A Universal Approach to Command and Control Heterogeneous Autonomous Robots

Christoph Sieber*, Luis Miguel Vieira da Silva, Alexander Fay
Helmut-Schmidt-University
Hamburg, Germany
*christoph.sieber@hsu-hh.de

Tobias Brogt*, Gero Strobel
IT-Objects GmbH
Essen, Germany
*tobias.brogt@it-objects.de

Stephan Berkowitz*, Lukas Zembrot
Third Element Aviation GmbH
Bielefeld, Germany
*sb@3rd-element.com

***Abstract*—The usage of a team of heterogeneous, autonomous mobile robots makes it possible to execute a broad variety of scenarios more efficiently and effectively than single robots or a swarm of homogeneous robots ever could do. At the same time, the heterogeneity of these robots makes it difficult to command and control them and to manage their specific capabilities. In addition, it is crucial to test the robot team and its strategies before deploying them to the real world. Since neither a suitable development environment nor a suitable simulation environment are available, this paper presents an approach to enable heterogeneous robots to execute abstract missions on their own in a simulated environment. For this purpose, a sophisticated development and simulation environment has been developed and is demonstrated in this paper. Through this simulation environment, various control strategies can be tested to enable efficient use of a team of heterogeneous robots.**

***Index Terms*—autonomous robots, mobile robots, multi-robot systems, autonomy, simulation, drones**

I. Introduction

Mobile robots gain popularity due to their versatility and relevance in many situations. Robots are used in various sectors such as industry, businesses, and households to perform everyday activities and services. The demands on robots are constantly increasing as they have to accomplish more and more difficult scenarios [1]. But increasingly complex scenarios become more difficult to accomplish with single robots. In addition, responding to a dynamic environment is very challenging for a single robot [2]. Developing a robot that can quickly and safely meet all requirements to cover every possible scenario, is a costly and inefficient challenge [3]. Accordingly, complex scenarios should be accomplished jointly with a team of robots rather than with a single robot. This allows robots to complement each other and compensate for deficiencies of other robots. In addition, a team of collaborating and interacting robots can perform even faster and more effectively [4]. Another advantage is the robustness and reliability of a team of robots: if one robot in the team fails, the remaining robots might still be able to accomplish the scenario. Therefore, multi-robot systems (MRS) are considered [5]. For a team of robots, a distinction is made between a homogeneous and a heterogeneous robot team. A team of homogeneous robots consists of robots that have the same characteristics and capabilities such as a swarm of multiple identical Unmanned Aerial Vehicles (UAV). Different robots with different capabilities are considered in a team of heterogeneous robots [6]. The collaboration of several robots with different, complementary capabilities allows the accomplishment of very complex scenarios and is therefore being investigated in various research projects [2], [3]. However, mainly a heterogeneous robot team of one modality is considered. Thereby, the most frequently investigated modalities are land and air [7]. On the contrary, in the following, we will focus on a heterogeneous robot team of multiple modalities, such as land, air and water. The collaboration of the members of such a team allows to accomplish complex scenarios like search and rescue.

Due to this heterogeneity of a team and the specific capabilities of the individual robots associated with it, the number of possible solution strategies for the accomplishment of complex scenarios increases. This increases the effort to command and control a team of heterogeneous robots [5]. The interaction of different types of robots in an overall system in the context of simulations is still an unexplored area and corresponding simulation tools are not yet available. Therefore, this work is concerned with it.

This paper demonstrates an approach that reduces the effort required to command and control a team of heterogeneous robots. For this purpose, a multimodal applicable system architecture is presented as well as a method that fulfills the accomplishment of scenarios through structured partitioning. Section II gives a brief review of the state of the art of autonomous mobile robots and their development. In Section III the proposed system architecture of a single robot and the integration into a heterogeneous team is presented as well as the integration into the simulation is shown. Using this setup, Section IV demonstrates an application of the approach where a transport scenario is accomplished by a heterogeneous robot team consisting of an UAV and an Unmanned Ground Vehicle (UGV). Finally, a summary and an outlook on further research and development is given in Section V.

II. State of the Art

This section gives a brief overview of autonomous mobile robots. Essential terminologies and definitions are given and

https://doi.org/10.24405/14564

conclusively the common software for robot software development is presented.

