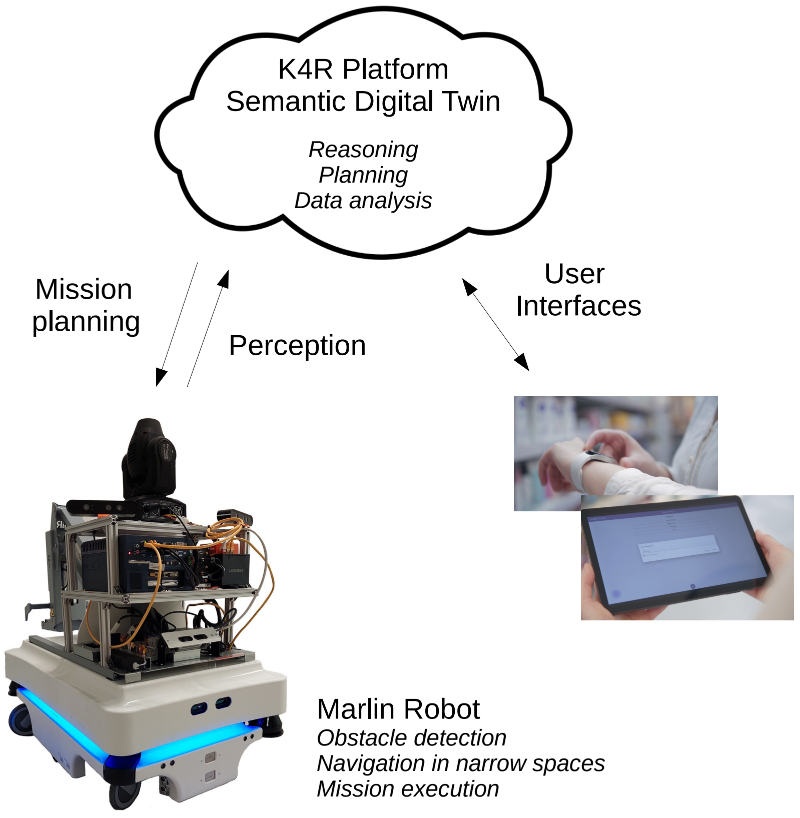
Graphical Abstract

MARLIN: A Cloud Integrated Robotic Solution to Support Intralogistics in Retail

Dennis Mronga, Andreas Bresser, Fabian Maas, Adrian Danzglock, Simon Stelter, Alina Hawkin, Hoang Giang Nguyen, Michael Beetz, Frank Kirchner



Highlights

MARLIN: A Cloud Integrated Robotic Solution to Support Intralogistics in Retail

Dennis Mronga, Andreas Bresser, Fabian Maas, Adrian Danzglock, Simon Stelter, Alina Hawkin, Hoang Giang Nguyen, Michael Beetz, Frank Kirchner

- The introduction of a semantically annotated digital twin (semDT) that stores information about the retail store and allows semantic queries
- A service robot that exchanges information with the semDT to increase its capabilities in retail intralogistics scenarios
- The evaluation of the service robot's capabilities and its synergistic relationship with the semDT in terms of obstacle detection, planning, and navigation.

MARLIN: A Cloud Integrated Robotic Solution to Support Intralogistics in Retail

Dennis Mronga[a], Andreas Bresser[a], Fabian Maas[a], Adrian Danzglock[a], Simon Stelter[c], Alina Hawkin[c], Hoang Giang Nguyen[c], Michael Beetz[c], Frank Kirchner[a,b]

[a]*Robotics Innovation Center of German Research Center for Artificial Intelligence GmbH (DFKI), Robert-Hooke-Str. 1, Bremen, 28359, Bremen, Germany*
[b]*Robotics group of the University of Bremen, Robert-Hooke-Str. 1, Bremen, 28359, Bremen, Germany*
[c]*Institute for Artificial Intelligence of the University of Bremen, Am Fallturm 1, Bremen, 28359, Bremen, Germany*

Abstract

In this paper, we present the service robot MARLIN and its integration with the K4R platform[1], a cloud system for complex AI applications in retail. At its core, this platform contains so-called semantic digital twins, a semantically annotated representation of the retail store. MARLIN continuously exchanges data with the K4R platform, improving the robot's capabilities in perception, autonomous navigation, and task planning. We exploit these capabilities in a retail intralogistics scenario, specifically by assisting store employees in stocking shelves. We demonstrate that MARLIN is able to update the digital representation of the retail store, autonomously plan and execute replenishment missions, adapt to unforeseen changes in the environment, and interact with store employees. Experiments are conducted in simulation, in a laboratory environment, and in a real store. We also describe MARLIN's capabilities in terms of obstacle detection and navigation in confined spaces.

Keywords: Service Robotics, Digital Twin, Retail, Task Planning, Knowledge Processing

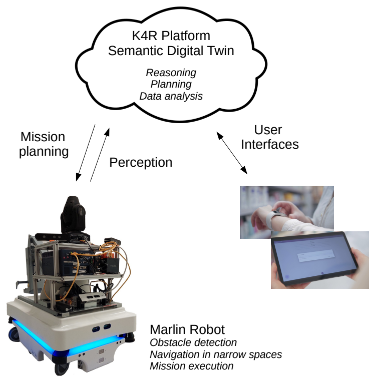
[1]K4R - Knowledge4Retail (`https://knowledge4retail.org`)

1. Motivation

In order to be competitive with respect to international online sellers, the stationary retailer has to combine its competencies in customer service and confidence with the possibilities of digitalization and robotics. In a future retail store, employees are providing advice to the customers, while robotic systems are in charge of stock-taking, replenishment and collecting scattered items. Smartphone apps direct the customer to the desired goods and answer their queries related to the assortment. Finally, the technology supports visually impaired or disabled people with shopping. Thus, AI and robotics in stationary retail have the potential to increase productivity by automating everyday tasks and improve customer experience. In order to put this vision into practice, detailed and comprehensive models of the store, the selling process and operation sequences need to be made available in machine readable form and provided to the robotic systems. The robots may benefit from this background information and improve their capabilities in terms of execution speed, autonomy, and safety.

In this paper, we describe the mobile service robot MARLIN (Mobile Autonomous Robot for intra-Logistics IN retail) and its integration with the K4R platform[2], an open-source cloud platform for AI and robotic applications in retail (see Figure 1a for an illustrative overview). At its core, this platform provides so-called semantic digital twins, a generic, machine-readable format for digital representation of retail stores. They allow the construction of realistic digital worlds and enable a variety of novel AI and robotics applications like data analysis, symbolic reasoning, and process planning. We exploit the potential of integrating MARLIN with the K4R platform and evaluate its capabilities in terms of perception, autonomous navigation, and task planning. The overall system is evaluated in a retail intralogistics scenario, namely the support of store employees in refilling shelves (see Figure 1b). The robot autonomously transports goods to the target shelf, assists employees with replenishment using a pointer unit, and interacts with them via a graphical interface. At all times, it exchanges information with the semantic digital twin, for example the position of products within the shelves, the whereabouts of store employees, as well as the location of obstacles in the corridors, which are perceived through 3D onboard sensors on the robot. This way MARLIN can safely navigate in a retail store and adapt to

[2]`https://github.com/knowledge4retail`



(a) System overview

(b) Application: Support of store employees in retail

unforeseen changes in the environment, e.g., crowded, impassable aisles.

The robot itself consists of a commercially available MiR100 platform with a mounted transport hook and additional equipment such as sensors and a pointer unit. The hook can be used to attach and transport trolleys, which are typically used in retail when stocking shelves. During transport of trolleys the robot represents a tractor-trailer system with variable and comparatively large footprint. To reliably navigate within the confined environment of a retail store, sophisticated navigation planning and execution skills are required for such a system. Thus, we also develop and evaluate an approach for autonomous navigation of articulated tractor-trailer systems, which overcomes certain limitations of classical navigation methods in confined spaces.

In summary, the main contributions of this paper are:

- We introduce MARLIN, a mobile robotic solution to support shelf intralogistics in retail stores.
- We describe a semantically annotated digital twin (semDT) that stores information about the retail store and allows semantic queries
- We demonstrate how MARLIN exchanges information with the semDT to increase its capabilities, e.g., for autonomous navigation or 3D obstacle detection.

The paper is structured as follows. In Section 2, we relate our work to the state of the art in robotics for retail and intralogistics. In Section 3, we describe the service robot MARLIN and its capabilities in terms of perception, and autonomous navigation. In Section 4, we first describe the semantic digital twin and its connection to MARLIN, followed by an explanation of our approach to autonomous task planning, as an example of the use of AI applications in the K4R platform. In Section 5, we present experimental results on obstacle classification, autonomous navigation of tractor-trailer systems, as well as task planning utilizing the semantic digital twin. Finally, Section 6 provides a short conclusion and outlook on future applications.

2. Related Work

This section provides a state-of-the-art overview on the different areas of research touched by this work, namely robotics for retail and intralogistics applications in general, the digital twin technology, autonomous navigation for tractor-trailer systems, as well as adaptive task planning.

2.1. Robotics in Retail

When related to the field of warehouse logistics the number of commercially available robotic systems for stationary retail is comparatively low. One reason is the greater complexity of the store environment and the associated challenges in terms of robotic perception, navigation, and manipulation. For example, the store might be filled with customers, which impede an autonomous robot from navigating efficiently and safely. Furthermore, the products on the shelves vary greatly in terms of size, shape and weight, which makes autonomous manipulation difficult. Finally, the stores in itself vary widely and a one-fits-all robotic solution does not exist. Therefore, only a few autonomous robots have been used efficiently and economically in stationary retail to date, although the growth is rapid, as mentioned by Bogue [1].

Positive examples of economically viable robotic applications in stationary retail exist in the area of inventory and out-of-stock detection, for example the Tally robot by simbe robotics [2], or the AdvanRobot system proposed by Morenza-Cinos et al. [3]. In the area of inventory and out-of-stock detection, the visual perception of articles is in focus. A survey of machine vision based retail product recognition systems is presented by Santra and Mukherjee [4]. The work of Kumar et al. [5] describes semi-automatic out-of-stock detection using mobile robots and virtual reality. The approach is evaluated in a mock retail store. The same application is targeted by Paolanti et al. [6]. The authors use deep convolutional neural networks to automatically detect out-of-stock events in a real store environment during working hours. In the REFILLS project[3] a mobile autonomous robot is presented to acquire models of retail stores, count the stocked products, and document their arrangements in the shelves [7].

