RRL2023 ALeRT Germany

Maximillian Kirsch, Shubham Pawar, Dennis Kroll, Christoph Gollok, Mahesh Vattekatt, Diksha Purohit, Stefan Schiffer, René Rütters, Alexander Ferrein

Info

Team Name:	ALeRT
Team Institution:	FH Aachen, MASCOR Institute
Team Country:	Germany
Team Email:	alert@fh-aachen.de
Team Leader:	Maximillian Kirsch
Team URL:	ALeRT Website
Qualification Videos:	MASCOR YT ALeRT
RoboCup Rescue TDP collection:	None

***Abstract*—This paper presents the progress made by Team ALeRT for the RoboCup RescueLeague using our quadruped Spot robot from Boston Dynamics. We are a team of research associates and students from the Mobile Autonomous Systems & Cognitive Robotics Institute (MASCOR) at FH Aachen University of Applied Sciences.**

Our development efforts focus on integrating Spot and all associated devices and software into ROS 2. Specifically, we aim to evaluate autonomy using `golog++`, which enables task planning for deliberation, a critical aspect for rescue scenarios.

We are currently integrating sensors, grasping, automated object labeling and human detection, image segmentation for semantic reasoning, and a web-based graphical user interface (GUI) for operators to facilitate autonomous behavior.

***Index Terms*—RoboCup Rescue, Team Description Paper, Spot, ROS 2, `golog++`.**

I. Introduction

ALERT, the Aachen Legged Rescue Team, is a team of students and research staff from the MASCOR Institute at the FH Aachen University of Applied Sciences that aims to compete in the RoboCup RescueLeague. Though newly founded, some of our members have prior experience in the RoboCup through the LogisticLeague. We did not yet participate in any regional tournaments or publish scientific papers, but we are actively planning to participate in the upcoming German Open as a stepping stone towards qualification. We also presented a poster on the current status of the development of Spot with ROS 2 at the last RoboCup Symposium in Bangkok.

To compete in the RescueLeague, we chose to use the quadruped robot, Spot, by Boston Dynamics. Spot comes with various functionalities such as teleoperation and navigation with collision avoidance. Some of Spots functions are not open including access to the low-level motor controls. Therefore we cannot develop our own kinematic for Spot and have to use the Spot SDK to control locomotion. We could setup our own navigation on Spot with the Robot Operating System, ROS 2 [1], [2], but due the high amount of requirements in terms of sensors, vision and manipulation, we opted to use the available functions of the Spot SDK, like navigation, where possible, to allow us to invest more time in the open issues.

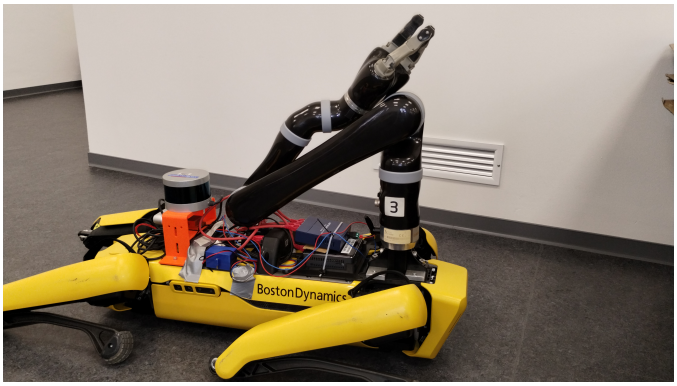


Fig. 1: Photo of our Spot with our major modifications

To prepare Spot for the Challenges in the RescueLeague we began by implementing a ROS 2 driver for Spot or respectively a wrapper between the Spot SDK and ROS 2 core functionalities.

Today, the Spot ROS 2 driver is an open-source package maintained by the Boston Dynamics AI Institute and the MASCOR Institute. Concurrently, we created a robot simulation for Spot in Webots, which enables us to test our software in a safe environment and facilitate new team members' understanding of Spot's architecture.

We then mounted and integrated various sensors and a manipulator to fulfill the challenges of the RescueLeague. To do this, we created and ported several packages to ROS 2, including the Spot driver, a driver for the Kinova Jaco, and *hector_slam* [3] for GeoTIFF maps.

While we have yet to decide whether to control Spot via teleoperation and autonomous background services or complete autonomy, our long-term goal is to create and evaluate complete autonomous behavior for Spot in rescue scenarios with `golog++` [4], [5], [6] agents. `golog++` is an interfacing and development framework for GOLOG [7], [8], [9], [10], [11] languages, which are logic action languages that use knowledge representation and reasoning for planning domains. High-level controllers based on GOLOG have been used for various domains such as robotic soccer [12] or domestic service robotics [13].

