Industrial Intralogistics – Developing an Autonomous Mobile Robot Learning Platform using the ROS2 Framework

Christian Schuster, Benjamin Massow (supervisor)

***Abstract*—This project aims to create a learning platform where students and engineers can learn to develop software using the Robot Operating System 2 (ROS2) to program Autonomous Mobile Robots (AMRs) in industrial intralogistics. Therefore, a course is designed to provide students and professionals with a solid foundation for using the ROS2 framework in mobile robotic applications. To ensure that learning is as hands-on as possible, an intralogistics use case is developed. The use case utilizes multiple TurtleBot4 (TB4) robots transporting parcels within an intralogistics setting. The learning platform is set up on a GitHub website to make the course materials available to the participants and to promote independent learning. To implement the use case, dummy machines are developed to replicate an intralogistics workflow. The goods are transported by the fleet of robots from the warehouse to the machines and finally to the shipping department. To be able to install the use case in a university setting, special attention is paid to university infrastructures. Therefore, dummy machines are developed that do not require a power supply. The packages are transported inside the dummy machine using an inclined plane. As the machines should be able to handle more than one package at a time, a mechanism to separate the packages is developed. The height difference caused by the inclined plane is overcome by constructing an elevator on the robots. In the use case, the TB4 educational robots are used as the AMRs to autonomously transport the goods. Based on the use case, learning modules are developed containing software for the single agents as well as Fleet Management software. In this study, the learning modules for the single agents are prepared for teaching. Thus, the implementation includes basic Nodes to explain ROS2 concepts, a vision-based docking algorithm, an undocking Action, and a Node to operate the elevator. These Nodes are prepared to be used by the Fleet Management later on in the course.**

***Index Terms*—ROS2, ROS2 Course, Industrial Intralogistics, AMR, Turtlebot4.**

C. Schuster is with the Department of Mechatronics, MCI, Innsbruck, Austria.

I. Introduction

AUTONOMOUS intralogistics transport of goods is getting more and more popular in companies. While warehouses are already automated to a large extent, transportation within companies is often still done manually. The increasing product individualization that companies are facing leads to strong demands on a highly flexible and agile warehouse and logistics. These needs open the door for mobile robots in intralogistics and can be accommodated by using Autonomous Mobile Robots (AMRs). After all, autonomous robots are capable of fulfilling these tasks and making the intralogistics ready for upcoming challenges. They cannot only be used in stocks and warehouses but also in other industries like hospitality, agriculture, and the health sector.

However, the usage and integration of AMRs within these different environments is a demanding and rather complex task. As AMRs are capable of driving fully autonomously through the respective environment they need multiple sensors. These sensors are needed for mapping the surroundings and navigating the AMR autonomously while avoiding obstacles. In addition, higher-level Fleet Management software is needed to organize the fleet and distribute tasks to the robots. In order to keep track of these complex systems, well-educated engineers are needed to meet the requirements of the industry. As the demand for those systems increases in all mentioned sectors, there is a need to train more engineers.

The Robot Operating System 1 (ROS1) framework is already widely used in robotics, especially mobile robotics, to program these robots. Furthermore, ROS1 is established as the de facto standard in robotics Research & Development [1]. To meet further requirements from the industry a new version, Robot Operating System 2 (ROS2) is released by Open Robotics and companies are just starting to migrate to this version. Due to the short time since ROS2 has been released, there are still few courses and resources available, making it difficult for future robotics engineers to get familiar with the ROS2 framework. Due to the mentioned needs in the industry, this study aims to develop an academic ROS2 robotics education platform with a focus on intralogistics. This is accomplished by using an intralogistics use case to demonstrate the ideas in a real-world setting.

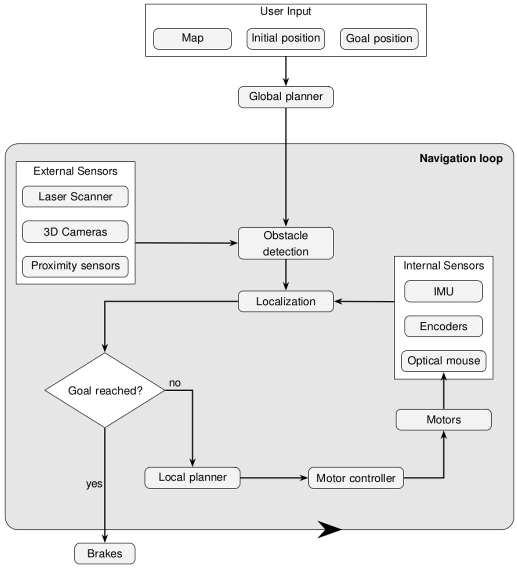


Fig. 1: Flow chart of the navigation in an AMR adapted from [3].