2.2. Digital Twins for Mobile Robots in Industry and Retail

Digital twins are increasingly used in industrial manufacturing and production to represent, simulate, monitor, analyze, and optimize production processes and product lifetime cycles [8]. However, in the area of stationary retail, the use of digital twins is less widespread. The digitization of stationary retail demands for an integration of various, disparate information like product data, article localization, customer routes and attention, as well as sales figures.

A considerable challenge for autonomous robots in retail is the perception and interpretation of the store environment (not the individual products in the shelves), which is different from store to store, but might also change within same store on a daily basis due to varying placement of articles, stock level, or locations of stand-up displays. A digital twin of the store environment can be used to collect, preprocess, and provide the perceived data from multiple sources in a machine-readable format. However, establishing a digital environment from scratch without considerable manual effort is a complex problem. Paolanti et al. [9] describe a semantic object mapping based on 3D point cloud data to analyze and map a store environment. The work by Beetz et al. [7] introduces semantic store maps, which are a special form of *semantic digital twins (semDT)* as described by Kümpel et al. [10]. A semDT is a semantically enhanced virtual representation of the physical retail store, which connects symbolic knowledge with a scene graph, allowing complex reasoning tasks. The work presented in this paper builds upon the concept of the semDT and showcases a robotic application to support store employees in shelf refilling. Specifically, we use the reasoning capabilities of KnowRob [11], a knowledge processing system for robots. KnowRob organizes the digitized store data coming from disparate sensor sources and allows a robotic task planner to pose semantic queries on this data.

The general idea of using an internal representation of the environment

[3]`http://www.refills-project.eu`

for robotic high-level reasoning and planning has been widely used before in robotics. For example, the work of Blumenthal et al. [12] introduces a representation based on scene-graphs, which represent the environment as directed acyclic graph of geometric entities. They evaluated their approach on a perception task using the KUKA youBot. Another approach is called Deep State Representation (DSR) [13], which combines geometric and symbolic information within the CORTEX architecture. In contrast to these works however, our approach explicitly makes use of a cloud infrastructure, which can be fed from various sensing sources like robots, stationary sensors, and the retail stores' ERP system.

The most widely spread use-case for a robot-internal representation of the environment is planning and execution of high-level actions. Task planning in robotics is concerned with deliberately deciding on a sequence of actions to take in order to achieve a given set of goals [14]. AI planning methods have a long history [15] and have been applied to increasingly complex robotic applications, for example household [16] or manufacturing [17]. Despite the long exploration of AI-based planning there are still open research problems, e.g., how to efficiently represent knowledge from disparate sensor sources, or how to bridge the gap between symbolic and numerical action representations. Task planning for robotics is closely linked to the fields of knowledge representation and reasoning. Planning complex tasks in real-world environments requires a powerful representation of knowledge acquired from disparate sources, as well as a means to reason about this information. Both can be provided by KnowRob [11], the knowledge representation and reasoning framework which we use in our work. We connect KnowRob to the knowledge base of ROSPlan [18], a framework for AI planning in robotics, which provides various planners.

In industry and retail most robotic mission or task planning approaches are concerned with intralogistics, e.g., managing a fleet of AGVs and other agents to optimize warehouse logistics [19, 20]. Although the primary goal in this research is to plan autonomous transport tasks for a robot navigating a retail store, our planning approach allows arbitrary tasks like interacting with the store employees, manipulation of products, or updating the digital store representation.

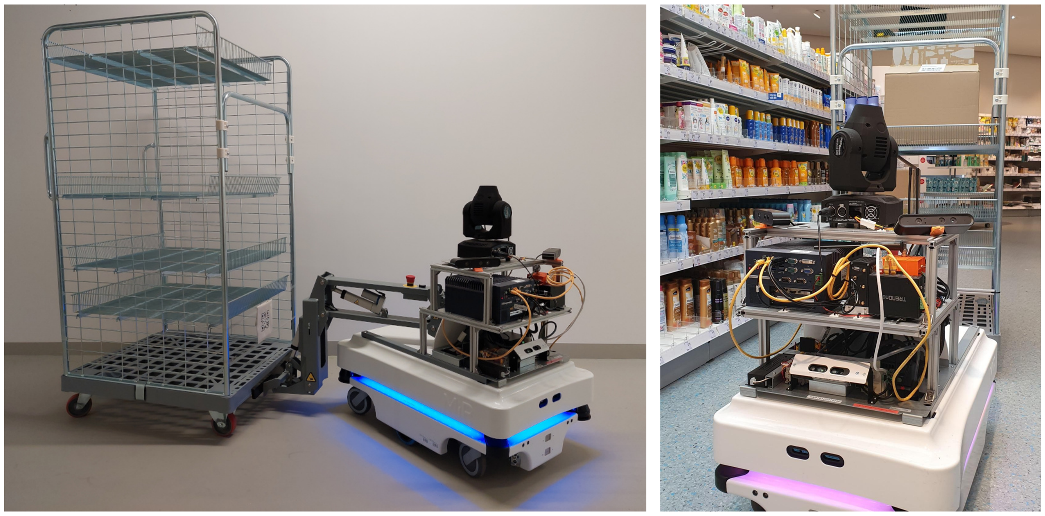


Figure 2: MARLIN: A mobile service robot for the support of store employees

3. MARLIN: A Service Robot to Support Intralogistics in Retail Stores

This section describes the service robot MARLIN and its capabilities regarding perception, autonomous navigation and interaction with the store employees.

3.1. System Description

The design of MARLIN as illustrated in Figure 2 has been led by the requirements of pilot application *Service Robotics to Support Store Employees* in the Knowledge4Retail (K4R) project. Within this project, a mobile, autonomous robot should be developed to support store employees in shelf refilling. The robot should be able to navigate efficiently and safely within a retail store. Apart from that it should (a) be integrated seamlessly with the K4R platform, a cloud solution to enable AI applications in stationary retail, (b) reuse existing structures of the stores, e.g., carts on wheels used for intra-logistics, and (c) provide user interfaces to interact with the store employees.

MARLIN consists of a commercially available MiR100 platform[4] with transport hook, equipped with an external PC, a pointer unit to guide the

[4]https://www.mobile-industrial-robots.com/solutions/robots/mir100/

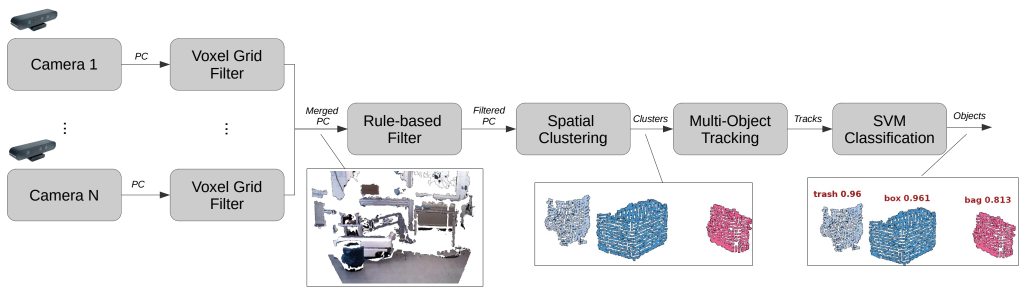


Figure 3: Sensor Processing Pipeline for obstacle detection on MARLIN. PC - Point Cloud.

store employee in the process of replenishment, as well as 4 RGB-D cameras, which provide points clouds in a 360 degree view. The system is able to autonomously pick-up, carry, and place transport carts on wheels, which are commonly used by retailers. Interaction with the user can be performed via an attached tablet, which is connected via the K4R platform as described in Section 4.

3.2. Obstacle Detection and Classification

By integrating MARLIN with the K4R platform, the robot continuously exchanges information with the retail store's semDT. Using its built-in sensors, the robot can detect obstacles, upload their position to the semDT, and reuse this information for future task planning and navigation. To handle static and dynamic obstacles, a pipeline was developed to detect and classify objects in the raw point cloud data. This pipeline, which is an extension of the multi-object tracking described by de Gea Fernández et al. [21], contains three main processing steps: (1) background removal, (2) clustering and tracking of objects, and (3) normalization & classification of tracked objects. Figure 3 shows an overview of the sensor processing pipeline.

Background Removal. The multi-object tracking approach we use to cluster and track obstacles in raw point cloud data has originally been developed for stationary robots [21]. In the original approach, the stationary background is removed from the point cloud data before clustering in order to increase performance. While this task is trivial for a stationary system, in a mobile robotics application the background filter must constantly adapt to the environment and be much faster to ensure that no artifacts remain in the filtered point cloud, even when the robot is moving fast. Thus, we use the following procedure for background removal. We first reduce the number of points in the original point cloud using a voxel grid filter. Points that are too far away or too close to the ground are also removed. Then, the point clouds from each camera are merged into a single point cloud and irrelevant areas are removed using a rule-based filter implemented in the Point Cloud Library [22]. This filter removes all 3D points corresponding to shelves and walls of the retail store using whitelist rules. The whitelist rules, defined by rectangles in the 2D map, are determined as follows: We iterate row-wise through the pixels of the 2D map until we find a free pixel. This defines the upper left corner of the first rectangle. Then we move in x- and then in y-direction until we hit a occupied pixels, which provides the bottom right corner of the rectangle. We repeat this procedure, until all the free pixels in the map are covered by rectangles. The filter then removes all 3D points whose xy coordinates are not covered by a whitelist rule. Figure 4 shows a subset of the rules defined for the 2D map of a retail store. The starting points of each rule are shown in white and the areas to be removed are shown in black. The enlarged image shows the whitelist rules, each in a different color. By using the rule-based filter, shelves and other permanent (known) obstacles in the store are filtered out of the point cloud, while dynamic obstacles remain.

Clustering and Tracking. The filtered point cloud is passed to an adapted version of the multi-object tracking described by de Gea Fernández et al. [21]. In this approach, the filtered 3D points are clustered according to their Euclidean distance. Small clusters are removed as they are assumed to result from sensor noise. In the subsequent object tracking, the remaining clusters are modeled as 3D ellipsoids, which can be used to estimate the full pose and spatial velocity of each cluster. For tracking, a track ID is assigned to each cluster. Given a set of new point clouds from the processing pipeline and a set of tracks, the tracking algorithm computes an updated set of tracks as follows. First, the states (poses and spatial velocities) of all tracks are updated using a Kalman filter. Then, each existing track is assigned to a cluster using an association measure based on the Euclidean distances of the 3D points in the cluster to the ellipsoid of the track. In this way, objects are tracked over multiple time frames. The method is robust to partial occlusion and sensor noise and allows tracking of multiple objects in cluttered scenes.