In the next chapter, we describe the system architecture and the hardware and software used.

II. System Description

A. Hardware

We use the Boston Dynamics Spot as robot platform, with an Intel NUC serving as the computing unit. The major

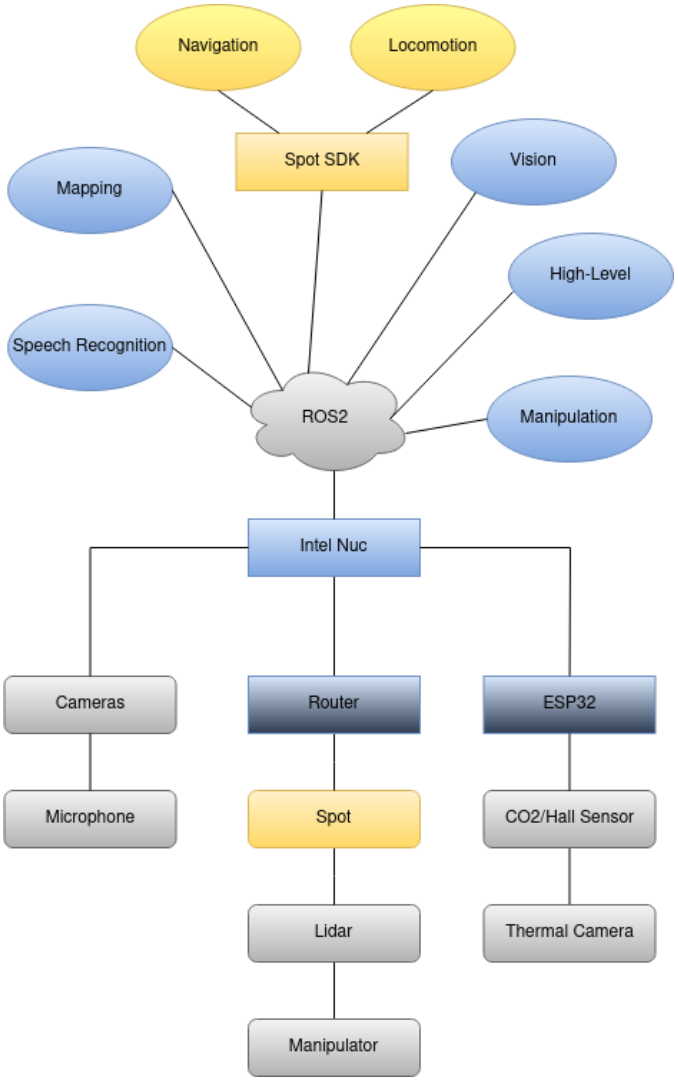


Fig. 2: Overview of the System Architecture.

modifications we did are to attach sensors, mount and integrate the Kinova Jaco and creating power supplies for the necessary devices. All network devices, including the LiDAR, manipulator, and Spot itself, are connected via ethernet ports through a router to the NUC. The hall, CO_2 sensors and thermal cameras are connected to the NUC through an ESP32, while devices such as cameras and microphones that only require a USB port are directly connected to the NUC. An overview of our system architecture is shown in Fig. 2.

1) Locomotion: Spot is a quadruped robot with a max speed of 1.6 $\mathrm{m/s}$, max slope of ±30°and a max step height of 300mm. Also Spot has a max payload mounting weight of 14 kg. All details of the platform are found in the specifications.

2) Manipulation: To complete the DEX challenges we have a Kinova Jaco (Tab. I) robotic arm. We want to evaluate if the lightweight Kinova gripper is an alternative to the available Spot Arm.

3) Power Supply: Spot has an unregulated DC voltage output of 35-75V and a maximum power output of 150W per port. At Port 1, we have hooked up a Boston Dynamics Spot-GXP, which provides us with two power rails - 24V and 12V.

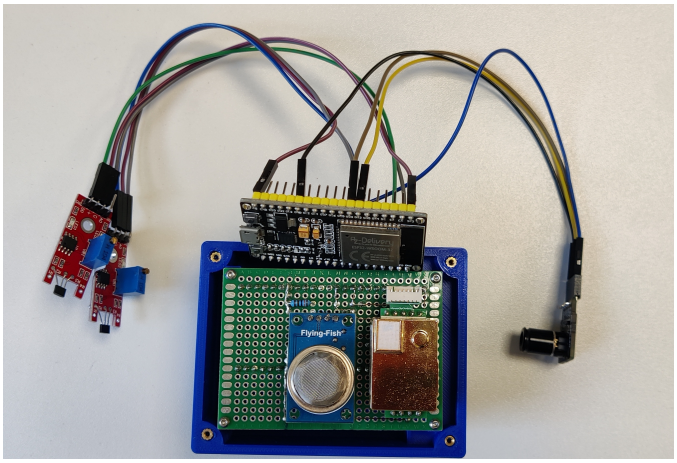


Fig. 3: From left to right, two A3144 hall sensor, an ESP32 microcontroller, below that MQ135 and MHZ-19B on right side MLX90640 thermal camera.

