Knowledge based control for autonomous mobile machines in mines

Niko Känsäkoski
Intelligent Robotics
VTT Technical Research Centre of Finland
Oulu, Finland
niko.kansakoski@vtt.fi

Jere Backman
Machine Intelligence
VTT Technical Research Centre of Finland
Oulu, Finland
jere.backman@vtt.fi

Tapio Heikkilä
Intelligent Robotics
VTT Technical Research Centre of Finland
Oulu, Finland
tapio.heikkila@vtt.fi

Abstract— In this paper we introduce knowledge-based cognitive control for autonomous mobile robots (AMR), targeting mining applications. Cognitive situation aware AMR control is demonstrated, relying on reasoning based on a knowledge base, which is maintained in real-time by robot perceptions. A use case is introduced with sharing perceptions of object detections via the knowledge base and with related AMR specific decision making about drivability of routes.

Keywords— Ontology, cognitive architecture, machine vision, autonomous mobile robot

I. INTRODUCTION

Robotic technologies have been entering mines, like with Autonomous Mobile Robots (AMRs) used as inspection robots [1]. In addition, robotic technologies have contributed to heavy mining machinery in semi-autonomous drill rigs, dozers, and load haul dumpers, and also to fully autonomous dump trucks [2,3]. Increasing autonomy still can improve the efficiency and working conditions of the mines, by reducing the amount of people required to work in the tunnels and enabling operations by adapting to exceptional situations, like falling rocks and broken vehicles in tunnels.

The level of autonomy can be increased by introducing cognitive capabilities to robotic machines. Using knowledge representation (KR) provides the robots and machines with cognitive skills for reasoning and perception, enabling them to autonomously perform a task, make decisions, and interact with a variety of environments ranging from static, structured, or fully observable to dynamic, unstructured, or partially observable [4].

Liu et al [5] have formulated a computational Semantic Reasoning Framework (SRF) to be composed from

- knowledge sources: provide raw data and extracted semantic knowledge,
- computational frameworks: define mathematical relationships between known concepts, and perform inference, and
- world representations: enable the robot to model its environment (objects, spaces, and agents) and behaviors (actions and tasks).

Ontologies are an essential part of knowledge bases, providing the conceptual backbones. Ontologies offer several benefits such as interoperability through shared understanding of the problem domain; formalization to make shared understanding machine-processable; and semantic representation to provide quality services in robot systems. Ontologies can be classified into upper ontologies, reference ontologies, domain ontologies, and application ontologies, where the level of concretization increases in the corresponding order. Domain ontologies like AuR [6] focus on certain realms of the real world, introducing specific detailed concepts supporting application development, like functional concepts in AuR.

Many developments have been reported using ontologies and knowledge bases to introduce cognitive capabilities for robots. Tosello et al [7] developed a proprietary Semantic Knowledge Base with descriptions of manipulation actions (grasp, push, place), objects (3D models and shapes), and environments (trajectories performed earlier), and tested it for object manipulation. Many others lay their solutions on existing upper, and domain ontologies, like Bermejo-Alonso et al [8] in developing an ontology-driven engineering methodology (ODEM) for developing self-awareness mechanisms into autonomous robots, based on a domain ontology for autonomous systems (OASys). Tenorth et al [9] developed a system to present and reason about the knowledge of detected objects by linking the object recognition output with the mapping system, relying on KnowRob domain ontology, which is based on the definitions of the OpenCyc upper ontology.

We have approached the challenges of introducing autonomy to mining machines by developing knowledge-based control for robotic machines. The environmental conditions concerning the mobility of mining tunnels are maintained in an ontological knowledge base, designed in OWL, and implemented as RDF with Jena ontology tools [10]. Systems like SANDVIK AutoMine® [11] provides both optimized route-based automation and intelligent teleoperation with operator-assisting automatic steering. We are focusing on the automatic route driving and our contribution is in introducing a system for shared, mobility related knowledge, and maintaining it by perceptions of the mobile robots in real time. Our focus

has been in application ontology and in developing a mechanism for real-time maintenance of knowledge to support robot operations. We have tested the approach by a working real-time demonstrator system, based on an AMR, with additional perception capabilities. In the following chapters we introduce our system architecture with its subsystems, and report on the first successful tests in our demonstrator. At the end we give short conclusions.

