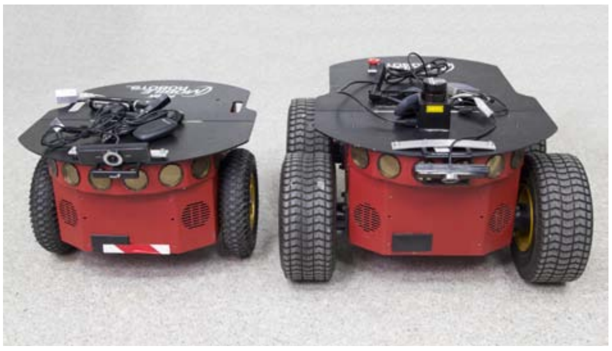


Fig. 3. The Pioneer robots: 3-DX on the left and 3-AT on the right

It is worth to mention that the serial port of the Pioneer robot is slightly different from the external one and requires logic high state on the DTR signal (pin 4) with positive voltage level (about 5V). To meet this requirement, we looped DTR (pin 4) with RI signal (pin 9) on the robot side.

2.3. Low-level Software

As many of the papers considered in the introduction, we also use the ROS framework [15] for software development. We have developed the basic packages (amigobot_driver and pioneer_drver) which can be used as low-level drivers and made them open-source [1]. One can find there:
- robot communicating program (RosAria),
- URDF model spawning program,
- Xbox joystick handling program,
- robot manual operation program,
- transformation publishing program (via ROS TF mechanism),
- laser scanner communicating program (for Pioneer robots).

After starting an appropriate roslaunch file all topics and services allowing for control of the robot and sensor state reading are available for a further usage.

3. Exercises

The design of the exercises starts with answering some basic questions about a main target audience and a required background of the participants. Being in touch with our students gives us the impression of their strong theoretical knowledge in many fields of science needed in mobile robotics, but one can feel their lack of practical experience in its implementation. The presented reasoning gives us the opportunity to prepare attractive exercises with assumptions of knowledge of participants about linear algebra, the basis of the various sensors, machine vision and control theory.

It is widely known that the UNIX operating system has many tools facilitating development of the software for above mentioned fields and it is hard to imagine a robot with a Windows instead. The same analogy one can find using a ROS framework – it is possible to create a robotic system without it, but it provides many useful tools and let a student or a scientist focus on a solution of the real-world problem.

A difficult part for our students with using the ROS framework is large entry-work. Many of them sees the UNIX system for the first time and is not fluent in finding and understanding articles written in English, so we had decided to prepare a script, which deals with this matter. The paper starts with a gentle introduction to LINUX and helps the reader to understand basic concepts of the ROS framework based on an example of simple package creation.

The second very important script is about different coordinate frames in the ROS, which knowledge we required of, before students can perform the exercise B2. A separate script is also provided for each exercise. All mentioned documents can be found on our website [1]. Educational materials are prepared in Polish, as this is the official language of our course. If you would be interested in other language versions, please contact us via an e-mail.

Having four robots described above, we can perform four exercises at once, where each task is prepared for maximum 4 participants (groups are up to 15 students). We believe it is a reasonable situation, where four-person team can discuss details of the problem within a group resulting in valuable experience of the whole process of the solution finding procedure. We assume 5 lesson units for each exercise (5 times 45 minutes) and this is a typical time, where every group could finish their solution to a problem.

Exercises are divided in two groups – one for each type of a robot. The first exercise for both groups (A1 and B1) is strictly guided so it introduces to a student the basis of robot programming – including dealing with a given set of sensors or programming tools. The student first learns how the correct robot program should look like – the program structure is given and an only student task is to "fill the gaps" in the code. In the second exercise (A2 and B2) only the scratch of the program code is given, but the problem solution is described in detail in the script. We hope that such approach keeps students of from wasting a time on technical problems and at once gives them possibly large space for testing their own ideas.

In the following sections of this chapter each exercise will be described in detail.