The 12V is used to connect the sensors and the router. The 24V is converted to 19.2V with a buck converter to power the NUC.

Due to the increased power consumption of the robot arm, it is connected to port 2 of the Spot via a buck converter. The voltage is 24V with a maximum current output of 6A.

The standby time for the batteries is 180 minutes, and their average runtime is 90 minutes, depending on the payload. However, we have not yet tested the power consumption when the gripper is attached.

4) Sensors:

a) Mapping: For mapping, we utilize a Velodyne VLP-16 3D-LiDAR. The Velodyne system enables larger scale mapping than that achievable with Spot's five internal grayscale stereo cameras.

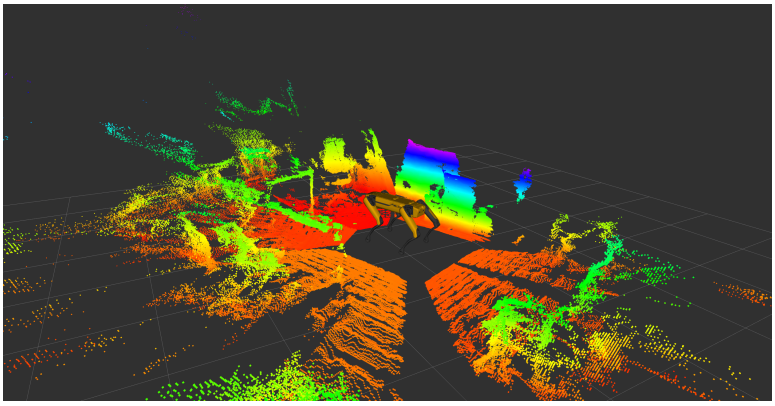
b) Thermal/Magnetic: The thermal and magnetic sensors are connected to the ESP32, which in turn is connected via USB to our NUC. The setup is shown in Tab 3. Currently, we plan to use data from the ESP32 pass it to the NUC and publish the data via a ROS 2 Node. We need to conduct some tests to determine wether micro-ROS is a viable alternative to directly publish the sensordata from the ESP32 to ROS 2 instead of implementing a ROS 2 Node ourselfs.

c) Object Detection: To perform object detection, we have installed three RealSense cameras, for which we have prepared the necessary mounting. Our Spot model has five grayscale cameras and by mounting additional RealSense cameras, we are able to capture color information for more accurate object detection.

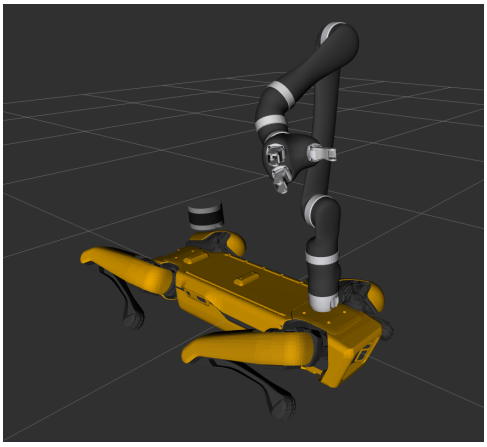
B. Software

The operating system Ubuntu 22.04 runs on the NUC and we use ROS 2 humble as middleware. Our objective is to create a ROS 2 interface for every hardware device we use.

In cooperation with the Boston Dynamics AI Institute, we developed a ROS 2 driver for Spot that enables the wrapping of functions of the Spot SDK into ROS 2 topics, services, and actions. With the ROS 2 Spot Driver, we can publish geometry



(a) The ROS 2 package *depth-image-proc* is used to transform depth images of Spots 5 internal cameras to a Pointcloud2



(b) *RobotModel* of Spot, Kinova Jaco and LiDAR in RViz2

Fig. 4: Spot visualized in RViz2

twist messages to the cmd_vel topic, start motors via services, and send navigation goals with collision avoidance and more. The Fig. 4a shows the visualization of Spot's *RobotModel* and the *Pointcloud2* generated by Spot's stereo cameras in RViz2.