II. KNOWLEDGE-BASED CONTROL SYSTEM

A. Architecture

The knowledge-based control system consists of three main parts: a cognitive platform, a perception platform, and a control platform. In Figure 1, the system architecture and the main information flow is shown. The role of each software platform is to control one of the main functionalities of the system. The perception platform detects objects in the environment and collects information about the detected objects, the cognitive platform maintains and uses the semantic knowledge, and the control platform controls the mission execution of the robot. Similar architecture is used with the open-source ORO platform [12], where the ontology is used with services providing access to the ontology. In our case the cognitive platform implements the same functionalities.

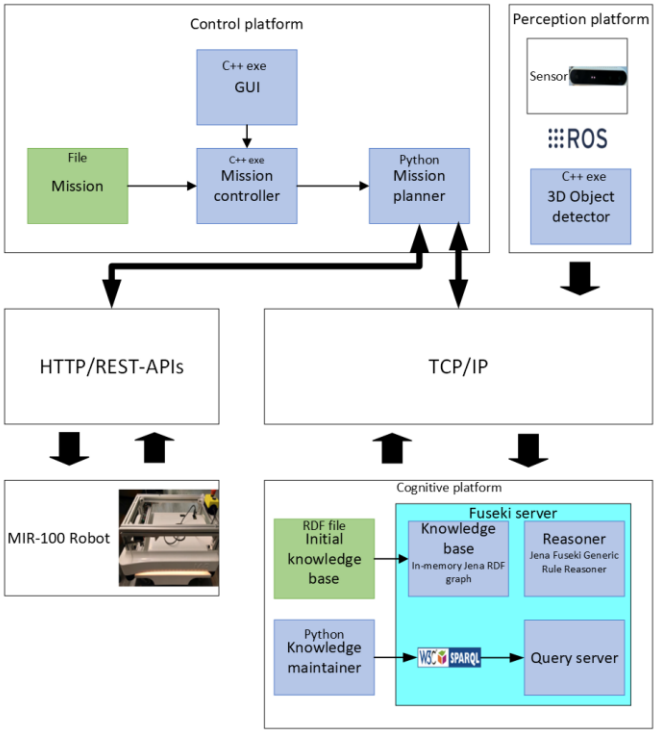

Fig. 1. System architecture.

B. Cognitive platform

The cognitive platform consists of a knowledge base, a Fuseki server with the Generic Rule Reasoner, and knowledge maintainer, a software component that transfers and maintains the knowledge in knowledge base. The Fuseki server has rules that change the content of the knowledge base based on robot perceptions. The ontology specifying the structure of the knowledge base defines maps, interest points in maps, objects, obstacles, and drivable routes. Based on the information in the knowledge base at a given time, an event can be triggered that can modify robot missions. An event relates to e.g., detection of a rock blocking the tunnel during a mission. The knowledge base is implemented as an RDF file that contains the structure, and the initial values. The knowledge base used during the experimentations is an in-memory Jena RDF graph. For robot mission control, the main concept in the ontology is the RouteSegment class. Each individual of the RouteSegment has the following relationships: hasEndpoint, which indicates which waypoints are the endpoints of this particular route segment; belongsToMap, which indicates which map the route segment belongs to, and the possible relationship; and hasSlowDownArea, which indicates whether there is some area in the route segment where the robot must drive slowly. In addition to these relationships, the individual RouteSegment has the data values length, width, height, and speedLimit. The structure of the ontology is shown in Figure 2.

The knowledge maintainer of the cognitive platform ensures that any messages to the knowledge base, either queries collecting information or messages to update the knowledge base based on robot perceptions, are in the correct format and relayed to the relevant party.

The adaptiveness of the system is dependent on the actor and situation, e.g. the type of perception, like detection of a possible obstacle object. This essentially means that the reasoner, and the rules, do not change the structure of the map, and instead they change attribute values of individuals, such as the width of a tunnel. The knowledge maintainer and mission planner are responsible for making sure that the information is used properly – because of an obstacle, a smaller robot may be able to drive through a tunnel, but a larger robot may not be able to drive through.