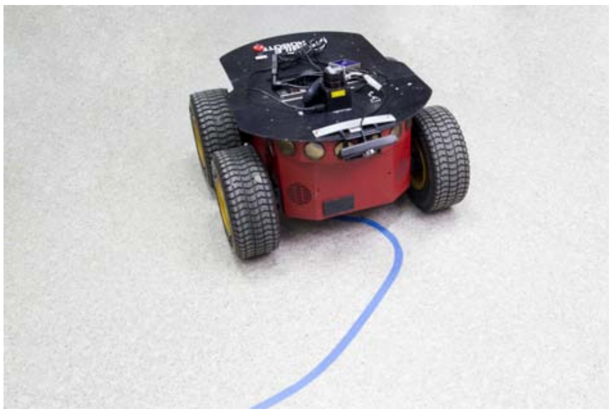


Fig. 4. The 3-DX robot in the camera based line-following exercise

3.1. Exercise A1: Visual Line-follower

In the very beginning of the visual line-follower exercise (figure 4) students learn how to work with a robot with the on-board computer. They can try a direct connection to a robot with a display, keyboard and mouse – this shows that the robot uses a common computer architecture. This is also a very basic way of programming robot, allowing to solve all low-level problems. For the majority of tasks this approach seems to be the most comfortable, due to a low latency and independence from a local net connection.

However, direct connection reduces a robot mobility. To overcome this limitation a mobile robot developer should know some remote access techniques. For this reason, two other approaches are introduced to students:

- system-level remote access, via SSH connection – most useful for development,
- ROS-level connection via topics, services etc. – most useful for runtime.

This exercise introduces several other basic tasks, such as:

- launching, stopping and restarting robot drivers,
- enabling and disabling robot control over an USB joystick,
- robot sensor data visualization in the rViz program.

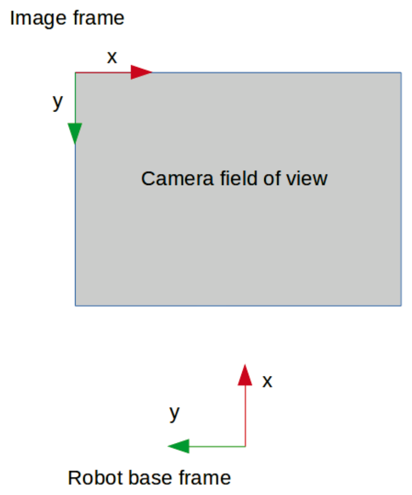


Fig. 5. Robot base and image frame scheme

After that, students start writing their own code, filling the gaps in prepared source files. To introduce communication between ROS nodes, the task of line following is divided between two programs. The first one finds the line based on the camera input and publishes it over the ROS in the form of a point cloud. The second one receives the point cloud and use it to calculate robot control commands. The relation between the robot base frame and the image frame is shown in the figure 5, which is a part of the exercise script. The resulting point cloud can be visualized in the rViz program.

As mentioned above, a source code is prepared and the only students' task is to fill some gaps. Communication with the camera, a robot controller and between nodes is ready. The remaining tasks are:

- image binarization for line finding,
- conversion from an image to a point cloud,
- point cloud data analysis and control calculation

With the obtained skills students can move forward and start more self-directed robot programming in the exercise A2.

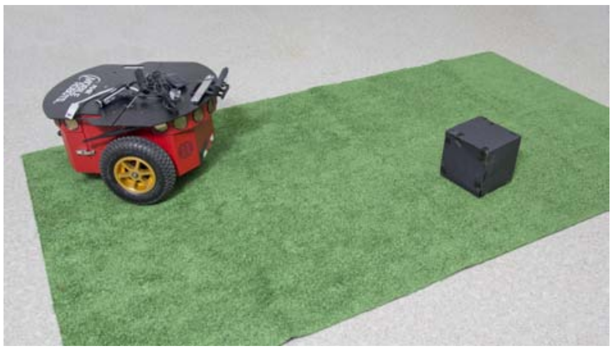


Fig. 6. The 3-AT robot in the camera based sumo exercise

3.2. Exercise A2: Visual Sumo

The aim of this exercise is development of a robot software for vision-based sumo as shown in the Figure 6. The robot should keep moving on a green mat, looking for adversaries in the form of black and red carton boxes. When it meets one, the robot should try to move it out of the mat. After a success looking for adversaries should be continued.