Normalization and Classification. We use obstacle classification based on 3D point clusters provided by the multi-object tracking. The idea is that permanent obstacles, e.g., display stands, can be added to the virtual environment

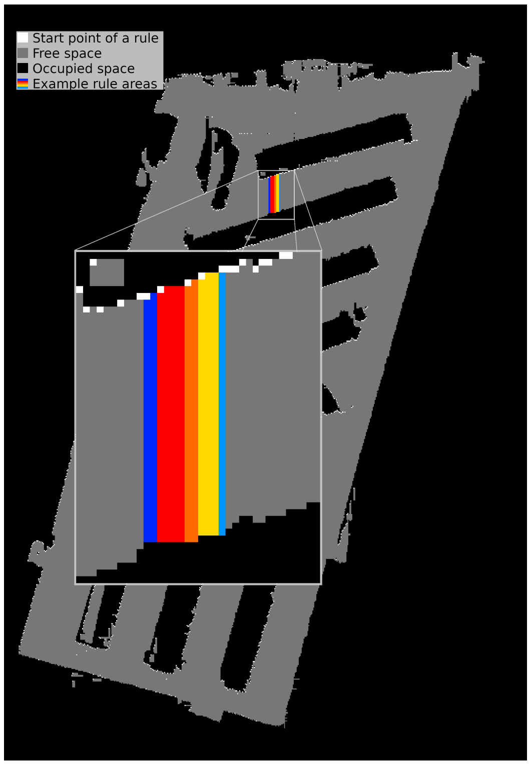
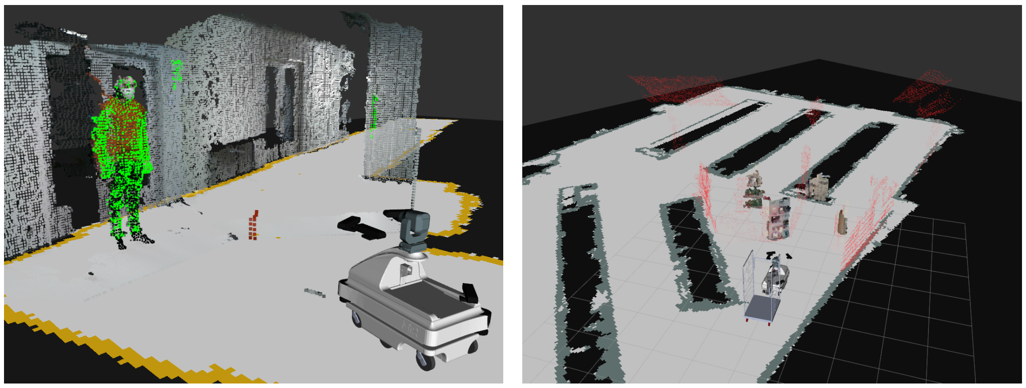


Figure 4: Visualization of the whitelist rules on a map of a retail Store



(a) Pointcloud without (normal color) and with filter(green dots)

(b) Filtered Pointcloud of an environment with multiple different obstacles (unfiltered - red, clustered points - original color)

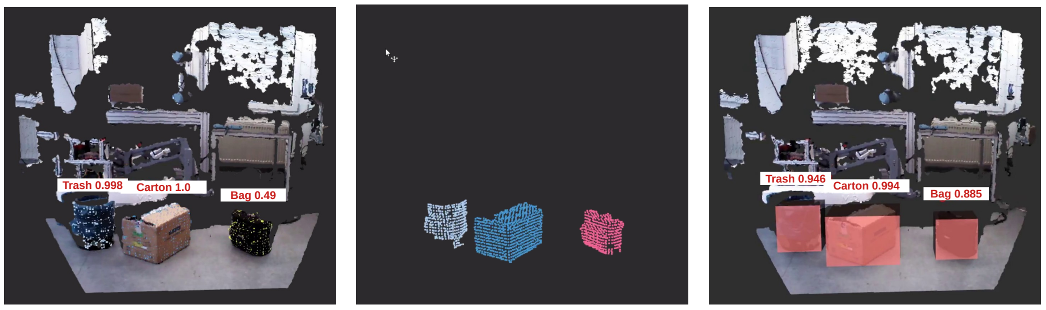
Figure 5: Results of the Pointcloud Filter.

in semDT to improve the autonomous navigation of the mobile service robot, while non-permanent obstacles such as customers or shopping carts can be ignored. The clusters are normalized to have zero mean and unit variance. For each cluster the surface normals are computed[5] and then forwarded to a prediction node as one-dimensional feature vectors, which have a fixed size $n = 10000$. If a cluster provides less features, the feature vector is artificially augmented. The prediction node computes the probability that a given cluster belongs to a given object class. We assume that when used in a store, the system must be able to cope with few different types of obstacles outside the shelves, so that most objects can be classified correctly. Nevertheless, the prediction probability for an object may be low. If it is below a certain threshold, it is still included in the semDT, with the object class ID set to "unknown". We compare different classification approaches such as random forests (RF), support vector classifier (SVC), a voting classifier (VC) consisting of a random forest and a support vector classifier, and stochastic gradient descent (SGD). We chose SVC above other approaches, because it provides good detection accuracy alongside with fast predictions on the recorded training dataset as illustrated in Table 1. The model was trained with a linear kernel. All implementations were from the Scikit Learn [23] library. To reduce computation time, multiple clusters are evaluated simultaneously. The models of the classifiers are trained offline using a relatively small dataset consisting of raw point clouds of each object class. Figure 6 shows initial results in a laboratory environment, including the original scene (left), background filtering and clustering (center), and classification (right) using the five different object classes.

3.3. Autonomous Navigation for Tractor-Trailer Systems

The proprietary navigation stack of the MiR100 platform provides navigation capabilities for indoor operation through a ROS (Robot Operating System) interface [24], both for operation with and without an attached trailer. Autonomous navigation with an attached trailer, however, requires enormous safety distances around the footprint of the robot (the recommended minimum corridor width is the total length of the system plus 50 cm safety distance). Therefore, the system is not able to navigate through narrow aisles typically found in the defined target environment, a retail store.

[5]`https://pcl.readthedocs.io/en/latest/normal_estimation.html`



(a) Original 3D scene (b) Segmented clusters (c) Result after classification

Figure 6: Clustering, tracking and classification of three different objects in a laboratory environment.

However, physically this is quite possible, as a human operator is able to remote control the system safely in the target environment. For this reason, we implement and integrate a novel approach for autonomous navigation of tractor-trailer systems, which provides improved capabilities compared to the proprietary approach.

Vehicle Kinematics. For the differential drive tractor with the trailer joint attached directly at the steering axis, a car-like controller with kinematic bicycle model can be applied, as shown in Figure 7. In our model, the trailer represents the rear axle of the model and the MiR100 represents the steering axle. The cart is attached to the robot via a transport hook, which has a passive rotational joint located at the rotation axis of the robot base. After picking up the cart, it is rigidly attached to the hook.

The trailer has one axle with fixed caster wheels and one axle with swivel caster wheels that can passively rotate around their vertical axis to follow the motion of the trailer. We define a coordinate frame in the center of the axle between the two fixed caster wheels, which coincides with the base frame of the bicycle model (positioned at $x(t), y(t)$ in Figure 7). The orientation of the coordinate frame in map coordinates is defined by the angle Θ. The distance between this frame and the center of rotation of the diff-drive tractor vehicle is the wheelbase L of the model. The angle of rotation of the tractor vehicle relative to the trailer is the effective steering angle δ.

Global Path Planning. We use the SBPL (Search-Based Planning Library [25]) Lattice Planner with a rectangular footprint tied to the fixed axis of the trailer. This planner finds the path to the requested goal pose by chain-

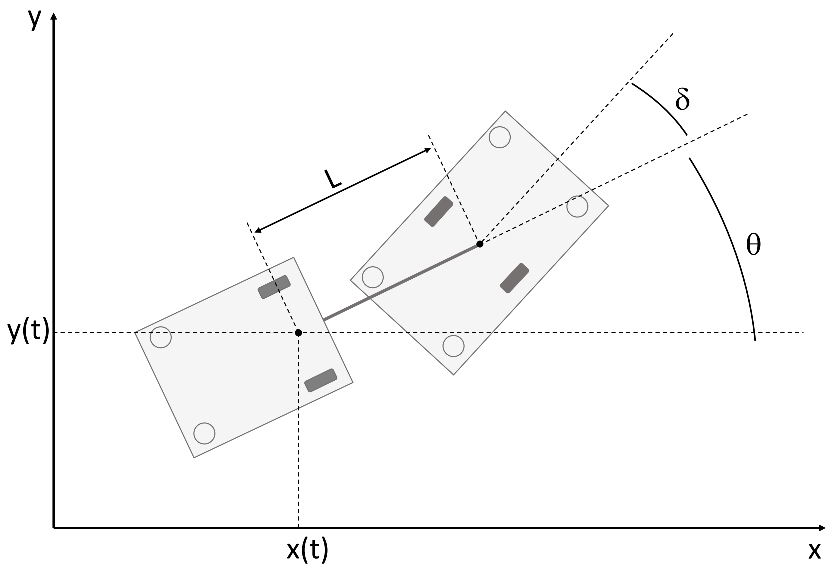


Figure 7: Model of the tractor-trailer system kinematics according to a bicycle model.

ing motion primitives with different lengths and curvatures. By limiting the maximum curvature of the motion primitives the resulting plan is expected to be feasible to be executed by the vehicle via the local planner. The path stubs are generated by a search algorithm and evaluated in an occupancy grid which covers the whole environment.

Local Path Planning. For the local path following we use the TEB (Time Elastic Band) Local Planner [26], which supports navigation for car-like vehicles. For evaluating the actual path, it tracks the obstacles around the vehicle in a smaller, local occupancy grid. It then computes the actual control commands in terms of linear and angular velocities ($\dot{x}$ and $\dot{\Theta}$) to be executed by the vehicle. For representing the system in the local path planner, we use the *Two Circles* footprint model. According to this model, the footprint of the vehicle is specified in terms of two circles that are defined by radius and offset from the vehicle's base frame, respectively (see Figure 8).