Despite ROS 2 having been released years ago, many packages are still only available for ROS. To address this, we ported packages such as the Kinova Jaco manipulator driver and *hector_slam* to ROS 2 enables us to create GeoTIFF maps.

1) Low level control: To control the locomotion of the quadruped robot Spot, we have to use the functions provided by the Spot SDK. However, the SDK does not offer all of the low-level functions necessary to directly control the robot's motors.

The Spot ROS 2 driver provides topics to control Spot's motion, including the cmd_vel topic, which enables the direct use of ROS 2 packages such as *teleop_twist_keyboard* and *teleop_twist_joy*. Moreover, the driver enables the ability to rotate Spot's body in roll, pitch, and yaw and adjust its height during movement from a minimum of 0.52m to a maximum of 0.7m. The different gait modes of the robot are also accessible via ROS 2.

Furthermore, Spot comes equipped with a handheld controller for tele-operation, allowing for easy remote control of the robot's movement.

2) Localization: The Spot SDK provides odometry data for Spot, which can be accessed via a ROS 2 topic. Currently, we are utilizing the internal odometry of Spot. However, we plan to develop our own method of calculating the odometry data in the future.

3) Mapping: Our mapping system leverages the Velodyne VLP-16 sensor mounted on Spot to capture high-resolution 3D data and generate maps. We use the efficient ROS 2 package *slam_tool_box* for real-time generation of high-quality maps. To produce a georeferenced map, we have adapted the *hector_geotiff* package for ROS 2, allowing easy integration with other geospatial data. A dedicated ROS 2 node locates and marks detected objects within the GeoTIFF map, performing the necessary transformation between the sensor and map frame. A GeoTIFF map generated by *hector_slam* is shown in Fig. 5.

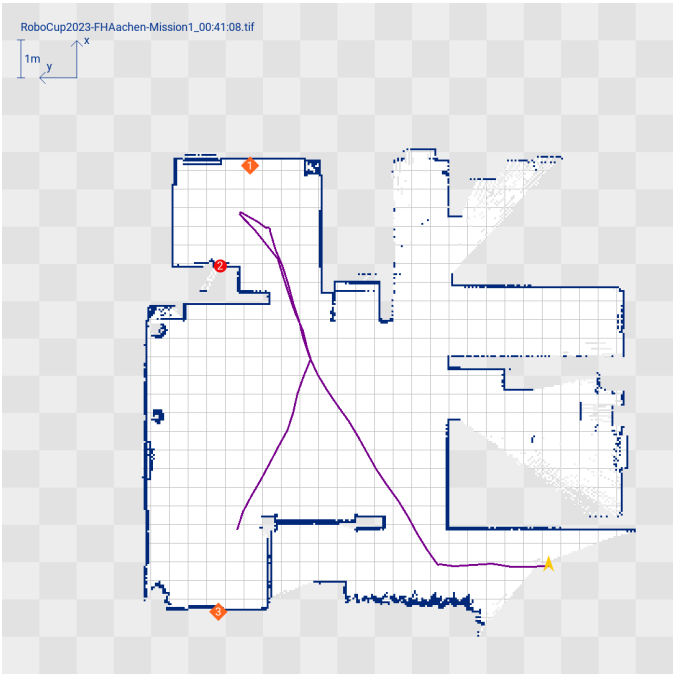


Fig. 5: GeoTIFF map generated by *hector_slam* for ROS 2 with different objects marked inside Webots.

4) Navigation: Spot offers an easily accessible navigation solution that can be controlled via a handheld device or the Spot SDK. The Spot ROS 2 driver's action server employs the Spot SDK functions to manage requests to move to a designated location. To cater to the RescueLeague's diverse requirements, we decided to use Spot's in-built navigation system with an interface through the Spot ROS 2 driver's action server.

In the long term, our goal is to incorporate open-source navigation and path planning methodologies, such as *navigation2* and *slam_tool_box*. With the integration of open-source tools,

we aim to expand the functionality of our system, ultimately increasing its versatility and effectiveness in various settings.

5) Autonomy: To achieve autonomous behavior, we intend to use the `golog++` framework, which offers a GOLOG language that can be executed on real robots. We had previously developed a ROS interface for `golog++` [4] and now we have successfully ported it to ROS 2.

The `golog++` language utilizes a Classical Planning Representation, provides a planner but also allowing imperative programming for simpler tasks that do not require a plan. To provide a brief explanation, a `golog++` program undergoes transformation into a C++ object model and is subsequently executed by the READYLOG [14], [12] interpreter. The `golog++` language provides instantaneous actions that are mapped to ROS 2 services, durative actions that are mapped to ROS 2 actions and exogenous events that are mapped to ROS 2 topic subscriptions.

