Proceedings of the European Robotics Forum 2018 Workshop "Teaching Robotics with ROS"

Stefan Schiffer,
Alexander Ferrein,
Mukunda Bharatheesha,
and Carlos Hernández Corbato
(Editors)

February 25, 2019

Preface

ROS is developing into the standard middleware for robotics applications. Researchers and practitioners world-wide contribute their results and publish software packages for ROS. This way, a large number of state-of-the-art robotics algorithms become available to be used freely. With the ROS-Industrial project, the ROS success story should be extended to industrial robots as well. To foster ROS-Industrial is also the mission of the ROSIN project (http://rosin-project.eu). In order to build a broad ROS-I community, one of the goals of ROSIN is to teach ROS-I to university students and professionals.

Many institutions offer education activities about the Robot Operating Systems. There are a number of Summer Schools, there are professional trainings for ROS-Industrial and even online courses are available for learning ROS. But how is ROS and, for that matter, how are some of the fundamentals of robotics being taught to the students? With the workshop "Teaching Robotics with ROS" we aim at bringing together educators involved in teaching robotics courses and/or ROS to discuss curricular topics, best practices and exchange about common problems with teaching robotics with ROS.

This volume contains the papers presented at TRROS 2018: Teaching Robotics with ROS (Workshop at ERF 2018) held on March 13-15, 2018 in Tampere, Finland. There were 7 submissions. Each submission was reviewed by at least 2, and on the average 2.1, program committee members. The committee decided to accept 7 papers.

Parts of the workshop organization was supported by the ROSIN project. The ROSIN project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732287. We also appreciate the support we received when using EasyChair for the workshop submission handling and in generating the proceedings.

February 25, 2019
Aachen

Stefan Schiffer
Alexander Ferrein
Mukunda Bharatheesha
Carlos Hernández Corbato

Table of Contents

Program Committee

Mukunda Bharatheesha	Delft University of Technology
Carlos Hernandez Corbato	Delft University of Technology
Alexander Ferrein	FH Aachen University of Applied Sciences
Stephan Kallweit	University of Applied Sciences Aachen
Nicolas Limpert	FH Aachen University of Applied Sciences
Stefan Schiffer	RWTH Aachen University & MASCOR Institute
Patrick Wiesen	University of Applied Sciences Aachen

Interactive ROS Tutorials with Jupyter Notebooks

Enric Cervera*

Robotic Intelligence Lab
Universitat Jaume-I de Castelló, Spain,
ecervera@uji.es

Abstract. Collaboration is a major strength of ROS, as it was designed specifically for groups with different expertise (mapping, navigation, vision, etc) to collaborate and build upon each other's work. The ROS Wiki is such an example,[1] with the official documentation for ROS packages, as provided by their developers, and every user has dived into its ROS tutorials for learning the basic concepts. Those tutorials consist of web pages with embedded examples of source code that the user can run in a separate terminal. We aim to transform the static tutorials into interactive documents where the user not only reads but also runs the code inside the same web browser, by using Jupyter Notebook technology.[2] Our goal is to make the tutorials more concise and effective, avoiding the tedious and error-prone copying-and-pasting of code. Also, the integration of execution results provides the user with step-by-step feedback, for a more illuminating learning experience.

Keywords: tutorial, interaction, literate computing

1 Introduction

The ROS Tutorials are the primary source of information for learning ROS. They consist of a collection of web pages about the installation and setup of the system, the core concepts, and the development of robot applications in the programming languages C++ or Python.

Each web page presents a specific issue, combining an informative description interleaved with snippets of code or commands. The code must be first copied and pasted into an editor, then run in a terminal. The commands can be copy-pasted directly into the terminal, and executed. Such a workflow requires several views simultaneously:

- Documentation (teaching materials)
- Edition of the source code

* ORCID id: 0000-0002-5386-8968. The views and opinions expressed in this article are those of the author, not affiliated with any of the projects mentioned.

[1] http://wiki.ros.org/

[2] http://jupyter.org/

– Execution and monitoring of results

For a proper management of these views, several windows are needed in the user desktop. A typical layout is shown in Fig. 1, with the browser window at the left, displaying the documentation, the editor window at the top right, and the terminal window for execution at the bottom right.

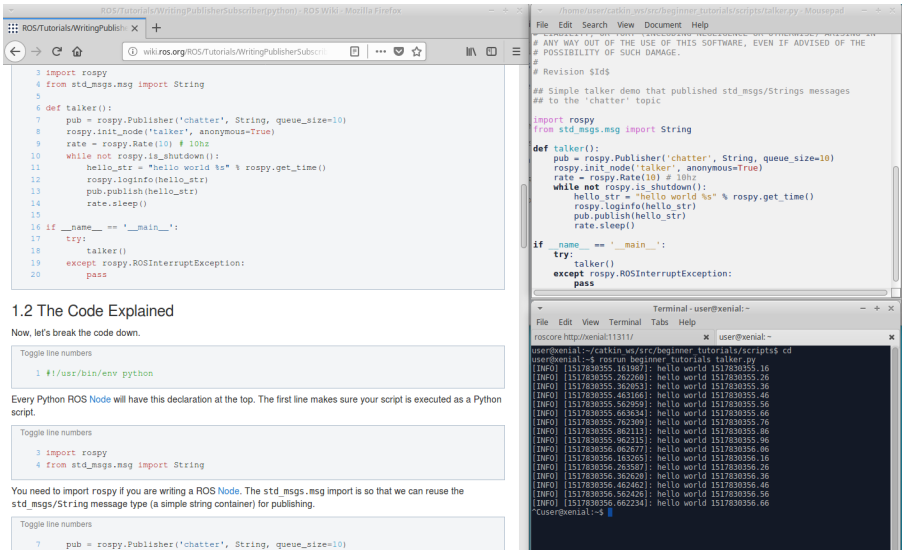


Fig. 1. Desktop layout: left window is the browser with the documentation web page; top-right is the editor window with the source code; bottom-right is the terminal window for the execution of the program.

There are some disadvantages in this arrangement, the most obvious one is the duplication of code: the programming statements are shown in the documentation, then they need to be copied into the development editor. In fact, some tutorials feature two versions of the same code, a complete one for easy copying and pasting, and a step-by-step version for the explanations.

In addition, the execution of the code takes place in a third window (the console), losing the context of the source code. Moreover, the complete program is run, making it difficult to execute the statements step-by-step, as opposed to the explanations in the tutorial.

Jupyter notebooks provide an integrated environment for the execution of code statements, interleaved in the same web pages of documentation. The code can be executed in cells, or snippets of code with a few statements, and the results are displayed below the same cell of code. Code cells can be edited and run freely by the user, interactively. As a result, a single window combines the three necessary views (teaching materials, edition, execution), as shown in Fig. 2.

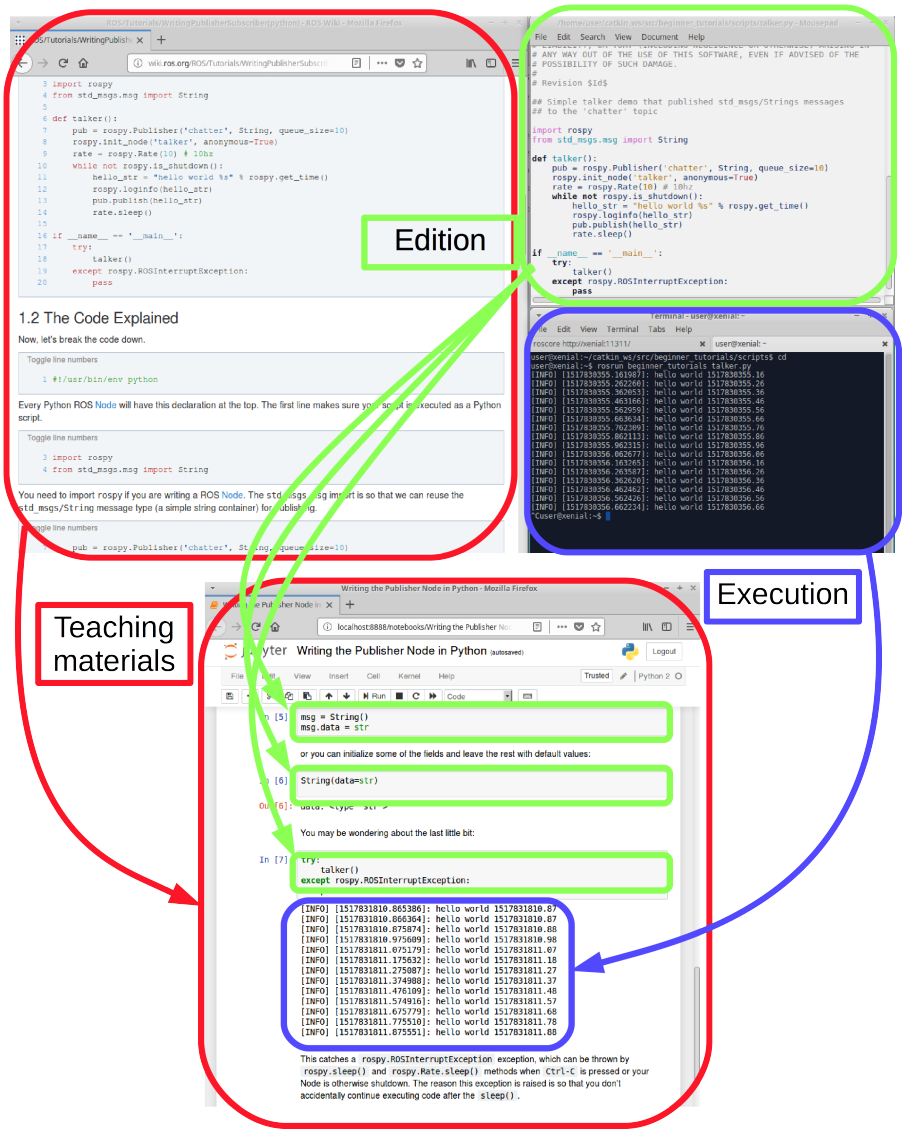


Fig. 2. Interactive ROS Tutorial on a Jupyter notebook: the teaching materials, source code, and execution results are all presented together in the browser window.

This paper describes how jupyter notebooks work (Section 2), how ROS and its tutorials can be integrated in the workflow (Section 3), and a couple of examples of representative tutorials regarding ROS topics and services (Section 4). Finally, Section 5 draws some conclusions and outlines the work in progress.

2 Jupyter Notebooks

The Jupyter Notebook is an interactive computing environment that extends the console-based approach in a new direction: it provides a web-based application suitable for capturing the whole computation process of developing, documenting, and executing code. Its roots lie in the Interactive Python environment [5], and the ideas behind the proprietary-based Mathematica Notebook environment [12].

Jupyter Notebooks are experiencing an immense success in recent years, applied to education and research [7] in a number of disciplines like computer vision, machine learning, or even the analysis of gravitational wave data [9].

Jupyter implements a "literate computing" scheme [4], inspired in the "literate programming" paradigm [3], a software development style pioneered by Stanford computer scientist, Donald Knuth, that emphasizes an approach where exposition with human-friendly text is punctuated with code blocks. Such an approach seems more adequate for demonstration, teaching and research purposes, especially for science and technology.

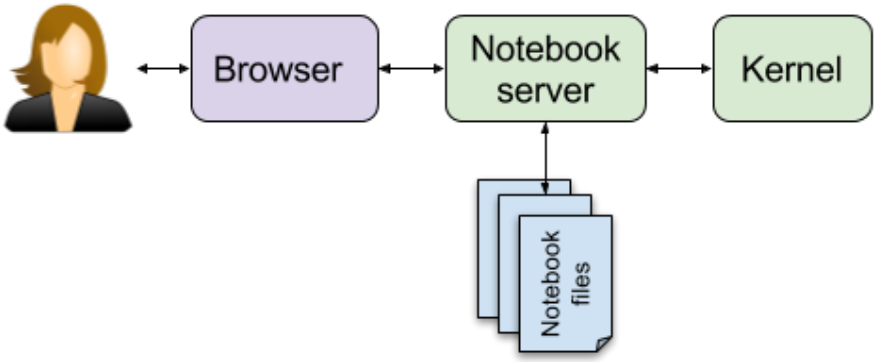


Fig. 3. Jupyter notebook elements.

As a server-client application, there are several processes involved in the user interaction, as seen in Fig. 3. The kernel is the separate process that actually runs the code. The notebook server stores code and output, together with markdown notes (a lightweight markup language that can be converted to HTML), in an editable document called a notebook. The user interacts with the browser, which

is connected to the notebook server for rendering notebooks, and sending user input to the kernel.

There are kernels available for many languages, starting from Julia, Python and R (the original languages Jupyter was developed for —hence the acronym) but also to other compiled languages like Java or C++.

3 Integration of Jupyter and ROS

Jupyter notebooks are being used in robotic Internet sites such The Construct [8], which also integrates 3D simulations with Gazebo. In our approach we will only focus on the programming task.

ROS development is mainly based on the languages C++ and Python. The former is perhaps the most widely used client library, and it is designed for high performance. The latter provides the advantages of an object-oriented scripting language: it favors implementation speed so that algorithms can be quickly prototyped and tested. Python is a core dependency of ROS, since some tools are developed in that language.

3.1 Local Install

Jupyter can be easily installed using Python's package manager,[3] or third-party distributions like Anaconda.[4] Once installed, it is launched from the terminal, where the server keeps running in the background. At the same time, a new browser window opens and displays the notebook folder, or a specific notebook.

The user can then interact with the notebook, with an environment similar to that of the console, i.e. all the C++ or Python libraries are accessible as in any ROS application. Practical examples of tutorials will be shown in Section 4.

Obviously, ROS is a prerequisite that should be available in the system, either in a local installation or as a Docker image.[5]

3.2 Notebooks in the Cloud

Since Jupyter Notebooks use a client/server approach, it is possible to run the server in a different computer than the client. Both machines surely must be connected, but the connection protocol (http - the WWW protocol) makes it possible to run the server in any computer connected to Internet.

Free cloud services for IPython and Jupyter already exist, where their basic functionality can be tested.[6]

A recent and interesting initiative is Binder,[7] which not only enables sharing of live notebooks in a computational environment, but authors can publish

[3] https://packaging.python.org/tutorials/installing-packages/
[4] https://www.anaconda.com/
[5] http://wiki.ros.org/docker/
[6] https://try.jupyter.org/
[7] https://mybinder.org/

notebooks in a source code repository along with an environment specification. By pointing the Binder web service at the repository, a temporary environment is automatically created with the notebooks and any libraries and data required to run them. This allows authors to publish their code in an interactive and immediately verifiable form [2].

An environment specification for ROS can be created based on the ROS docker images.[8] These images provide a simplified and consistent platform to build and deploy applications, with an easy way to develop, reuse and ship software [11].

We have upgraded the ROS docker images with the necessary packages for running Jupyter along ROS, so that the example tutorials presented in the next section are stored in a GitHub repository that can be run either in a local environment, or in the cloud using the Binder service.[9]

4 Tutorial Examples

For demonstration purposes, we have adapted the tutorials for some of the most fundamental concepts in ROS: topics and services [6]. Topics are named buses over which nodes exchange messages. They have anonymous publish / subscribe semantics: nodes that are interested in data subscribe to the relevant topic; nodes that generate data publish to the relevant topic. There can be multiple publishers and subscribers to a topic.

The publish / subscribe model is a very flexible communication paradigm, but its many-to-many one-way transport is not appropriate for request / reply interactions. For that purpose, ROS features another paradigm called Service, which is defined by a pair of messages: one for the request and one for the reply. A providing ROS node offers a service under a string name, and a client calls the service by sending the request message and awaiting the reply. Client libraries usually present this interaction to the programmer as if it were a remote procedure call.

We have developed the Jupyter Notebook versions of the ROS tutorials for writing a simple publisher and subscriber of topics, and a simple service and client. The primary programming language is Python, but a working example in C++ will also be presented.

4.1 Writing a Simple Publisher and Subscriber

This tutorial consists of one section for the publisher node, and another section for the subscriber node.[10] Each section is divided into two sub-sections, a first part with the code, and a second one with the "code explained". This last part consists of the code split into chunks of a few lines, followed by paragraphs of description about each code fragment.

[8] https://hub.docker.com/_/ros/
[9] https://github.com/RobInLabUJI/ROS-Tutorials
[10] http://wiki.ros.org/ROS/Tutorials/WritingServiceClient(python)

We first divide the materials into two notebooks, for the publisher and the subscriber. The first section is not necessary, since the code in the "explanatory" section can be readily executed in the notebook. So this second section is transferred to the notebook by simply moving the source code into "code cells" and the description into the "markdown cells" (Fig. 4).

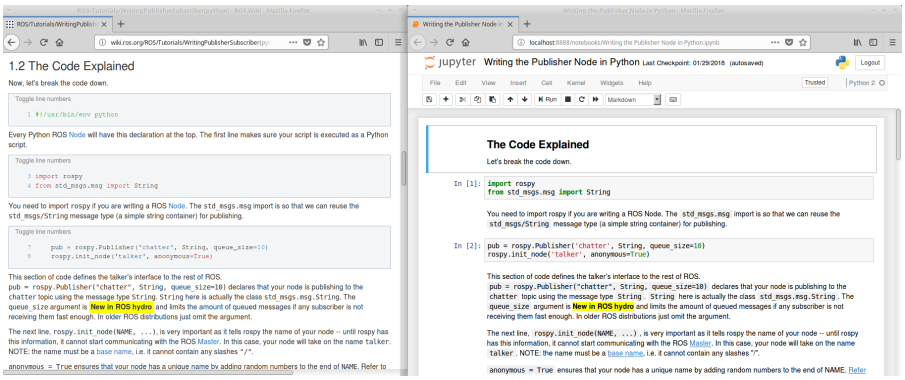


Fig. 4. Adapting the ROS Topics tutorial webpage (left) to a Jupyter Notebook (right). The source code is transferred to code cells, and the explanatory text is copied into markdown cells. The layout remains very similar, and the links are kept.

Some additional simplifications apply: the initial declaration for Python scripts is not necessary, since the Jupyter environment is already running a Python kernel. For the same reason, it is not necessary to add the lines of code for checking that a "main" application is running. One should keep in mind that for the code to be re-useable in another execution environment, the full source code should be used.

The subscriber code is adapted in a similar way. The resulting document has fewer lines than the original, since there is no duplicated code, and some unnecessary statements have been removed. In addition, it can be readily executed from the same browser, without need to launch the script in a console terminal (Fig. 5).

4.2 Writing a Simple Service and Client

The structure of this tutorial is similar to the previous one.[11] Again, the tutorial is split into two notebooks, for the service and client respectively. Each notebook consists of the contents of the section "code explained", either as code or markdown cells.

The code can be readily executed on each notebook in parallel, and the results of the remote call are shown in both notebooks too (Fig. 6).