A. Autonomous Mobile Robots

A robot is defined by ISO 8373 as a "programmed actuated mechanism with a degree of autonomy to perform locomotion, manipulation or positioning" [8]. Nevertheless, there are a variety of different interpretations of the definition of robots. Robots are often utilized when their environment is fully known and controlled. All their actions are known in advance and thus they do not need to act independently [9].

On the other hand, robots, and especially mobile robots, are considered to need to sense their environment and choose their actions accordingly. The mobile robot does not know the environment in advance, which is also not controllable, and thus does not have a fixed sequence of actions but must act and react in accordance with the environment. Such a robot operates according to certain rules corresponding to the sensed data [8]. Moreover, a mobile robot is a "robot able to travel under its own control" [8]. Finally, an autonomous mobile robot is a robot that acts without remote control [9]. Autonomy is considered a key capability of mobile robots, enabling increasingly complex missions, including teaming with other manned and unmanned entities to accomplish the overall scenario [10]. In the context of this article, a scenario refers to a setting of conditions and circumstances such as the currently available robots as well as an overall goal to be achieved. Missions, on the other hand, are derived from a scenario and consist of one or multiple tasks. Each mission requires one robot. The execution of each mission derived from the scenario provides a possible way to accomplish a scenario.

Thus, there are numerous scientific and legal definitions of autonomy, as e.g. in [9]. However, each of these definitions applies only to specific areas of law or selected technical applications. Accordingly, there is a lack of a unified and universally accepted definition applicable to a team of heterogeneous, multimodal robots. An attempt at a unified legal definition was made in [11]: "Autonomy is a state in which a robot system, once activated, is capable of autonomously performing some or all of the mission tasks, for a specified period of time or continuously, in specified areas or everywhere, while it must remain possible at all times for a technical supervisor to shut down the system to a low-risk, operable state in situations that cannot be foreseen by human judgement." (translated from [11]). The authors of this article endorse this definition, as it ensures human governance by setting the goal and operational framework, while it gives the robot a maximum of self-determination to operate.

B. Software Development Framework ROS2

The Robot Operating System (ROS) is an open-source middleware that is widely used for robotics applications and has become a de facto standard. The increasing demand for MRS has brought the awareness that ROS is not sufficient. ROS was designed to develop a single robot and does not provide a platform for MRS. Additionally, ROS is not real-time capable, which is necessary in complex applications of MRS. Other difficulties include vulnerability to failure due to lack of data encryption as well as lack of reliability [12]. These challenges have brought the development of ROS2. In this process, the successful concept is transferred into a new architecture. The big difference is the use of the Data Distribution Service (DDS) standard for communication, which enables data transfer between processes even on distributed heterogeneous platforms. There is a global data space that can be accessed by all applications.

In ROS2, nodes are used to represent independent computational processes. The use of nodes promotes modularity, reusability, and faster development. Communication between nodes can be done through topics or other methods such as services. Communication using topics follows a publish/subscribe model, where messages are passed through a clearly assignable topic. Nodes can subscribe to a topic by the name of that topic and receive the message when a node publishes on that topic [12]. Every node that publishes or subscribes to data is a participant that is allowed to write to and read from the data space. In contrast services represent a call/response model, where a service provides information only when it is called by a client [13].

At the same time, a node remains responsible for a specific, modular objective and can be treated and programmed in isolation [13]. Further, a node cannot be forced from an external source to behave in a certain way. The independence of the nodes and the use of DDS for communication allow MRS to be implemented while ensuring the autonomy of the individual robots and at the same time providing a capability to communicate and interact with the environment and especially other robots.

III. System Architecture & Integration

In the context of the deployment of a team of heterogeneous robots, a distinction can be made between three simulation stages of a single robot. On the one hand, there is a purely software-based simulation of all components, which is referred to as software in the loop (SiL) [14]. Furthermore, hardware components can be integrated into the simulation. This is referred to as hardware in the loop (HiL) [15]. The last step, which does not involve simulation in the strict sense, describes the deployment of a robot to the real world. This includes the installation of the necessary software on the real hardware components of the robot and the execution of these software modules, to control the robot in a real-world mission.