II. Methods

A. Related Work

AMRs form a paradigm shift in terms of navigation, as the mobile robots move autonomously through facilities. Therefore, no reconstruction work is required in the operating area, which can significantly reduce downtime in warehouses, assembly lines, etc. In recent years, AMR applications have evolved from industrial settings to more dynamic scenarios like surveillance, emergency response, and residential applications [2]. The navigation of an AMR is a complex task that needs several sophisticated algorithms and sensors in the background. A typical flow chart of the components used to navigate an AMR is shown in Fig. 2.

In order to start the navigation of AMRs, a map of the environment is needed, which is generated by driving through the facility with the robot once using the Simultaneous Localization and Mapping (SLAM) algorithm. Within this map, an initial position for the robot to localize itself is required. The map and the initial position, together with a goal position, are the boundary conditions for running the navigation. Therefore, the last input from the user is the goal position in the map containing the position and the heading of the robot. Then, the global planner suggests a path based on the initial map. If an obstacle arises in the area that is not recorded in the virtual map, the global planner is planning the path despite this obstacle. Therefore, the navigation loop containing a local planner is entered. The local path planner is capable of rerouting the path based on the detected obstacles and is responsible for bypassing the obstacles detected by the external sensors [4].

As these requirements place high demands on sophisticated algorithms such as localization, sensor fusion and path planning, ROS2 framework is used. ROS2 is offering multiple packages for actuators, sensors, navigation algorithms, etc., made available by community support. These packages can easily be integrated, which enables developers to solely program the custom overlay. Thus, the development time for new robots is reduced massively. The next generation of the widely adopted ROS1 framework also offers numerous advancements and improvements over its predecessor. The new communication middleware Data Distribution Service (DDS) used by ROS2, leads to better security, supports real-time computing and embedded systems, facilitates communication among multiple robots, and enables operations in challenging networking conditions. Moreover, the compatibility is enhanced in ROS2 as it supports not solely Linux but also Windows and macOS.

To teach ROS2 the concept of Project-Based Learning (PjBL) is used, as this approach is widely adopted in the engineering field [5][6]. The idea of PjBL is to teach students the concepts to be learned with the help of a project. Many universities place great emphasis on the development of students' research skills, with the disadvantage that practical relevance is pushed into the background. This can lead to a lack of transferable knowledge and professional skills. These skills can be improved by using the PjBL approach [7]. PjBL is a very effective way to acquire lasting, contextual knowledge, particularly, for computer science students [8]. This can also be assumed for the course to be developed since the course is conducted with ROS2 and is mainly based on programming. Furthermore, this method especially leverages learning in multidisciplinary areas such as robotics [5][6].

B. Development of the Use Case and a Corresponding Learning Platform

First, before starting to design the course with the use case, the requirements have to be set, this can be seen in Tab. I.

TABLE I: Requirements for the use case.

Requirement
Containing the main ROS2 concepts
Practical relevance
Using real hardware
Motivating participants
Suitable in a University environment
Using minimal requirements for electrical infrastructure
No reconstruction work should be necessary
Lightweight design for easy rearranging
Space requirement below $50\,\mathrm{m}^2$
Low cost

The use case is developed based on the requirements and the topics defined in Tab. II. As the use case should contain built hardware and be related to intralogistics, stock, shipping, and machines to process goods are needed. The idea is that the robots pick up the goods from stock and deliver them to machines to be processed, depicted in Fig. 2. These could be only one processing step, e.g., milling the unmachined part, or several processing steps such as milling, polishing, and packaging. This is done to achieve different workflows and complicate robot movements to be more practical. The cycle is finished by delivering the product to the shipment. As this setup is used within the university, the machines are not processing anything but are there for the sake of implementing the process flow. To simplify the development of the hardware the same dummy machines should be used for all components: the machines themselves, stock, and shipment.