[11] http://wiki.ros.org/ROS/Tutorials/WritingServiceClient(python)

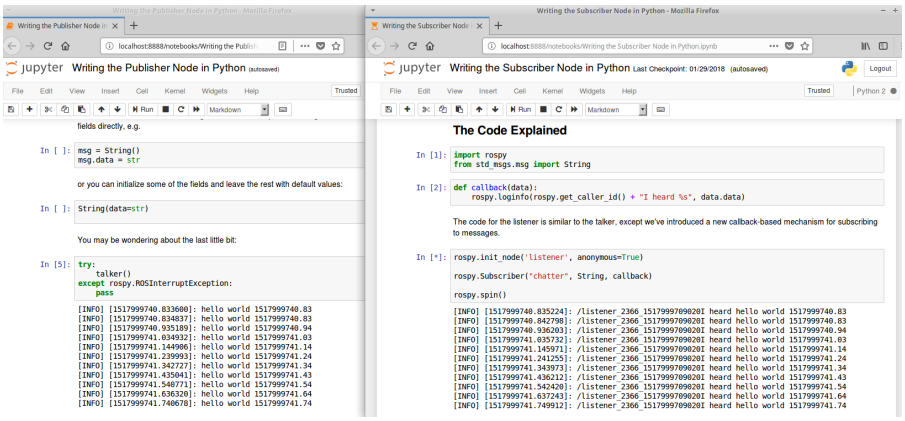


Fig. 5. Published and subscriber code running in Jupyter notebooks. The output of each node is displayed in the corresponding notebook.

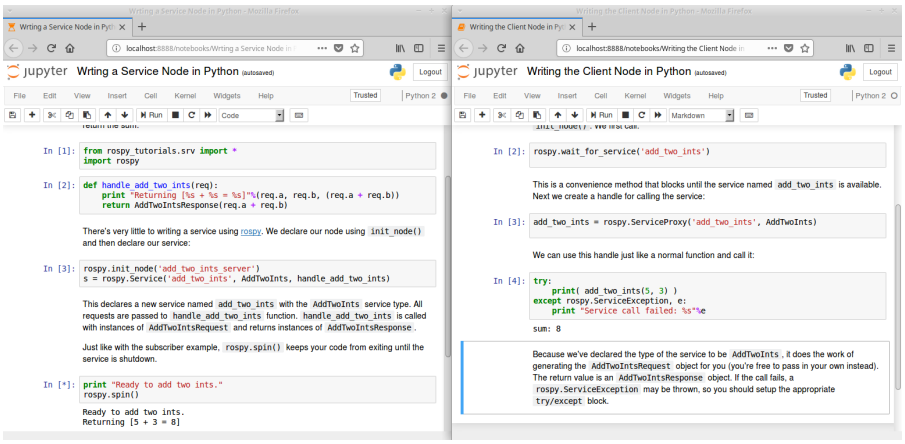


Fig. 6. Service and client code running in Jupyter notebooks. The output of each node is displayed in the corresponding notebook.

4.3 Adapting Tutorials in C++

Though Jupyter has evolved from IPython, much effort has been devoted to the separation between the kernel running the code, and the notebook frontend. Nowadays, more than 40 different kernels are available, each for a huge variety of programming languages.[12]

One of such kernels is based on Cling, an interactive C++ Interpreter [10] developed within ROOT, a framework for data processing created at CERN [1].

We have also adapted the C++ versions of the topics and services tutorials to jupyter notebooks. Similar to Python, slight changes are needed: the main function is removed, since the code is run interactively step-by-step; blocks of code cannot be split across cells, e.g. loops or if-else constructs. Care must be taken with the declarations, since any second run raises an error. The solution is to restart the kernel, which eliminates all declared variables for a fresh start.

The results of a C++ notebook for the tutorial on ROS services is shown in Fig. 7. These results are coming from a local install, since at this moment the Binder version of the tutorial only features the Python kernel.

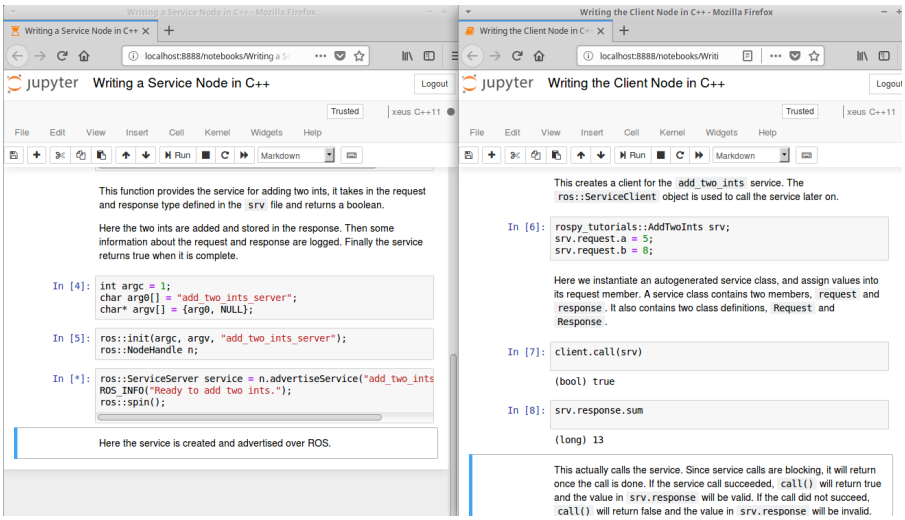


Fig. 7. Service and client C++ code running in Jupyter notebooks.

5 Conclusions and Future Work

Jupyter notebooks bring new live to ROS tutorials; instead of copying-and-pasting code and running it in a terminal console, the code is interleaved in

[12] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels

the same tutorial, where it can be run, and its output is displayed in context. Tutorials become more concise, and the user workflow is greatly simplified, thus enhancing the learning process.

Jupyter is an open-source, cross-platform project that has become enormously popular in the recent years. A variety of programming languages is supported, including C++ and Python, the most widely-used languages in ROS. In future extensions, we plan to test languages like Lisp, Go, Haskell, Java, Node.js, Julia, and Ruby: all of them are available both in Jupyter kernels and ROS client libraries.[13] [14]

The notebook platform is easily installed in a local computer alongside with ROS, or it can be deployed in a cloud server. The Binder web service allows any author to publish a tutorial in a public web repository that becomes live instantly, allowing other users to try ROS without even installing it in their computers.

Though the adaptation of existing tutorials to interactive notebooks can be time-consuming, as it is done manually by now, we encourage the ROS community to use these tools for the development of new materials, since teaching of robotics and ROS can greatly benefit from them.

In the future, we plan to evaluate groups of students for quantitative measurements of the user proficiency in comparison with the classical tutorial approach.

Acknowledgments

Support of IEEE RAS through the CEMRA program (Creation of Educational Material for Robotics and Automation) is gratefully acknowledged. This paper describes research done at the Robotic Intelligence Laboratory. Support for this laboratory is provided in part by Ministerio de Economia y Competitividad (DPI2015-69041-R), by Generalitat Valenciana (PROMETEOII/2014/028) and by Universitat Jaume I (P1-1B2014-52).

References

1. Antcheva, I., Ballintijn, M., Bellenot, B., Biskup, M., Brun, R., Buncic, N., Canal, P., Casadei, D., Couet, O., Fine, V., et al.: Root—a c++ framework for petabyte data storage, statistical analysis and visualization. Computer Physics Communications **182**(6), 1384–1385 (2011)
2. Kluyver, T., Ragan-Kelley, B., Pérez, F., Granger, B.E., Bussonnier, M., Frederic, J., Kelley, K., Hamrick, J.B., Grout, J., Corlay, S., et al.: Jupyter notebooks-a publishing format for reproducible computational workflows. In: F. Loizides, B. Schmidt (eds.) Positioning and Power in Academic Publishing: Players, Agents and Agendas, pp. 87–90 (2016)
3. Knuth, D.E.: Literate programming. CSLI Lecture Notes, Stanford, CA: Center for the Study of Language and Information (CSLI), 1992 (1992)

[13] https://github.com/jupyter/jupyter/wiki/Jupyter-kernels
[14] http://wiki.ros.org/Client%20Libraries

4. Millman, K.J., Pérez, F., Stodden, V., Leisch, F., Peng, R.: Developing open-source scientific practice. Implementing Reproducible Research **149** (2014)
5. Pérez, F., Granger, B.E.: Ipython: a system for interactive scientific computing. Computing in Science & Engineering **9**(3) (2007)
6. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA workshop on open source software, vol. 3, p. 5. Kobe, Japan (2009)
7. Shen, H.: Interactive notebooks: Sharing the code. Nature News **515**(7525), 151 (2014)
8. Tellez, R.: A thousand robots for each student: Using cloud robot simulations to teach robotics. In: M. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh (eds.) Robotics in Education, pp. 143–155. Springer International Publishing, Cham (2017)
9. Vallisneri, M., Kanner, J., Williams, R., Weinstein, A., Stephens, B.: The ligo open science center. In: Journal of Physics: Conference Series, vol. 610, p. 012021. IOP Publishing (2015)
10. Vasilev, V., Canal, P., Naumann, A., Russo, P.: Cling–the new interactive interpreter for root 6. In: Journal of Physics: Conference Series, vol. 396, p. 052071. IOP Publishing (2012)
11. White, R., Christensen, H.: Ros and docker. In: Robot Operating System (ROS), pp. 285–307. Springer (2017)
12. Wolfram, S.: Mathematica: a system for doing mathematics by computer. Addison-Wesley (1991)

The potential of a robotics summer course On Engineering Education

N. M. Fonseca Ferreira[1,2,3,4,*], André Araujo[5]
M.S. Couceiro[5] and David Portugal[5]

[1]*Engineering Institute of Coimbra (ISEC)*
[2] *RoboCorp, I2A, Polytechnic of Coimbra (IPC),* Portugal
[3]*Knowledge Research Group on Intelligent Engineering and Computing for Advanced Innovation and Development (GECAD) of the Institute of Engineering, Polytechnic Institute of Porto, Portugal*
[4]*INESC TEC – Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência (formerly INESC Porto), Portugal*
[5]*Ingeniarius, Coimbra, Portugal*

[*]*nunomig@isec.pt*

Abstract

RobotCraft is an international internship with a summer course in robotics designed especially for BSc to PhD students. The students attending this 2-months program have the opportunity to work in robotics, focusing on several state-of-the-art approaches, technologies and learned how to design, build and program their robots throughout multiple activities, carefully prepared to provide a wide range of skills and knowledge in the topic. This paper describes the methodology used to introduce participants to a hands-on technical craft on robotics and to acquire experience in the low-level details of embedded systems.

Keywords*: Engineering education, Project-based learning, educational robotics.*

1. Introduction

Robotics is a very attractive subject in the field of engineering. More frequently, educators find robotics a suitable project-based learning tool. Using robots as a teaching tool, can lead to the acquisition of knowledge and skills in several engineering areas, such as electrical, mechanical and computer engineering areas. As can also provide the students with problem solving, teamwork and self-taught skills. With the educational benefits in mind, world-widely, some educators have been creating for students extra-curricular activities involving robotics, such as Robotics Summer Camps and Robot Competitions [1-5]. Robot contests present several successful designs for projects surveyed by students in universities, colleges and schools. These contests can offer engineering assignments of different levels, from a high-school competition [6-7] to advanced research programs such as the robotic

soccer initiative, or pose a challenging problem, designing a robot that can navigate autonomously through a maze, find a lit candle, and extinguish it in minimum time.

As a multi-disciplinary subject, robotics involves physics, mathematics, control, programming, computer-aided design and hands-on technical skills. The primarily focus of the robotics programs are different, while a Computer Science robotics program may focus on the high-level algorithms used for image recognition and navigation, a mechanical engineering program may focus on the manipulation of servos and motors to complete specific tasks. For college students looking to become involved in robotics, however, it can be difficult to find an introductory course that empowers them with the knowledge to construct and operate their own autonomous robots. The RobotCraft is an international internship with a summer course in robotics designed especially for BSc to PhD students. The students attending this 2-months program have the opportunity to work in robotics, focusing on several state-of-the-art approaches and technologies. The summer course, now in its second edition and entitled as the 2nd Robotics Craftsmanship International Academy.

RobotCraft 2017 received around 100 applications, but just 84 attended the summer course. The attendants came from a wide range of countries, namely Egypt, Spain, Jordan, Lebanon, Palestine, Portugal, Sweden, Turkey, Germany, Algeria, Estonia, Finland, United Kingdom, Greece, Hungary, Italy, Morocco, Malaysia, Netherlands, Romania, Russia, Kazakhstan Syria and Kosovo.

2. International Summer School Program

This summer school program designed to bring engineering students from all over the world as a way to experience life and learning hands-on technical skills. The program provided a solid learning opportunity for international students and presented two challenges. The first challenge was the wide range of educational backgrounds from the students. As a result, this course had to be accessible to students who had never worked with embedded systems before, while at the same time, it needed to engage and challenge those students who already had some robotics project experience. This was the second major challenge faced; all of the presented material had to be interesting and engaging enough to keep participants interested on the course subjects, meeting the different needs of the international students.

In order to support the wide range of background and skills level of the students, the course was layout into six different topics, each with the duration of approximately one week. The topics are summarized in Table 1. For each of these topics, the participants attended a seminar, lectures and several practical sessions (Table 2.) The seminars presented were on enthusiastic topics and this learning activity allowed the participants to have contact with researchers referred to each expertise field. Also as part of their learning activities, as shown on Table 3, the existence of practical assignments, in order to see results early on in the learning process, while introducing concepts, allow the more advanced participants to customize their systems [8-9]. The methodology used on this course allowed participants to accelerate their learning processes, and also the development of systems thinking and the skills

of intensive purposeful teamwork; reducing the gap between background, theoretical and practical activities.

Table 1. Course Schedule and Outline

Schedule	Topic	Brief Description
First and second week	Introduction to Robotics	✓ History of robotics and its evolution
		✓ Mobile robot morphologies (namely sensors and actuators)
		✓ Brief literature review (basic theoretical concepts)
Third week	Computer-Aided Design (CAD)	✓ 3D modelling tools
		✓ 3D printing
		✓ Model a 3D structure for the mobile robotic platform
		✓ 3D print the personalized 3D structure
		✓ Assemble the mobile robotic platform
Fourth week	Arduino Programming	✓ C language applied to Arduino programming
		✓ Features of Arduino solutions (e.g., hardware architecture, cycles, communications)
		✓ Identify different wireless communication technologies
		✓ Low-level algorithms, flowcharts and pseudocode
		✓ Develop a typical differential kinematic application
Fifth and sixth week	Robot Operating System (ROS)	✓ ROS features (e.g., packages publish-subscribe, topics)
		✓ ROS-compatible simulators (Stage)
		✓ High-level algorithms, flowcharts and pseudocode
		✓ Develop a typical remote sensing application
Seventh and eighth week	Artificial Intelligence (AI)	✓ Different paradigms and some real applications
		✓ Integrating biologically-inspired models
		✓ Formalizing a biologically-inspired approach
		✓ Develop a streaming architecture to exchange all necessary data (e.g., sensor readings, encoder's readings, actuators control, etc.)
Last day	Competition	✓ Mobile robot platform maze competition
		✓ Mobile robot Patrol competition: algorithm testing
		✓ Prize delivery

The practice is fundamental in the learning process and can offer educational advantages: the participants acquired skills are required in many professional fields and various science methods studied, can be apply on robot navigation and other functions. The assignments provided to the students were creative and involved instructive activities. The course schedule planning accounted the following factors: Each topic should be preceded by its prerequisite topics; Each topic should be learned in parallel with the linked topics; Combination of subjects and balance of theoretical, seminaries and lab studies are desired; Seminaries presented by researchers in the specific field of each workshop is extra motivation to the participants, this stimulate the creative and guided by innovation, which suggests a professional who is capable of maintaining the skills and knowledge updated to recent scientific–technological advances. The team assignments given in each week, allowed the participants to cooperate as a team and to work more independently. Table 3 shows the learning activities used to achieve the objectives described.

The final competition, in the end of RobotCraft, had two different goals: maze solving and patrolling attributes. In the maze scenario, the robot needs to find its

way through the maze; where the evaluation contemplates several conditions: the distance to the maze's exit elapsed, the time and the number of wall collisions.

Table 2. Seminar, lectures and practical sessions

	Description	Methods used	Objectives	Assessments
Seminar	Invited Talk (45 min + 30 min)	Audio and visual materials. Discussion between Oral Speaker and participants.	Engage students to this particular area of knowledge. Provide students with the state-of-the-art developments.	Feedback from the audience. Pertinent questions and students interaction . Interest shown during the presentation.
Lecture (theoretical lesson)	Talk given by one of the resident teachers (1hour + 20 min)	Content well organized and structure. Audio and visual materials. Discussion between teacher and participants.	Provide students with the basic theoretical contents. Promote parallel learning with linked topics.	Oral Questioning. Tutorial exercises.
Pratical sessions (lab practice)	4 to 8 hours per day of Lab practice, supervised by 2 to 4 teachers	Active involvement, through hands-on projects. Challenging team assignments.	Emphasize concept application. Foment team-learning activities. Foster and develop critical thinking.	Oral Questioning. Team and individual capabilities on solving problems and developing critical thinking.

Table 3. Learning Activities.

Objectives	Learning Activities
Implementation of basic system functions	Work with instructional modules. Lectures provided in the context of each module and the tutorials provide structured information for the participants.
Design and construction of the system	Teamwork on practical project assignment.
Implementation, control and communications	Work on research and Lab practice. Participants need to develop the proposed assignments and to conclude the final project. System of extra point's reward, to increase motivation and development of all the proposed tasks.
Adaptation of the system to the real environment and prepare to the competition	Lab practice and assignments.

And in the patrol mission, the robot needs to patrol, cooperatively, a given region, minimizing the idleness of all points of interests; therefore, the evaluation of this patrol mission is on the average idleness. Table 4 shows for each subject approached during the course, the intended learning objectives and the observed outcomes, as well as an example of a proposed assignment given to the participants.

Table 4. Subjects - Learning Objectives, Assignments and Outcomes.