In this exercise only a short scratch of the program is prepared, so a list of student's tasks elongates, including:

- camera input handling,
- ROS communication setup,
- reading joystick data – our robo-fighter keeps moving, so motors-enable control is necessary,
- computer vision algorithm development,
- control algorithm development.

The computer vision algorithm development is divided into several tasks, sketched in the exercise script. First, the student's algorithm should control whether the robot is on the green mat or not. This is achieved in four steps:

- input image size is reduced, then the image is median blurred and converted to the HSV colour space,
- image is binarized by a HSV in-range thresholding,
- erosion is performed to reduce noise,
- the number of mat pixels is counted for left and right halves of the image. Absence of the mat in either of halves means that the robot reached the mat's edge.

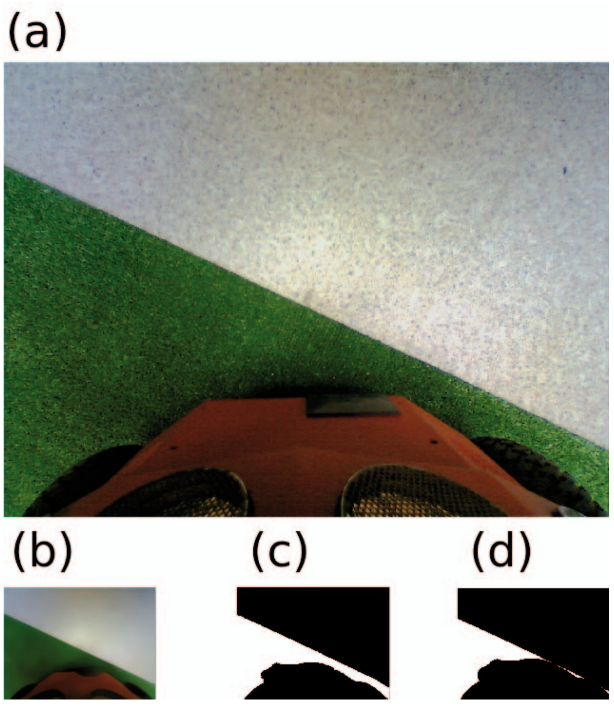


Fig. 7. Steps of the green mat detection algorithm: (a) the input image (b) the resized, median blurred image (c) result of the HSV in-range thresholding (d) the eroded binary image

Algorithm steps are visualized in the figure 7.

The other task is a development of an adversaries' detection algorithm. This task is quite like the mat detection, but the similarity of the robot and the adversary colours causes the necessity of the input image masking. The result is shortly presented in the Figure 8.

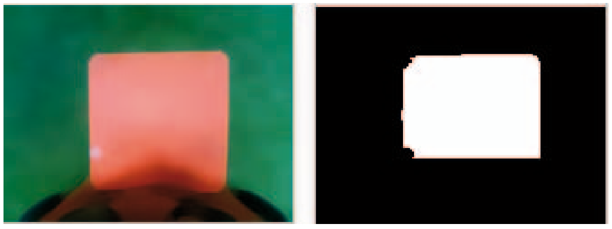


Fig. 8. The result of the adversary search algorithm sketched in the exercise script

3.3. Exercise B1: Sonar Corridor-Follower

The exercise B1 is quite like the A1, presented in 3.1, but this time a robot moves along a corridor with carton walls. The main sensors are sonars, which are less reliable than the camera, so the robot way is much simpler – it resembles a curve from the beginning of a driving license test.