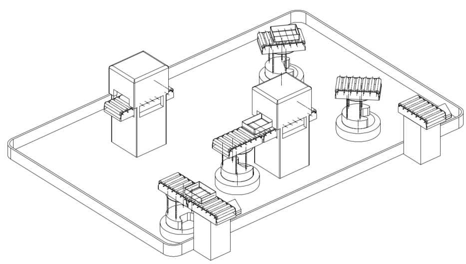


Fig. 2: First draft of the intralogistics use case.

TABLE II: Topics to be taught in the course.

Main Topics	Sub Topics
ROS2 in General	Get familiar with Ubuntu
	Installation of ROS2
	ROS2 Concepts
	Establish a workspace
	Creating Packages
	Creating nested messages
	Parameters
Publisher/ Subscriber Nodes	Publisher/Subscriber message flow
	ROS2 messages
	Publishing to Nodes from CLI
	Subscribing to Nodes from CLI
	Publishing to a Node from another Node
	Subscribing to a Node from another Node
	Creating a custom ROS2 message
	Learning about QoS
Service Nodes	Message flow of Services
	ROS2 Service messages
	Calling a Service Node from CLI
	Calling a Service Node from a Node
	Writing a Service Server
	Creating a custom ROS2 Service message
Action Nodes	Action message flow
	ROS2 Action messages
	Sending goals to Actions from CLI
	Sending goals to Actions from a Node
	Creating a custom ROS2 Action message
Launch files	Creating Launch directory
	Creating Launch file
	Launching Parameters
Multiple Robots	Concepts
	Namespace Nodes
	Communicating with a Robot fleet
ROS2 deep dive	Setting up complex custom messages
	Complex interaction between different ROS2 Concepts

Furthermore, a suitable AMR is needed to enable the use case to work. Therefore, the TurtleBot4 (TB4) Standard from Clearpath Robotics (Fig. 3) is chosen and is explicitly advertised as a training robot. Furthermore, the TurtleBot robots are already well-known in the academic field due to their predecessors. The TB4 is the newest generation of the most used open-source robotics platform for education and research [9]. Another big advantage is the large community developing these robots. Thus, questions can be answered quickly via GitHub.

Fig. 3: TB4 Standard from Clearpath Robotics [9].

The robot is built based on the mobile robot development platform Create3 Educational Robot from iRobot [10]. The TB4, as well as the Create3 base platform, are fully ROS2 based. Furthermore, the TB4 is still affordable for educational institutions with a price of around €2,500. In order for the robot to become an AMR, additional components are mounted to the Create3. The TB4 is equipped with an onboard computer to process the data stream of the add-on sensors. Furthermore, this computer runs on Ubuntu Server with ROS2 to distribute the sensor data into the ROS2 space. The additional hardware of the TB4 can be seen in Tab. III.

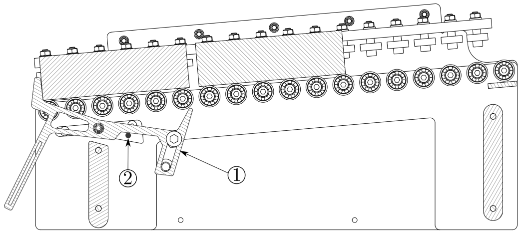
TABLE III: Hardware extensions of the TB4.

Qt.	Hardware and Sensors
1	Raspberry Pi 4B 4GB (onboard computer)
1	LiDAR Sensor (RPLIDAR A1M8)
1	Camera (Oak-D Pro)

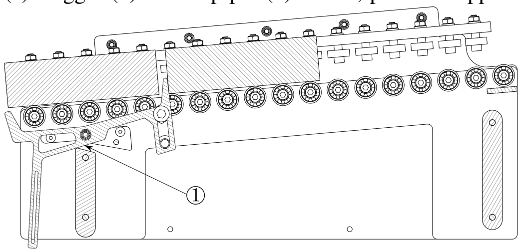
Based on this AMR, the dummy machines are developed. To make the assembly and the manufacturing as easy as possible, the dummy machines are made of acrylic glass, which is cut with a laser cutter. To ensure that the use case is as independent as possible from the environment, the dummy machines are built passively without the need for electricity. Therefore, the dummy machine is built with the principle of an inclined plane. The inclination of the roller track is chosen with five degrees. The transportation of the parcels in the machine is realized with a roller track. Furthermore, the machine is built with a running-in to ensure clearance when loading and unloading. The roller track as well as the running-in are constructed with open ball bearings to ensure low starting resistance. The running-in supporting plate is designed with a putting-in mechanism and is secured with elbow connectors. The machine is built without the necessity of threading. Thus, the parts do not need any post-processing after cutting. Furthermore, a separation mechanism is required to ensure that only one package at a time leaves the machine. In Fig. 4, the separation mechanism constructed with a lever is shown. The resting position of the lever is depicted in Fig. 4a. The pin (Nr. 2 in Fig. 4a) prevents the lever from turning too far. In Fig. 4b, the trigger is activated by the robot and releases the first box while preventing the second box from moving. To enable this principle to work, the boxes have to be separated by a gap. Therefore, stoppers are fixed on the front and back sides of the boxes. Two screws are mounted on the back of the lever in an elongated hole. This increases the weight in the back to ensure that the lever goes back to the resting position after triggering.