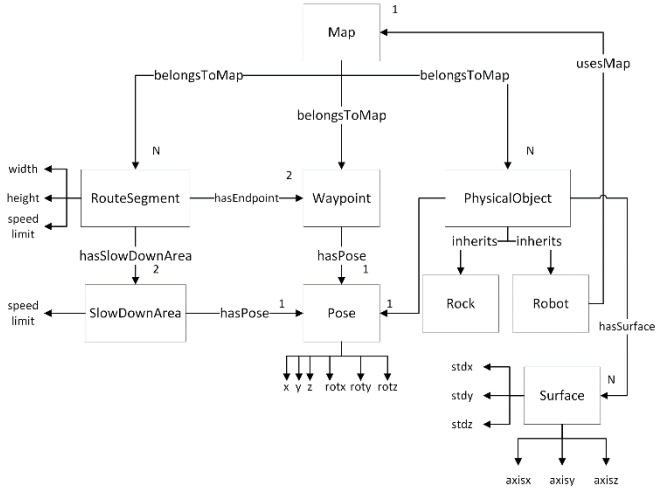

Fig. 2. Ontology structure.

C. Perception platform

The perception platform uses an Intel RealSense D415 3D camera, from which both 2D image and 3D point cloud streams are utilized. The object detection system is currently based on VTT's proprietary 3D computer vision SW [13], which filters and segments the 3D point clouds. Later on we integrate an object recognizer utilizing a convolutional neural network.

This work has been funded by Business Finland and VTT.

Detected objects are currently recognized as rocks or humans, based on the dimensions and shape of the object surfaces. From the object detection we get information about a surface, which is essentially the visible part of the object as a part of the acquired point cloud. From the point cloud, we segment surfaces and calculate the principal component analysis (PCA) for them. We get the shape and dimensional information in the form of a center point, eigenvalues, and eigenvectors, based on which we approximate the dimensions of the object. These approximated dimensions are used to determine whether the detected surface object would be representing a new object or be part of an existing object. If it is a new object, the information is sent to the knowledge maintainer that creates a SPARQL update message, which is then sent to the knowledge base. In the knowledge base, a rule is triggered that is used to update the new tunnel parameters based on the detected object information.

D. Control platform

The control platform is a combination of the mission selection, mission control, and mission planning. In mission selection, the operator selects the mission the robot should execute. The mission can be anything from moving to a specific location to a looping mission where the robot goes repeatedly between locations and performs tasks.

The control platform is used to control the AMR, in our case a MIR100, based on the mission parameters from the operator and the possible events during the mission. The control platform consists of a GUI, a mission controller, a mission planner, and mission parameter files. The GUI provides the operator with a mission list to choose a mission and mission parameter file. The mission planner plans the exact way the robot needs to act to achieve the goal of the mission. This includes path planning and adaptation to different situations alongside the cognitive platform.

The mission controller triggers and monitors the mission execution, ensuring that the mission is either executed successfully, or if something goes wrong, reporting to the operator about what happened. The mission controller essentially has a list of high-level actions that the robot is expected to perform at a given time, and the mission controller gives this information when needed to the mission planner, where the low-level actions are determined.

A key part in the mission planner is the path planner. Path planning is divided into two parts: path planning before mission execution, and reactive path planning during mission execution. The path planning is implemented as a graph search using Dijkstra's algorithm to find the optimal path. The graph used in path planning is a mobility graph created from the information stored in the knowledge base. The graph is created starting from the drivable routes, and nodes are added according to the endpoints of the RouteSegments that have been selected. We use high level graph-based path planning in addition to the MIR robot's own path planning. This is to simplify the process and allow for an easy way to essentially block certain parts of the map for a robot, since the obstacle could be in a position where the robot might have to get close to it, before noticing that there is no way to get around it.

Path planning before mission execution starts with collecting the current map from the knowledge base, then creating a graph where each RouteSegment individual is an edge in the graph, and the weight of the edge is based on the estimated time taken to travel through the RouteSegment. The created graph is practically a mobility graph, where only the RouteSegments that the current robot can travel along are added to the graph. The ability to travel through a RouteSegment is decided by finding if the current robot's dimensions are less than the width and height of the RouteSegment.

To control the robot, we used the MIR REST-API [14] to send it commands. These are simple commands, such as loading and executing a mission, changing mission parameters, pausing and resuming a mission, and creating areas where the robot acts differently, such as driving slowly. The API is used to control the robot but provides its clients with no knowledge of the overall mission properties. The mission controller itself is also unaware of the MIR REST-API functionality, and only provides an API adapter with high level commands.

The MIR robot has its own laser scanners that it uses to create a map around it and navigate using the map. Figure 3 shows the map that is used in our test environment. Blue circles with arrows indicate the waypoints. While the arrow indicates the orientation of the robot in the goal position, the waypoints are dynamic, and the correct position and orientation will be determined in the mission planner.