Again, students start with getting known with the mobile robot. They turn it on, perform a self-driving test, establish an USB or Bluetooth communication and run the robot driver. Then they get familiar with robot sensors and a control over the ROS, especially the rViz program, which is presented in the figure 9.

Finally, student fill gaps in a prepared source code, in a similar way like in the exercise A1. Their job consists of the following tasks:

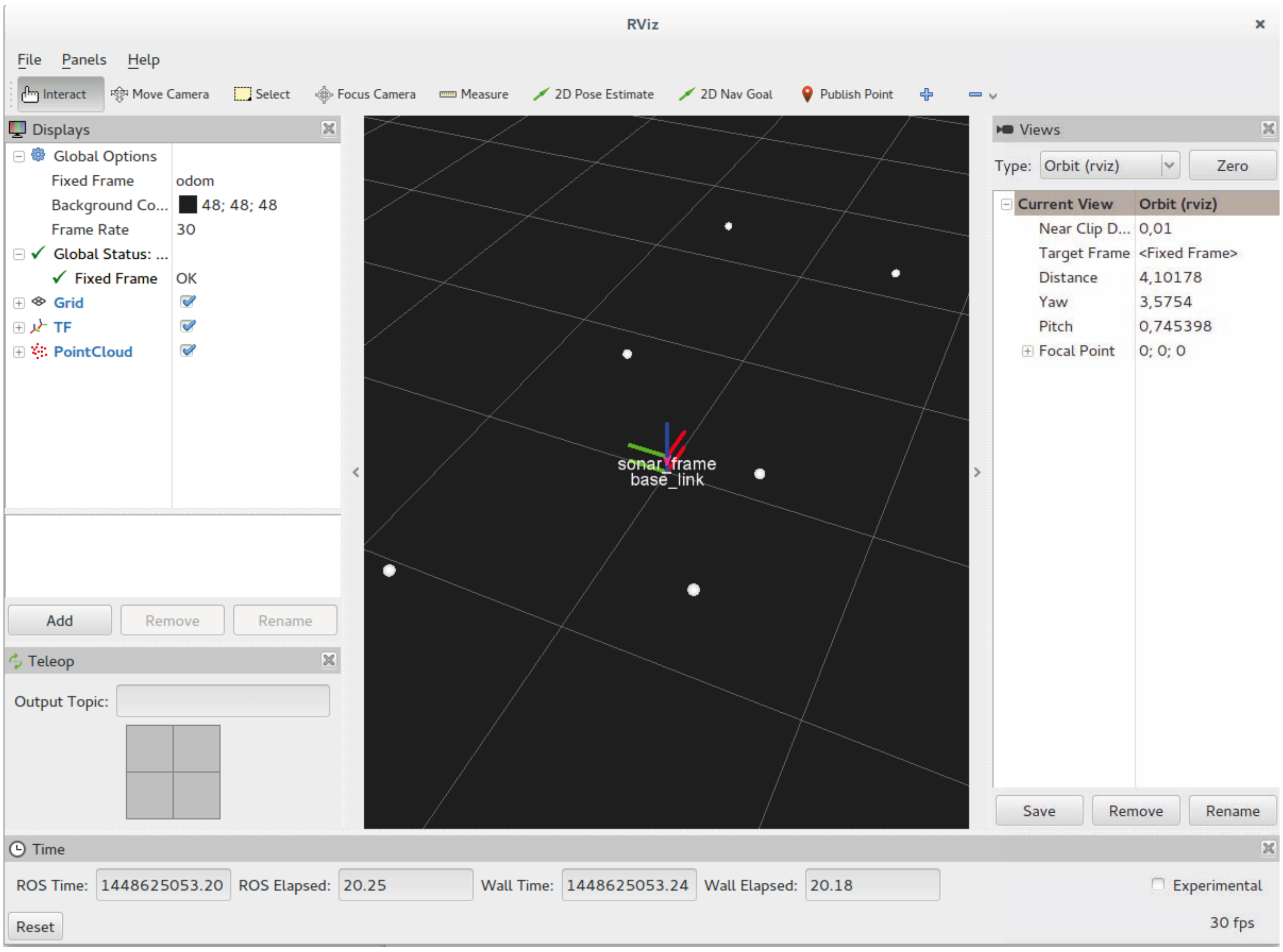
- understanding a sensor data structure,
- implementation of a stop condition – reaching the end of the corridor,
- calculation of a robot moving direction,
- development of a PID-based robot direction controller,
- optionally reducing a robot speed at curves and increasing it on straight lines.