As the inclined plane of the dummy machine leads to a height difference between the loading and unloading positions, an elevator must be developed to overcome this. Just like the dummy machine, the elevator is also made of acrylic glass. However, the elevator needs a tray on top to carry one box. This is achieved with a similar concept as for the dummy machine with a smaller roller conveyor. The roller conveyor as well as the running-in is constructed with open ball bearings, the same as for the dummy machine. To unload the parcel from the robot, the tray is supported off-center by a clevis joint and ball bearings. This enables the tray to be tilted to unload the parcel from the robot. The eccentric supporting leads to a resting position of -10°. The elevator is driven by a NEMA17 stepper motor. The idea behind unloading the parcel is to use the linear movement of the elevator to tilt the roller track. To achieve this, two levers are mounted on the roller track (Nr. 3 in Fig. 5).

When the elevator is lifting, these levers should come in contact with a plate at the end of the linear movement and thus tilt the roller conveyor and unload the package. As the height of the elevator on the robot is higher than the dummy machine, the dummy machine has to be heightened. This is achieved by mounting a rack, as depicted in Fig. 5. The loading and unloading mechanism is shown with the rack to raise the dummy machine. The robot needs the hang-



(a) Trigger (1) and stop pin (2) to rest, parcels stopped.



(b) Trigger (1) activated, one parcel released.

Fig. 4: Cross-section views of the dummy machine showing the separation mechanism with the trigger active and at rest.

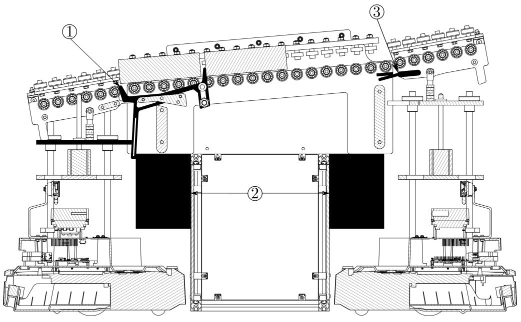


Fig. 5: Cross-section view of the loading and unloading of parcels with the separation mechanism (1), the tilting mechanism (3), and the hang-on parts (2) for LiDAR detection.

on components shown in Fig. 5 in order to detect the dummy machines at their actual size. To be properly recognized, these hang-on components must be at the Light Detection and Ranging (LiDAR) sensor's height.

TABLE IV: Table of contents of the learning modules for the ROS2 intralogistics course.

Topic
1. Home
2. Project Introduction
3. Getting started
4. What is ROS2?
5. Lighting up the TurtleBot (Publisher/Subscriber)
6. Measuring distances with the TurtleBot (Service)
7. Estimate a Pose with the Camera (Service)
8. Let's undock the TurtleBot (Action)
9. Let's dock the TurtleBot (Action)
10. Starting up multiple Nodes (Launch files)
11. Let's drive the elevator (additional, Action)
12. How to use multiple robots?
13. Creating a Fleet Management Architecture
14. SLAM and the NAV2 Toolbox
15. Set up the communication between the Fleet and the Server
16. Updating of the Fleet states (fleet updater)
17. Searching the best-suited Agent for a task (task allocator)
18. Instruct the best-suited Agent (task dispatcher)

C. Development of the Learning Modules

The software development requirements are defined based on the guiding idea of the hardware implementation. Therefore, the following software packages need to be developed within the course:

1) Docking the AMR
2) Undocking the AMR
3) Driving the Elevator
4) Fleet Management