6) Manipulation: Initially, we found only a ROS package available for the Kinova Jaco. As we wanted to avoid the use of the *ros_bridge*, we had to port the package to ROS 2. The Kinova Jaco utilizes its own library to calculate inverse kinematics and reach a desired end position for all its joints. Our next objective is to integrate the Kinova Jaco with MoveIt! [15], which will allow us to create a collision box for the arm in conjunction with Spot.

7) Vision: We are detecting landolt C orientations in five steps: i) An image of a video stream is first converted to grayscale. It is then filtered by adding blur. ii) The image is converted to a binary mask with respect to threshold and individual contours are found. iii) If the ratio of contour's (convex hull area)/(minimum enclosing area) is greater minimum circle ratio, defect of the contour is found. iv) Again, if the depth of this defect is greater than minimum depth, it is marked as a gap. v) The orientations of gaps are calculated wrt to the largest one always being 'Top'.

Also, a list of the orientations of landolt c's (in descending size order) is displayed in red. The Following orienations are possible: Top (T), TopLeft (TL), Left (L), BottomLeft (BL), Bottom (B), BottomRight (BR), Right (R), TopRight (TR) A detection of landolt C orientation is shown in Fig. 6.

Finding hazmat signs: i) An object detection model is fine tuned with YOLOv5 by Ultralytics, and then converted to ONNX format. ii) The ONNX model is loaded with an OpenVINO toolkit inference engine, and the bounding boxes are drawn on the image for visualization. iii) For pose estimation of a detected sign, the median depth point within the bounding box is being used, which we get from RealSense cameras.

Finding victims: i) An instance segmentation model is fine tuned with YOLOv5 by Ultralytics, and then converted to ONNX format. ii) The ONNX model is loaded with an OpenVINO toolkit [16] inference engine, and the bounding boxes and masks are drawn on the image for visualization. iii) For pose estimation of a detected victim, the median depth point within the mask is being used, which we get from RealSense cameras.

Finding QR codes: i) Image is first converted to grayscale. It is then filtered by adding blur. ii) *zbar* library is used to find QR codes in the image. iii) Using its 4 corners and *solvepnp*

Fig. 6: Landolt Detection: The green circles in output images refer to the contour defects.

Fig. 7: Hazmat Detection of a non-flammable and dangerous sign

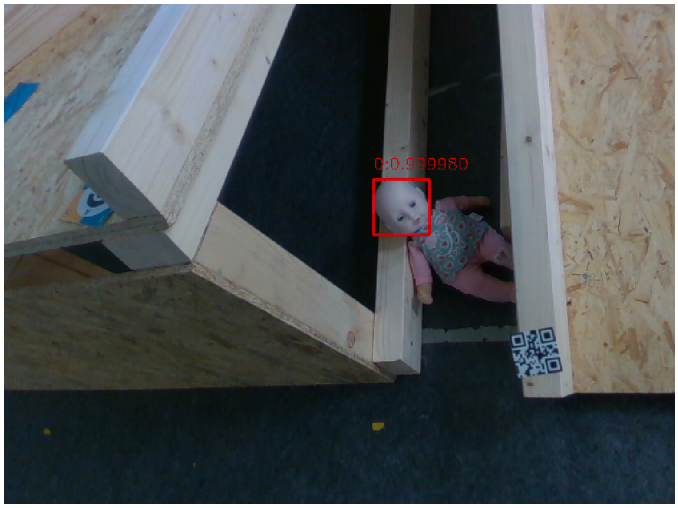


Fig. 8: Baby Doll Detection with a Realsense using YOLOv5.

Fig. 9: Qr-Code Detection: i) In the right image, the red dot on the bounding box of the qr code denotes the top left corner of the tag. ii) In the left image, the published transformation is visualised with respect to the camera frame.

from OpenCV [17], the pose of the individual QR code is found and a transform is published as well.

8) Communication protocol: All our devices have an interface to ROS 2. The LiDAR and the manipulator are connected via RJ45. Our RealSenes cameras are simply connected via USB.

C. Communication

We use a Wifi connection to establish communication between the robot and the operator station. Unfortunately, we have not yet determined any further details at this time.

D. Human-Robot Interface

Spot can be controlled using the handheld controller or through ROS 2. For tele-operated tasks, we aim to incorporate the SteamDeck into the tasks. In addition to the controller, the operator can interact with Spot via the ROS 2 Spot driver. After establishing an SSH connection from the operator's laptop to the NUC and starting the Spot driver, launch files can be executed to initiate the necessary nodes for a given challenge. Also topics can be visualized mit RViz2.