In addition to these stages of the deployment of a single robot, it is possible to form a team of robots from both simulated and real robots, for example based on a real robot and a simulated robot. This means that individual robots are used which are based on the different simulation stages described above. This makes it possible to evaluate different performance aspects at different levels reaching from the individual robot to the entire robot team. To enable such use cases for a broad range of heterogeneous robots in a simulation, a

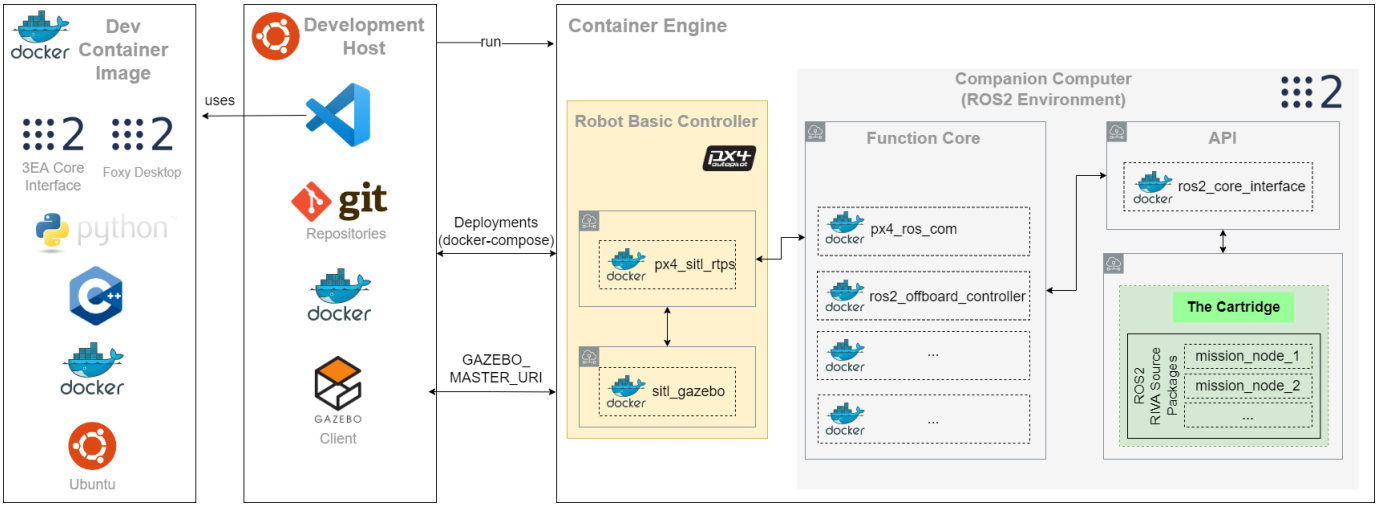

FIGURE 1. OVERVIEW DEVELOPMENT ENVIRONMENT.

powerful tool is required that enables the integration as well as the simulation of hardware components and real robots. In addition, a development environment integrated into the simulation is advantageous to quickly integrate new software modules into the existing infrastructure, for example, to be able to quickly evaluate adaptations to algorithms.

In order to fulfill the stated requirements, a simulator was developed. It is made up of various components and individual systems that make it possible to represent the various mentioned forms of simulation. The components and their interactions are shown in Fig. 1.

The initial aim was to make the installation and configuration of the simulator as simple as possible despite the large number of components. Container technology is used for this purpose. With the help of Docker[1], essential components are flexibly assembled and can be executed independently of the conditions on the host system. The integrated simulation tool runs on an Ubuntu based host system (Ubuntu 20.04 LTS) with a Docker installation. In addition the authors selected Gazebo[2] as the simulation environment because this tool is used extensively in research and practice, has many extensions, is under continuous development and has an active community of users as well as developers [16]. In addition, a separation of server and client is enabled, which reduces the total computing effort in a distributed system such as an MRS. The Gazebo client runs directly on the host system, while the Gazebo server is deployed using a Docker container. The client automatically connects to the server so that the simulation results calculated on the server can be easily displayed without requiring the server on the host system.

The Gazebo server is supplemented by a container that includes the Robot Basic Controller (RBC), which in the case of chosen UAV systems is represented by the px4 platform[3].