Typically, local planners (e.g., in the ROS navigation stack) provide the output command as the longitudinal speed and angular velocities (here: $\dot{x}, \dot{\Theta}$). To map these commands to the car-like vehicle model we need to add another intermediate controller stage that derives the commands for the tractor vehicle ($\dot{x}', \dot{\Theta}'$), which correspond to the desired behavior of the

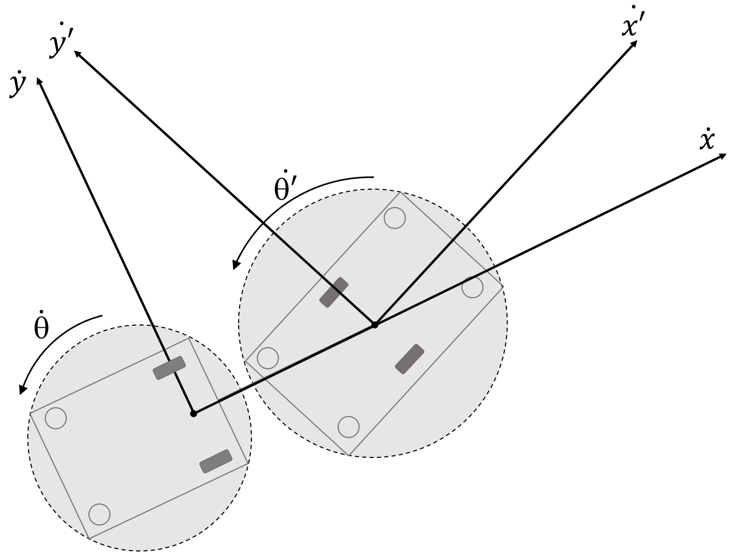


Figure 8: *Two Circles* footprint model for the TEB Local Planner (highlighted circles) and the controlled longitudinal and rotational velocities for the car-like vehicle model $(\dot{x}, \dot{\Theta})$ and the tractor vehicle $(\dot{x}', \dot{\Theta}')$. The lateral velocities $(\dot{y}, \dot{y}')$ are zero as they cannot be actuated.

tractor-trailer system. Given the vehicle's wheel base L, the steering angle δ can be obtained from the target velocities via the curvature of the requested path κ with the following formula:

$$\kappa = \dot{\Theta}/\dot{x} \tag{1}$$
$$\delta = \arctan(L\kappa) \tag{2}$$

To control the steering angle, we use a P-controller, which generates the output rotational velocity that would minimize the deviation between the current and the target steering angle:

$$\dot{\Theta}' = K_p \Delta\delta_{normalized} \tag{3}$$

To avoid issues with angular wrap around, we use the following normalization term to make sure the angle is always in the half-open interval $[-\pi, \pi)$:

$$\Delta\delta = \delta_{target} - \delta_{current} \tag{4}$$

$$\Delta\delta_{normalized} = (\Delta\delta + \pi) \bmod 2\pi - \pi \tag{5}$$

where mod is the modulo operation. The current steering angle can be retrieved from the sensor attached to the MiR hook joint. In order to prevent self-collisions, the local planner limits the maximum steering angle the controller will command. Furthermore, we specify the maximum allowed angular velocity to limit the controller output.

In analogy to the angular velocity, the longitudinal velocity is not directly applied to the tractor system. To avoid drift before the desired steering angle is regulated, we add a Gaussian activation function to limit the output longitudinal velocity $\dot{x}'$ based on the current normalized deviation of the steering angle $\Delta\delta_{normalized}$:

$$\dot{x}' = \dot{x} \exp^{-\frac{\Delta\delta^2}{\alpha^2}} \tag{6}$$

By adjusting the activation factor α we can influence the width of the admissible band of steering angle deviations. Values of around 10 or larger will practically allow the full longitudinal velocity regardless of the deviation. Smaller values gradually reduce the bandwidth of the allowed range.

4. The K4R Platform for AI Applications in Retail

The K4R platform is an open-source software platform to enable AI and robotics application for retail. At its core, it provides so called semantic digital twins, which are illustrated in the following section.

4.1. Semantic Digital Twin

A semantic Digital Twin (semDT) is a digital representation of a retail store, which connects a *scene graph* to a *semantic knowledge base* as is described in [10]. The *scene graph* contains a 3D model of the store, which is semantically annotated, and holds information like the relative location of the store shelves and products. This data can be automatically generated by a robot driving through the store and scanning all the products within the shelves.

In the context of this paper, it is assumed that robots specialized for the task are employed to create such semDTs on a nightly bases as described in [7]. This ensures that MARLIN always has updated information to work with.

The acquired information is connected to an ontology-based *semantic knowledge base*, which is based on interlinked ontologies providing further information on the products, like their ingredients and classifications, 3D models, product taxonomies, product brands or labels. This facilitates semantic reasoning on the semDT, visualization of the 3D environment in various applications, and it allows a human user or a robot to request information using semantic queries. These queries are processed by *KnowRob* [11], which is the underlying knowledge representation and reasoning framework.
The Digital Twin itself is constructed using concepts defined in the OWL (Web Ontology Language) format [27]. Internally, everything is represented as a triple, which essentially describes how *entities* are *related* to each other and which *properties* they posses. E.g., *entities*, *relations* and *properties* are the building blocks of the triples. Queries like "which shelves contain empty facings" and "where is a product of type X from the brand Y located" can then be answered in order to help a robot transport products for restocking to the correct locations, or to help guide a customer to a searched product.

```
query:
findall(Shelf,
(has_type(Facing, 'http://knowrob.org/kb/shop.owl#ProductFacingStanding'),
\+ triple(Facing, 'http://knowrob.org/kb/shop.owl#productInFacing', _ ),
triple(Facing, 'http://knowrob.org/kb/shop.owl#layerOfFacing', Layer),
triple(Shelf, soma:hasPhysicalComponent, Layer)), Shelves).

result:
Shelves=[http://knowrob.org/kb/shop.owl#Shelf_ejfydt,
         http://knowrob.org/kb/shop.owl#Shelf_jehtnm,
         http://knowrob.org/kb/shop.owl#Shelf_egvvkt,
         ...].
```

Listing 1: An example query which finds all shelves which contain empty standing facings. The result is returned as a list of unique identifiers of shelf individuals.

```
query:
subclass_of(ProductType, 'http://knowrob.org/kb/shop.owl#Product'),
has_type(Item, ProductType),
triple(Facing ,'http://knowrob.org/kb/shop.owl#productInFacing', Item),
triple(Facing, 'http://knowrob.org/kb/shop.owl#layerOfFacing', ShelfLayer),
triple(Shelf, soma:hasPhysicalComponent, ShelfLayer).

result:
Shelf=http://knowrob.org/kb/shop.owl#Shelf_pothbl,
ShelfLayer=http://knowrob.org/kb/shop.owl#ShelfLayer_yuoqzx,
```

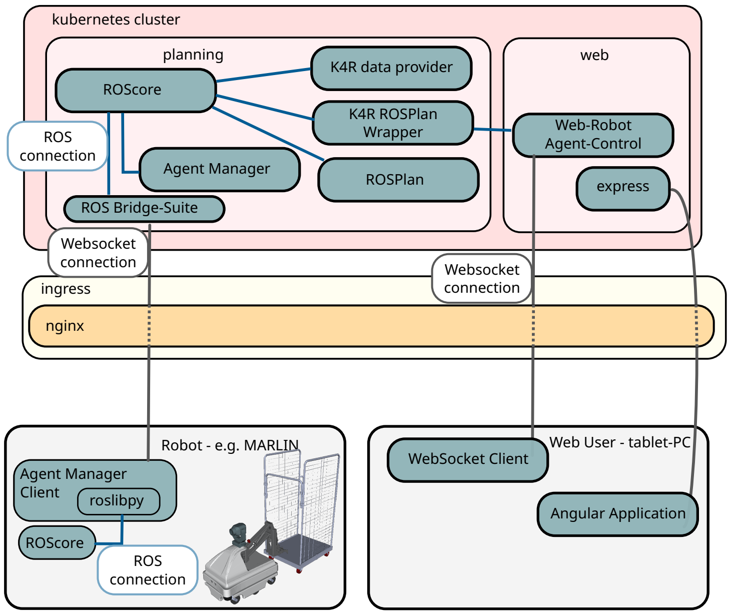


Figure 9: Architecture and interaction of the Web and Planning systems from the users tablet-pc to the robot

```
Facing=http://knowrob.org/kb/shop.owl#Facing_tjeobt,
Item=http://knowrob.org/kb/shop.owl#Product_urmvvv.
```

Listing 2: A query which returns the location of a product from a certain type (which can be substituted by a brand, for example) in terms of which shelf, layer and facing it is in.

4.2. K4R Platform Architecture and Robot Interfaces

The K4R infrastructure is developed in containers deployed as pods within a Kubernetes cluster[6]. The two main pods described here are called *planning* and *web*. The *planning* pod uses the ROS WebSuite [28] to connect to robots and other devices via WebSockets. The purpose of the *planning* pod is to use ROSPlan [18] to create and execute high-level action plans for robotic agents, control the agents, and exchange data with them. The ROSPlan knowledge base is constantly synchronized with the semDT, using the semantic reasoning capabilities of KnowRob.
The *web* pod is used for human-robot interaction. It hosts a custom Express.js TypeScript application with an Angular front-end through which the

[6]`https://kubernetes.io/`

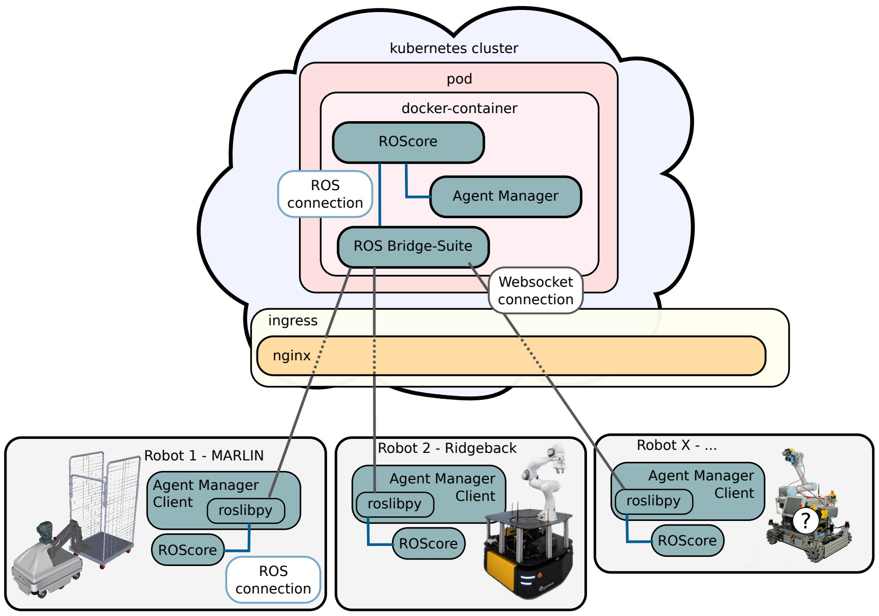


Figure 10: Cloud architecture of Agent Management

user can retrieve information about the agents, launch missions, or directly control one of the robots. The *web* pod sits behind an OAuth2 Keycloak installation to ensure secure communication over the Internet. To allow storing each agent's state, we use certificate-based authentication with tokens and ROSAuth. An overview of the architecture is shown in Figure 9. The K4R platform architecture is described in more detail in [29].