To provide a graphical user interface for the operator, we are working on a web-based solution for Spot. Our objective is to use ROSWebTools such as ROSboard to visualize all collected data. Then, we intend to integrate buttons within the web GUI which enable executing launch files, starting of nodes on Spot and an emergency stop button.

III. Application

A. Set-up and Break-Down

Both Spot and Kinova come with their own travel cases. Unfortunately, we do not yet have travel cases for the sensors, operator station and other hardware, which is why we do not have measure setup and breakdown times at the moment.

B. Mission Strategy

Following the latest team leader call and updated rulebook, we must discuss our strategy for the RoboCup RescueLeague within our team in more detail.

Our current idea involves utilizing a semi-autonomous system, using multiple `golog++` agents that can be executed through a web GUI by the operator. Specifically, we aim to completing the easier DEX tasks, detecting landolt objects, identifying objects, finding baby dolls and mapping within the new combo lane. We also plan to utilize additional sensors like CO_2, magnetic and a microphone for the challenges in the RescueLeague.

Our strategy involves deploying Spot into the lane via a `golog++` agent, allowing it to search and solve possible tasks through the execution of further `golog++` agents executed by the operator via the web interface. If Spot successfully solves all tasks, the operator will send him back to the starting point and begin a new repetition.

Given our lightweight gripper's limited length and payload, we must prioritize the easier DEX tasks that can realistically be achieved. Our primary objective is to demonstrate our system's planning capabilities and integration with ROS 2 by evaluating and integrating `golog++` agents into various challenges and tasks.

C. Experiments

For the qualification video, we constructed two lanes to test Spot's mobility. Spot performed well in the TER 2 challenge, which involved navigating K-Rails on a Crossover Slope. However, in the MAN 2 challenge, where the robot has to step over two planks while maintaining ground contact, Spot's default configuration made it difficult to complete the challenge successfully. We performed both challenges via tele-operation using a keyboard.

In addition to the qualification video, we also tested our developed Spot ROS 2 driver at an event last summer. Visitors had the opportunity to move Spot through a parkour using gestures, as shown in Figure 10.

To test our software, we also use our Webots Spot simulation whenever possible.

D. Application in the Field

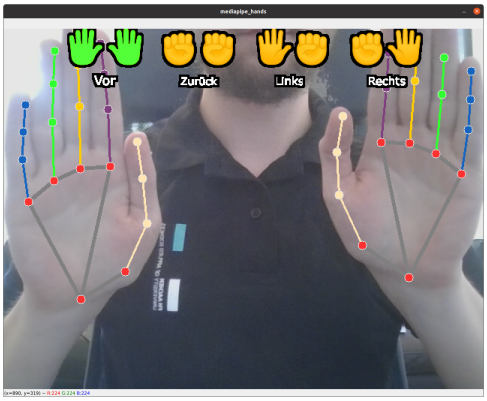
Quadruped robots are becoming increasingly popular due to their impressive mobility, speed, and payload. As stated by the manufacturer, Spot has a wide range of applications, including remote inspection of hazardous environments, rescue operations, and logistics operations.

A rescue scenario involves complex tasks and interactions in different environments, including humans. Therefore, robots require a higher level of autonomy for efficient deliberation. `golog++` allows for task planning on real robots, which is a key element of deliberation.

We are currently developing a web GUI that is adapted for the SteamDeck, which will make our system more accessible and user-friendly. This web GUI will provide an interface through which users can access Spot's functions via a ROS 2 interface.

Once the web GUI is implemented, operators should be able to control Spot in three different modes: tele-operated, semi-autonomous and fully autonomous. This will offer them greater flexibility in using the robot in various scenarios. In addition, the produced data will be visualized, providing operators an intuitive and easy-to-understand representation of Spot's actions.

(a) Spot walks with collision avoidance through a parkour with ramps while being controlled by an operator's gestures.



(b) GUI that visualize gesture detection with *mediapipe* to recognize open and closed hand gestures for teleoperating Spot.

Fig. 10: Showcase of Spot moving through a maze controlled by adults and kids.

We believe that choosing ROS 2 over ROS has both strengths and weaknesses. ROS has been used for years and has become the state-of-the-art in terms of middlewares. Additionally, there exists a ROS package for Spot, which was first maintained by Clearpath Robotics but has since been canceled.

Although ROS 2 has been around for several years, some features have still not been ported. Nevertheless, we believe that ROS 2 has more advantages in the long run. ROS will reach its end of life in 2025, and there will be no official support from Open Robotics. The ROS-Industrial Consortium has expressed its intention to continue supporting and maintaining ROS, but it remains to be seen how much support ROS will receive from the community after the official end-of-life date.