Subject	Intended Learning Objectives	Proposed Assignment	Observed Learning Outcomes
Robotics	Identify mobile robot morphologies Implement, develop for functional architecture to a mobile robot.	Simple tasks where both circuit and program needed to be changed, e.g. modifying the communication protocol start code.	All the participants achieved the intended learning objectives. All groups completed the assignment with good remarks by the teachers.
Computer-Aided Design (CAD)	Identify 3D modelling tools and printers Execute a 3D modelling tool (FreeCAD) Create and print a 3D structure	Participants must design a creative robot housing. The robot housing should hold the 2-ultrasound sensors (left and right sensors), 1 infrared sensor (front sensor) and 4 LEDs.	All the participants achieved the intended learning objectives. All teams showed creativity in the design of the 3D structure.
3D mobile Robot	Assemble the printed 3D structure Assemble all mechanical components	Participants must follow a given hardware architecture in order to construct their mobile robot platform	All groups assemble their mobile platforms. All participants understood the hardware architecture.
Arduino Programming	Apply C language in Arduino programming Create the interface to link the Arduino board with the sensors and actuators	Create a function that reads the ultrasound sensors and converts its measurements in millimeters. Create a function that reads the difference between the numbers of pulses counted by the encoders on each wheel since last request.	The participants shown good response to the Arduino module. All groups were able to plan, organize and execute the tasks.
Kinematics and Control	Relate kinematics with the robot control system Create and implement a kinematic model of a differential drive robot	Adapt and merge the codes to the real hardware, comprising linear and angular velocities on the control of speed and the direction of both wheels.	The evaluation of all participants was positive, highlighting the interpersonal help between each team.
ROS Architecture	Interpret and operate in a ROS environment Explore ROS features Relate Arduino task with ROS architecture	Create a ROS package, that contains a node capable of subscribing 3 topics provided by the code developed in the previous task in Arduino.	All participants shown some difficulties upon the introduction of ROS. The assistance and help of the teachers were fundamental and on this module, they overcome most of their drawbacks by team interaction.
Simulating with Stage and ROS	Sketch a robotic simulation setup and implement the mobile robot platform in ROS. Execute Stage software in ROS and evaluate the mobile robot performance.	In a ROS package, create the needed files to simulate a virtual world with a robot in Stage. The extra goal is to have the robot mapping the environment with laser scans, in parallel with other tasks.	Almost all groups achieved the intended learning objectives. Robot design creativity used in Stage, rewarded with extra points.
Artificial Intelligence (AI)	Illustrate and label different AI approaches Implement and compare AI algorithms	Implement a simple algorithm inspired on biological systems, e.g. an ant algorithm.	Almost all groups developed an ant algorithm. 2-3 groups developed and implemented a more advanced AI algorithm.
Competition	Operate the mobile robot platform in a real 3D scenario maze). Assess the performance of the surveillance algorithm (patrol).	Conclude the algorithm development of the mobile robot platform. Evaluate and carry out final improvements.	All groups were able to develop a full operating mobile robot platform. 10 of 15 groups enter the maze final competition and just 3 teams concluded a successful surveillance algorithm.

3. Robot Craftsmanship

The course developed to be a practical hands-on experience for students of various backgrounds; and to engage students on robotics, met some specific criteria: the use of hardware and software supported by large communities, allowing students the benefit of finding help and examples online, both during and after the course.

All the devices used were relatively affordable, so that students could easily purchase their own components to tinker with, after the course. Although simplistic, the mobile robotic platform assembled, needed to comprise all relevant components inherent to mobile robotics (Figure 1).

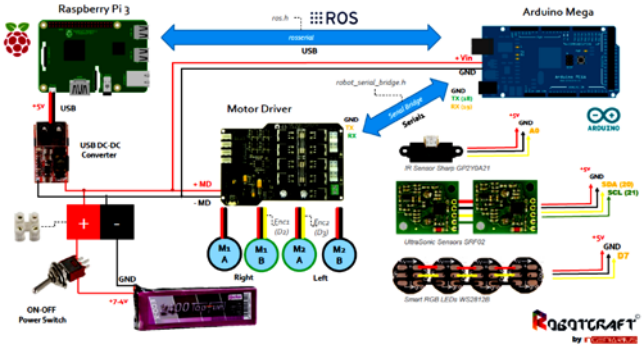


Fig. 1. Main hardware parts of the robotic system.

After the assembly of the platforms, students were introduced to C language and to some common algorithms in mobile autonomous robotic topics, such as mobile robotic kinematics, motion control, localization, path planning, among others. They started merging the developed algorithmic into systems capable of basic autonomous functionality and evaluate it considering the robot performance and then, improving the developed code.

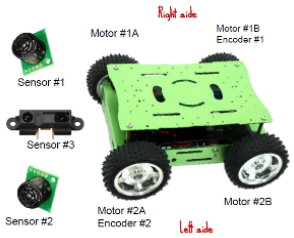


Fig. 2 The mobile robot platform.

As they develop skills working with ROS (Robot Operating System), writing robot software in a flexible framework, they acknowledge that several kinds of robot bases have common points: wheels, motors, odometry, among others. The interprocess communication is an important feature to the overall process. The robot

needs to see obstacles and decide where to go next (reactive walk). For this, it continuously needs to read laser scans to make decisions, where through a simple algorithm; it sends commands to the base. This is a kind of service used on any mobile robot. Simple service, like navigation consists on the determination of a valid trajectory between two points, provided by a map. Knowing the robot position, the localization of the robot in space is possible. Synchronous communication is an important issue when defining goals for the robot to move, for determining the possible paths and for knowing when the robot got there.

In order to avoid harming the robot or oneself, they simulated their approach before attempting it in the real robot platforms. They used Stage (OpenSource software), a standalone robot simulation program, on the ROS platform and were able to simulate multi-robot tasks in a ROS packages (e.g., coverage, patrolling, formation control, exploration, mapping, and it can include robots, sensors, actuators, moveable and immovable objects). The attendants learn to configure properly a workspace, to set up and run the simulation program, and to create a ROS package for the simulations. They were able to test and validate their project.

In the final week of the course, participants worked together on the development and improvement of their mobile robot platforms. They gained experience in how to accomplish tasks, in problem solving and in design decisions. Instructional time was primarily spent guiding attendants through the implementation of algorithms, and working through the difficulties and pitfalls of real hands-on development. Their skills in scheduling timelines, teamwork and compromise were improved. One noteworthy event was by the end of the last week, some teams realized that they would not be able to complete the project in time to enter the competition. In order to meet this goal, opposing teams worked together and even shared algorithms and code. At the end of the week, all teams had developed robots that could autonomously compete.

In the final day, the competition took place, and comprised two different objectives: first, the maze solving and second, the patrolling attributes (Fig. 3).

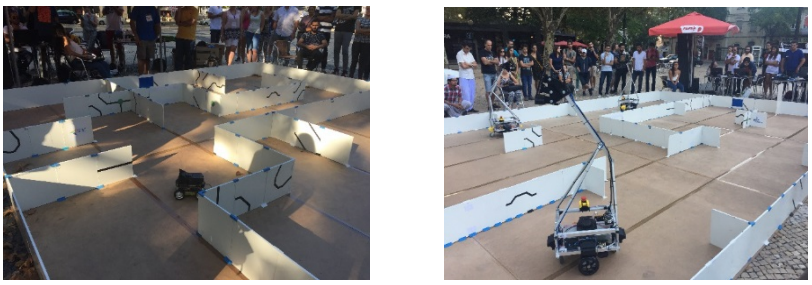


Fig. 3. Competition day: maze solving (left) and patrolling scenario (right).

Figure 3 shows the maze scenario, where the robot needs to find its way through the maze and the patrol mission, where robots needed to patrol cooperatively a given

region, minimizing the idleness of all points of interests. The maze scenario assessment was through the distance elapsed, time and number of collisions and for the patrol scenario was through the average idleness.

4. Surveys

To obtain a formalized feedback of the course, participants took two surveys. The first was answered by 96% of enrolled attendants. The main purpose of this survey was to identify the overall knowledge, of each participant, in different related topics. The second, taken in the last seminar by 77% of enrolled participants, aimed to get feedback from the attendants, about their expectations and to provide a useful overall evaluation of the course.

4.1. Participants

During the first seminar, 81 participants answered the initial survey, corresponding to 96% of enrolled attendants and came from twenty different countries. Being an intensive summer course in English language and disseminated in several information channels, Portugal (the host country) is second with just 7% of student participation behind Turkey, representing 51% of enrolled students.

The attendants became aware of the existence of this summer course through several channels of information. The more important ones were through friends and colleagues, social media and Erasmus channels, representing 70% of the enquiries.

From the 81 attendants that answered the initial survey, 92.5% were university students in their home countries, 79% had ages between 20 to 24 years old and 75% of them were male. BSc, MSc and PhD students, corresponded to 80%, 10% and 2.5% of participants, respectively. Figure 4 shows the distribution of participants according to the area of specialization. The others 7.5% already concluded their studies and were not involved in a university course.

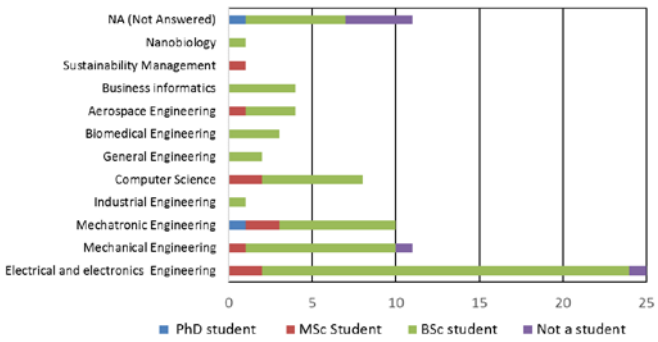


Fig 4. Number of participants according to their area of specialization.

As is it shown on figure 4, 80% of the participants have a background on, or are attending, a university course on engineering. Electrical and electronics engineering

is the area with most participants, 31%, against 26% of participants with a mechanical or mechatronics engineering background (14% and 12% respectively); 10% are attending a Computer science course, 5% and 4% of them, are students on Aerospace and Biomedical engineering, respectively.

When asked, what were the main reasons (up to 3) for enrolling in this course; participants gave different and diverse reasons. Some wanted to have an educative summer, others to learn more on ROS, C# and/or Artificial Intelligence; others the main purpose was to make an internship, or visit Portugal (9%), or to improve their English. Most of them, around 42% shown to have personal interest in acquire experience in robotics. Around 47% of the attendants said they had already built a robot before.

4.1.1 Women participation

From the last decades the number of women in engineering courses has been increasing [10]. This edition, has been no exception, there was an increase of the percentage of women involved. There were 84 attendants, 25% of the enquiries were female, corresponding to an increase of 20% of female participation from last year edition. These female attendants came mainly from Turkey, followed by Hungary and Morocco with 40%, 20% and 15% of participation, respectively. 80% of them are BSc students, with ages between 20 and 24 years old. Their areas of specialization are mostly on engineering, with 25% on Electrical and Electronics Engineering, 20% on Business Informatics and 15% on Computer Science.

4.2. Participants knowledge

The initial survey had a series of questions, aimed to access the overall knowledge of the participants in some areas, such as Computer-Aided Design, 3D Printing, Mechatronics, Arduino Programming, Kinematics, Control, ROS and Artificial Intelligence. Figures 5 and 6 illustrate the responses to six of the survey questions, based on a five point Likert Scale [11]. Likert Scales have the advantage that they do not expect a simple answer (yes or no, good or bad) from the respondent, but rather allow degrees of opinion, and even no opinion at all. For example, there are Agreement, Frequency, Importance and is assumed that the experience is linear. The left and right extremes, correspond to numbers 1 and 5, respectively. And it is assumed that there is a continuum of possible answers from the left to the right of the scales, that is, from Never to Very Frequently, or from Unimportant to Very Important, and a choice of five pre-coded responses can be given, with the neutral point being occasionally or moderately Important [12]. Figure 5 shows the current understanding on the topics and reveals that most students do not understand a large part of these topics. In fact, only 4 participants worked with ROS before starting the course.

Fig. 5. Initial current understanding on RobotCraft topics.

Also the background in some subjects like electronic, computer, assembly language, show that the participants have an overall poor knowledge and lack of hands-on experience.

4.3. Participants reactions

Figure 6 illustrates a comparison made with the initial and final surveys taken by the participants, the topics, which they had, a non-relevant initial understating are ROS with 67%, Artificial Intelligence with 49%, followed by Kinematics, Mechatronics, Control and 3D printing with a percentage of around 40%. The topics where the seminars were more important in the context of the course were the lectures within Arduino, Kinematics, ROS, Control and Artificial Intelligence, with 55%, 57%, 66%, 62% and 68%. These were also the topics where the evaluation of the seminar lectures were more relevant, with 43%, 38%, 40%, 45% and 49%, considers that the evaluation was positive. When comparing the initial and current understanding on each topic, when comparing the initial and current understanding are ROS topic with a 29% drop, from 67% to 38%, Mechatronics with a 17% drop from 42% to 25%, followed by Kinematics and 3D printing with a 15% and 14% drop. In fact, ROS, Kinematics and Arduino topics had a very subtle increase of 10%, 2% and 2% of participants with a relevant current knowledge on the topic. When asked about the difficulty of these topics, the ones that had more percentage of non-relevant knowledge and higher relevancy of the seminars lectures to their understanding, ROS, Control and Artificial Intelligence appear with 51%, 46% and 48% of percentage of participants alleging they were difficult topics to learn. In fact, about ROS the participants felt this was a very important topic of the robotics course, but it is very difficult to learn in just two weeks. Based on formal and informal feedback, the course was successful in providing the participants with a meaningful introductory, yet comprehensive robotics experience. In addition, their feedback is important to improve the overall quality of this course.

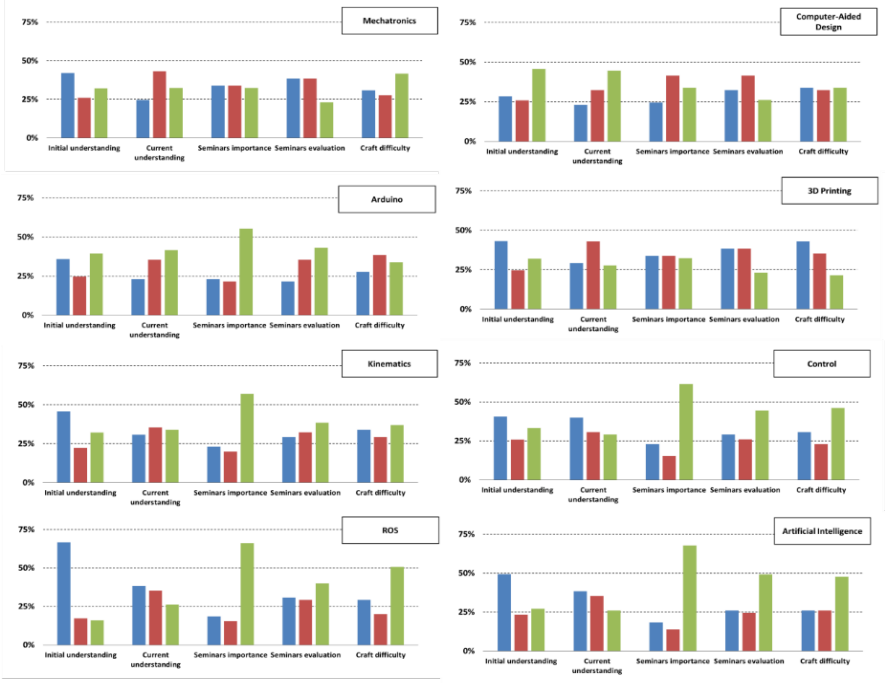


Fig 6. Participants opinion on the topics address

5. Conclusions

A two months robotics course, aimed for international students from varying engineering backgrounds, with the advantage of coupling various skill levels, was successful. The methodology used, had the ability to give to participants an appropriate introduction to a complete robotics design experience. The participants saw their academic knowledge on some engineering subjects improved. The methodology used, developed not just their technical skills but social also, through teamwork. Even a moderate knowledge increase on some approach subjects is a finding that robotics, if well approached, can be a multi-disciplinary learning platform.

Acknowledgments

This work is financed by the ERDF – European Regional Development Fund through the Operational Programme for Competitiveness and Internationalisation - COMPETE 2020 Programme within project «POCI-01-0145-FEDER-006961», and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) as part of project UID/EEA/50014/2013.

References

[1] E. Kolberg and N. Orlev, "*Robotics learning as a tool for integrating science technology curriculum in K-12 schools*," presented at the Frontiers in Education Conference, 2001.

[2] F. B. V. Benitti, "*Exploring the educational potential of robotics in schools: A systematic review*" Computers & Education, vol. 58, no. 3, pp. 978-988, 2012.

[3] A. Melchior, F. Cohen, T. Cutter, T. Leavitt and N. Manchester, "*More than robots: An evaluation of the first robotics competition participant and institutional impacts*," Heller School for Social Policy and Management, Brandeis University, 2005.

[4] S. Grover, "*Robotics and Engineering for Middle and High School Students to Develop Computational Thinking*," in Annual Meeting of the American Educational Research Association, 2011.

[5] W. W. Lau, G. Ngai, S. C. Chan and J. C. Cheung, "*Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students*," SIGCSE Bull., vol. 41, no. 1, pp. 504-508, March 2009.

[6] A. Welch and D. Huffman, "*The Effect of Robotics Competitions on High School Students' Attitudes Toward Science*," School Science Mathematics, vol. 111, no. 8, pp. 416–424, Dec. 2011.

[7] 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, no. Part B, pp. 686–691, Jan. 2016.

[8] Igor M. Verner, Shlomo Waks and Eli Kolberg, "*Educational Robotics: An Insight into Systems Engineering*", European Journal of Engineering Education, 24:2, 201-212, DOI: 10.1080/03043799908923555, 1999

[9] Moshe Barak, "*From 'doing' to 'doing with learning': reflection on an effort to promote self-regulated learning in technological projects in high school"*, European Journal of Engineering Education, Volume 37, Issue 1, DOI: 10.1080/03043797.2012.658759, 2012.

[10] Duyen Q. Nguyen, "*The Status of Women in Engineering Education*", Int. J. Eng. Ed. Vol. 16, No. 4, pp. 286-291, 2000.

[11] Likert, R., "*A Technique for the Measurement of Attitudes*". Archives of Psychology, 140, 1–55, 1932.

[12] Paulhus, D. L., "*Two-component models of socially desirable responding*, Journal of personality and social psychology, 46(3), 598, 1984.

Hands-on Robotics Teaching with ROS

Roel Pieters, Reza Ghabcheloo and Minna Lanz

Tampere University of Technology, Tampere, Finland,
roel.pieters@tut.fi

Abstract. This paper describes the teaching efforts in several courses in our newly started robotics major in Tampere University of technology. While the fundamentals and theory in robotics are covered by traditional courses, hands-on experiments and student projects use ROS as software platform. ROS is introduced by an overview lecture with practical demonstration and aims to point students towards information and resources instead of providing ready-made solutions. This approach leads to much trial-and-error and a steep learning curve that, we believe, is highly valuable. We reflect on an early evaluation of our approach and conclude that the advantage of ROS as tool to educate robotics is due to its holistic nature. From low-level concepts (e.g. interfacing, communication, diagnostics) to high-level functionality (e.g. visualization, manipulation, SLAM), all are included in ROS and required to educate the next generation roboticists.