Agent Management. Robots and other agents connect to the K4R platform using the agent manager as shown in Figure 10. It registers with the planning system and provides information about the connected robot. The connection also uses the ROS WebSuite, which is commonly installed on the robotic agent. However, this requires the robot to be fully connected within the network, which is complicated to accomplish when being connected via a cloud platform. Instead of developing a custom server to handle data from the different agents, we deploy the ROS WebSuite on the K4R platform itself, which also brings benefits for robotics software developers, as they can stick to ROS topics and services.

User Interfaces. The user interface was developed as a responsive Angular web application that focuses on touch input for tablets, but can also be

used with a PC. It allows the user to select a robot, configure it, display information about the system, and start different missions, such as "attach trolley", "go to charging station", or "start refilling the shelves".
Beyond the web user interface, we are developing a smartwatch application for the task of picking. The app notifies the employee when the robot arrives at a delivery point, allowing the employee to place products from the transport cart on the shelves, and confirm when the task is complete.

Task Planning. For mission planning of MARLIN and other robots, we use the ROSPlan planning system from Cashmore et al. [30], which builds on the POPF (Partial Order Planning Forwards) planner from Coles et al. [31]. The planning framework runs on the K4R platform. Thus, it is possible to plan missions with multiple agents and resources. ROSPlan maintains it own knowledge base, which is synchronized with the semantic digital twin (and its underlying scene graph representation) via ROS topics. The planning domain is modeled in PDDL (Planning Domain Definition Language [32]).
The focus of mission planning here is on the replenishment of the shelves, as well as the related autonomous transport of goods. The goal of such a mission is that all products are autonomously delivered to their target shelves, where they are filled into the shelves by a store employee. In the corresponding domain definition, all robots, trolleys, waypoints for unloading the trolley, as well as the products and their positions must be defined. However, the desired product positions as well as the unloading points can be derived from the store geometry provided by the semDT using KnowRob. The robot autonomously attaches the trolley loaded with products, then moves to the first shelf so that a store employee can replenish it and subsequently confirm the execution on the GUI. This process is repeated until all products are unloaded. The scenario is described as a PDDL domain in Listing 3. A problem definition for this domain can be found in Listing 4.
The planner does not solve the scheduling and coordination problem, so currently only one agent can be used in a plan. It also ignores the capacity of a cart, so for this scenario we assume that all products can be loaded onto a single cart.

Listing 3: PDDL domain definition

```
(define (domain pick-and-place-domain)

(:requirements :strips :typing :disjunctive-preconditions :durative-actions)

(:types
        agent
        waypoint
```

```
                product
                trolley
        )

        (:predicates
                ; agent reached a waypoint
                (agent_at ?a - agent ?wp - waypoint)
                ; defines the order in which the waypoints are visited
                (trolley_at ?t - trolley ?wp - waypoint)
                (docked ?a - agent ?t - trolley)
                (has_trolley ?a - agent)
                (loaded_on ?t - trolley ?p - product)
                (product_at ?p - product ?wp - waypoint )
        )

    (:durative-action dock_trolley
        :parameters (?a - agent ?t - trolley ?wp - waypoint)
                :duration (= ?duration 10)
        :condition (and
                        (at start (agent_at ?a ?wp))
                        (at start (trolley_at ?t ?wp))
                )
        :effect (and
                        (at end (docked ?a ?t))
                        (at end (has_trolley ?a))
                )
        )

    (:durative-action load_on_trolley
        :parameters (?a - agent ?t - trolley ?wp - waypoint ?p - product)
                :duration (= ?duration 1)
        :condition (and
                        (at start (docked ?a ?t))
                        (at start (trolley_at ?t ?wp))
                        (at start (product_at ?p ?wp))
                )
        :effect (and
                        (at end (loaded_on ?t ?p))
                        (at end (not (product_at ?p ?wp)))
                )
        )

        ; move to waypoint, without trolley
        (:durative-action move_to_waypoint
                :parameters (?a - agent ?from ?to - waypoint)
                :duration ( = ?duration 10)
                :condition (and
                        (at start (agent_at ?a ?from))
                )
                :effect (and
                        (at start (not (has_trolley ?a)))
                        (at start (not (agent_at ?a ?from)))
                        (at end (agent_at ?a ?to)))
        )

        (:durative-action move_to_waypoint_with_trolley
                :parameters (?a - agent ?t - trolley ?from ?to - waypoint)
                :duration (= ?duration 10)
                :condition (and
                        (at start (agent_at ?a ?from))
                        (at start (trolley_at ?t ?from))
                        (at start (docked ?a ?t))
                )
                :effect (and
                        (at start (not (agent_at ?a ?from)))
                        (at end (agent_at ?a ?to))
                        (at start (not (trolley_at ?t ?from)))
                        (at end (trolley_at ?t ?to))
                )
        )

        ; confirm all products have been placed
        (:durative-action confirm
                :parameters (?t - trolley ?wp - waypoint ?pr - product)
                :duration ( = ?duration 1)
                :condition (and
                        (at start (trolley_at ?t ?wp))
                        (at start (loaded_on ?t ?pr))
                    )
                    :effect (at end (product_at ?pr ?wp))
            )
) ; end define
```

Listing 4: PDDL problem definition

```
(define (problem pick-and-place-problem)
  (:domain pick-and-place-domain)
  (:objects
    marlin - agent
    trolley0 - trolley
    prod0 - product
    start_area load_area dock_area - waypoint
    wp0 - waypoint
    )
  (:init
    ; in the beginning all agents are in the starting area
    (agent_at marlin start_area)
    ; in the beginning all trolleys are in the docking area
    (trolley_at trolley0 dock_area)
    ; in the beginning all products are in the loading area
    (product_at prod0 load_area)
    )
  (:goal
    (and
      (product_at prod0 wp0)
        )
  )
)
```

Listing 5: Basic plan from defined PDDL

```
0.000: (move_to_waypoint agent0 start_area dock_area)  [10.000]
10.001: (dock_trolley agent0 trolley0 dock_area)  [10.000]
20.002: (move_to_waypoint_with_trolley agent0 trolley0 dock_area load_area)  [10.000]
30.003: (load_on_trolley agent0 trolley0 load_area prod0)  [1.000]
30.004: (move_to_waypoint_with_trolley agent0 trolley0 load_area wp0)  [10.000]
40.005: (confirm trolley0 wp0 prod0)  [1.000]
```

5. Experimental Evaluation

In this section, we evaluate the capabilities of MARLIN in terms of obstacle detection, autonomous navigation, and task planning. Results are provided in simulation, in laboratory environment, and in a retail store.

5.1. Obstacle Detection and Classification

First, we compare different classifiers for obstacle classification to justify our choice of SVC. Next, the computational effort of the obstacle detection and classification pipeline is evaluated by measuring the computation time for the different processing steps, as shown in Figure 3. Experimental data is recorded as MARLIN navigates a retail store with various obstacles in the aisles (see Figure 5b). We use two depth cameras with a resolution of 640 x 480 pixels and evaluate the pipeline on MARLIN's onboard PC with 8 x 3.6 GHz, 32 GB RAM. Second, we evaluate the classification performance by

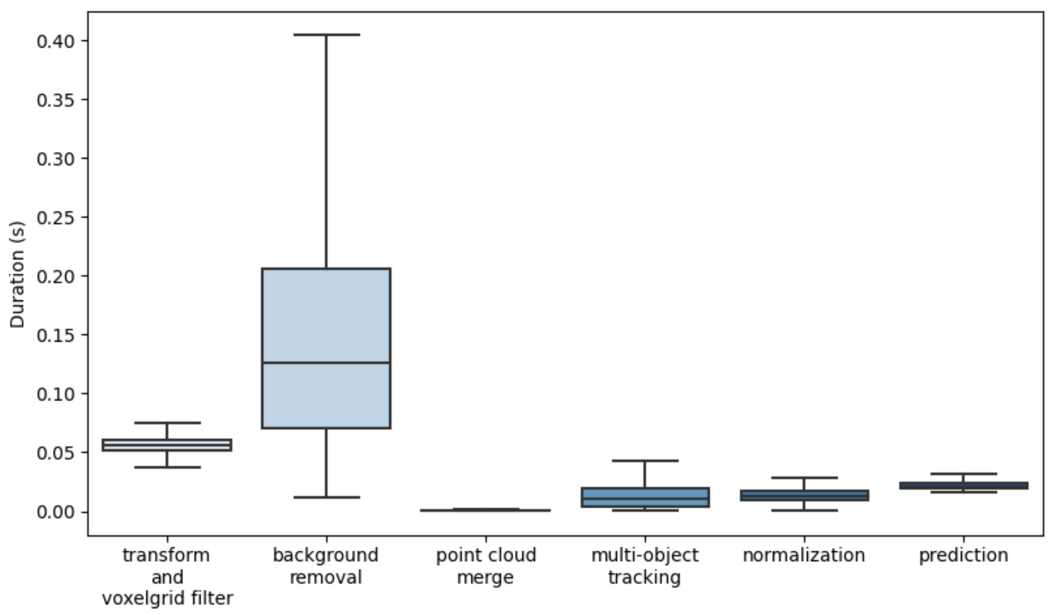


Figure 11: Computation time of the individual processing steps in the obstacle detection and classification pipeline.

measuring the prediction error rate of the classifier using training and test data obtained in a laboratory environment. We use a single depth camera with 640 x 480 pixels in this experiment.

5.1.1. Classifier Comparison

	Accuracy	Runtime [ms]
SVC	0.9127	3.8501 +/- 0.2010
RF	0.97069	45.4708 +/- 15.3428
VT	0.9610	15.6017 +/- 1.5021
SGD	0.6125	1.9475 +/- 1.8217

Table 1: Comparison of the Accuracy and the runtime (in seconds) of the different classifiers. SVC - Support Vector Classifier, RF - Random Forest, VT - Voting Classifier, SGC - Stochastic Gradient Descent

Table 1 shows the result on 4 different classifiers from the SciKit Learn ML Library.

5.1.2. Computational Efficiency

To evaluate computational performance of the obstacle detection and classification pipeline, we measure the average computation time of the individual processing steps given $n = 1000$ sample points clouds collected when navigating with the MARLIN robot in a retail store. The store environment was filled with artificial obstacles like boxes and shopping carts. Figure 11 illustrates the results. It can be seen that background removal requires the largest computation time, which is due to the large number of rules that is created for the map of the retail store (see Figure 4). The poor quality of the map, the diagonal orientation of the shelves, and the way the condition filter works, where rules can only be created parallel to the x- and y-axis, result in the creation of > 1700 rules in total. The second most time-consuming process is the transformation and voxel grid filter, which depends mainly on the size of the incoming point cloud. The computation times of the other processing steps, namely point cloud merging, tracking, normalization, and preparation, are low in comparison.