IV. Conclusion

In conclusion, the Aachen Legged Rescue Team (ALeRT) is a newly founded team of students and research staff from the MASCOR Institute at FH Aachen University of Applied Sciences, whose aim is to compete in the RoboCup RescueLeague. Although we have not yet participated in any regional tournaments or published scientific papers, we are actively planning to participate in the upcoming German Open as a stepping stone towards qualification. Additionally, some team members have participated successfully in the RoboCup LogisticsLeague for several years.

To compete in the RescueLeague, we chose to use the quadruped robot Spot by Boston Dynamics. We created a ROS 2 driver for Spot, mounted and integrated various sensors, and created and ported several packages to ROS 2, including the Spot driver, a driver for the Kinova Jaco, and *hector_slam* for GeoTIFF maps.

The team's long-term goal is to create and evaluate complete autonomous behavior for Spot in rescue scenarios with `golog++` agents.

The mission strategy involves utilizing a semi-autonomous system that uses multiple `golog++` agents, which can be executed through a web GUI by the operator.

TABLE I: Manipulation System

Attribute	Value
Name	Kinova Jaco
Reach	900 mm
Grip force	25 - 40 N
Degrees of freedom	4 or 6
System Weight	5.2 kg
Weight including transportation case	? kg
Transportation size	? x ? x ? m
Unpack and assembly time	?
Startup time (off to full operation)	?
Power consumption	25 W
Cost	? USD

Appendix A
Team Members and Their Contributions

- Maximillian Kirsch — Spot ROS 2 driver, High-Level
- Shubham Pawar — Object Detection, Mapping, Gripper
- Christoph Gollok — System Integration
- Dennis Kroll — Power Supply
- Mahesh Vattekatt — Sensors
- Diksha Purohit — Sensors
- Stefan Schiffer — Advisor
- René Rütters — Advisor, Sponsor
- Alexander Ferrein — Advisor, Sponsor

Appendix B
Lists

A. Systems List

Unfortunately, we have not yet developed a concept for an operator station.

B. Hardware Components List

Hardware components list of our robot.

C. Software List

List of all relevant software packages we use.

TABLE II: Robot Platform

Attribute	Value
Name	Spot
Locomotion	quadruped
System Weight	32.7kg (72.1 lbs)
Weight including transportation case	47.6kg
Transportation size	927 x 546 x 464 mm
Typical operation size	1100 x 500 x 610 mm
Unpack and assembly time	? min
Startup time (off to full operation)	? min
Battery endurance (Average Runtime/ Standby)	90 / 180 min
Maximum speed	1.6 m/s
Payload	14 kg
Cost	60000 €

TABLE III: Operator Station

Attribute	Value
Name	?
System Weight ? kg	
Weight including transportation case	? kg
Transportation size	?x?x? m
Typical operation size	? ?x?x? m
Unpack and assembly time	? min
Startup time (off to full operation)	? min
Power consumption (idle/ typical/ max)	? (60 / 80 / 90 W)
Battery endurance (idle/ normal/ heavy load)	? (10 / 5 / 4 h)
Any other interesting attribute	?
Cost	? USD

ACKNOWLEDGMENT

We would like to express our gratitude to our sponsors and advisors Alexander Ferrein, René Rütters, and Stefan Schiffer for giving us the opportunity to realize this project and for supporting us in all aspects.

We also extend our appreciation to the BDAI Institute, especially Jennifer Barry, Jiuguang Wang, and Daniel J. Gonzalez for their continuous support and maintenance of the Spot ROS 2 driver, and to everyone else who has contributed to it.

Furthermore, we would like to thank Andrei Alexeev for creating a concept for the documentation of our team.

Last but not least, we are grateful to FH Aachen, MASCOR Institute, and everyone else who has supported us in any way.