Keywords: robotics, education, Robot Operating System

1 Introduction

Since the advent of robotics, considerable effort has been put in the development of theory, algorithms and their implementation. Similarly, such effort should also be represented in the education of robotics at academic institutes [1–4]. The balance between theoretical foundations and practical work is, however, not trivial to find, and often complicated by the multitude of available software. Traditionally, robotics education has gone hand-in-hand with programming (e.g. C/C++) or higher-level computing environments such as Matlab®. In recent years, however, research has moved slowly to adopt ROS as middleware [5], and many other libraries and interfaces offer some form of integration towards it (e.g. ROS in Matlab). Education has caught up to this and many higher institutes of education now offer ROS in course form, or require ROS to be used as programming interface for project work[1]. Additionally, professional education on ROS that is not part of a curriculum can be found in summer schools [6], conferences[2], online courses and in bookform[3]. The take-up of industry, however,

[1] https://wiki.ros.org/Courses
[2] https://roscon.ros.org
[3] https://wiki.ros.org/Books

is slower but initiatives are ongoing to provide the reliability and robustness that is required in industrial environments (EU H2020 project ROSIN[4] [7]).

As educational design is mostly done behind closed doors, with this brief paper we report the route taken at Tampere University of Technology, in the recently started robotics major. By providing an overview of our approach and an early evaluation of outcomes, we hope to open up a discussion on best practices with respect to robotics education and ROS.

2 Robotics major

The robotics major at Tampere University of Technology started in September 2017, and offers MSc students to specialize in robotics and the technology related to it. The degree is offered by the Faculty of Engineering Sciences. The most significant contributions come from the laboraties of Automation and Hydraulic Engineering (AUT) and Mechanical Engineering and Industrial Systems (MEI), although other units of the university, for example Signal Processing, also contribute. The two year program (120 ECTS) requires students to have solid mathematical background and good programming skills. Studies include the fundamentals of robotics, control of robotics, sensing/perception systems for robotics, robot programming and project work. After the studies, students are able to develop robot-based systems. The major consists of four mandatory courses complemented with elective courses that go deeper in a certain topic. In several of these courses, experimental work includes the use of ROS to connect to and control robotic systems, as described in Table 1 and 2.

2.1 TUTRobolab

While all traditional lectures are given in lecture rooms, practical work and demonstrations make use of a laboratory environment specifically designed for students. This TUTRobolab offers a place for students to work with robotic equipment and experiment without major restrictions. Robots and related technology are for example industrial manipulators (Universal robots, Franka, PRob), mobile robots (turtlebot, in-house developed robot), a multitude of sensors (2D/3D ToF cameras, lidar, GPS, IMU, etc.) and different processing platforms (PCs, embedded PCs, Raspberry Pi). While giving preference to course students, the lab is available 24/7 to all students interested in robotics and aims to create a casual learning environment.

2.2 Mandatory courses

The core of the major consists of four mandatory courses, as described in Table 1, designed to educate students on the basics and practicals of robotics and includes project-based work to prepare them for future professional work life. All courses emphasize practical implementation and require programming to be part of the work.

[4] http://rosin-project.eu/

Table 1: Mandatory MSc courses as part of the robotics major

Mechatronics and Robot Programming	*5 ECTS*	*Lab exercises*
Provides an introduction to sensors and robots. Student groups (2-4 persons) complete exercises ranging from sensor interfacing and read-out, visualization, installation and set-up for different processing platforms (PC, Raspberry Pi, wireless networking, SSH) and robot platforms (motors, turtlebot, crane), to advanced algorithms (simulation, SLAM, navigation). All exercises are demonstrated by each group, questioned to assess each member's understanding and finalized by a report.		
Robotics Project work	*6 ECTS*	*Group project*
Provides project-based group work. Student groups (2-4 persons) pick a topic and develop a robotics project, emphasized on interdisciplinary collaboration. Lectures are provided by speakers from industry and academia. The project requires students to develop project planning and management skills, and collaborate on multiple disciplines. Typical projects include LEGO house building with a UR5 robot, indoor delivery robot and upgrading an industrial robot manipulator.		
Robot Manipulators: Modeling, Control and Programming	*4 ECTS*	*Theory, practicals*
Provides a traditional introduction to robot manipulation. In this course students obtain knowledge on robot modeling (kinematics and dynamics, Denavit-Hartenberg notations), and programming for industrial manipulators. Exercises use Matlab and the course is finalized with an exam.		
Fundamentals of Mobile Robots	*5 ECTS*	*Theory, practicals*
Provides the fundamentals of mobile robotics including localization, mapping and planning. Extensive exercises use Matlab and ROS.		

2.3 Elective courses

Besides the four mandatory courses, additional credits have to be taken to complete the major. These courses can be on specific topics such as computer vision and machine learning, or more focused on specific fields within robotics, such as planning and advanced control (e.g. see Table 2). In both mandatory and elective courses, where possible, ROS is used to ease the education of robotics and robotics related technology.

2.4 Teaching philosophy

The approach we take to educate robotics on a MSc level, is to find an appropriate balance between theory and practice. While the fundamentals can only be taught by introducing the relevant mathematical theory, a deeper understanding is obtained when such theory can be tried and tested in simulation or experiments. Take for example coordinate systems and their transformation.

Table 2: Elective MSc course as part of the robotics major that includes ROS

Advanced Robotics	*5 ECTS*	*Theory, practicals*
Provides advanced theory on motion control for robots, such as dynamical systems, stability theory, visual servoing, force control, etc. Theory is supported by practical implementation and integration on an advanced robot manipulator (Panda, Franka Emika GmbH).		

Understanding the theory and its use can be assisted by tools such as Matlab[5] and ROS[6] that ease implementation and visualization.

A choice was made to not have a separate course purely on ROS, as other institutes offer (such as e.g. ETH Zürich[7]), but only provide an introductory lecture. This 1.5 hour lecture provides the basics of Linux and ROS, and shows students where to find information and how to develop their skills. In case of difficulties or problems student assistants are available to assist and help out with practical matters.

3 Evaluation

The first intake of the major attracted a significant number of students, where a selection had to be made based on background and motivation. Such decisions are difficult as grades and background do not always represent the best selection criteria. For example, giving preference to students that have experience and background with ROS might be suitable for elective courses (e.g. Advanced Robotics), but not for mandatory courses, as ROS will be introduced there. Despite this, the majority of students successfully passed, or are about to pass, the courses. Evaluation of (ROS) skills was assessed by oral examination and practical demonstration of the developed software.

3.1 Lessons learned

Lessons learned for the educators of the robotics major are mostly related to the time and effort necessary to prepare course work and practical experiments. It is very easy to underestimate the time required to set up simulations and experiments and verify that all will function properly on different platforms. This applies similarly for laboratory systems that are in continuous use for different projects. Software versions and hardware differences might simply not be compatible. It is difficult to anticipate for this and to estimate how much time and effort it takes for students to master such realization.

[5] https://se.mathworks.com/help/robotics/gs/coordinate-systems-in-robotics.html
[6] https://wiki.ros.org/tf2
[7] http://www.rsl.ethz.ch/education-students/lectures/ros.html

Even though ROS is a major step forward in providing a standard in robotics education, this also is a drawback. It is very convenient to install a package, execute the algorithms it provides and demonstrate a functioning system. This does not imply, however, that the fundamentals behind the algorithms and robot are understood. Such understanding has to come from studying the algorithms and experimenting with different parameters, or by developing and/or extending the methods. Again, we believe that by providing the tools and problems, and requiring from students a solution, will lead to a deeper understanding of the theory.

Additionally, not all students have the same background, leading to gaps in programming capabilities and hands-on experiments with software coding. This leads to big differences in outcomes and progress throughout a course. One solution to this is to require a certain level of programming skills in order to be accepted in the major. Despite this, we found good results in creating group work with different skill sets. Even though this might cause one person to take the lead in project work, this can be compensated to ensure all participants have to demonstrate the learned skills. Group work also leads to students learning valuable skills necessary in professional work life, such as educating colleagues, team work, effctive time and project management, and communication.

Feedback that was obtained from students after finalizing a course gave valuable information on how to improve the education. Issues that were raised were for example, the lack of a lab technician for quick questions and small practical issues, and the deployment of a virtual workspace where students can collaborate, discuss and share ideas (e.g., Slack[8]).

3.2 Post-graduate requirements

In Finland, it is common to do a MSc thesis in a company, where ROS is most likely not being used. Why should an academic institute then spend so much effort on educating their students with ROS? Our answer to this is that it's impossible to ensure a common framework that is supported by all involved partners. Even the most common tool for studying engineering (Matlab), is still underrepresented in industry due to its high cost. Our main aim is to educate future engineers with the skills and capabilities to design robotic systems. This whole overview of robotics ranges from interfacing sensors and actuators, to control architectures and high-level industrially relevant applications. With so much diversity a common ground is found in ROS, as it supports all. Moreover, different programming languages can be used, to accomodate different educational backgrounds. Through ROS, students practice C/C++ or Python which is commonly used in industry, and learn what a complex robot control system should include.

[8] https://slack.com/

4 Conclusion

This short paper described the efforts made in Tampere University of Technology with respect to robotics education. Our approach is to balance theory with significant hands-on experimental work, by utilizing the Robot Operating System. The advantage that ROS has over other robotics software, such as Matlab, is that it provides a complete overview and implementation of required tools. From low-level concepts such as communication and interfacing to high-level algorithms, it educates roboticists the basics and pushes the state-of-the-art. Drawbacks of ROS are found in the limited adoption in industry and in compatibility or support issues (e.g., unsupported versions, hardware/software version mismatch). Despite this, a first evaluation of our robotics major suggests that students benefit greatly from a community-supported initiative such as ROS. The large user-base and growing support for robotic equipment, means that information can be found and asked for on-line. This brings a steep learning curve that, we believe, is highly valuable.

Acknowledgements

The work presented is financed by the Academy of Finland project: "Competitive funding to strengthen university research profiles", decision number 310325.

References

1. Touretzky, D.S.: Computer Science Education: Robotics for computer scientists: whats the big idea? Computer Science Education, vol. 23, no.4, pp. 349-367, 2013.
2. Correll, N., Wing, R. and Coleman, D.: A One-Year Introductory Robotics Curriculum for Computer Science Upperclassmen. IEEE Transactions on Education, vol. 56, no. 1, pp. 54-60, 2013.
3. Esposito, J.M.: The State of Robotics Education: Proposed Goals for Positively Transforming Robotics Education at Postsecondary Institutions. IEEE Robotics & Automation Magazine, vol. 24, no. 3, pp. 157-164, 2017.
4. Sergeyev, A., Alaraje, N., Parmar, S., Kuhl, S., Druschke, V. and Hooker, J.: Promoting industrial robotics education by curriculum, robotic simulation software, and advanced robotic workcell development and implementation. Annual IEEE International Systems Conference (SysCon), Montreal, QC, pp. 1-8, 2017.
5. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R. and Ng, A.Y.: ROS: an open-source Robot Operating System. In ICRA workshop on open source software, Vol. 3, No. 3.2, pp. 5, 2009.
6. Ferrein, A., Kallweit, S., Scholl, I. and Reichert, W.: Learning to program mobile robots in the ROS summer school series. In Proceedings of the 6th International Conference on Robotics in Education (RIE-15), 2015.
7. Ferrein, A., Schiffer, S. and Kallweit, S.: The ROSIN Education Concept. In Iberian Robotics conference, Springer, Cham, pp. 370-381, 2017.

Learning Advanced Robotics with TIAGo and its ROS Online Tutorials

Jordi Pages, Luca Marchionni, Francesco Ferro

PAL Robotics S.L.
c/ Pujades 77-79, 4-4, 08005 Barcelona, Spain
jordi.pages@pal-robotics.com, luca.marchionni@pal-robotics.com,
francesco.ferro@pal-robotics.com

Abstract. Learning Robotics in these recent years has become more easy thanks to ROS and its worldwide success [1]. ROS has managed to reduce the learning curve for newcomers as results can be obtained even before going into the theoretical basis like differential steering systems, forward and inverse kinematics and motion planning. This paper presents the comprehensive set of on-line tutorials featuring TIAGo, the mobile manipulator of PAL Robotics, which use Gazebo simulator and ROS to take a widespread audience to a trip to discover in a practical way a broad range of areas like control, motion planning, mapping and navigation and 2D/3D perception.

1 Introduction

Dynamic simulations of different robots are very common nowadays. Accurate simulation models provide an easy way for rapid prototyping and validation of robotic tasks. Focusing in ROS based simulations, the abstraction layer provided by frameworks like **ros_control** [1] [2], ensures that what you get in simulation is what you will get in the real robot, at least in terms of interfaces. Furthermore, working in simulation is very important when learning robotics, as programming a real robot may be dangerous for the robot itself, the environment or people around. Simulation then is an essential tool for learning robotics.

PAL Robotics has published a comprehensive set of on-line tutorials based on its mobile manipulator TIAGo [2]. The tutorials are organized in different blocks which are depicted in Figure 1. The first section explains in detail how to get a computer ready to follow the tutorials presented afterwards. The installation instructions for an Ubuntu computer and a ROS distribution are explained along with all the required ROS packages from PAL Robotics in order to have the dynamic TIAGo's simulation in Gazebo up and running.

All the tutorials are based on spawning the model of TIAGo in a given simulated world in order to perform a specific task. The tutorials are structured in a way that provides the basic ROS instructions in order to have the simulation

[1] http://wiki.ros.org/ros_control
[2] http://tiago.pal-robotics.com

running, how to run specific ROS nodes performing the target task and also providing some theoretical background of the robotic task when needed.

The first block of tutorials addresses basic control aspects of the robot. Afterwards tutorials about laser-based mapping, localization and autonomous navigation are presented. The next block teaches different ways to perform motion planning with TIAGo including a pick and place demo. Right after a set of tutorials showing different computer vision applications are included. Then, tutorials about processing point clouds show basic perception tasks based on 3D data obtained with the depth camera of the robot.

2 Control

These tutorials aim to teach how to command the different Degrees of Freedom (DoF) of TIAGo.

2.1 Differential Drive Base

The first two tutorials in this block show how to control the differential drive base of TIAGo in order to have the robot moving in the ground plane. The main teaching of these tutorials is that a differential drive system can perform a Twist [3], i.e. the composition of a linear velocity along the axis of the robot pointing forward and a rotation speed about the center of the robot base.

One of the tutorials instruct on how to send the appropriate message. For example, the message to have TIAGo moving at 0.5 m/s forward and at the same time rotating at 0.2 rad/s about its vertical axis can be seen in the following command line instruction:

```
rostopic pub /mobile_base_controller/cmd_vel \
geometry_msgs/Twist "linear:
  x: 0.5
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.2" -r 3
```

2.2 Upper body joints

The goal of these tutorials is to provide the reader with knowledge about the typical controllers in ROS based robots to control them in joint space and what can be achieved with these basic controllers.

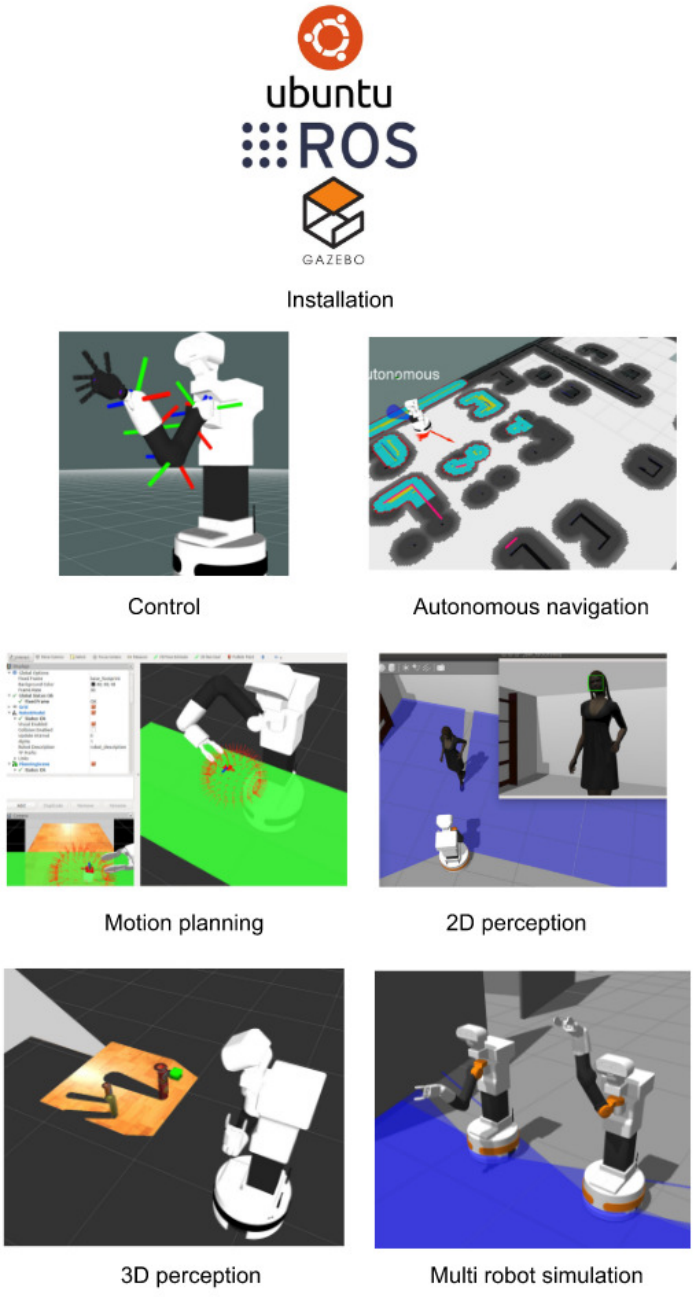


Fig. 1. Overview of the on-line tutorials of TIAGo

Joint Trajectory Controllers Two tutorials are provided so that the reader can have a grasp of the different ROS Joint Trajectory Controllers [3] defined in the upper body of TIAGo each one controlling the following groups of joints:

- **arm_torso**: group composed of the prismatic joint of the torso and the 7 joints of the arm
- **gripper** or **hand**: groups of the 2 joints of the gripper or the 3 actuated joints of the humanoid hand of TIAGo
- **head**: the 2 joints of the head composing the pan-tilt mechanism

The tutorials provide links to the official ROS documentation explaining how a Joint Trajectory Controller is useful to execute joint-space trajectories defined by a set of waypoints so that the trajectory results from interpolation using different strategies. The use of TIAGo's joint trajectory controllers is exemplified with a simple C++ code that the user can easily modify and produce complex motions with the robot in simulation.

This tutorial can be used when teaching robotics in order to quickly validate forward kinematics computations of the arm given a joint space configuration by comparing the pose of the end-effector obtained in simulation. For example, in order to obtain the cartesian coordinates of TIAGo's arm tip frame the following ROS instruction can be used

```
rosrun tf tf_echo /torso_lift_link /arm_1_link
```

which will print the transformation from the arm tip link, i.e. arm_7_link, with respect its parent link, i.e. torso_lift_link, which is presented in the following form

```
- Translation: [0.155, 0.014, -0.151]
- Rotation: in Quaternion [0.000, 0.000, 0.020, 1.000]
            in RPY (radian) [0.000, -0.000, 0.039]
            in RPY (degree) [0.000, -0.000, 2.242]
```

This is also a good point to introduce the ROS Transform Library [4] to the students and explain the two rotation representations provided, i.e. the Quaternion based and the Roll-Pitch-Yaw representation.