To decrease overall computation time, one could trade off accuracy versus the resolution of the original depth images, manually preprocess the 2D map to produce a lower number of rules in the background filter, or make the clustering step in the multi-object tracking more discriminating to decrease the number of processed clusters.

5.1.3. Classification Accuracy

To evaluate the performance of obstacle classification we record data in a laboratory environment similar to Figure 6. We train 5 different objects (bag, carton, hook cover, human, and thrash can) using raw point cloud data. To evaluate the error rate of the trained SVC model, we consider both the individual predictions of the model (Figure 13a) and the output prediction (Figure 13b), where ten predictions are combined into one. The validation of the trained model was performed with live data from objects of all classes shown in Figure 12. Except for the classification of the hook cover (which is sometimes split into two clusters due to its shape, leading to false classifications), the precision of the classification can be improved by considering ten classifications.

5.2. Tractor/Trailer Navigation

The capabilities of the approach for tractor-trailer navigation are first evaluated in simulation. In the next step, we reproduce the results on the real

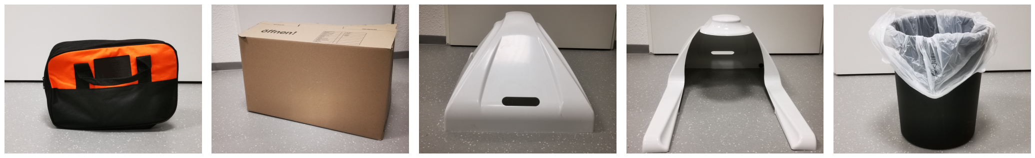
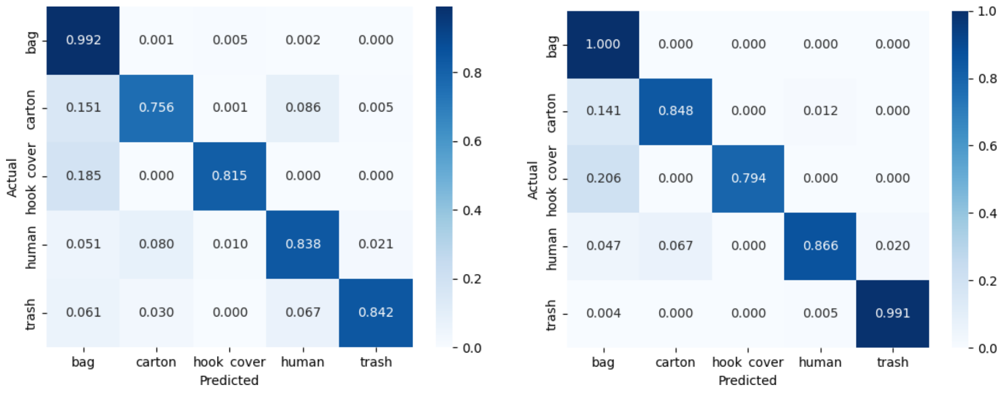

(a) Bag (b) Carton (c) Hook cover front (d) Hook cover rear (e) Trash bin

Figure 12: Objects used for validation of the obstacle classifier (without human)



(a) individual predictions (b) most probable class of a tracked cluster

Figure 13: Confusions matrices of the SVC (weighted with the probability of the predictions)

system in a similar environment and compare our approach with the capabilities of the proprietary navigation stack.

5.2.1. Evaluation in Simulation

We first create a simple simulation environment in which the robot is to navigate along a corridor. Figure 14 shows a schematic top view of the environment, with the walls in black and the empty floor space in white. The central square wall is resized in steps of 0.1 m to create different corridor widths between 1.4 m and 2.0 m. The values were selected based on the corridor widths typically found in retail stores. The corridor length is kept constant at 10 m. The target points P_0 - P_3 are located in the middle of the respective sides and are aligned so that the robot only has to move forward.

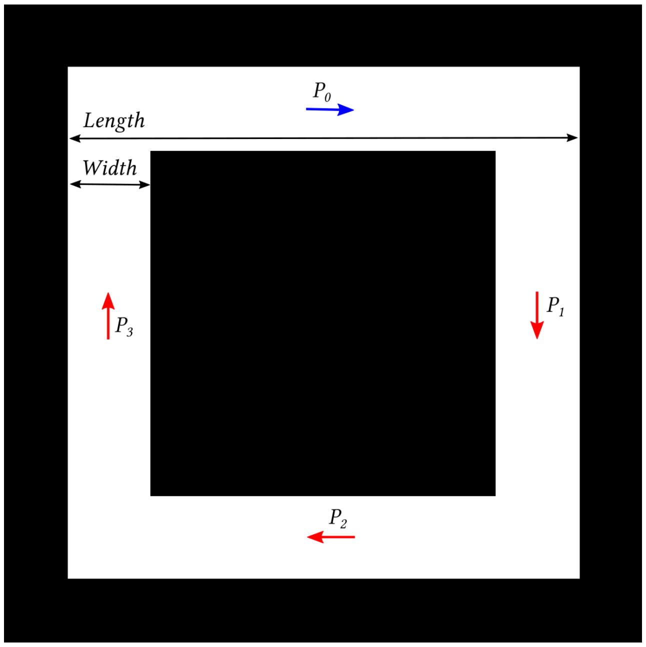


Figure 14: Schematic top view of the artificial simulation environment for evaluation of the navigation capabilities.

Procedure. Initially, the robot is placed at position P_0 and is then asked to navigate to the positions P_1, P_2, P_3, and P_0 in order. The goal tolerance of the navigation approach is selected to allow 0.5 m of translational and 0.2 rad of rotational deviation. We run the experiment 25 times for each configuration.

If the robot fails to reach a position, we register this failure. In this case, the next position is chosen nevertheless, so that all goal positions are evaluated on each run. We justify this procedure as follows: Sometimes, despite an error at one position, the robot still manages to reach the subsequent positions. This happens either because the robot drives past an unreachable position, or because it drives backwards from a corner where it was stuck before. The entire run is aborted if the laser scanner detects an obstacle in the circular safety zone around the towing vehicle that is slightly larger than its actual footprint. We record the trajectories of the tractor $x'(t), y'(t)$ and the trailer coordinate system $x(t), y(t)$ during the experiments, as well as the execution time, and whether each position has been reached or not. The goal of evaluation is to measure the success rate (in terms of the number of positions reached successfully) and the average duration for each run as a function of corridor width to get an idea of the expected performance on the real system.

Results. Figure 15 shows the trajectories of the tractor and trailer positions along the track. It can be seen that the tractor overshoots the center path at the corners, because otherwise the trailer would collide with the inner walls.

Figure 16a illustrates the number of intermediate targets reached along the route over the corridor widths. It can be seen that the approach works reliably down to a corridor width of 1.6 m. With a corridor width of 1.5 m, 92 out of 100 intermediate targets are still reached during the 25 passes. At a corridor width of 1.4 m, the success rate drops sharply. Here, the vehicle reaches the first intermediate target in only four of 25 passes.
In addition, we evaluate the time required to navigate around a single corner, which naturally increases as corridor width decreases. This is evident for corridor widths of 1.5 meters and less, while the time required to navigate the route is relatively constant for corridor widths of 1.6 meters and more (see Figure 16b).

5.2.2. Real-World Evaluation

For the evaluation on the real system, we create a similar track as in simulation. However, unlike the simulation, the corridor has only one corner with critical width, while the rest of the path is wide enough to easily return to the starting position before the next pass. Figure 17b shows a schematic overview of the setup with the walls in black and the free area in white.

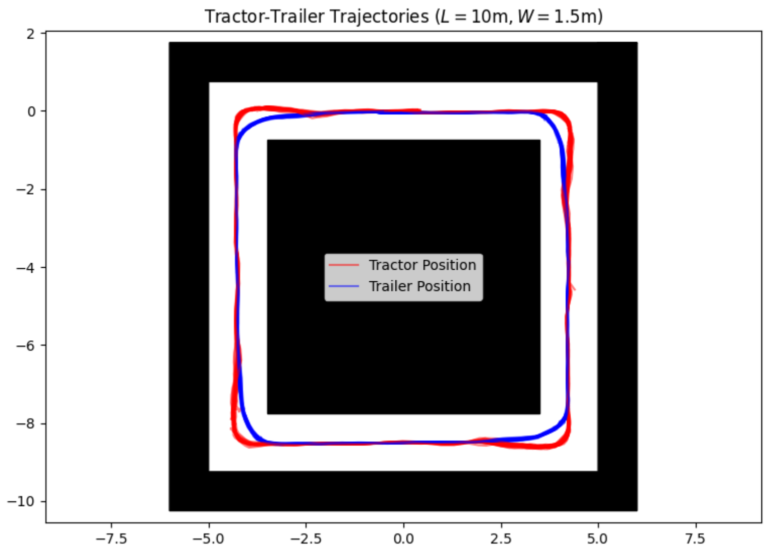
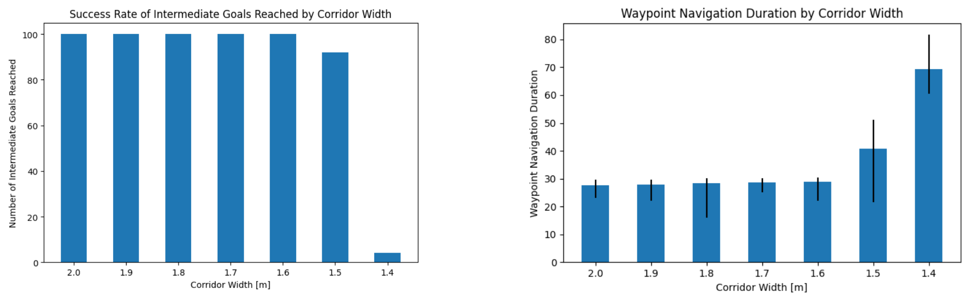


Figure 15: Visualization of tractor and trailer trajectories.

The central wall is moved to create different corridor widths. The positions P_0 and P_1 are adjusted accordingly to always be in the center of the corridor. The goal of this evaluation is to reproduce the performance observed in the simulation and compare our approach with the proprietary navigation method of the MIR robot, which is provided by the manufacturer.