REFERENCES

[1] C. Ken, "ROS high-level concepts documentation," Open Source Robotics Foundation, Tech. Rep., 2012. [Online]. Available: http://wiki.ros.org/ROS/Higher-LevelConcepts

TABLE IV: Hardware Components List

Part	Brand & Model	Unit Price	Num.
DC/DC	Converter 24V-19V	?	1
Battery Management	Spot Battery Charging System	-	1
Micro controller	ESP32	20 €	1
Computing Unit	Intel NUC7i5BNK	670 €	1
WiFi Adapter		?	1
IMU	Xsens MTi-100	1.849 €	1
LiDAR	Velodyne VLP-16	5000 €	1
Cameras	RealSense	300 €	3
Thermal Camera	MLX90640	120 €	1
CO_2 Sensor	MHZ-19B	35 €	1
Magnetic Sensor	A3144	1 €	1
Router	RUTX50	560 €	1
Rugged Operator Laptop	Latitude 5421	1000 €	1
Batteries	Spot Explorer Battery	-	4

TABLE V: Software List

Name	Version	License	Usage
Ubuntu	22.04	open	
ROS 2	humble	BSD	
ROS 2 Spot Driver	humble	BSD	
Spot SDK	?	?	
ROS 2 Kinova Jaco Driver	humble	-	Manipulation
OpenCV	4.0+	BSD	landolt detection
yolo	v5	?	Hazmat detection
ROS 2 *hector_slam*	-	BSD	GeoTIFF map
golog++	?	GPL-3.0 license	High-Level Control

[2] 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, may 2022. [Online]. Available: https://doi.org/10.1126%2Fscirobotics.abm6074

[3] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, "A flexible and scalable slam system with full 3d motion estimation," in *Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR)*. IEEE, November 2011.

[4] V. Mataré, T. Viehmann, T. Hofmann, G. Lakemeyer, A. Ferrein, and S. Schiffer, "Portable high-level agent programming with golog++," 01 2021, pp. 218–227.

[5] V. Mataré, S. Schiffer, and A. Ferrein, "golog++: An integrative system design." in *Proceedings of the 11th Cognitive Robotics Workshop 2018 (CogRob@KR 2018), co-located with 16th International Conference on Principles of Knowledge Representation and Reasoning*, October 2018, pp. 29–36.

[6] M. Kirsch, V. Mataré, A. Ferrein, and S. Schiffer, "Integrating golog++ and ros for practical and portable high-level control," 01 2020, pp. 692–699.

[7] H. J. Levesque, R. Reiter, Y. Lespérance, F. Lin, and R. B. Scherl, "GOLOG: A logic programming language for dynamic domains," *Journal of Logic Programming*, vol. 31, no. 1–3, pp. 59–84, April-June 1997. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0743106696001215

[8] G. De Giacomo, Y. Lespérance, and H. J. Levesque, "Reasoning about concurrent execution, prioritized interrupts, and exogenous actions in the situation calculus," in *IJCAI*, vol. 97, 1997, pp. 1221–1226. [Online]. Available: http://www.dis.uniroma1.it/degiacom/papers/1997/DeLL97ijcai.pdf

[9] G. De Giacomo, Y. Lespérance, H. J. Levesque, and S. Sardina, "Indigolog: A high-level programming language for embedded reasoning agents," in *Multi-Agent Programming*. Springer, 2009, pp. 31–72.

[10] A. Ferrein, G. Steinbauer, and S. Vassos, "Action-based imperative programming with YAGI," in *Workshops at the Twenty-Sixth AAAI Conference on Artificial Intelligence*, 2012.

[11] A. Ferrein, C. Maier, C. Mühlbacher, T. Niemueller, G. Steinbauer, and S. Vassos, "Controlling logistics robots with the action-based language YAGI," in *International Conference on Intelligent Robotics and Applications*. Springer, 2016, pp. 525–537.

[12] A. Ferrein and G. Lakemeyer, "Logic-based robot control in highly dynamic domains," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 980–991, 2008.

[13] S. Schiffer, A. Ferrein, and G. Lakemeyer, "CAESAR: An intelligent domestic service robot," *Intelligent Service Robotics*, vol. 5, pp. 259–273, 2012.

[14] A. Ferrein, "Robot controllers for highly dynamic environments with real-time constraints," *KI - Künstliche Intelligenz*, vol. 24, no. 2, pp. 175–178, 2010. [Online]. Available: http://dx.doi.org/10.1007/s13218-010-0041-3

[15] S. Chitta, I. Sucan, and S. Cousins, "Moveit![ros topics]," *IEEE Robotics & Automation Magazine*, vol. 19, no. 1, pp. 18–19, 2012.

[16] A. Demidovskij, A. Tugaryov, A. Kashchikhin, A. Suvorov, Y. Tarkan, F. Mikhail, and G. Yury, "Openvino deep learning workbench: Towards analytical platform for neural networks inference optimization," *Journal of Physics: Conference Series*, vol. 1828, p. 012012, 02 2021.

[17] I. Culjak, D. Abram, T. Pribanic, H. Dzapo, and M. Cifrek, "A brief introduction to opencv," in *2012 Proceedings of the 35th International Convention MIPRO*, 2012, pp. 1725–1730.