[1]https://docs.docker.com/
[2]https://gazebosim.org/home
[3]https//px4.io/

The RBC takes over all the necessary control mechanisms of a robot's sensors and actuators, such as controlling the different UAV rotors. The associated px4 autopilot-software is integrated with the Gazebo simulation to send commands to the robot and receive simulation data (e.g., sensor data and GPS positions).

The px4 uses a bridge module inside a further container, which maps messages between the RBC and the ROS2 Environment. ROS2 is chosen, because ROS is de facto the standard for the development of robots and for the development of a team of robots the further development ROS2 must be used. The ROS2 Environment contains on the one hand the robot specific nodes, the so-called Function Core. It controls the functional processes on the robot at a higher abstraction level than the RBC. The Function Core contains multiple ROS2 nodes, which are embedded in Docker containers. For each robot type, the RBC and the Function Core form a unit with basic functions that enable the fundamental operation of a robot. This unit also prevents aspects of unsafe behavior, e.g., planning robot trajectories outside defined safety zones. The number of basic functions is significantly dependent on the capabilities of a particular robot and is therefore finite or can only be changed by modifying the robot. It is therefore suitable to encapsulate them and to offer only one peripheral interface through which these basic functions can be executed as often as desired and in any order. Since this encapsulation is achieved by both hardware and software, reproducibility and modularity are also easily enabled.

On the other hand, the Cartridge is part of the ROS2 Environment. The Cartridge serves as an interface for the developers and allows individual testing and code variations, such as mission planning. The Cartridge is composed of nodes in containers that can be implemented in various approaches and programming languages like Python or C++. Several Cartridges can be deployed on a robot so that they can be tested alternatively or competitively. Here, for example, communication modules can be implemented to enable collaboration

in a team of heterogeneous robots. All communication with the external world is done via the Cartridge. The Cartridge is connected to the robot specific nodes via the Core Interface. It provides an application programming interface for connecting the basic functions of the Function Core and RBC with creative functions, such as mission planning, which can be integrated here in the respective ROS2 nodes.

The number of creative functions depends only on the input of the developer and is therefore infinite. They can be individually extended or removed for the respective scenario, as well as for a broader range of use cases. Main advantage of the separation of creative functions from basic functions is the possibility for the developer to create and integrate software-modules to command and control the individual robot, without the need to address specific actuators or sensors, enabling the developers to target a broad range of heterogeneity.

In order to support the development of software in the Cartridge, Visual Studio Code[4] is used as the development environment (IDE). It integrates well with the components used, is open source and there are also many plug-ins for extending the IDE. In contrast to classic software development, the development environment does not run natively on the operating system of the host system, instead it is integrated into the system landscape of the simulation using container technology. The container contains all required dependencies, IDE extensions and mounts the source code repository, containing the development artifacts.

The container-based orchestration of the individual components of the simulator makes it possible to use the different simulation stages and their advantages. Gazebo represents the real world as a physics engine and simulates the behavior of the robots and provides corresponding sensor data, such as speeds or position data. The RBC simulates the control unit used for basic robot movement. A real robot for example can use a px4 autopilot-software, running on an associated hardware unit on the robot. The ROS2 Environment, running in containers on the host system, can be deployed in the same way on the real robot with the help of a companion computer. The two simulation stages of a single robot SiL and HiL both use Gazebo. For the SiL simulation, the RBC container and the ROS2 Environment containers are used. When running a HiL simulation, the RBC container is replaced with the real hardware used on the robot, which is then integrated with Gazebo and the ROS2 Environment. When forming a team of robots from both simulated and real robots, the above-mentioned procedures are combined in a flexible way. This allows for easy deployment and evaluation of mission execution.

IV. Application

In this chapter, the applicability of the presented approach is demonstrated by means of a simple transport scenario. This scenario simplifies a real-world problem, where a factory hall and a warehouse are separated from each other by difficult terrain. It is visualized in the simulation environment of Gazebo.

The goal of the scenario is to transport a box from a pick-up point to a drop-off point. Therefore, a team of heterogeneous robots is provided in a simulated setup. The team consists of one UAV and one UGV. As the robots are activated, they shall be capable of autonomously performing all mission tasks required to execute the transport. Due to environmental obstacles, the transport can neither be performed by the UAV nor by the UGV alone. The pick-up point is located on a flat surface on this side of a wall that the UGV cannot cross. The drop-off point is located on the other side of the wall, inside a building the UAV cannot enter. At the beginning, the UAV is located at the pick-up point and the UGV is near the drop-off point. Figure 2 shows this environment and the initial scenario setup.