Procedure. We start with a corridor width of 1.9 m and reduced it in increments of 0.1 m. The goal tolerance was set to allow 0.5 m translation and 0.2 rad rotation error. For each corridor width, we start the experiment with the robot in pose P_1 and send it to the target poses P_0 and P_1 alternately, regardless of whether the previous target was reached or an error occurs. We repeat the evaluation five times without manual intervention for both our custom tractor-trailer navigation approach and the proprietary navigation stack of the MIR robot. The run is aborted if the system hits an obstacle and is halted by its own safety stop system or if the emergency stop has to be pressed by the human operator. We record the timestamps at which the target positions are reached or, in case of a target position is not reached, we register an error.



(a) Success rate in reaching the target positions. (b) Navigation time per target

Figure 16: Results on tractor-trailer navigation in simulation.

Results. Table 2 shows the success rate of the approaches with respect different corridor widths.

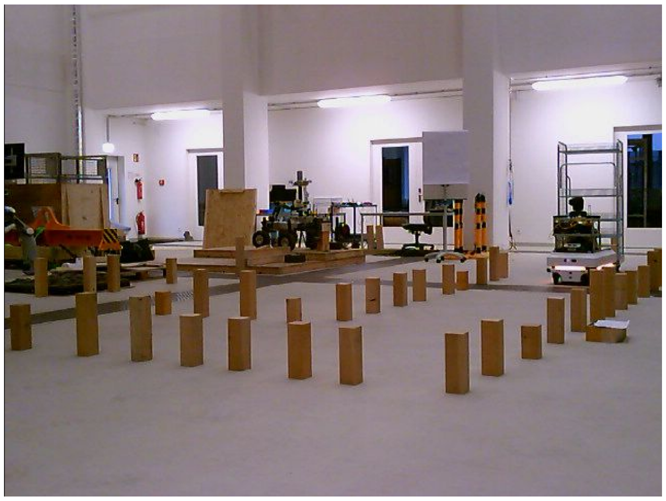
Corridor Width	Custom	Proprietary
1.9 m	4 / 5	5 / 5
1.8 m	5 / 5	5 / 5
1.7 m	5 / 5	5 / 5
1.6 m	4 / 5	5 / 5
1.5 m	5 / 5	0 / 5

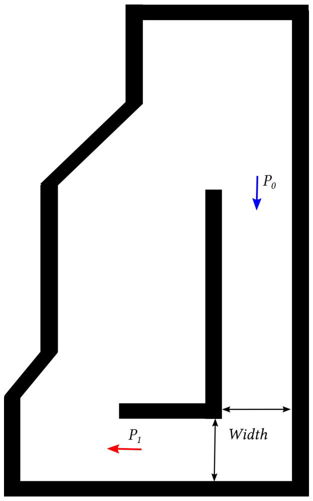
Table 2: Success rate for reaching the goal pose P_1 in real-world evaluation for custom and proprietary navigation approach.

We find that the proprietary navigation approach is able to reliably navigate the course down to a corridor width of 1.6 m. However, at a corridor width of 1.5 m, the planner is no longer able to find a path through the course. The custom navigation approach, on the other hand, is able to navigate the course up to and including a width of 1.5 m.
Also, in the custom navigation approach, we observe two error cases at corridor widths of 1.9 m and 1.6 m. Both occur at the turn just before the start position P_1, where the vehicle gets stuck and aborts the run. It then continues with the subsequent targets, which are again successfully reached.

5.2.3. Discussion

Using the proposed method for tractor-trailer navigation, the robot can navigate reliably up to a corridor width of 1.6 m in the simulated environment. Below 1.5 m, navigation time increases noticeably and the system begins to


(a) Perspective view of the actual setup. The walls are made up of wooden blocks that can be moved to adjust the corridor width.



(b) Schematic top view of the evaluation environment.

Figure 17: Real-World Evaluation.

occasionally get stuck at a corner. At even lower corridor widths, the success rate drops to near zero, which is consistent with the manufacturer's stated limitations for the system.
In the real-world robot application, the custom approach to tractor-trailer navigation was able to navigate narrower corridors than the MIR robot's proprietary approach. However, the proprietary navigation approach tends to deliver smoother and more robust execution, possibly due to better fine-tuning of navigation parameters. With manual control of the robot, even much smaller corridor widths are possible, leaving room for further optimization for the autonomous navigation.

5.3. Task Planning

In this section, we evaluate the performance of the planner with respect to the size of the store and the number of products. For this purpose, we use the domain description shown in Listing 3 and the problem definition shown in Listing 4. Note that the PDDL problem description is generated for our experiments - to keep the listing short we only show a simple scenario here with one product, one trolley and one waypoint. This definition describes the task "Replenish all items loaded on the cart" in PDDL. We assume here that the robot can approach all available unloading points without exceptions. All computation is performed on an Intel i7-8550U CPU. We use the

POPF planner from ROSPlan and evaluate different transportation scenarios. As can be seen in Figure 18 (blue bars), the planning time increases exponentially with the number of products and shelves. Thus, a completely free definition, where the agent can move to any waypoint is not practical in a realistic scenario with hundreds of products and more than one hundred shelves, and narrowing the search space is required. Based on the store layout we can pre-calculate the distances of each waypoint to each other using the Distance Matrix Service[1] developed as part of the Knowledge4Retail Platform. This service uses a simplified version of the store layout to quickly compute shortest paths between two points in the store. We can use this service to sort the list of waypoints for unloading.
If we give the planner the ordered list of waypoints, the planning time is reduced considerably, as can be seen in Figure 18 (orange bars).
If an unloading point cannot be approached (e.g., because an obstacle is in the way), and the navigation planner of the robot cannot find an alternative route, the sequence of unloading points is adjusted. In this case, the robot initially omits the next unloading point, adds it to the end of the list as an additional destination, and proceeds to the next unloading point. It is assumed that the unloading point is only temporarily blocked (e.g. due to high customer traffic in an aisle) and will be available again after some time. It will be very beneficial to provide additional information about the specific store in the knowledge base of the planner. If the obstacle can not move the robot might need to call a shop employee for help but if there is a human in the path it could ask the human politely to move or ask a shop employee for help. An option to improve the robots' behavior would be to use Reinforcement Learning to figure out where these areas are and optimize the order in which the waypoints are approached accordingly. A possible implementation for this has been investigated by [33] and shows very promising results, but they need to be validated outside of simulation and are not integrated into the K4R platform yet.

5.4. Use Case: Support of Shelf Refilling

We evaluate the shelf replenishment application in a drugstore. MARLIN starts in the charging station. We load different products from the store on the cart and manually pass the list of products to the robot. If the store

[1]`https://github.com/knowledge4retail/k4r-distance-matrix-api`

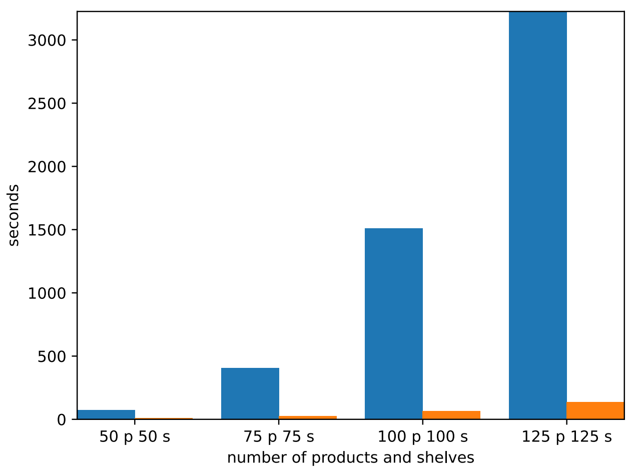
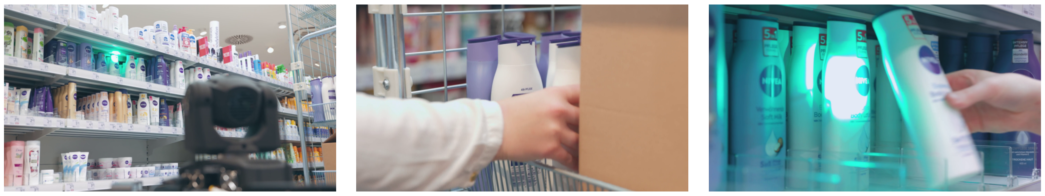


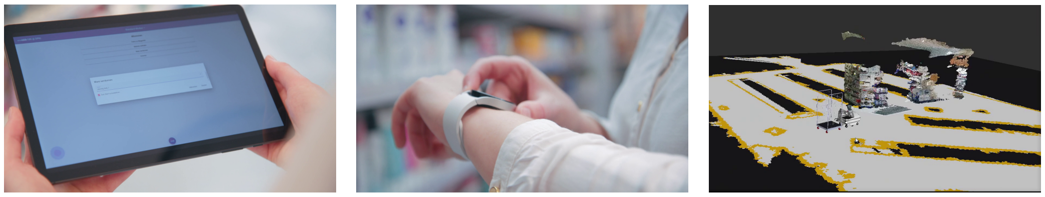
Figure 18: Planning time based on amount of products (p) and shelves (s). Blue bars - No prior knowledge on the order of the unloading points, orange bars - Order between the unloading points is pre-defined.

employee selects the replenishment mission on the graphical user interface, MARLIN picks up the cart with the products. Using the information in the semantic digital twin, the robot can infer the product locations, and, from the known store geometry, calculate feasible unloading points in front of the shelves where the products are located. Feasible means here that, firstly, the points have to be reachable for the robot navigation system. Second, if multiple products are located nearby in the same shelf they can be replenished using a single unloading point. And, third, the unloading point must be selected such that the robot does not block the target location in the shelf for the store employee. At every unloading point, the robot contacts the store employee, which receives a note on a smartwatch or tablet to start replenishment. If the task is finished, the employee must confirm this on the graphical user interface, and the robot will move to the next unloading point. This process is repeated until all products are unloaded from the cart. Afterwards, the robot will return to the charging station. During the entire process, MARLIN perceives the environment through its onboard sensors, detects and classifies obstacles, and stores permanent obstacles in the digital twin. Figure 19 shows screenshots from a representative video demonstrating the use case.

(a) MARLIN navigating narrow passages in a retail store.



(b) Support replenishment by directing the store employee to the target location of the product.



(c) User interfaces, GUI (*left*), Smartwatch (*middle*), Visualization of MARLIN's sensor data (*right*).

Figure 19: Screenshots from a video taken during evaluation in a retail store.