Tasks presented above are supported with the schemes in the exercise script. The Figure 10 presents two of them: one for sonar data structure, and the second for the calculation of the robot moving direction.

3.4. Exercise B2: Sonar Micro-Mouse

In the exercise B2 students develop software for a micromouse-like robot shown in the Figure 11. The aim of the robot is to explore the labyrinth with a right-hand rule – no mapping or shortest path finding is required. Students start with a short code skeleton and implements their own ideas. The main tasks are:

- obtaining information about walls from the sonar range sensors,
- understanding the role of coordinate frames in a robot movement programming,
- teaching the robot to move exactly one step forward (the step is the size of the labyrinth's cell),



Global Options
Fixed Frame odom
Background Co... 48; 48; 48
Frame Rate 30
Global Status: ...
Fixed Frame OK
Grid ✓
TF ✓
PointCloud ✓

Output Topic:

Type: Orbit (rviz) Zero
Current View Orbit (rviz)
Near Clip D... 0,01
Target Frame <Fixed Frame>
Distance 4,10178
Yaw 3,5754
Pitch 0,745398
Focal Point 0; 0; 0

ROS Time: 1448625053.20 ROS Elapsed: 20.25 Wall Time: 1448625053.24 Wall Elapsed: 20.18 Experimental
30 fps

Fig. 9. A screenshot from the rViz program, enabling the sensor data visualization and the robot velocity control

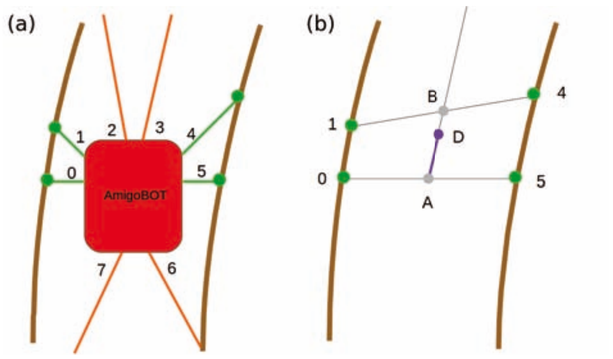


Fig. 10. Schemes from the exercise script. (a) The robot sensor data structure explanation – the robot is equipped with 8 sonars and they readings are formed as a cloud of ordered points. Knowledge of a sonars' order simplifies development of the control algorithm. (b) Simple calculation of the robot moving direction – the algorithm developed by students calculates points A and B presenting the corridor's centre respectively at a robot position and in front of the robot. Finally the point D is calculated to normalize the distance ahead the robot. Y-coordinate (Y is oriented from the left to the right) of the point D is then used to control the robot movement direction.

- teaching the robot to rotate exactly 90°left or right,

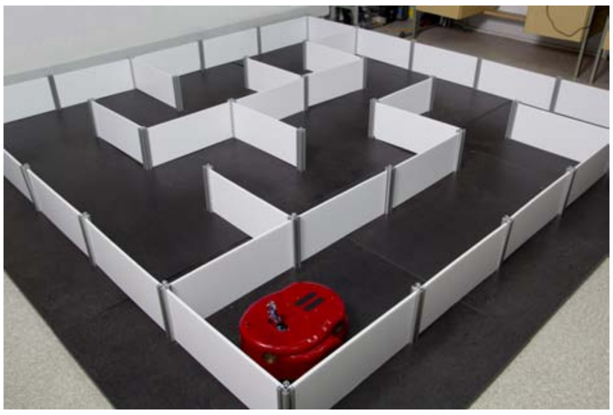


Fig. 11. The labyrinth with the Amigobot robot inside, used for both micro-mouse and corridor follower tasks.