The docking procedure must be developed to enable the robot to accurately approach the dummy machines as well as stock and shipment. Hence, an undock procedure has to be implemented to safely undock the robot without unexpected rotations. Furthermore, software to drive the elevator must be developed, to unload the parcels from the robot. Therewith the required software for the single client is completed. To finish the implementation of the use case, the Fleet Management software has to be developed which organizes the movement of the single agents. Therefore, a table of contents for the course is developed which can be seen in Tab. IV. The course material is made available using GitHub Pages. The GitHub Page is implemented using Material for MkDocs [11]. The biggest advantages of a webpage compared to classical course material are the embedding of videos, as well as interactive learning with coding exercises, promoting self-paced learning. Furthermore, a major facilitation for the maintainer and the learners is to easily unroll updates for all users. In the following the two main modules designed for the single agents are shown, namely the pose estimation and the docking procedure. The modules for the Fleet Management are developed in [12]. As the docking procedure of the robot is crucial for the use case to work, it is explained in more detail. The docking procedure is built on two modules, the pose estimation which estimates poses by image processing of an ArUco marker (Fig. 6) and the docking based on the estimations.

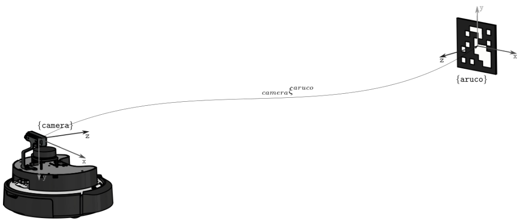


Fig. 6: Pose estimation of the ArUco marker.

OpenCV provides packages for the detection of the markers as well as the pose estimation [13]. The ArUco markers are used because of their robustness to be detected and the fast recognition of the markers in an image [14]. Furthermore, they can be easily fixed to the dummy machines. The pose estimation is developed by utilizing a ROS2 Service Node. The program flow is shown in Fig. 7. Since the ArUco markers can be detected with a unique `ID`, it can be checked if the pose estimation is determined for the correct marker.

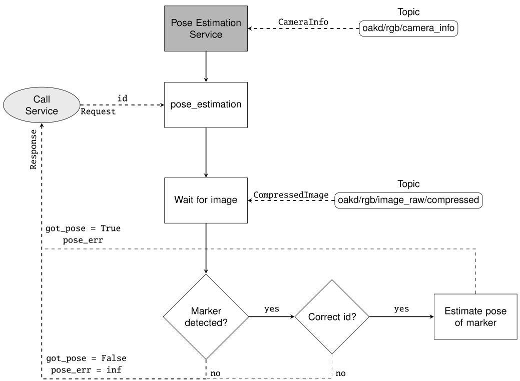


Fig. 7: Finite-state machine of the pose estimation Service.

To make the docking of the robot work, a ROS2 Action Node is developed, as shown in Fig. 8. The pose estimation Service is called in a loop to get the current translational errors in the plane and the heading error of the robot. The docking is executed in three main steps. First, the rough docking uses a combination of driving Actions to roughly align the robot in front of the marker. Second, the controller docking uses a PID controller to calculate the rotational velocities based on the pose errors while driving toward the dock with a constant velocity. The last step is the fine docking, which is used for the last few centimeters driving straight to safely trigger the mechanism of the dummy machines. The developed Action Node can be easily called from the Fleet Management software to dock the agents.

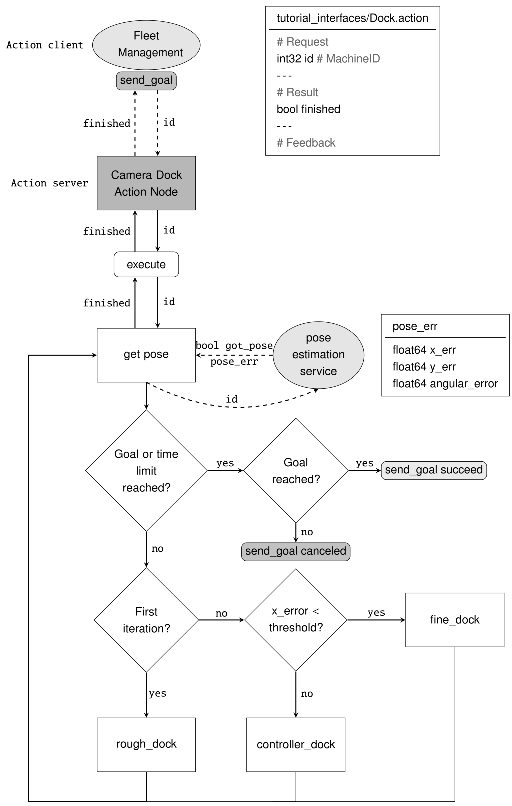


Fig. 8: Finite-state machine of the docking procedure toward the ArUCo marker.