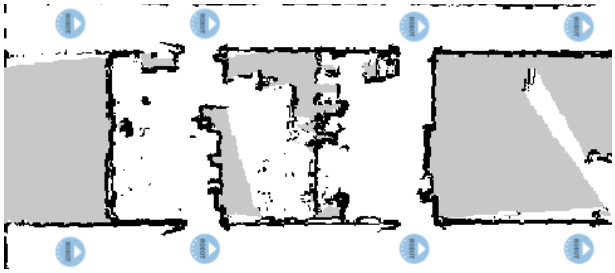

Fig. 3. Robot map and waypoints.

The robot control structure is designed to use a business as usual type of control: any deviation from the original mission should be done in a way that the mission controller does not need to react to it in any way. This means that any event is seen, and reacted to, automatically during the mission. Unless something important happens during the mission execution, for example no possible route being found, the robot being stuck, or the robot having entered an error state, the mission execution continues normally. If something abnormal happens, the mission controller will get information about this and then communicate it to the operator.

In the mine context, the information we have of the map is often changing and there may be areas in the map that are not constantly supervised. Any changes occurring in these areas are therefore only seen when some actor, AMR or human arrives at the scene and finds something. In this example, if a rock is blocking a tunnel, it is likely known only after it is reported by some means. One of the types of reactive path planning is therefore changing the path according to unexpected events during the mission. In terms of the rock, we have two possible scenarios: either the AMR can avoid the obstacle, or the

obstacle blocks the road. If the obstacle blocks the road, a new path will be calculated.

Since in the mine context multiple AMRs, and potentially multiple types of AMRs, can be used simultaneously, each of these AMRs may gain information about the surrounding environment and then send that information to the knowledge base to be used also by the other AMRs. For example, when detecting an obstacle potentially blocking a route, we only change the parameters of the tunnel so that different types of AMRs can decide whether the tunnel is drivable or not, depending on their own dimensions.

III. Demonstration use case

To simulate the mine environment and tunnels, we used two connected corridors in an office as a space where the robot can travel. We created a set of waypoints for the robot using the MIR map. These waypoints were then used to define route segments, which are the potentially travelable parts of the map. These route segments, and waypoints, were added to the Apache Jena Fuseki based knowledge base.

With a simple example we demonstrated the mission planning and control of a cognitive, situation aware mobile robot. The routes were parts of an office corridor, and the rock was a cardboard box. The following three scenarios were tested and demonstrated:

1. The route is clear the entire duration of the mission.
2. A rock is detected at some point during the mission execution, and the rock can be avoided.
3. A rock is detected at some point during the mission execution, and the rock blocks the route.

In each scenario, the operator must first select a mission. For this, we used a simple user interface with a drop-down box, where the operator can select the correct mission. The mission mainly consists of the target location(s) corresponding to ore loading and unloading areas, and possible loops between these. For each location, the mission controller asks the path planner to calculate the ideal path to the location. In the knowledge base, each of these waypoints has a MIR mission ID as a parameter that is linked to this waypoint. For example, Waypoint01 would have a MIR mission GoToWaypoint01. The GoToWaypoint01 MIR mission only has two actions: set (and reset) register, and move to Waypoint01. This move action is dynamic, and if Waypoint01 changes, so does the move action. After loading the MIR mission, the mission changes the register value to indicate the mission execution. Once the move action is done, the register is reset to a value that we check in a while loop to determine when the MIR mission is done.

In the first use case scenario we simply showed that the AMR can perform the task without interruptions, meaning that the AMR does not stop or change behavior without good cause. The second and third scenario showed that the AMR can change behavior during the mission. Especially in the third scenario it is important to show that the path planning can be triggered by an outside event.

In the second use case scenario, a rock is detected during the mission. First, for safety, the mission controller pauses the current MIR mission. While the robot is stationary, the cognitive platform and the mission planner determine what to do next. In this scenario, the decision is that we can drive around the rock, but more slowly. A speed limit in the form of a SlowDownArea instance is created in the knowledge base, and in the MIR map according to the information of the observed obstacle. Similarly, a Rock object is created in the knowledge base. Both objects are then linked to the current RouteSegment, and finally the MIR mission is resumed.