- development of a right-hand rule labyrinth solving algorithm.

4. Students' Opinion Poll

To measure the students' feedback in a systematic way, after finishing the course students were asked to fill the anonymous poll. Only 28 students have participated in it. In seven questions students were asked to give the mark (from 1 to 10) marking different aspects of the course, results are presented in the

Table 1. Another two questions concerned particular exercises – a student marked the most interesting one and one which needs to be improved in the first order, see the Table 2. Finally, a student divided 100 points between 7 aspects of future work, results are presented in the Table 3.

Tab. 1. Pool results - questions marked from 1 to 10

Question	Average Mark
General course mark	8.5
Hardware state mark	8.0
Educational materials (instructions)	7.9
Essential background of the lecturer	9.2
Cooperation with the lecturer	9.0
Improvement of student's skills	7.8
Student's self-activity mark	8.1

Carrying out this pool improved our plans. Keeping in mind that some high scores can be caused by students' courtesy, relative values present which aspects may be modified.

Tab. 2. Pool results - which exercise is the most interesting and which should be improved firstly in students' opinion

Exercise	Is interesting?	Should be improved?
A1	15%	21%
A2	35%	50%
B1	35%	21%
B2	15%	8%

Tab. 3. Pool results - voting for aspects of future work. Each student divided 100 points between aspects.

Aspect	Votes
Purchase of modern robots (same type)	10.5%
Purchase of other types of robots (drones, walking robots)	31.2%
Educational materials improvement	4.2%
Improving essential background of the lecturer	0.1%
Improvement of cooperation with the lecturer	2.5%
Development of new exercises	16.5%
Increasing number of laboratory hours in the course	35.0%

5. Conclusions

We wrote this paper just after the end of the second edition of described laboratory exercises. About 50 students have taken part in the course. The presented exercises have received remarkably positive feedback from the course participants according to the educational level, as well as skills and knowledge they obtained. We also tried to fit their remarks about hard and not well-explained tasks. We consider that the resulting set of educational materials, scripts, robot software and hardware is worth attention. That is why we decided to share with the community as much as possible.

Future work is based on the pool results and our own perception. Some students feel a lack of time during the laboratories, which suggests that they may be not well-prepared. To prevent this, the scripts should be further improved. Exercises presented in this paper must be kept up to date (e.g. with the ROS updates). In our opinion, no more exercises can be performed during the limited didactic hours. This turns us to focus on improving other courses, presented in the introduction of the methodology.

The developed set of exercises is a part of the methodology presented in the article introduction. It gives not only specialized skills, useful for future mobile robots' developers, but also a set of general skills, enriching the experience of every engineer.

AUTHORS

Łukasz Chechliński – Institute of Automatic Control and Robotics, Faculty of Mechatronics, Warsaw University of Technology, ul. Boboli 8, 02-525 Warsaw, e-mail: lukasz.chechlinski@gmail.com, www: www.wutrobotics.mchtr.pw.edu.pl.

Daniel Koguciuk* – Institute of Automatic Control and Robotics, Faculty of Mechatronics, Warsaw University of Technology, ul. Boboli 8, 02-525 Warsaw, e-mail: daniel.koguciuk@gmail.com, www: www.wutrobotics.mchtr.pw.edu.pl.

*Corresponding author

REFERENCES

[1] "WUTRobotics research group website". wutrobotics.mchtr.pw.edu.pl. Accessed on: 2017-06-09.

[2] L. Armesto, P. Fuentes-Durá, and D. Perry, "Low-cost Printable Robots in Education", *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, 2016, 5–24, DOI:10.1007/s10846-015-0199-x.