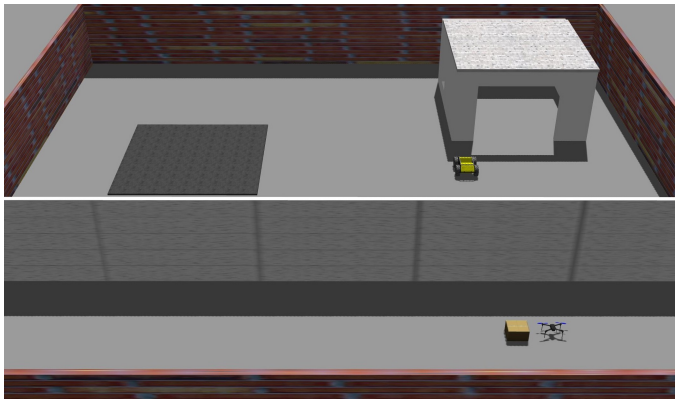

Figure 2. Initial Scenario Setup.

Consequently, the transport must be conducted sequentially with the use of both robots and by transferring the box at a specific handover point. Hence, the creative function of mission planning is used. This yields to one specific mission for the UAV and one specific mission for the UGV. These robot-specific missions are planned in an external mission-planning node and communicated to the individual robot. A mission is formulated abstractly, making it independent from basic functions. It therefore consists of abstract tasks, such as “move to a specific coordinate”, instead of “fly to a specific coordinate”.

After the UAV and the UGV each receive their specific missions, an internal mapping procedure is started. This mapping analyzes the mission by processing it task by task and identifies the basic functions required to execute the mission via the Core Interface.

The result is a sequence of basic functions called execution-list. Example: The UAV maps one abstract task “move to a specific coordinate” into two sequential basic functions “take-off” and “fly to a specific coordinate”. To ensure maximum autonomy in the execution of the mission, the exact flight trajectories are planned by the UAV.

As soon as the mapping is finished, the created execution-list is processed through the Core-Interface in a structured way.

[4]https://code.visualstudio.com/

In the given scenario, the UAV picks up the box, takes off, flies to the defined handover point, lands, drops the box, messages the UGV, takes off and flies back to the pick-up point, where it finally lands. The UGV starts driving to the handover point as soon as it receives the message from the UAV. There it picks up the box, drives to the drop-off point and unloads the box. FIGURE 3 shows the box lying at the handover point while the UAV flies back to the pick-up point and the UGV drives to the handover point.

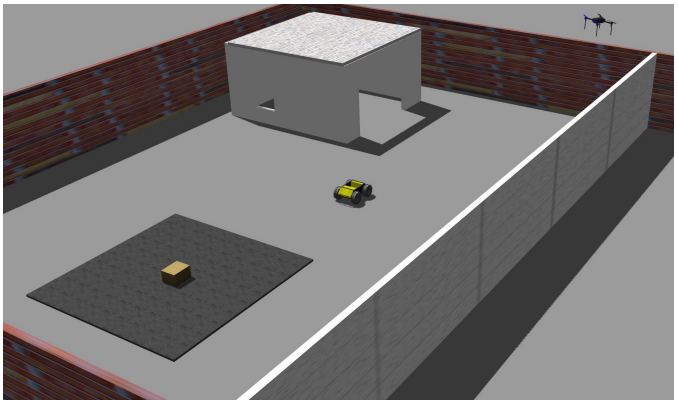

FIGURE 3. SCENARIO EXECUTION PROGRESS.

The scenario ends when the rover unloads the box. Both robots are now idle and available for following missions.

V. CONCLUSION AND OUTLOOK

In this article, an approach was presented that enables it to command missions to mobile robots, independent of their basic functions, and to let them execute those missions autonomously. For this purpose, an operative separation of basic functions and so-called creative functions is conducted in the system architecture of the robots. This architecture was presented and a simple transport scenario was used to demonstrate its functionality and effectiveness.