Head control This tutorial provides an example of a more sophisticated controller that can be build on top of the joint-space trajectory controller of the 2 joints of the head. This new controller is the Head Action controller [5]. The goal of this controller is to have the robot's head pointing to a given direction, i.e. looking at an specific cartesian point. Therefore, the input of the controller is not in joint-space but in cartesian space. The source code of the controller,

[3] http://wiki.ros.org/joint_trajectory_controller
[4] http://wiki.ros.org/tf2
[5] http://wiki.ros.org/head_action

available in [6], is also a good example on how to implement inverse kinematics using the Kinematics and Dynamics Library (KDL) [7].

The tutorial is based on a simple ROS node implemented in C++ which also shows how to use the actionlib interface of ROS [8]. The C++ example opens a window with the RGB image providing from TIAGo's head camera so that the user can click on any pixel and the robot in simulation then moves the head so that it looks at the given direction, see Figure 2.

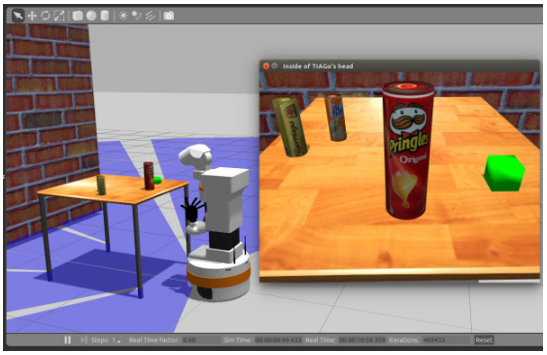


Fig. 2. Head control of TIAGo defining a sight direction

The C++ example also shows how to make use of the intrinsic parameters of the camera in order to compute a cartesian point in the line of sight defined by the pixel clicked by the user.

Playing back pre-defined motions The last tutorial in the Control block explains how to store in a yaml file upper body motions defined in joint space so that they can be played back at any time with the Play Motion library in ROS [9]. The tutorial also explains how to change the velocity of the motions by varying the time given to reach each waypoint of the trajectory.

3 Autonomous Navigation

Two tutorials are presented in order that lead to autonomous navigation of TIAGo in simulation. The first tutorial is devoted to explain how to generate a map with the laser range-finder of the robot. This tutorial can be used as demonstration of Simultaneous Localization and Mapping (SLAM) by using the gmapping algorithm wrapped in ROS [10], see Figure 3.

[6] https://github.com/pal-robotics/head_action
[7] http://www.orocos.org/kdl
[8] http://wiki.ros.org/actionlib
[9] http://wiki.ros.org/play_motion
[10] http://wiki.ros.org/gmapping

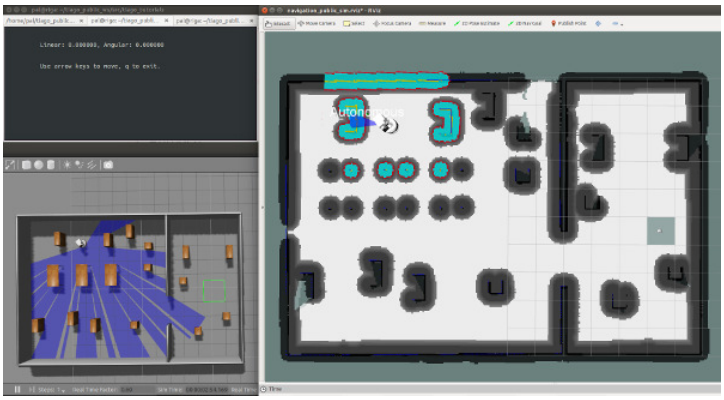


Fig. 3. Laser-based map generated by TIAGo

Once the user has created a map a tutorial showing the autonomous navigation of TIAGo is presented. This tutorial embraces theoretical concepts like Adaptive Monte Carlo Localization [4], particle filtering to track the pose of the robot in the map, the costmap concept, laser scan matching and obstacle avoidance with motion planning.

The navigation tutorial also introduces a more sophisticated approach to obstacle avoidance which consists in using not only the laser scan but also a virtual scan produced from the point cloud of the depth camera of TIAGo's head.

In Figure 4 a global trajectory that TIAGo is trying to follow to reach a destination point in the map is shown in blue. Furhtermore, the costamps appear overlayed in the map defining inflated areas that produce repulsion points taken into account by the local planner that reactively corrects the robot trajectory in order to avoid obstacles detected by the laser and the camera.

4 Motion Planning

The block of Motion Planning focues on MoveIt! [11] [5], which supports several motion planning libraries like OMPL [6] the state of the art software in motion planning for manipulation. The different tutorials are briefly explained hereafter.

4.1 Motion planning in joint space

This is the basic tutorial of motion planning. It shows how to define the kinematic configuration to reach in joint space and how to generate a plan and execute it in MoveIt!. This is a good example on how to obtain collision-free and joint limit avoidance when moving the robot upper body in joint space.

[11] http://moveit.ros.org

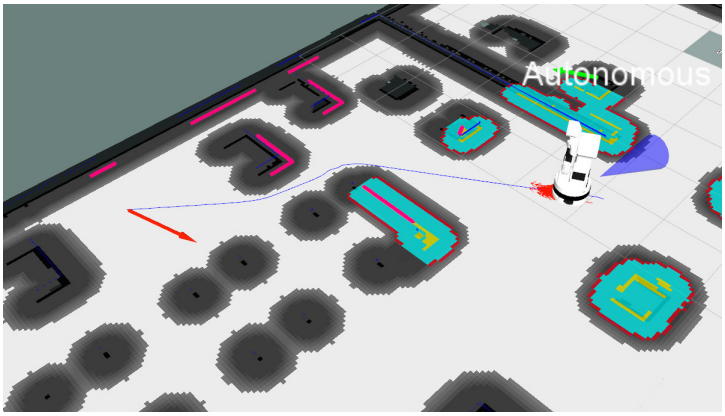


Fig. 4. AMCL-based localization of TIAGo and path planning

4.2 Motion planning in cartesian space

This tutorial allows the user to define the goal kinematic configuration in cartesian space, i.e. by defining the goal pose of the given frame of the robot's upper body. Then, it is an example on how to run inverse kinematics using MoveIt!

4.3 Motion planning taking Octomaps into account

This tutorial is an extension of the previous one in order to add vision-based collision avoidance with the environment. Indeed, it is an example on how MoveIt! can handle a 3D representation of the environment based on Octomap's occupancy grids [7], see 5.

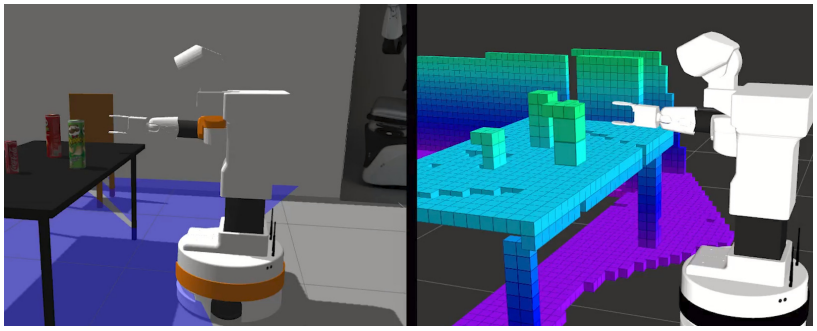


Fig. 5. Motion planning including Octomap occupancy grid

4.4 Demonstration of a pick & place application

The last tutorial of motion planning includes a full pipeline to have TIAGo picking an object from a table, raising it and then place it back to its initial position. The tutorial shows an example on how to use monocular model-based object pose estimation and how to use MoveIt! to perform the pick and place operations by avoiding collisions with the table. A snapshot of the demo is shown in Figure 6.

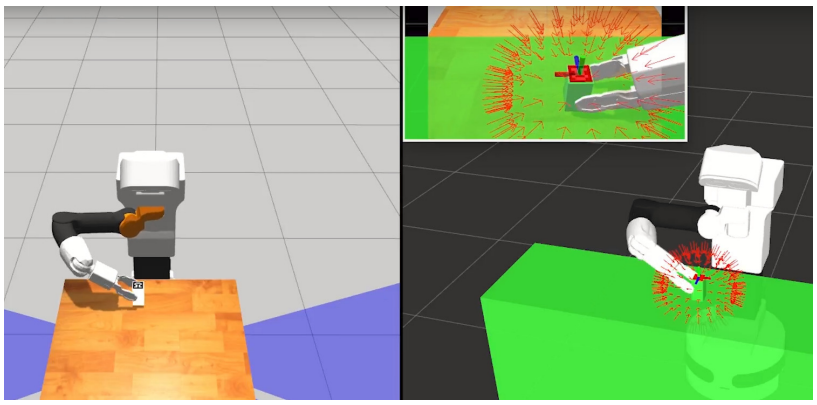


Fig. 6. Pick and place demo using MoveIt!

5 Perception

5.1 2D perception

A collection of tutorials based on OpenCV [8] are included in order to show typical tasks in robotics involving computer vision. These tutorials cover the following areas:

- Color-based tracking
- Keypoint detection and descriptors computation
- Fiducial markers detection and pose estimation
- Person and face detection
- Planar textured object detection based on homobraphy estimation

Figure 7 shows a couple of simulations. On top, the output of the tutorial showing how to perform person detection. On the bottom, the example of textured planar object detection and pose estimation.

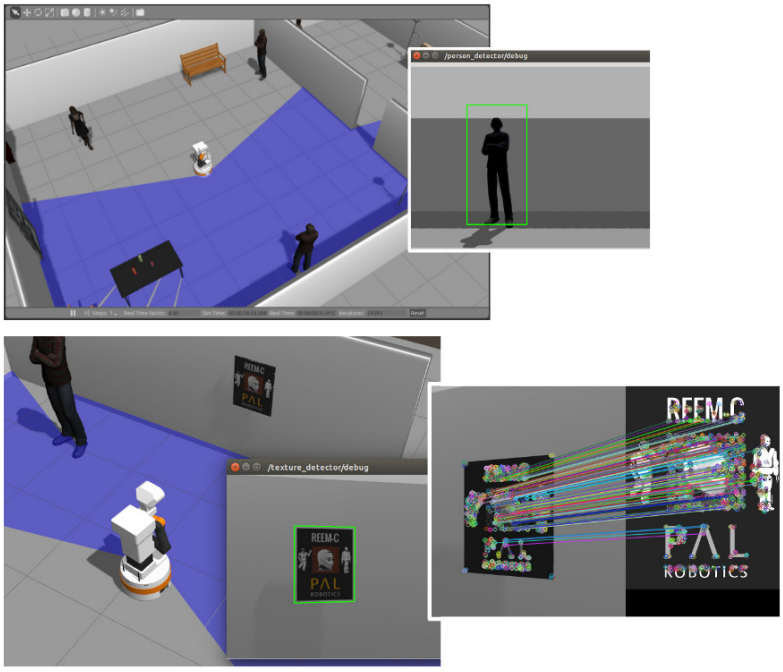


Fig. 7. Some examples of 2D perception in simulation

5.2 Point cloud perception

TIAGo is provided with a RGBD camera placed in its head. Nowadays this kind of low cost depth sensors are of common use in robotics applications. Furthermore, the existance of the Point Cloud Libray [9] provides a useful set of tools to process the data from the depth sensors and obtain semantic data out of it.

A set of tutorials showing point cloud processing are presented. The tutorials aim to show how to detect, segment and estimate the pose of and object on top of a table for robotic manipulation tasks. Figure 8 show the result of the tutorial providing cylindrical object detection and pose estimation.

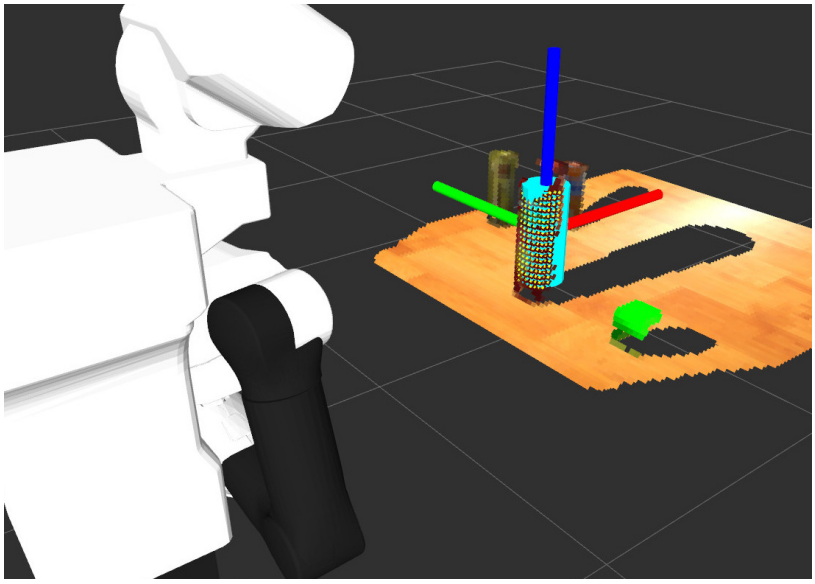


Fig. 8. Table top cylindrical object detection and pose estimation

The tutorials show step by step how to obtain this result starting from the detection of the plane of the table, following with the segmentation of the point clusters on top of such a plane, and finally identifying the clusters that fit on the cylindrical model.

The result of these tutorials can be then combined with the motion planning tutorials in order to implement grasping tasks with integrated perception.

6 Conclusions

This paper presents the on-line tutorials of ROS for the mobile manipulator TIAGo. The paper has shown how this tutorials can be used as practial examples

when teaching fundamental topics in Robotics and how they can be used as demonstrators of the different theoretical approaches involved.

The authors are convinced that these simulation-based tutorials can reduce the learning curve of robotics students as well as being powerful tools for teachers in order to engage easily their students by providing them with a framework where practical results are achieved very fast.

Since the publication of the tutorials, on early October 2016, a number of users, from teachers to PhD/under-graduate students, have adopted them to teach or learning some aspects of robotics. One important feedback provided by some of the users was that the installation part of the tutorials was a bit hard to meet in all existing Operating Systems and/or versions of ROS. In order to partially overcome this limitation the tutorials were also published in the one-line site of The Construct[12], so that the tutorials can be followed just by using a Web browser. The on-line course also offers exercises to practise the different topics taught in the tutorials.

References

1. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software. Volume 3., Kobe (2009) 5
2. Chitta, S., Marder-Eppstein, E., Meeussen, W., Pradeep, V., Rodríguez Tsouroukdissian, A., Bohren, J., Coleman, D., Magyar, B., Raiola, G., Lüdtke, M., Fernández Perdomo, E.: ros_control: A generic and simple control framework for ros. The Journal of Open Source Software (2017)
3. Davidson, J., Hunt, K.: Robots and Screw Theory: Applications of Kinematics and Statics to Robotics. Oxford University Press (2004)
4. Thrun, S., Fox, D., Burgard, W., Dellaert, F.: Monte carlo localization for mobile robots. In: In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA. (1999)
5. Chitta, S., Sucan, I., Cousins, S.: Moveit![ros topics]. IEEE Robotics & Automation Magazine **19**(1) (2012) 18–19
6. Şucan, I.A., Moll, M., Kavraki, L.E.: The Open Motion Planning Library. IEEE Robotics & Automation Magazine **19**(4) (December 2012) 72–82 http://ompl.kavrakilab.org.
7. Wurm, K.M., Hornung, A., Bennewitz, M., Stachniss, C., Burgard, W.: Octomap: A probabilistic, flexible, and compact 3d map representation for robotic systems. In: In Proc. of the ICRA 2010 workshop. (2010)
8. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools (2000)
9. Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl. In: In Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE 1–4

[12] http://www.theconstructsim.com

Teaching ROS efficiently to mixed skill classes
A blended learning approach with gamification elements

Staehle Benjamin and Ertel Wolfgang

Institute for Artificial Intelligence University of Applied Sciences
Ravensburg-Weingarten, Germany staehle@hs-weingarten.de

Abstract. In this work in progress report, we illustrate how to efficiently teach a mixed skill class the foundations of ROS within one semester in a single course. The goal is to equip students with the basic knowledge and tools to join our Robocup@Home team, start a scientific project or thesis. To achieve this in a mixed skill setting we combine blended learning with gamification elements supported by a code versioning system. We believe that this approach is not only the most efficient way to teach this kind of matter but also bridges the gap between academic and industry working concepts.

1 Introduction

Developing Service Robots for over a decade, we experienced how frustrating it can be for students having to deal with multiple barriers before being able to start working with their actual field of interest. Many students came to our lab with big visions but then had to spend months understanding the depending software frameworks and setting up their development environment. We noticed that such tasks can consume a lot of the initial motivation which can even lead to a dropout. This also reflects back on researchers and employees of the university as they have to invest a serious amount of time explaining and assisting novice students. In order to leverage this problem and to increase efficiency inside our institute, we decided to start a practical oriented course to create a balanced foundation for students interested in robotics. This also gives us the opportunity to include students from non-pure computer-science fields such as electrical engineering and mechatronics whose curriculum usually covers merely basic education in complex software development.

1.1 Robotic Frameworks

[HC11] describe the most relevant toolkits and frameworks for robotic development. The diversity in the late 2000s made it difficult to build applications upon other research group's software. In many cases, this led to the development of isolated solutions resulting in redundant applications even among different departments inside the same university. Established frameworks like ROS[1], their

[1] Robot Operating System http://www.ros.org/

vivid communities and broad support of robotic hardware have improved the collaboration and communication among robotic research groups drastically over the past years. Yet the complexity of hardware and software on robot systems remains extraordinary challenging but the foundation laid down by ROS makes entering the world of robotics much easier and therefore interesting for teaching.

1.2 Teaching Concepts

The vision of this work is motivating students to transform their already acquired, but mostly theoretical, knowledge into solutions for real-world problems. Service Robots are an ideal platform for this transition as they incorporate aspects of various study fields and therefore can be used to address a broad audience. Already evolved courses like [Yim+08], [CWC13] and [Cap13] state that the most effective and fun way to acquire this kind of knowledge is a practical approach that involves interaction and experimenting along learning the theoretical concepts. Classic lectures, that separate theory from practical exercises, cannot meet this requirement as each student has to pass an individual point of understanding which cannot be forced from outside. This requires an asynchronous teaching method as proposed in [Kel+06]. Methods like flipped classroom likewise demand that the student can freely choose when to learn and at which speed. As the name indicates the method flips the lecture to the students free time and homework or exercises to the original class session. Still, the method emphasizes a strict separation between teaching and exercising [Chr16]. Blended learning, however, transforms lecture times into hybrid teach and exercise sessions in which students can choose how to approach their current problems while benefiting from group discussions in a workshop-like atmosphere. In this approach, the teacher is no longer the singular source of knowledge but a mentor to whom the students can directly address specific questions. A very important component in this rather loose way of teaching is the strict definition of deadlines and a consequent reaction if they are not respected. Another way to achieve this is the usage of gamification elements such as live exercise rankings during the sessions to trigger a competitive motivation among the students. Also, the course can be separated into major parts that act similar to an achieved goal in a computer game. In this work, we combine the mentioned concepts into a practically oriented course which we describe in the following section.