In the third use case scenario, the information about the rock is added to the knowledge base, but now the decision is to change the route. First, the current MIR mission is deleted and a new mobility graph is created. In this mobility graph, the RouteSegment that has a blocking rock is not present. Another path is then calculated if one exists. If the new path exists, the mission planner sends this information to the robot, which will continue its mission

We also tested a scenario where a rock was earlier detected and the knowledge base updated accordingly. In this case, either the robot moved slowly around the rock, or if the rock was blocking the path, the RouteSegment was not added to the mobility graph and the robot found another path to the goal.

IV. Conclusions

In this paper a simple use case for knowledge-based control with ontology-based reasoning and robot control in a mine context was introduced. We showed promising results of cognitive situation aware adaptive mobile robot control, relying on robot perceptions from the environment conditions. Future research should be done to add new different situations the robot can manage. This includes other obstacle types, human interactions with the robot and dynamic maps. A final future challenge will be to test the capabilities of the system in real underground situations with real mobile machines.

References

[1] A3 Robotics Marketing Team, How are Autonomous Mobile Robots Used to Inspect Mines? 10/22/2019. In: https://www.automate.org/blogs/how-are-autonomous-mobile-robots-used-to-inspect-mines. (06.04.2023)

[2] Automine® Equipment Automation and Teleoperation systems. In: https://www.rocktechnology.sandvik/en/products/automation/automine-equipment-and-teleoperation-systems (06.04.2023)

[3] Jörgen Andersson, Pioneering Mine Automation in South Korea. 30 January 2023. In: https://solidground.sandvik/pioneering-mine-automation-in-south-korea (06.04.2023)

[4] Manzoor, S.; Rocha, Y.G.; Joo, S.-H.; Bae, S.-H.; Kim, E.-J.; Joo, K.-J.; Kuc, T.-Y. Ontology-Based Knowledge Representation in Robotic Systems: A Survey Oriented toward Applications. Appl. Sci. 2021, 11, 4324. https://doi.org/10.3390/ app11104324

[5] Weiyu Liu, Angel Daruna, Maithili Patel, Kartik Ramachandruni, Sonia Chernova, A survey of Semantic Reasoning frameworks for robotic systems, Robotics and Autonomous Systems, Volume 159, 2023, https://doi.org/10.1016/j.robot.2022.104294.

[6] IEEE Standard for Autonomous Robotics (AuR) Ontology, IEEE Std 1872.2-2021. 47 p.

[7] Elisa Tosello, Zhengjie Fan and Enrico Pagello, A Semantic Knowledge Base for Cognitive Robotics Manipulation. I-COGROB2015 Workshop on: "Towards Intelligent Social Robots – Current Advances in Cognitive Robotics" , in Conjunction with Humanoids 2015.

[8] Bermejo-Alonso, J.; Hernández, C.; Sanz, R. Model-based engineering of autonomous systems using ontologies and metamodels. In Proceedings of

the 2016 IEEE International Symposium on Systems Engineering (ISSE), Edinburgh, UK, 21–26 July 2016; pp. 1–8.

[9] Tenorth, M.; Kunze, L.; Jain, D.; Beetz, M. Knowrob-map-knowledge-linked semantic object maps. In Proceedings of the 2010- IEEE 10th IEEE-RAS International Conference on Humanoid Robots, Nashville, TN, USA, 6–8 December 2010; pp. 430–435.

[10] Jena Ontology API. In: https://jena.apache.org/documentation/ontology (06.04.2023)

[11] THE POWER OF AUTOMATION TAKING YOU MILES FURTHER - SANDVIK AUTOMINE® FOR TRUCKS. Sandvik, 2019. In: https://www.rocktechnology.sandvik/en/campaigns/automine-for-trucks/ (15.5.2023)

[12] S. Lemaignan, R. Ros, L. Mösenlechner, R. Alami and M. Beetz, "ORO, a knowledge management platform for cognitive architectures in robotics," *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Taipei, Taiwan, 2010, pp. 3548-3553, doi: 10.1109/IROS.2010.5649547

[13] T. Seppälä, J. Saukkoriipi, T. Lohi, S. Soutukorva, T. Heikkilä and J. Koskinen, "Feature-Based Object Detection and Pose Estimation Based on 3D Cameras and CAD Models for Industrial Robot Applications," 2022 18th IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications (MESA), Taipei, Taiwan, 2022, pp. 1-5, doi: 10.1109/MESA55290.2022.10004402.

[14] MIR 2019 - 2.7.1 MIR100 REST API, MIR Robotics, 2019, 330 p.

.