[3] S. Atmatzidou and S. Demetriadis, "Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences", *Robotics and Autonomous Systems*, vol. 75, Part B, 2016, 661 – 670, DOI:10.1016/j.robot.2015.10.008.

[4] C. Avsar, W. Frese, T. Meschede, and K. Brieß, "Developing a Planetary Rover with Students: Space Education at TU Berlin", *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 8, no. 1, 2014, 20–29, DOI:10.14313/JAMRIS_1-2014/3.

[5] A. Eguchi, "Educational robotics to promote 21st century skills and technological understanding among underprivileged undergraduate students". In: *Integrated STEM*

Education Conference (ISEC), 2015 IEEE, vol. 1, 2015, 76–82, 10.1109/ISECon.2015.7119949, DOI:10.1109/ISECon.2015.7119949.

[6] A. Eguchi, "RoboCupJunior for promoting *STEM* education, 21st century skills, and technological advancement through robotics competition", *Robotics and Autonomous Systems*, vol. 75, Part B, 2016, 692 – 699, DOI:10.1016/j.robot.2015.05.013.

[7] A. El-Fakdi, X. Cufí, N. Hurtós, and M. Correa, "Team-Based Building of a Remotely Operated Underwater Robot, an Innovative Method of Teaching Engineering", *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, 2016, 51–61, DOI:10.1007/s10846-015-0203-5.

[8] E. Fabregas, G. Farias, S. Dormido-Canto, M. Guinaldo, J. Sánchez, and S. D. Bencomo, "Platform for Teaching Mobile Robotics", *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, 2016, 131–143, DOI:10.1007/s10846-015-0229-8.

[9] M. Kandlhofer and G. Steinbauer, "Evaluating the impact of educational robotics on pupils technical- and social-skills and science related attitudes", *Robotics and Autonomous Systems*, vol. 75, Part B, 2016, 679 – 685, DOI:10.1016/j.robot.2015.09.007.

[10] M. Kandlhofer, G. Steinbauer, S. Hirschmugl-Gaisch, and J. Eck, "A cross-generational robotics project day: Pre-school children, pupils and grandparents learn together", *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 8, no. 1, 2014, 12–19, DOI:10.14313/JAMRIS_1-2014/2.

[11] F. M. López-Rodríguez and F. Cuesta, "Andruino-A1: Low-Cost Educational Mobile Robot Based on Android and Arduino", *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, 2016, 63–76, DOI:10.1007/s10846-015-0227-x.

[12] S. Michieletto, S. Ghidoni, E. Pagello, M. Moro, and E. Menegatti, "Why teach robotics using ros", *Journal of Automation, Mobile Robotics and Intelligent Systems*, vol. 8, no. 1, 2014, 60–68, DOI:10.14313/JAMRIS_1-2014/8.

[13] G. Nugent, B. Barker, N. Grandgenett, and G. Welch, "Robotics camps, clubs, and competitions: Results from a *US* robotics project", *Robotics and Autonomous Systems*, vol. 75, Part B, 2016, 686 – 691, DOI:10.1016/j.robot.2015.07.011.

[14] R. Pérula-Martínez, J. M. García-Haro, C. Balaguer, and M. A. Salichs, "Developing Educational Printable Robots to Motivate University Students Using Open Source Technologies", *Journal of Intelligent & Robotic Systems*, vol. 81, no. 1, 2016, 25–39, DOI:10.1007/s10846-015-0205-3.

[15] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "*ROS*: an open-source Robot Operating System". In: *ICRA Workshop on Open Source Software*, vol. 1, 2009.

[16] H. Yi, C. Knabe, T. Pesek, and D. W. Hong, "Experiential Learning in the Development of a DARwIn-HP Humanoid Educational Robot", *Journal of Intelligent and Robotic Systems*, vol. 81, no. 1, 2016, 41–49, DOI:10.1007/s10846-015-0200-8.