2 Method

2.1 Code Versioning and Issue Based Working

Being familiar with a code versioning system such as git and platforms that are building functions around it is an essential skill for any software developer these days. Popular platforms for code versioning are github[2], bitbucket[3] or source-

[2] https://github.com
[3] https://bitbucket.org

forge[4]. We still experience that sometimes even high semester students are not familiar with these tools. Our approach is centered around a university-hosted version of gitlab[5] but there are no functions used that an online platform such as github could not easily replace. A core functionality of such platforms aside from supporting the development process are issue trackers. In general, these mechanisms are used for bug reports or feature requests. They can have rich descriptions including code samples or images and have a comment function to discuss specific topics in the scope of the current issue. We use these mechanisms to distribute course materials and exercises. Additionally, it is an effective communication channel to our students as they can ask questions individually and in relation to their current exercise. Each repository also contains an own wiki area which the students can use for personal notes. The built-in CI[6] pipelines, similar to the popular jenkins[7] and travis[8] used in many open-source projects, evaluate the work in progress of the students at each commit and provide instant feedback that is visualized in the web frontend as shown in Fig. 1a.

2.2 Course Overview

Our prototype lecture *Introduction to Autonomous Mobile Robots* is a 5 ECTS[9] course with two lab sessions per week. It is separated into four tiers with a rising degree of difficulty. Also, it is a mixed skill course that masters and bachelor students of different study fields can attend in parallel. The exercises and course materials cover the same topics but masters have more challenging exercises and also have to dig deeper into the theoretical background than bachelors. Each session is started with a 10min wrap up of the current progress stating how many students have already reached which level and how the overhaul class performs. All statistics are anonymized to protect the student's privacy.

Tier1. The entry part of the course aims to balance the basic skills among the participants necessary to work in a robotic environment. This includes an introduction to the Linux shell, git code versioning and Python. For this purpose we use the well-known, free online learning platform codecademy[10] which provides comprehensible tutorials at a beginner level and live code evaluation. In parallel, the students get an introduction to our gitlab platform and their tier1 repository. Here they have to solve additional exercises covering relevant topics from codeacademy and with extra tips and tricks for daily use which are not part of the online-tutorials. Students that claim to have sufficient skills are allowed to skip the codecademy courses and only solve the gitlab exercises but in return have to work more autonomously than other students in this tier.

[4] https://sourceforge.net
[5] https://about.gitlab.com
[6] Continous Integration
[7] https://jenkins.io
[8] https://travis-ci.org/
[9] European Credit Transfer and Accumulation System for students
[10] https://www.codecademy.com

Tier2. The next level consists of plain python exercises and explains the fundamentals of a robotic system such as actuators and sensors. This tier's primary goal is to familiarize students with the gitlab platform and python. All course materials and exercises are organized inside the gitlab platform from this point on. The students have to do basic calculations on a given set of simulated laser data as seen in Fig. 1b and are introduced to creating own tests for the CI engine.

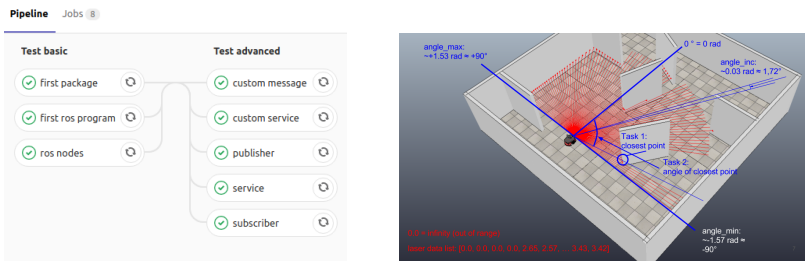
Tier3. In this chapter, the students learn about the most important ROS concepts such as topics, services, actions and helper tools. Beforehand the students are introduced to a virtualized development environment. We provide virtualbox[11], docker[12] and kvm[13] images from which the students can choose based on their personal preference. All exercises can be conducted inside this environment using gazebo[14], a popular simulation framework. In this environment, the students only have to do minor adjustments such as generating ssh keys to start working. The degree of difficulty rises constantly while progressing to encourage the students to discuss the materials rather than just solving the tasks. This tier ends with a mini project where the students have to navigate a turtlebot robot through a gap in a wall. The project is evaluated individually and acts as a midterm exam.

Tier4. The course ends with the so-called maze challenge in which the students get a laser-equipped turtlebot and a training maze which they can use to test their algorithms. Before, we reflect common mistakes that occurred during the tier3 mini project and point out what could be improved. After this, the students get a clean repository which they have to organize and document on their own. The final grading does not only depend on the performance of their robot in the challenge, but also on the quality of code and development process (e.g. using issues, commit messages, tests). The best students of this course are given the opportunity to become a member of the universities RoboCup@Home[15] Team alongside with project and thesis offerings.

2.3 Comparison to Online Platforms

Flipped and blended learning approaches strongly depend on online teaching resources and exercise frameworks. Before starting to run an own infrastructure which has to be set up and maintained it is advisable to check the currently available online platforms. When we started the course 2016 we enrolled our students at the *robotIgniteAcademy*[16] offered by the company *The Construct*.

[11] https://www.virtualbox.org
[12] https://www.docker.com
[13] https://www.linux-kvm.org
[14] http://gazebosim.org
[15] http://www.robocupathome.org
[16] http://www.theconstructsim.com



(a) (b)

Fig. 1: CI result and Python laser exercise

Starting from scratch this is one of the most comfortable ways to set up a class. Besides the great tutorials, the web-based coding and simulation environment is the key benefit of this approach, as students only need a web browser to work. Nevertheless, we decided to run our own setup in the end due to flexibility and cost reduction reasons. Also, we experienced connection issues from time to time and other small problems that tarnished the experience for our students. In Table 1 we compare the two approaches. The results only reflect our personal experience and may depend on external factors, such as connection speed, which the provider cannot influence.

Table 1: Comparison of online platform and self hosted solution

Criteria	**Online platform**	**Self hosted**
Setup effort	++	-
Maintenance effort	++	o
Flexibility (e.g. self defined exercises and challenges)	-	++
Costs	- -	+
Reliability	o	+

++ very good, + good, o neutral, - bad, - - very bad

3 Conclusion and Outlook

At the time of writing this work is still in progress and will be evaluated at the end of 2018. Nevertheless, we can already state that those students which have completed this course during the last semesters were able to achieve a homogeneous skill level in the context of ROS basics regardless of their background. This is also reflected by the composition of our current RoboCup@Home Team where the amount of mechatronic and computer-science students is equally distributed.

Currently, we are investigating evaluation instruments such as [CWC13] to document and compare the learning quality among multiple classes and study fields. Also, we are continuing to improve the instant feedback mechanisms and want to apply the proposed methods in other lectures to continuously enhance the learning quality at our university. Further, we plan to establish follow up classes that are focused on specialized topics of robotics such as manipulation, object recognition or navigation.

4 Acknowledgements

The Authors would like to thank Martin Preussentanz, Jochen Weissenrieder, Joerg Wendorff, Benjamin Kathan, Christopher Bonenberger and Markus Schneider for the inspiring discussions. In addition, we want to thank Sashidhar Reddy Kanuboddi, Simon Bucher and Igor Chernov for supporting the lecture as tutors and coworkers. Special thanks go out to Steffen Pfiffner for helping to run this course in the start phase. This work was conducted within a practically oriented curriculum development program named WILLE[17] founded by the MWK[18] State Department of Baden-Wuerttemberg Germany.

References

[Cap13] D. J. Cappelleri. "A Novel Lab and Project-Based Learning Introductory Robotics Course". In: *IEEE Transactions on Education* 56.1 (2013), pp. 73 –81.

[CWC13] N. Correll, R. Wing, and D. Coleman. "A one-year introductory robotics curriculum for computer science upperclassmen". In: *IEEE Transactions on Education* 56.1 (2013), pp. 54–60. ISSN: 00189359. DOI: 10.1109/TE.2012.2220774.

[HC11] A. Harris and J. M. Conrad. "Survey of popular robotics simulators, frameworks, and toolkits". In: *Conference Proceedings - IEEE SOUTHEASTCON*. IEEE, 2011, pp. 243–249. ISBN: 9781612847399. DOI: 10.1109/SECON.2011.5752942. arXiv: 0412052 [cs]. URL: http://ieeexplore.ieee.org/document/5752942/.

[Kel+06] J. O. Kelly et al. "A non-prescriptive approach to teaching programming". In: *ACM SIGCSE Bulletin* 38.3 (2006), pp. 217–221.

[Yim+08] M. Yim et al. "A practice-integrated undergraduate curriculum in mechanical engineering". In: *ASEE Annual Conference and Exposition, Conference Proceedings* (2008). ISSN: 21535965.

[Chr16] Christopher Pappas. *Blended Learning vs Flipped Learning: Can You Tell The Difference? - eLearning Industry*. 2016. URL: https://elearningindustry.com/blended-learning-vs-flipped-learning-can-tell-difference (visited on 02/19/2018).

[17] Wissenschaft lernen und lehren, trans: learning and teaching science

[18] Ministerium fuer Wissenschaft, Forschung und Kunst, trans: state department for science, research and art

Learning by Doing - Mobile Robotics in the FH Aachen ROS Summer School

Patrick Wiesen, Heiko Engemann, Nicolas Limpert, Stephan Kallweit

University of Applied Sciences Aachen, Institute for Mobile Autonomous Systems and Cognitive Robotics (MASCOR), Aachen, Germany
Technical University of Tshwane, TUT, Pretoria
{wiesen, engemann, limpert, kallweit}@fh-aachen.de

Abstract. Mobile Robotics is one of the highest rated future technologies and a fast growing market. For a modern approach of this topic - mobile robotics - the most accepted framework is the Robot Operating System ROS. Learning ROS is still a challenging topic, so an educational concept for qualifying students is presented. The ROS Summer School is a "hands-on" approach for learning robotics with a high degree of practical aspects. The large number of participants shows the demand for educational programs covering the need for open source technologies in Robotics.

Keywords: ROS, Summer School, robotics education, mobile robotics, autonomous navigation

1 Introduction

Autonomous mobile robotics is a key technology for the upcoming digital revolution. In the future, the manufacturing industry will use mobile robotics for flexible, customized production. In the service sector, mobile robots are useful for logistic tasks, like warehousing, are able to support facility management and are useful for service economy. As a result, a lot of today's jobs will be affected by automated solutions. Current studies show that 47 % of the jobs in the USA, 57 % of the jobs in the OECD and 77 % jobs in China could be affected by this change [1]. One of the major challenges of the society is to shift workers into newly emerging jobs and to prepare students for the upcoming needs of the job market.
It might be an opportunity to achieve this shift by teaching the Robot Operating System (ROS) [2]: one of the most accepted frameworks in the area of mobile robotics. It has become the standard in research and is increasingly introduced into industry. Research scientists worldwide are exchanging their results in form of ROS

modules, called ROS packages. The open source concept is supporting the development and the dissemination of this framework, so the importance of ROS increases constantly. The community, the number of packages, the questions and the wiki pages of ROS are growing exponentially since its release in 2009 [3], which can be proofed by the annual ROS metrics report[1]. The digital revolution leads to a demand in education in robotics and especially in the ROS framework .

1.1 The FH Aachen ROS Summer School

In 2012, the University of Applied Sciences Aachen started to satisfy this demand by organizing the ROS Summer School for an international audience on an annual basis. Since the very beginning, the concept was to learn ROS by using it with real robots. The Aachen ROS Summer School was one of the first of its kind and became quite popular. The number of participants was growing quickly as shown in figure 1 (left). The Aachen ROS Summer School started initially with ten participants and multiplied the number quickly up to around 50 external participants within the last years, coming from between 15 to 25 different countries (s. Fig. 1).

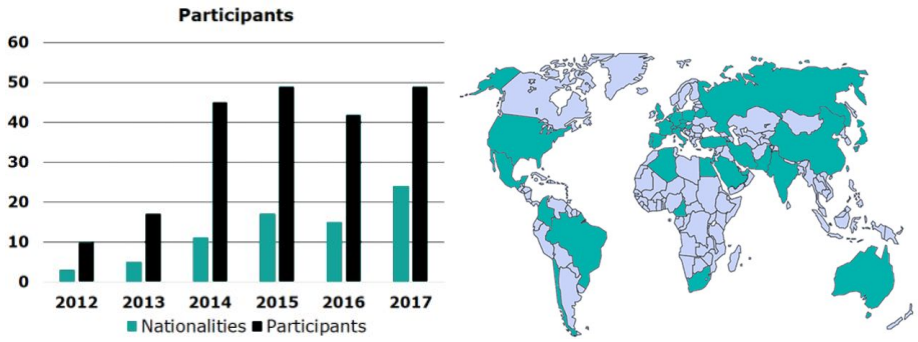


Fig. 1. Left: Number of participants and nationalities in FH Aachen ROS Summer School. Right: World map highlighting the ROS Summer School participants origin.

During the last six years the didactic concept and the used hardware has been continuously improved. The goal is to increase the theoretical knowledge about mobile robotics as well as handling real robots with the framework ROS in a short amount of time.

[1] http://download.ros.org/downloads/metrics/metrics-report-2017-07.pdf

1.2 FH Aachen Rover

A dedicated hardware for the Summer School was developed, in order to generate a suitable educational platform and to allow the participants to easily copy the hardware for their own purpose. The FH Rover is based on a common RC-Crawler chassis and uses a large amount of different hardware components. The hardware is attached to the rover using a customized mount. Figure 2 gives an overview about the different components of the FH Rover. The complete system is powered by just one LiPo battery.

Fig. 2. Hardware components of the FH Rover. Clockwise, starting from the top left: 2D Laser Range Finder SICK Tim 571, Embedded PC Odroid XU4, Gamepad, Infrared Sensor Sharp GP2Y0A21YK0F, LiPo battery Turnigy nano-tech 5800 mAh, Battery beeper, Webcam Logitech C270 and Megapirate AIOP v2 Flight Controller.

The Megapirate AIOP v2 flight controller interfaces the motor controller of the RC car and the additional infrared sensor mounted at the front of the chassis. The flight controller is based on an ATMega 2560 microcontroller. In addition it is equipped with the MPU6050 IMU.
An embedded PC with ARM-architecture is used for high level processing running ROS under Ubuntu. The Odroid XU4 interfaces the 2D laser range finder SICK TIM571, the Logitech C270 webcam, the flight controller and the optional the gamepad.
During the ROS Summer School, the participants will learn how to handle all the different hardware components. The participants go through the complete pipeline starting from the motor controller commands on microcontroller level, up to the high level tasks of path planning. This results in a complete bottom up study of a mobile robot.

2 Intended Learning Outcomes

The main intended learning outcome (ILO) from the ROS Summer School is to achieve core competencies in the field of mobile robotics exemplary on a car like robot. The theoretical concepts are practically applied using ROS and handled within different subjects, which are separated in two parts: Basic ROS concepts and Higher Level Mobile Robotics. The Basic ROS concepts are learned within the practical sessions: Software Hierarchy, Communication, Interfacing Hardware and Development of Robot Applications. During the sessions about Higher Level Mobile Robotics, the participants can use their Rover platform to experience 3D (x, y, θ) robot simulation using Gazebo [4], Image processing with OpenCV [5] and Alvar for Marker Tracking [6], map generation and SLAM using hector_slam [7], navigation using move_base [8] and State Machine development for complex robot behaviours using SMACH [9]. In addition, the cultural exchange and an inspiring atmosphere are key elements for a successful training. Table 1 shows an overview about the covered concepts, ROS packages and ROS tools in form of a time schedule.

Table 1. Concepts, ROS packages and tools covered in the ROS Summer School.

Day	Concept	related ROS packages and tools
1	**Welcome Day** Registration, ROS Show, ROS Introduction	
2	**Basics** Linux Introduction, ROS Filesystem, ROS Nodes	catkin, turtlesim
3	**Communication** Publisher/Subscriber, Services, Parameter Server	rqt_graph, rviz, joy
4	**Hardware** Microcontroller, Transformations, Hardware Interfaces	rosserial, tf2, sick_tim, rqt_tf_tree
5	**Simulation/ Image Processing** Robot model description URDF, Gazebo, AR-Tags	ar_track_alvar, gazebo_ros
6	**Trip to Paris**	
7	**Localization and Mapping**	laser_scanmatcher, amcl, hector_slam, gmapping
8	**Path Planning**	move_base, smach
9	**Industrial Exhibition**	
10	**Exam and challenge preparation**	
11	**Challenge day**	

2 Didactic concept

There are multiple concepts on how to teach ROS, like MOOCs, the ROS Wiki or e.g. books like "A Systematic Approach to Learning Robot Programming with ROS" by Newmann and Wyatt [10]. The didactic concept of the ROS Summer School is based on [11] by A. Ferrein et al. The aim is to build up knowledge in the field of mobile robotics and to bring the participants to an intermediate ROS level in just two weeks.
The didactic concept is based on the SOLO Taxonomy concept [9] by J. Biggs et al., where SOLO means "Structure of the Observed Learning Outcome", which was re-engineered by C. Brabrand et. al in [10] to clarify the five different SOLO levels by classifying them with verbs describing the Learning Outcome of the students.The intended learning outcomes are related to the daily structure of the sessions (s. Fig. 3).

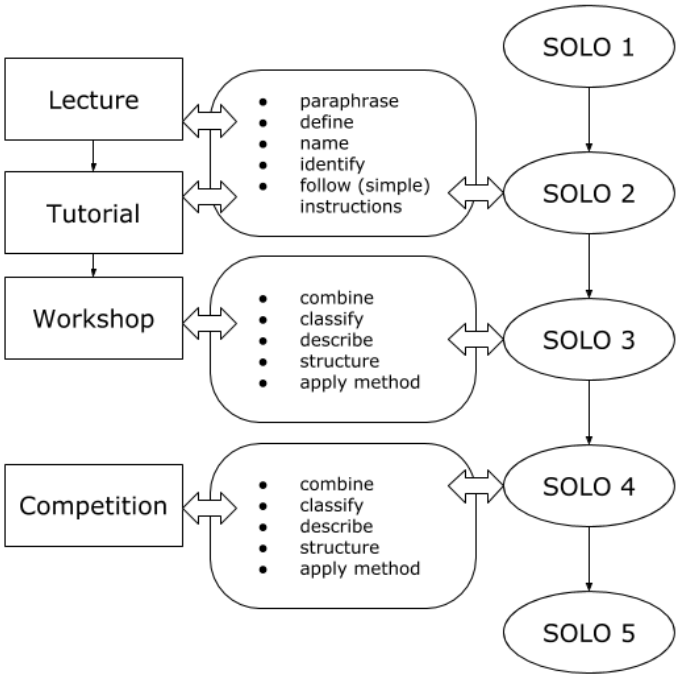


Fig. 3. ROS Summer School SOLO Taxonomy based on [10].