III. Results

All topics that are to be communicated, as elaborated in Tab. II, can be taught by the developed use case. Some topics are covered more often as they are repeated in the course, e.g., there are two Service Nodes, one for easy entry and a more sophisticated one that is needed for the use case. It should be mentioned that the Nodes are namespaced, starting with the Service Node to measure distances with the LiDAR, as this is best practice in ROS2 and necessary for the course participants to work with the robots at the same time.

IV. Conclusion

In this study, a Learning Platform is designed based on an intralogistics use case. The use case is utilizing a TB4 AMR to rebuild an intralogistics scenario. To replicate the practice as closely as possible, dummy machines, stock, and shipping hardware must be built. The software for the robot is developed using the cutting-edge ROS2 framework widely used in the robotics field. The software required for the use case is developed in learning modules, at the same time the concepts of ROS2 are introduced and completed. This includes the development of a docking procedure for the robot by using image processing and the operation of the elevator built on top of the robot. These software packages will be used by the fleet management, developed in [12]. The course is intended to lead from the introduction to ROS2 to the development of sophisticated Nodes.

References

[1] Open Robotics, "Why ROS 2?" Product Information, 2022.

[2] A. Loganathan and N. S. Ahmad, "A systematic review on recent advances in autonomous mobile robot navigation," *Engineering Science and Technology, an International Journal*, vol. 40, p. 101343, 2023, doi:https://doi.org/10.1016/j.jestch.2023.101343.

[3] Mobile Industrial Robots A/S, "MiR 100," User Guide, 2020.

[4] R. Hercik, R. Byrtus, R. Jaros, and J. Koziorek, "Implementation of Autonomous Mobile Robot in SmartFactory," *Applied Sciences*, vol. 12, no. 17, 2022, doi:10.3390/app12178912.

[5] U. Qidwai, "Fun to Learn: Project-Based Learning in Robotics for Computer Engineers," *ACM Inroads*, vol. 2, no. 1, p. 42–45, 2011, doi:10.1145/1929887.1929904.

[6] S. Farzan, "Project-Based Learning for Robot Control Theory: A Robot Operating System (ROS) Based Approach," 2023, doi:10.48550/arXiv.2305.11279.

[7] P. Guo, N. Saab, L. S. Post, and W. Admiraal, "A review of project-based learning in higher education: Student outcomes and measures," *International Journal of Educational Research*, vol. 102, p. 101586, 2020, doi:https://doi.org/10.1016/j.ijer.2020.101586.

[8] K. Gary, "Project-Based Learning," *Computer*, vol. 48, no. 9, pp. 98–100, 2015, doi:10.1109/MC.2015.268.

[9] Clearpath Robotics Inc., "Turtlebot4 User Manual," accessed July 20, 2023. [Online]. Available: https://turtlebot.github.io/turtlebot4-user-manual/

[10] iRobot corp., "Create® 3 Educational Robot," Product Information, 2022.

[11] M. Donath, "Material for MKDocs," 2023, accessed June 15, 2023. [Online]. Available: https://squidfunk.github.io/mkdocs-material/

[12] M. Carlone, "Fleet Management of AMRs in Intralogistics Using ROS2," *Master Thesis*, 2023.

[13] G. Bradski, "The OpenCV Library," *Dr. Dobb's Journal of Software Tools*, 2000.

[14] S. Garrido-Jurado, R. Muñoz-Salinas, F. Madrid-Cuevas, and M. Marín-Jiménez, "Automatic generation and detection of highly reliable fiducial markers under occlusion," *Pattern Recognition*, vol. 47, no. 6, pp. 2280–2292, 2014, doi:https://doi.org/10.1016/j.patcog.2014.01.005.

Christian Schuster studies Mechatronics & Smart Technologies Master at MCI | The Entrepreneurial School in Innsbruck, Austria. His academic pursuits and professional interests are rooted in the fields of software development, robotics, and automation. In his bachelor's thesis, he dealt with the automation of a dispensing process for drinks utilizing an industrial robot.