6. Conclusion and Outlook

In this paper, we present the MARLIN service robotic system and its integration with the K4R platform, a cloud computing solution that enables AI and robotics applications for retail. By connecting with the K4R platform, MARLIN's capabilities in perception, navigation, and mission planning are enhanced. We demonstrate that MARLIN is able to detect and classify unknown obstacles, navigate through the narrow aisles of a retail store, and plan and execute missions that assist the store employee in replenishing the shelves.
The potential of AI solutions and autonomous robotics in retail is huge. However, today the barriers to entry for retailers in such solutions are still quite high. For example, setting up a robotic system to support store employees requires a lot of expert knowledge and customized, expensive hardware. The idea of the K4R platform is to reduce these barriers to entry by providing retailers with the infrastructure and general-purpose AI functionalities. By centralizing AI approaches such as planning, reasoning, or machine learning in the K4R platform, it is possible to integrate commercially available robotic systems such as MARLIN into complex AI applications or even orchestrate entire fleets of AGVs. In this context, a possible extension of our work is to simultaneously use multiple robots for store intralogistics, for example by implementing a similar approach as in [34] or scheduling methods as described in [35]. Moreover, integrating different sensor sources, e.g., cameras to monitor the flow of customers, into the planning system can improve the reliability and speed of autonomous navigation in crowded stores. Finally, we would like to evaluate the proposed solution in a large-scale subject study (i.e., with store employees) to obtain realistic statements on usability and feasibility.

Acknowledgement

This work has been performed in the Knowledge4Retail (K4R) project, funded by the German Federal Ministry for Economic Affairs and Climate Action (grant number 01MK20001B).

References

[1] R. Bogue, Strong prospects for robots in retail, Industrial Robot: the international journal of robotics research and application 46 (2019).

doi:10.1108/IR-01-2019-0023.

[2] S. Robotics, Simbe robotics website, 2022. URL: https://www.simberobotics.com/platform/tally/.

[3] M. Morenza-Cinos, V. Casamayor-Pujol, J. Soler-Busquets, J. L. Sanz, R. Guzmán, R. Pous, Development of an RFID Inventory Robot (AdvanRobot), Springer International Publishing, Cham, 2017, pp. 387–417.

[4] B. Santra, D. P. Mukherjee, A comprehensive survey on computer vision based approaches for automatic identification of products in retail store, Image and Vision Computing 86 (2019) 45–63. URL: https://www.sciencedirect.com/science/article/pii/S0262885619300277. doi:https://doi.org/10.1016/j.imavis.2019.03.005.

[5] S. Kumar, G. Sharma, N. Kejriwal, S. Jain, M. Kamra, B. Singh, V. K. Chauhan, Remote retail monitoring and stock assessment using mobile robots, in: 2014 IEEE International Conference on Technologies for Practical Robot Applications (TePRA), 2014, pp. 1–6. doi:10.1109/TePRA.2014.6869136.

[6] M. Paolanti, L. Romeo, M. Martini, A. Mancini, E. Frontoni, P. Zingaretti, Robotic retail surveying by deep learning visual and textual data, Robotics and Autonomous Systems 118 (2019) 179–188. doi:https://doi.org/10.1016/j.robot.2019.01.021.

[7] M. Beetz, S. Stelter, D. Beßler, K. Dhanabalachandran, M. Neumann, P. Mania, A. Haidu, Robots collecting data: Modelling stores, in: Robotics for Intralogistics in Supermarkets and Retail Stores, Springer, 2022, pp. 41–64.

[8] M. Singh, E. Fuenmayor, E. P. Hinchy, Y. Qiao, N. Murray, D. Devine, Digital Twin: Origin to Future, Applied System Innovation 4 (2021). URL: https://www.mdpi.com/2571-5577/4/2/36. doi:10.3390/asi4020036.

[9] M. Paolanti, R. Pierdicca, M. Martini, F. Di Stefano, C. Morbidoni, A. Mancini, E. S. Malinverni, E. Frontoni, P. Zingaretti, Semantic 3d object maps for everyday robotic retail inspection, in: M. Cristani,

A. Prati, O. Lanz, S. Messelodi, N. Sebe (Eds.), New Trends in Image Analysis and Processing – ICIAP 2019, Springer International Publishing, Cham, 2019, pp. 263–274.

[10] M. Kümpel, C. A. Mueller, M. Beetz, Semantic Digital Twins for Retail Logistics, Springer International Publishing, Cham, 2021, pp. 129–153. URL: `https://doi.org/10.1007/978-3-030-88662-2{\_}7`. doi:`10.1007/978-3-030-88662-2_7`.

[11] M. Beetz, D. Beßler, A. Haidu, M. Pomarlan, A. K. Bozcuoğlu, G. Bartels, Know rob 2.0—a 2nd generation knowledge processing framework for cognition-enabled robotic agents, in: 2018 IEEE International Conference on Robotics and Automation (ICRA), IEEE, 2018, pp. 512–519.

[12] S. Blumenthal, H. Bruyninckx, W. Nowak, E. Prassler, A scene graph based shared 3D world model for robotic applications, Proceedings - IEEE International Conference on Robotics and Automation (2013) 453–460. doi:`10.1109/ICRA.2013.6630614`.

[13] P. Bustos, L. J. Manso, A. J. Bandera, J. P. Bandera, I. García-Varea, J. Martínez-Gómez, The CORTEX cognitive robotics architecture: Use cases, Cognitive Systems Research 55 (2019) 107–123. doi:`10.1016/j.cogsys.2019.01.003`.

[14] B. Siciliano, O. Khatib, Springer Handbook of Robotics, Springer Handbook of Robotics, Springer Berlin Heidelberg, 2008. URL: `https://books.google.de/books?id=Xpgi5gSuBxsC`.

[15] R. E. Fikes, P. E. Hart, N. J. Nilsson, Learning and executing generalized robot plans, Artificial Intelligence 3 (1972) 251–288. URL: `https://www.sciencedirect.com/science/article/pii/0004370272900513`. doi:`https://doi.org/10.1016/0004-3702(72)90051-3`.

[16] M. Beetz, D. Jain, L. Mösenlechner, M. Tenorth, Towards performing everyday manipulation activities, Robotics and Autonomous Systems 58 (2010) 1085–1095. URL: `https://www.sciencedirect.com/science/article/pii/S0921889010001119`. doi:`https://doi.org/10.1016/j.robot.2010.05.007`, hybrid Control for Autonomous Systems.

[17] J. Huckaby, S. Vassos, H. I. Christensen, Planning with a task modeling framework in manufacturing robotics, in: 2013 IEEE/RSJ International

Conference on Intelligent Robots and Systems, 2013, pp. 5787–5794. doi:10.1109/IROS.2013.6697194.

[18] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, M. Carreras, Rosplan: Planning in the robot operating system, Proceedings of the International Conference on Automated Planning and Scheduling 25 (2015) 333–341. URL: https://ojs.aaai.org/index.php/ICAPS/article/view/13699. doi:10.1609/icaps.v25i1.13699.

[19] A. Bolu, O. Korçak, Adaptive task planning for multi-robot smart warehouse, IEEE Access 9 (2021) 27346–27358. doi:10.1109/ACCESS.2021.3058190.

[20] D. Shi, Y. Tong, Z. Zhou, K. Xu, W. Tan, H. Li, Adaptive task planning for large-scale robotized warehouses, in: 2022 IEEE 38th International Conference on Data Engineering (ICDE), 2022, pp. 3327–3339. doi:10.1109/ICDE53745.2022.00314.

[21] J. de Gea Fernández, D. Mronga, M. Günther, T. Knobloch, M. Wirkus, M. Schröer, M. Trampler, S. Stiene, E. Kirchner, V. Bargsten, T. Bänziger, J. Teiwes, T. Krüger, F. Kirchner, Multimodal sensor-based whole-body control for human–robot collaboration in industrial settings, Robotics and Autonomous Systems 94 (2017) 102–119. URL: https://www.sciencedirect.com/science/article/pii/S0921889016305127. doi:https://doi.org/10.1016/j.robot.2017.04.007.

[22] R. B. Rusu, S. Cousins, 3D is here: Point Cloud Library (PCL), in: IEEE International Conference on Robotics and Automation (ICRA), Shanghai, China, 2011.

[23] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, E. Duchesnay, Scikit-learn: Machine learning in Python, Journal of Machine Learning Research 12 (2011) 2825–2830.

[24] E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, K. Konolige, The office marathon: Robust navigation in an indoor office environment, in:

2010 IEEE International Conference on Robotics and Automation, 2010, pp. 300–307. doi:10.1109/ROBOT.2010.5509725.

[25] S.-B. P. Lab, Search-based planning library (sbpl), 2023. URL: http://wiki.ros.org/sbpl.

[26] C. Roesmann, W. Feiten, T. Woesch, F. Hoffmann, T. Bertram, Trajectory modification considering dynamic constraints of autonomous robots, in: ROBOTIK 2012; 7th German Conference on Robotics, 2012, pp. 1–6.

[27] W. O. W. Group, Web ontology language (owl), 11.12.2013. URL: https://www.w3.org/OWL/.

[28] J. Mace, Rosbridge suite, 2023. URL: http://wiki.ros.org/rosbridge_suite.

[29] N. Leusmann, G. Mir, H. G. Nguyen, S. Stelter, K. Dhanabalachandran, C. Odabasi, M. Malki, A. Hawkin, F. Bazlen, M. Beetz, Retail semdt collection knowledge-base, a platform architecture, 2023 IEEE 3nd International Conference on Digital Twins and Parallel Intelligence (DTPI) (2023). Accepted for publication.

[30] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtos, M. Carreras, Rosplan: Planning in the robot operating system, in: Proceedings of the international conference on automated planning and scheduling, volume 25, 2015, pp. 333–341.

[31] A. Coles, A. Coles, M. Fox, D. Long, Forward-chaining partial-order planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, volume 20, 2010, pp. 42–49.

[32] M. Ghallab, C. Knoblock, D. Wilkins, A. Barrett, D. Christianson, M. Friedman, C. Kwok, K. Golden, S. Penberthy, D. Smith, Y. Sun, D. Weld, Pddl - the planning domain definition language (1998).

[33] Y. Liu, Using Reinforcement Learning for Multiple Waypoints Path Planning of Single Mobile Robot in the Dynamic Obstacle Environment, Master's thesis, Tallinn University of Technology, 2022. URL: https://digikogu.taltech.

ee/en/Download/143e218c-e1bb-4e08-91cb-e0d79e71f695/Stiimulpperakendaminemobiilserobotimitmepunkt.pdf.

[34] D. S. Silva Miranda, L. E. de Souza, G. Sousa Bastos, A rosplan-based multi-robot navigation system, in: 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018, pp. 248–253. doi:10.1109/LARS/SBR/WRE.2018.00053.

[35] F. Maas genannt Bermpohl, A. Bresser, M. Langosz, Experimental evaluation of agv dispatching methods in an agent-based simulation environment and a digital twin, Applied Sciences 13 (2023) 6171.