Each day of the Summer School is mainly splitted in three different sessions: Lecture, Tutorial and Workshop. The daily procedure of the ROS Summer School is that every day a new topic is handled. The average participants usually arrive to the ROS Summer School with the pre-structural SOLO level 1. That means they have no detailed idea of the Robot Operating System, because they never used it before. Some participants might have worked already with ROS and possibly reached SOLO 2 or SOLO 3 for a couple of items. For other specific content they are trying to extend

their knowledge. The goal is to bring all students to SOLO 3 after each one day session for each specific topic.
This starts in the morning with a 90 - 120 minutes interactive lecture where a subject in theory, e.g. ROS Communication is introduced. Key terms of this subject will be named and defined, e.g. ROS Master, Topics, Publisher, Subscriber or Services. The key words “define” and “name” are related to the uni-structured SOLO level 2.
After the lecture the participants start with a practical Tutorial, which is done from the very first day using the robots. The Tutorial is a step by step instruction of how to implement the theoretical parts learned during the lecture in practice, using ROS. The participants just have to follow simple instructions. The students can work in their own pace and the Tutorials usually take 120 - 240 minutes depending on the complexity of the daily subject and the pace of the students. “Following simple instructions” is also related to SOLO 2, but in the Tutorials it is a practical implementation instead of theoretical knowledge. An example is e.g. to create a first Publisher and Subscriber for a simple “Hello ROS” node based on a given template.
In the afternoon the participants can start with the Workshop. This Workshops requires to “apply the methods” from the Lecture and the Tutorial. The students have to solve a practical problem on their own, without step by step instructions. They have to “describe”, “structure” and “combine” the learned concepts. The keywords used to describe the Workshop are related to the multi-structural SOLO level 3. Depending on the topic and the students pace, the Workshops usually take 120 - 300 minutes. Based on the example for the Tutorial, the students would be asked in a Workshop to create a Publisher and Subscriber node to be able to teleoperate the robot. In this case they won’t get any template, but have to use the example from the Tutorial. The transfer effort is e.g. to choose the correct Message Type, which is different from the one in the Tutorial.
The single days of the ROS Summer School are structured in such a way, that the content of the next day depends on the content of the day before. To be able to solve the new subject, the students have to reach SOLO 2, but not SOLO 3. Finally, on the last day of the ROS Summer School all participants compete in a challenge. This challenge invites the participants to “analyze”, “compare” and “integrate” all the knowledge they have gathered within the complete Summer School and transfer this knowledge to solve a real world problem. These keywords lead to a point, where participants can reach the relational SOLO level 4 during the challenge preparation and execution.

3 The Challenge

To consolidate the learned aspects, the participants take part in a challenge. The challenge is a typical Find & Rescue scenario. It requires the robot to drive throughout a prepared arena (s. Fig. 4 top) and report position and identification of fiducial markers, called AR tags (s. Fig. 4 bottom left), that are commonly used for augmented reality purposes. The groups can decide to participate in the “real world” league or in the “simulation” league, which are separated to each other, but follow the

same rules. The simulation arena is a 1:1 rebuild in Gazebo from the real world scenario (s. Fig. 4 bottom right).

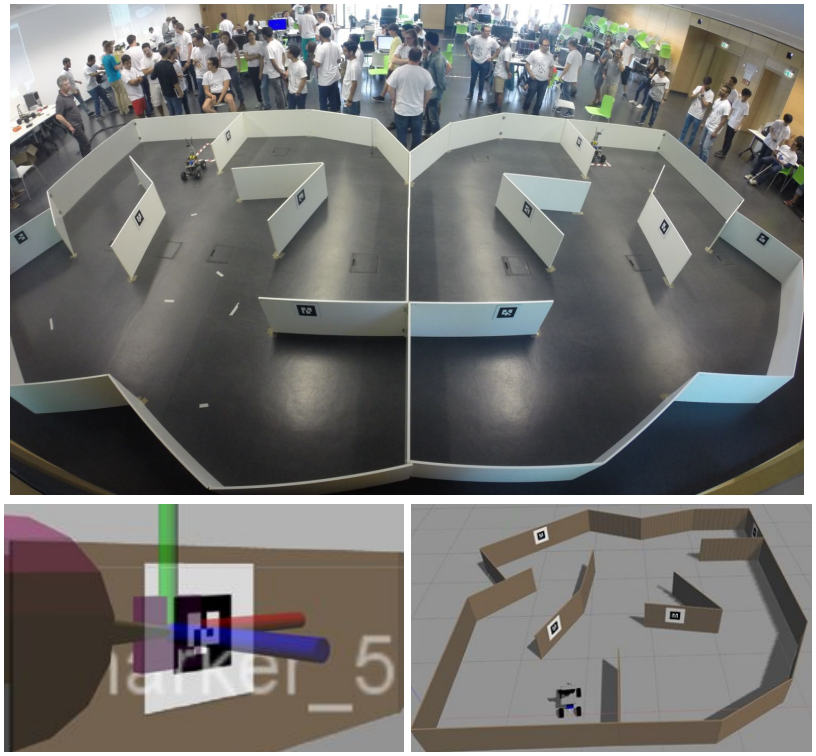


Fig. 4. Top: The challenge arena from ROS Summer School 2017. Bottom right: Fiducial marker - AR tag - detection in simulation. Bottom left: Gazebo simulated challenge arena.

The rules are as follows: each group gets a time frame of 300 seconds to solve the task. The scoring does also start with 300 points. Every used second will subtract one point from the score. It is not allowed to finish before finding all victims, which are represented by the AR tags. Identification of a victim will add additional 10 points, reporting the correct position related to the world frame will add another 50 points. The participants have four options of autonomy level to perform through the arena, where autonomy will be honored by a multiplication factor of the total score:

- Teleoperated x1
- Reactive x2
- Semi autonomous x3
- Fully autonomous x4

In teleoperated mode participants are allowed to manually control the robot to move throughout the course using a game controller. They still have to send the information about the victims before they are allowed to finish.
Reactive mode means, the robot can make use of a combination of distance information, e.g. laser range finder and infrared sensor, to be able to calculate motion directions and avoid collisions.
In the semi autonomous level the participants are allowed to send goal positions manually, e.g. via command line or a GUI like rviz, but the robot has to plan and execute the path autonomously.
The fully autonomous level requires the participants to make use of motion planners allowing to send navigational goals. In addition, they need to perform some task level behaviour. This approach is mainly based on the move_base ROS node and the SMACH task level architecture taught during the Summer School.

The different levels of experience and motivation of the participants represent several approaches that lead to success or failure. The most successful Team in 2017 decided to work with several abstractions. High-level decisions made by their approach took care of a clean definition of a behaviour, which decided to only send navigation goals and react according to detected fiducial markers. Lower components, e.g. fiducial marker reporting or navigation, were only responsible to do their particular task.

5 Worldwide export

In addition to the annual local ROS Summer School in Aachen, there was a demand for more educational ROS activities in other regions.
In March 2016, the University of Applied Sciences Aachen started to export the Summer School concept to a partner university, the Tshwane University of Technology TUT in Pretoria, South Africa. The Hardware was shipped from Aachen to Pretoria and 20 participants from TUT joined the course. The training material was compressed to make it a “one-week” event. This time issue was the reason to remove the fully autonomous approaches including the subjects move_base and SMACH.
In November 2017 another export was generated again at TUT. This time it was decided to use only simulation due to logistical and financial issues. The use of simulation allowed including more content, because the students did not need extra time to work on the hardware.
In between a couple of ROS School and ROS Industrials trainings have been performed based on a similar agenda. All exports can be seen in Table 2.

Fig. 5. Round course arena for ROS Winter School 2018 at MCI in Innsbruck

The latest export took place in February 2018: the ROS Summer School in Innsbruck, Austria. It was hosted at the Management Center Innsbruck (MCI) and part of their Winter School program. The Winter School Program at MCI is another international event, 20 participants from six different countries participated. This ROS School lasted for five days, not including the option of a full autonomous approach during the competition. However, given a similar task compared to the other Summer Schools, the participants managed to move their robots throughout the arena. The main achievement was to move reactively through a round course (s. Fig. 5) and stop in front of a STOP shield represented by an AR tag.

Table 2. FH Aachen ROS Summer School exports

Date	Location	Type	Number of participants
March 2016	TUT, Pretoria, South Africa	ROS School	20
November 2016	Fraunhofer IPA, Stuttgart, Germany	ROS-i Training	15
May 2017	TU Delft, Delft, Netherlands	ROS-i Training	10
July 2017	Fraunhofer IPA, Stuttgart, Germany	ROS-i Training	5
August 2017	Tartu, Estonia	ROS School	20
October 2017	FabLab, Venice, Italy	ROS-i Training	10
November 2017	TUT, Pretoria, South Africa	ROS School	10
February 2018	MCI, Innsbruck, Austria	ROS School	20

6 Evaluation

To identify issues or ways for improvement, the participants are supposed to fill a feedback questionnaire. The outcome from the questionnaire is consolidated in Figure 6. Throughout the years, the ROS Summer School hosted at FH Aachen continuously improved. The feedback from participants throughout the different Summer Schools resulted in several changes according to the participants needs.

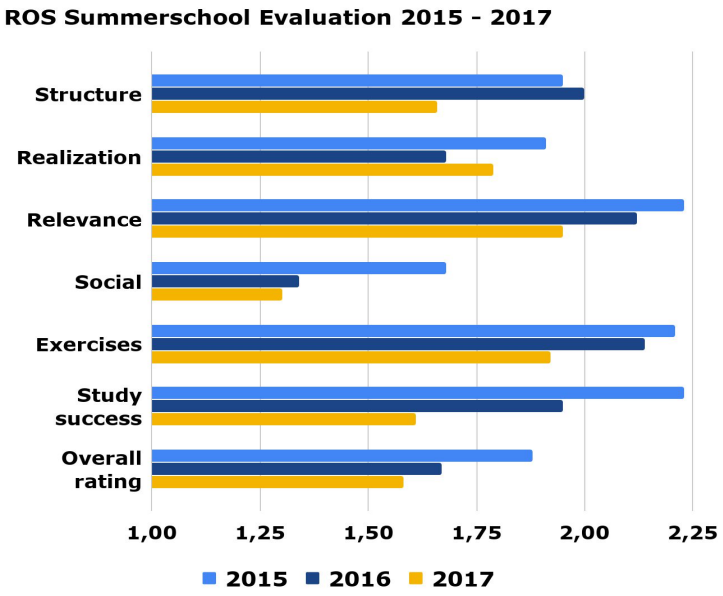


Fig. 6. Evaluation of feedback questionnaire from 2015 - 2017. Average ratings filled by participants are in the range of 1 (very good) to 5 (very bad).

Free text fields with info and comments for further improvement help to get an anonymous perspective from participants. In general, participants seem to prefer the increasing difficulty throughout the Summer School. In particular the approach to consolidate the learned aspects by the end of the Summer School was well received. According to the questionnaire the pace of lectures seems to be well met.

Example comments from the free text fields in 2017 have been:

- "It was very interesting course with application on robot which helps learn and apply simultaneously."
- "The lecture is a must for students interested in robotics."
- "The amount of practical approach in relative to the seminar is much [...] to my liking."

These comments implement that the practical approach to learn ROS is very much accepted by the students and we will intensify and improve the practical sessions.

7 Conclusion

The Aachen ROS Summer School is one of many ways to learn ROS. Nevertheless it is very well established and accepted by the ROS Community, proven by the evaluations of the event. With a regular participation number of 40 - 50 international students (s. Fig. 7), the presented Summer School fulfills the request in ROS education. Since the capacities of the FH Aachen are exhausted with a total of 50 external participants at a time, the international export concept has created another opportunity to grow. In spite of that, the great demand creates enough room in the field of ROS related robotics education for upcoming MOOCs and other ROS learning platforms.

Fig. 7. Group photo ROS Summer School 2017.

The evaluation also gives still some space to improve the ROS Summer School and keep it up to date, which is done every year in the phase before the Summer School starts. That includes: Documentation, Hardware and Software.
In 2017 the evaluation feedback was better than all years before, but the main criticism was the difficulty in mastering the Ackerman kinematics of the RC car. This issue is related to the specific RC car model itself and will be solved for the ROS Summer School 2018 through further improvements.
The later editions of the ROS Summer School are part of the ROSIN project. The ROSIN project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No. 732287.

Bibliography

[1] Carl Benedikt Frey and Michael A. Osborne „The Future of Employment: How Susceptible Are Jobs to Computerisation? | Publications“. Oxford Martin School. Zugegriffen 20. Februar 2018. https://www.oxfordmartin.ox.ac.uk/publications/view/2279.

[2] Quigley, Morgan, Ken Conley, Brian Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, und Andrew Y. Ng. „ROS: an open-source Robot Operating System“. In *ICRA workshop on open source software*, 3:5. Kobe, Japan, 2009.

[3] Boren, J., und S. Cousins. „Exponential Growth of ROS [ROS Topics]“. *IEEE Robotics Automation Magazine* 18, Nr. 1 (März 2011): 19–20. https://doi.org/10.1109/MRA.2010.940147.

[4] Koenig, N., und A. Howard. „Design and use paradigms for Gazebo, an open-source multi-robot simulator“. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, 3:2149–54 Bd.3, 2004. https://doi.org/10.1109/IROS.2004.1389727.

[5] Bradski, Gary, und Adrian Kaehler. „OpenCV“. *Dr. Dobb's journal of software tools* 3 (2000).

[6] Kato, H., und M. Billinghurst. „Marker tracking and HMD calibration for a video-based augmented reality conferencing system“. In *2nd IEEE and ACM International Workshop on Augmented Reality, 1999. (IWAR '99) Proceedings*, 85–94, 1999. https://doi.org/10.1109/IWAR.1999.803809.

[7] Kohlbrecher, S., O. von Stryk, J. Meyer, und U. Klingauf. „A flexible and scalable SLAM system with full 3D motion estimation“. In *2011 IEEE International Symposium on Safety, Security, and Rescue Robotics*, 155–60, 2011. https://doi.org/10.1109/SSRR.2011.6106777.

[8] Marder-Eppstein, E., E. Berger, T. Foote, B. Gerkey, und K. Konolige. „The Office Marathon: Robust navigation in an indoor office environment“. In *2010 IEEE International Conference on Robotics and Automation*, 300–307, 2010. https://doi.org/10.1109/ROBOT.2010.5509725.

[9] Bohren, J., und S. Cousins. „The SMACH High-Level Executive [ROS News]“. *IEEE Robotics Automation Magazine* 17, Nr. 4 (Dezember 2010): 18–20. https://doi.org/10.1109/MRA.2010.938836.

[10] Newman, Wyatt. *A Systematic Approach to Learning Robot Programming with ROS*. CRC Press, 2017.

[11] Ferrein, Alexander, Stephan Kallweit, Ingrid Scholl, und Walter Reichert. „Learning to Program Mobile Robots in the ROS Summer School Series“. Zugegriffen 20. Februar 2018. http://opus.bibliothek.fh-aachen.de/frontdoor/index/index/docId/7418.

[12] John, Biggs, and Tang Catherine. *Teaching For Quality Learning At University*. McGraw-Hill Education (UK), 2011.

[13] Brabrand, Claus, und Bettina Dahl. „Using the SOLO Taxonomy to Analyze Competence Progression of University Science Curricula“. *Higher Education* 58, Nr. 4 (1. Oktober 2009): 531–49. https://doi.org/10.1007/s10734-009-9210-4.

Teaching Robotics with Robot Operating System (ROS): A Behavior Model Perspective

Martin Cooney, Can Yang, Abhilash Padi Siva, Sanjana Arunesh, and Jennifer David

Halmstad University, PO Box 823, Kristian IV:s väg 3, 30118, Sweden
martin.daniel.cooney@gmail.com

Abstract. Robotics skills are in high demand, but learning robotics can be difficult due to the wide range of required knowledge, increasingly complex and diverse platforms, and components requiring dedicated software. One way to mitigate such problems is by utilizing a standard framework such as Robot Operating System (ROS), which facilitates development through the reuse of open-source code—however this also raises a challenge, in that learning curves can be steep for students who are first-time users. In the current paper, we suggest the use of a behavior model to structure the learning of complex frameworks like ROS in an engaging way. A practical example is provided, of integrating ROS into a robotics course called the "Design of Embedded and Intelligent Systems" (DEIS), along with feedback suggesting that some students responded positively to learning experiences enabled by our approach. Furthermore, some course materials, videos, and code have been made available online, which we hope might provide useful insights.

Keywords: Robotics Teaching, ROS, Behavior Model

1 Introduction: ROS-Based teaching of Robotics

The current paper reports on some of our recent experiences with teaching robotics at the postgraduate university level through Robot Operating System (ROS), leveraging a behavior model to encourage learning.

Robotics is a popular subject for which an explosion in applicability and demand is occurring [1]; but, robotics can also be demanding to learn, in encompassing knowledge in mechanics, electronics, statistics, arts, and software [2]. For software, to avoid "reinventing the wheel", robotics practitioners can use ROS, a standard framework which offers support for typical robotics capabilities such as interprocess communication and navigation[1]. One challenge with frameworks such as ROS is usability [3]: in particular steep learning curves facing first time users [4]. In conjunction with the vast amount of material which must be typically covered in learning robotics, this can dissuade teachers from incorporating such frameworks into their courses.

To facilitate the study of robotics using ROS, we suggest considering knowledge from human science, specifically behavior models, as a way to effectively structure

[1] http://www.ros.org/

learning experiences. From this perspective, we report on some of our experiences, positive and negative, in adding ROS to an existing robotics course, and provide some reference materials online (some course materials[2], videos[3], and code[4]), in the hope that they might be useful for other educators.

2 Design: Structuring Course Content via Behavior Models

One important factor which has been identified as facilitating learning of challenging material is engagement [5]. Educators have sought to increase engagement in various ways, such as by seeking to foster active learning of meaningful topics and adequate support systems which allow students to feel a sense of membership [6]. In the current paper, we turned our attention to a behavior model proposed by Fogg, whose usefulness has been described for various applications (e.g., persuading people to use social media) [7].

The Fogg Behavior Model highlights the importance of three aspects in facilitating behaviors (referred to here as requirements R1-3): motivation, ability and triggers. Students should feel motivated to learn (R1). Learning challenges should reflect students' abilities (R2). Furthermore, students require opportunities to engage in learning (R3). To address these requirements we adopted an approach comprising three facets (hereafter referred to as A1-3), as depicted in Fig. 1: demonstrations, classes, and independent project work.