While the basic feasibility of the approach with respect to mission planning and execution has been demonstrated here, it also offers great potential for further research. An extension of the use cases to target a larger robot team of greater heterogeneity and modality is in progress. Additionally, other use cases are being considered due to the diverse applicability of a robot team. Since the approach allows the robots not only to be simulated, but also to be deployed in the real world, or to perform a hybrid simulation, these techniques are also under further investigation.

Mission planning is currently carried out by humans and only communicated automatically to the robots. With respect to the specific capabilities of the individual robots, it is intended to automate this planning process and only provide a common goal for the robot team. The presented separation of the basic functions and the creative functions represents a suitable prerequisite for this.

In addition, until now, the robots have carried out their missions sequentially in order to accomplish the given scenario. This poses challenges, especially in the case of unforeseen events. These can be overcome with interaction and collaboration between the individual robots and are empowered by the approach presented. The necessary replanning of missions is also part of the authors' research.

ACKNOWLEDGMENT

This Paper is funded by dtec.bw – Digitalization and Technology Research Center of the Bundeswehr which we gratefully acknowledge [project RIVA].

REFERENCES

[1] M. B. Alatise and G. P. Hancke, "A review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods," *IEEE Access*, vol. 8, pp. 39 830–39 846, 2020.

[2] Y. Rizk, M. Awad, and E. W. Tunstel, "Cooperative Heterogeneous Multi-Robot Systems," *ACM Computing Surveys*, vol. 52, no. 2, pp. 1–31, 2020.

[3] Y. Carreno, È. Pairet, Y. Petillot, and R. P. A. Petrick, "Task allocation strategy for heterogeneous robot teams in offshore missions," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 222–230.

[4] Yifan Cai and Yang Simon X., "A survey on multi-robot systems," in *World Automation Congress 2012*, 2012, pp. 1–6.

[5] E. Tuci, M. H. M. Alkilabi, and O. Akanyeti, "Cooperative Object Transport in Multi-Robot Systems: A Review of the State-of-the-Art," *Frontiers in robotics and AI*, vol. 5, p. 59, 2018.

[6] H. Asama, "Trends of Distributed Autonomous Robotic Systems," in *Distributed autonomous robotic systems*, H. Asama, Ed. Tokyo: Springer, 1994, pp. 3–8.

[7] A. Schweim, M. Zager, M. Schweim, A. Fay, and J. Horn, "Unmanned Vehicles on the rise: A review on heterogeneous teams of cooperating robots," submitted for publication.

[8] ISO 8373:2021(E), "Robotics – Vocabulary," 2021.

[9] J. Hertzberg, K. Lingemann, and A. Nüchter, *Mobile Roboter: Eine Einführung aus Sicht der Informatik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[10] T. Petermann and R. Grünwald, "Stand und Perspektiven der militärischen Nutzung unbemannter Systeme," *Büro für Technikfolgenabschätzung beim Deutschen Bundestag, Arbeitsbericht*, vol. 144, 2011.

[11] C. Worpenberg, "Level 3, 4 oder 5? "Autonome" Fahrzeuge zu Land, Wasser und in der Luft - ein Plädoyer für eine verkehrsbereichsübergreifende Begriffseinführung von Automatisierungs- und Autonomiegraden," *InTeR Zeitschrift zum Innovations- und Technikrecht*, 2022.

[12] Y. Maruyama, S. Kato, and T. Azumi, "Exploring the performance of ROS2," in *Proceedings of the 13th International Conference on Embedded Software*, ser. ACM Digital Library. New York, NY: ACM, 2016, pp. 1–10.

[13] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot Operating System 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022.

[14] S. Demers, P. Gopalakrishnan, and L. Kant, "A Generic Solution to Software-in-the-Loop," in *2007 IEEE Military Communications Conference*. Piscataway, NJ: IEEE Service Center, 2007, pp. 1–6.

[15] M. Bacic, "On hardware-in-the-loop simulation," in *2005 44th IEEE Conference on Decision and Control & European Control Conference*. Piscataway, NJ: IEEE Operations Center, 2005, pp. 3194–3198.

[16] N. Koenig and 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)*, vol. 3. Piscataway, NJ: IEEE Operations Center, 2004, pp. 2149–2154.