Demonstrations can show what positive possibilities exist, thereby eliciting pleasure or hope, or suggest how negative outcomes can be avoided, e.g., as in so-called "fear appeals" [8] (A1, addressing R1). Lectures and labs can be used to scaffold core learning, by first dedicating sufficient time to considering simplified standardized concepts and tasks, thereby promoting a perception of self-efficacy (A2, addressing R2). Project work promotes autonomy, giving students a chance to make knowledge their own by using it in practice and making decisions (A3, addressing R3).

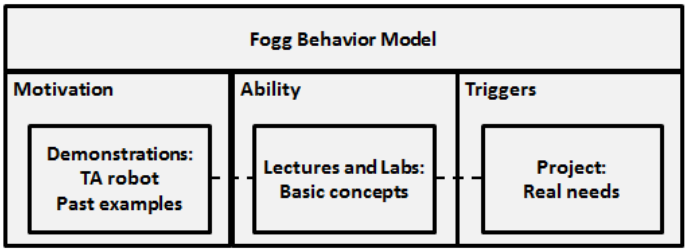


Fig. 1. Applying the Fogg Behavior Model to identify some desired components of a robotics course to facilitate incorporation and learning of complex tools such as robotics frameworks.

[2] http://islab.hh.se/mediawiki/images/7/72/Deis_course_description_2017.zip
[3] https://www.youtube.com/watch?v=B8nvrw7IjCI
[4] https://github.com/martincooney/BaxterDemo/

3 Implementation: The "Design of Embedded and Intelligent Systems" (DEIS) course

Informed by the behavior model, our approach was taken into account in adding ROS to an existing robotics course, "DEIS", which is a half-year intensive (double-credit) compulsory course targeting second year master's students at our university in Sweden. In accordance with the Bologna Process used in most European universities [9], a formal statement of examinable learning outcomes was made available. The core learning outcome is that the students should be able to improve both the breadth and depth of their conceptual and practical robotics knowledge in a collaborative, creative, and critical manner.

3.1 A1: Demonstrations

We conducted some demonstrations seeking to engage students to learn, which had not been performed in the previous year. Before the DEIS course started, students took an introductory course to robotics which did not use ROS. On the last day of this course, it was demonstrated how ROS can be used to avoid some of the challenges the students had faced (e.g., difficulty interfacing programs written in different programming languages).

Additionally, in the first three weeks at the start of the DEIS course, a robotic "teaching assistant" was introduced to show students an example of what positive things can be done with ROS, as shown in Fig. 2. This robot, composed of a Baxter humanoid upper body[5] attached to a Ridgeback mobile base[6], demonstrated abilities such as reading quizzes, speech and face recognition, and handing out materials. Some code for this robot has been uploaded to the internet, and details will be discussed separately [10].

Fig. 2. A robot teaching assistant was used in the DEIS course to demonstrate some robotic tasks which can be accomplished by using ROS, toward engaging students.

[5] http://www.rethinkrobotics.com/
[6] https://www.clearpathrobotics.com/

3.2 A2: Lectures and Labs

Content for the main part of the course supporting students' abilities to learn, the lectures and labs, was mostly retained from the previous year, while adding some material related to ROS. In total, eight teachers covered a wide range of topics, comprising statistical inference, sensors and actuators, sensor fusion, embedded programming, motion planning, simulations, communication, and image processing. In one of the first lectures, we described some of the merits and demerits of using ROS (e.g., the large community and useful tools, vs. the complexity and official support only for Ubuntu); we also defined some typical concepts (e.g., node, package, publisher, subscriber), listed some commands and tools (e.g., catkin_make, rostopic list, roscore; MoveIt!, Gazebo), and provided some "Hello world" examples in C++ and Python. In a follow-up lab, the students were asked to create catkin workspaces and use the ROS talker/listener tutorial to communicate between robots. Time spent for A2 was similar to that for A1. Concepts and tasks were kept simple to allow the students to perceive high ability.

3.3 A3: Project

Basic Concept. Students were also given a problem-solving project to work on in small groups, about platooning robots, which formed the core opportunity for learning robotics with ROS. The project topic, like in the previous year, was generally inspired by the Grand Cooperative Driving Challenge (GCDC), an international contest held between university teams, in which our university placed first in 2016[7]. Furthermore we focused on a scenario of cleaning, which we felt would have practical uses: For example, platoons of snow machines or snow plows are used at some ski resorts and on roads to remove snow. After disasters such as earthquakes, fires, floods, or landslides, teams cooperate to remove debris. Multiple lawn mower robots could remove grass from large open areas such as golf courses or the sides of highways, and vacuum cleaner robot teams could clean large venues such as sports arenas or hotels. Thus, we felt that such a scenario would offer various challenges and engaging opportunities to trigger learning. The main difference with the previous year, aside from the cleaning scenario, was the incorporation of ROS as a required component for the robots and infrastructure. We estimate that, although times were not recorded, the students spent more time on A3 than on A1 and A2, due to the importance of the trigger facet in allowing opportunities to make knowledge their own.

Learning environment. To implement platooning robots, the students worked in five groups, with one robot per group, in a 7.2 x 10.8m project room (80m^2 area) with a set up shown in Fig. 3. The robots ran on top of a 2.5 x 3.7 x 0.8m table with 0.3m walls in the middle of the room. The table was intended to be easy for students to work with robots without having to bend down, to keep robots' wheels from becoming dirty, andto stand at a desired distance (2.5m) from the ceiling to ensure that an overhead

[7] http://www.gcdc.net/en/

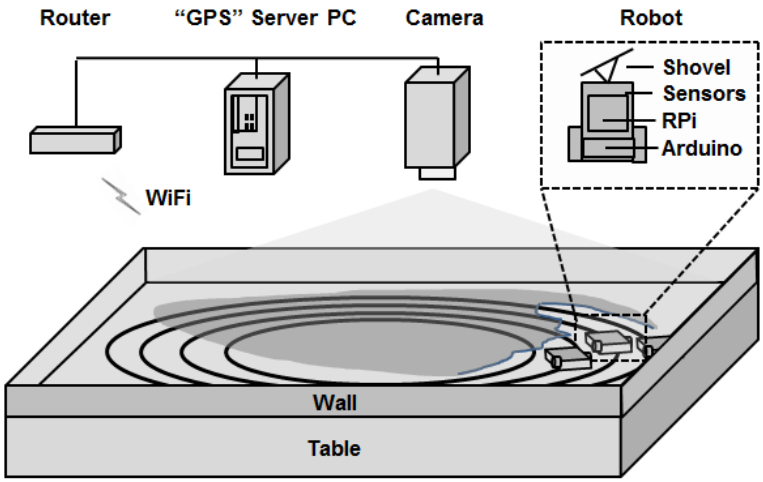


Fig. 3. Project set-up.

network camera (with 3 megapixel resolution at 20 fps, day and night) could capture the entire surface of the table. Concentric elliptical rings were added with black electrical tape to the table-top by the class (students and teachers), as tracks for the robots to run on. The room was well-lit with nine windows, and also featured 13 computers, with monitors equipped with HDMI cables to be able to also work with small computers on the robots; outside the project room was an area with tools such as 3D printers and soldering irons.

Robots. Inside the project room, students assembled and augmented some small differential drive robots from a commercially available kit using an Arduino Uno microcontroller. Sensors included an array of three line following sensors for following tracks, wheel encoders, an accelerometer, and mechanical bumpers; actuators included two 140 rpm gearmotors attached to 65mm rubber wheels, and a buzzer. After assembly, students added some additional components: a small single-board computer with in-built WiFi (Raspberry Pi 3, hereafter RPi, running a Linux operating system, Raspbian Jessie), and an 8-megapixel camera supporting 1080p30. An overhead camera, in conjunction with markers attached to the tops of robots, was also used to detect robot locations and identities.

Thus, the restriction on students was the general project theme and infrastructure, comprising a base platform using a RPi and Arduino and overhead camera. Students were also encouraged to be creative in designing their systems' appearances and capabilities. For example, the students freely selected extra components such as infrared range sensors, sonars, and servo motors to add creative features, and fashioned 3D

printed connectors and shovels for their robots. They could use school desktops, their own computers, or rely entirely on microcomputers and microcontrollers for processing. Students selected and set up power solutions using, for example, lithium polymer batteries and voltage regulators. Each group was also free to develop algorithms for detection; although in general groups started by finding colors and contours, and moved on to a more robust approach of using rotation and scale invariant log spiral markers [11]. During the course, students' choices, and their progress, were continuously monitored via a series of "tollgates", comprising reports on system design choices, presentations by individuals about topics of personal interest (the "research step"), and demonstrations of robot behavior such as lane changing.

ROS. To communicate between robots and with the overhead camera, ROS was used. All robots in the course used ROS Indigo; RPis had the minimal ROS-Comm variant of ROS installed which features basic communication libraries but not GUI tools. Additionally, a server program was also set up for the students on a "GPS" server PC to stream images from the overhead camera; images were used to estimate the x and y coordinate positions of robots, like how automotive navigation systems can use the real Global Positioning System to estimate their positions (in other words, real GPS was not used, but we referred to the system as "GPS" due to its analogous function in allowing localization for navigation).

Within groups, one member was also designated to be a representative who would be responsible for communication and meet with the other groups to decide on a shared protocol. Teachers did not interfere in this process. It would have been possible to define a protocol for the students to use, but the choice was given to the students as a chance to foster creative thinking via problem-solving. This resulted in a set-up with four channels as shown in Table 1, allowing behaviors like in Fig. 4.

Table 1. Communication channels.

No.	Channel	Purpose
1	Heartbeat	Used by each of the robot to indicate its position in coordinates
2	Platoon Position	Used to indicate relative order (the platoon leader was -1)
3	Fan out	Used to move between a single file and diagonal formation (for traveling or cleaning respectively). Messages could also be sent from infrastructure such as an outside computer. Upon receiving a command, the leader sent commands to its followers in the platoon to change lanes.
4	Lane Change	Used when a command is sent to the "fan out channel" to send instructions to each robot, or when obstacles are detected.

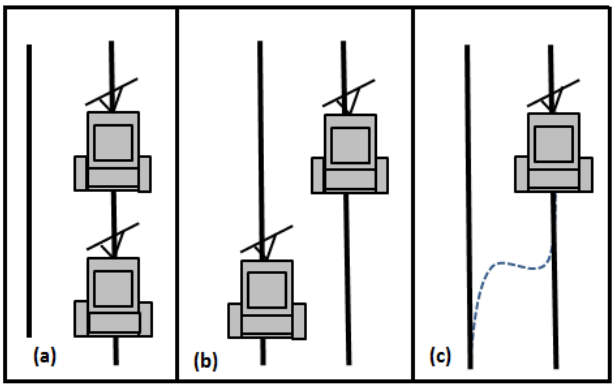


Fig. 4. Platooning examples: (a) straight line formation for traveling, (b) diagonal formation for cleaning, (c) changing lanes

The first two channels support messages which are in the style of Cooperative Awareness Messages (CAM), and the latter two, in describing traffic events, resemble Decentralized Environmental Notification Messages (DENM) [12]. Thus, ROS was used to enable some platooning behaviors in the project robots.

3.4 Examination

Overall, as in the previous year, students were graded 50% based on conceptual knowledge and 50% based on practical knowledge, through an oral exam and written report respectively. Criteria were as follows:

- Grade U (Fail): Basic requirements not met
- Grade 3: The student demonstrated collaboration, to apply basic concepts
- Grade 4: + Creativity
- Grade 5: + Critical thinking/excellent methodology

This year, 24 students (average age = 26.8 years, SD = 4.7, 8 female, 16 male) participated.

4 Experiences and Discussion

We gained some feedback on our course using ROS, in the final demonstration of the course project, through an anonymous survey conducted by our school, and by asking students. We note that in teaching it is generally difficult to conduct rigorous evaluations controlling only a single facet such as the usage of ROS, as there are typically many lessons learned throughout a course and many changes made toward allowing for the best possible learning experiences. Nonetheless we also present some comparison with feedback from the previous year when ROS had not been incorporated, in the estimation that the general trends can be informative.

4.1 Final Demonstration

In this year's course with ROS, students were able to develop additional functionality when compared to the previous year: moving in diagonal formation, avoiding obstacles, and clearing debris. Fig. 5 shows some scenes from the final demonstration of students' work with ROS, and some videos have been made available online[8].

4.2 Survey

Additionally, a survey was conducted by our university, obtaining feedback from 9 participants in regard to the questionnaire items below (using a six point scale, where 0 meant strongly disagree, and 5 meant strongly agree):

- The design of the course (teaching and examinations, etc.) has enabled me to attain the learning outcomes of the course.
- The content of the course (required reading, lectures, etc.) has enabled me to attain the learning outcomes of the course.
- Through the course, I was able to take part in research relevant for the field.
- Through the course, I developed my ability for critical thinking.
- The course encouraged me to actively search for and acquire new knowledge/abilities/skills within the field.

The average result was 4.0 (80%), which was an improvement from the previous year's score when ROS was not used, 3.5 (70%). Students also described some positive and negative experiences. Over half of the respondents described the project using ROS as the most worthwhile element in the course, with one mentioning the robot teaching assistant; this represented an increase from the previous year in which only two students mentioned the project. In terms of improvement, students suggested the course design could be structured to allow further freedom to select topics of interest, and that platooning had been difficult because it was hard to find times to share robots with other groups, among other comments (e.g., that the course room could be larger and that better hardware could be helpful).

4.3 Additional feedback

The survey yielded some useful information but did not specifically relate to ROS; therefore, the students were also asked for feedback about any problems they had experienced with ROS. Familiarization with basic concepts and installations were described as time-consuming, such as installation of the cv_bridge package on the RPis, or various versions of OpenCV. Delays were also reported as a problem. One example referred to synchronization with Matlab for image processing while using many nodes. Another example reported not knowing how to select a preferred protocol for messages: e.g., assuming latency could be more important than reliability for heartbeat messages, UDPROS could have been used instead of TCPROS. As well,

[8] https://www.youtube.com/watch?v=B8nvrw7IjCI

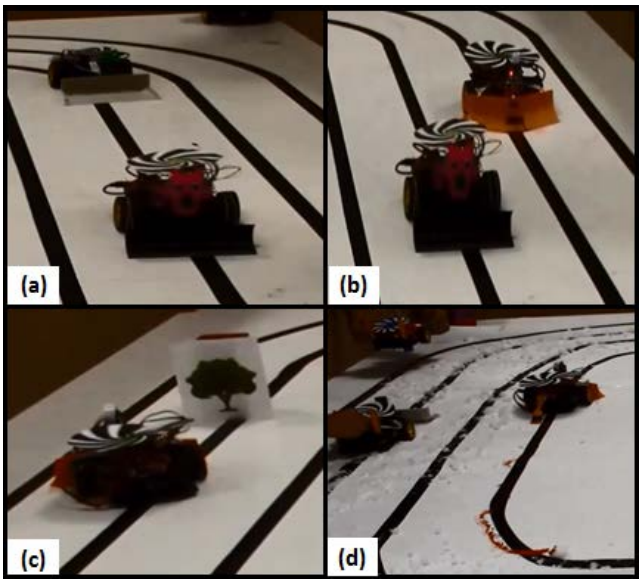


Fig. 5. Results of ROS-based project work: (a) Platooning in a straight line formation to reach a destination, (b) Diagonal formation to clean, (c) Swerving out of the way of a detected obstacle (here a "tree"), and (d) a final task in which robots had to deal with some artificial "snow" (movements were visualized by attaching a paint brush to the backs of the robots).

some tasks like line-following did not involve ROS, so some students, especially the communication "representatives" in each group, received more time to work with ROS than others. Despite such considerations, a number of students also voiced positive comments about their experiences.

4.4 Conclusions

In summary, we observed that incorporating ROS into an existing robotics course, guided by considering a behavior model, appeared to have allowed students to develop more capabilities with their robots, and feedback from students was more positive than in the previous year. As a result, we have decided to continue to use ROS in our course. Next year, we will consider how to further incorporate ROS functionalities, such as *rosserial* for the robots' microcontrollers or *bag* files for sensor data, and also to allow more freedom for students to explore topics on their own. We also plan to take into account lessons learned in the current year; for example, each group will receive two robots instead of one, which will let more students get hands-on with ROS, toward achieving more effective learning.

We expect that effective learning of ROS in university courses, also leveraging behavior models and knowledge of how to engage students, will contribute to robotics

in both academia and industry, as students bring their knowledge and engagement with them to new endeavors.

5 Acknowledgements

We would like to thank everyone who helped, including the students of the DEIS course! The authors received funding from the Swedish Knowledge Foundation (Sidus AIR no. 20140220 and CAISR 2010/0271).

References

1. Pratt, G. A.: Is a Cambrian Explosion Coming for Robotics? Journal of Economic Perspectives, 29(3): 51-60 (2015). DOI: 10.1257/jep.29.3.51
2. King, K. P., Gura, M. (Eds.): Classroom robotics: Case stories of 21st century instruction for millennial students. IAP, 117 (2007).
3. Toris, R., Kammerl, J., Lu, D. V., Lee, J., Jenkins, O. C., Osentoski, S., et al.: Robot web tools: Efficient messaging for cloud robotics. In: Intelligent Robots and Systems (IROS), 2015 IEEE/RSJ International Conference on, pp. 4530-4537. IEEE (2015).
4. Coleman, D., Sucan, I., Chitta, S., Correll, N.: Reducing the barrier to entry of complex robotic software: a MoveIt! case study. arXiv preprint arXiv:1404.3785 (2014).
5. Hake, R. R.: Interactive-engagement versus traditional methods: A six-thousand-student survey of mechanics test data for introductory physics courses. Am. J. Phys. Vol. 66, 64-74 (1997).
6. Marks, H. M.: Student engagement in instructional activity: Patterns in the elementary, middle, and high school years. American educational research journal, 37(1), 153-184 (2000).
7. Fogg, B. J.: A Behavior Model for Persuasive Design. In: Proceedings of the 4th International Conference on Persuasive Technology. Persuasive '09. New York, NY, US: ACM: pp. 40:1–40:7 (2009). doi:10.1145/1541948.1541999. ISBN 9781605583761.
8. Maddux, J. E., Rogers, R. W.: Protection motivation and self-efficacy: A revised theory of fear appeals and attitude change. Journal of experimental social psychology, 19(5), 469-479 (1983).
9. Wächter, B.: The Bologna Process: developments and prospects. European journal of education, 39(3), 265-273 (2004).
10. Cooney M., Leister W.: Designing an Engaging Robotic Teaching Assistant Using the Engagement Profile. (2018) (working paper).
11. Karlsson S., Bigun J.: Synthesis and detection of log-spiral codes. In: SSBA Symposium i bildanalys, pp. 4 (2011).
12. Santa J., Pereniguez F., Moragón A., Skarmeta A. F.: Vehicle-to-infrastructure messaging proposal based on CAM/DENM specifications. In: Wireless Days (WD), 2013 IFIP pp. 1-7. IEEE (2013).