Demonstrating Adaptive Mobile Manipulation in Retail Environments

Max Spahn, Corrado Pezzato, Chadi Salmi, Rick Dekker, Cong Wang,
Christian Pek, Jens Kober, Javier Alonso-Mora, Carlos Hernández Corbato, and Martijn Wisse

***Abstract*—Although autonomous robots have great potential to boost efficiency and throughput across the whole retail chain, they are mostly being deployed in large warehouses and distribution centers. Deploying robots in stores with customers, such as supermarkets, requires substantially more development efforts since they need to safely operate around customers and reliably cope with various uncertainties and disturbances, such as misplaced products. We present our recent efforts in developing a mobile manipulator platform for order picking in realistic supermarket settings. Our robot platform uses state-of-the-art perception and planning algorithms to robustly pick items in the presence of disturbances. In particular, it successfully demonstrates adaptive decision making and rapid replanning. Our robot allows adding new products and teaching new picking maneuvers from demonstrations. We validated our robot in a recreated supermarket in our lab and in a test supermarket of a large Dutch retailer. Our results show how our robot successfully recovers from various disturbances, including misplaced products, errors in picking, and from human interaction. We summarize our lessons learned to bring autonomous robots into real retail environments with customers.**

I. Introduction

Ageing has started to impact the labour markets profoundly, and robotic labour shortage relief is becoming a necessity in all industries [1], [2]. Yet, there are still surprisingly few robots operating outside of structured environments. To bring robots successfully to human-occupied environments, the main challenge still is handling human-instilled disturbances [3], e.g., misplaced products in shelves, newly added products, blocked aisles, or interactions from humans. Such disturbances should be handled adaptively, with little development effort and short response times for natural effective interaction. Focusing on these challenges, this paper demonstrates a combination of two novel methods, one for adaptive decision making and one for rapid motion planning, embedded in a state-of-the-art integrated robot system.

The demonstrator is a mobile manipulator for order picking in realistic supermarkets, see Fig. 1. The increase in online shopping induces an equal decrease in store visits. Especially in dense urban areas, these costly but conveniently located stores could have a dual use as distribution centres for flash delivery of online orders [4]. This requires the order picking to occur during opening hours, amongst store customers. We have taken this as our demonstrator scenario, but the methods are generally applicable for any scenario involving human-disturbed mobile manipulation tasks. In such scenarios, we assume that humans can block the robot's path, physically push or hold the manipulator, and move the pick-able items, before, during, or after a pick. Even if items are taken away from the robot's suction gripper, it should recover by picking another item of the same product.

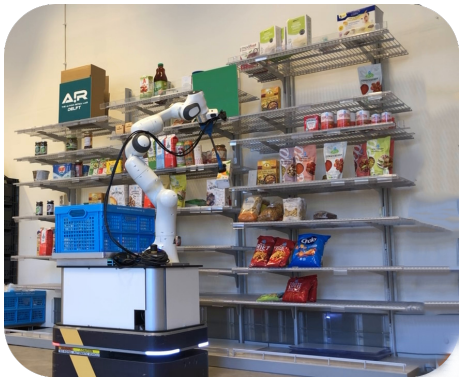


Fig. 1: We validated our mobile manipulator in our AIRLab lab environment (shown here) and a realistic supermarket environment of a large Dutch retailer.

This work presents our mobile manipulation solution in human-shared environments that aims at being adaptive and fault-tolerant. We focus on three main questions that are most relevant when deploying robots in supermarkets: How can we

1) generate *safe* and *robust* trajectories for manipulation?
2) ensure *fault-tolerant* task planning and execution?
3) easily *adapt* the robot to pick new products?

Our solution addresses these questions with specific decisions on the robot's capabilities for decision-making, trajectory generation, and perception. Our contributions are thus an integrated system with adaptive decision-making, fast motion planning and easy-to-use teaching from demonstration.

II. Related Work

Robotic mobile manipulation stands as a dynamic and expansive field of research, spurred by diverse potential applications and further fueled by prestigious international competitions, such as DARPA's Robotics Challenge [5], RoboCup@Home [6], the Amazon Picking Challenge [7], and

This research was supported by Ahold Delhaize. All content represents the opinion of the author(s), which is not necessarily shared or endorsed by their respective employers and/or sponsors.

All authors are with the Department of Cognitive Robotics, Delft University of Technology, 2628 CD, Delft, The Netherlands {m.spahn, c.pezzato, c.salmi, c.wang-17, c.pek, j.kober, j.alonsomora, c.h.corbato, m.wisse }@tudelft.nl, h.dekker@student.tudelft.nl

RoboCup@Work [8]. These competitions are tailored to address distinct challenges and performance criteria. Numerous research projects have yielded a significant number of various mobile manipulation platforms. For an exhaustive overview of wheeled mobile manipulation systems and the associated challenges, readers can refer to [9], [10]. In contrast to the aforementioned platforms, we focus on combining commercial off-the-shelf components with little to no modifications to address the specific application.

A supermarket mobile manipulator has been presented in [11] with a special focus on metrics in real-world settings and quantitative field experiments. Similar long-term fetch and carry experiments, yet in different environments, were carried out by Domel et al. [12] in a factory environment and by Stibinger et al. [13] in an outdoor competition to pick up and place simulated construction materials. Instead of relying on a Model Predictive Control formulation, such as [14] for motion planning and control, we deploy a reactive trajectory generation method [15], and a task planning and execution approach that is adaptive in the presence of disturbances. Additionally, we use learning from demonstration to easily teach new products. This approach seems extendable to very different tasks in the long run.

Object picking with manipulators is a well-studied problem. Early approaches rely on engineered components and split detection and grasping into separate tasks. An adapation of STOMP [16] for mobile manipulation was presented in [17]. Here, motions of base and arm are sequenced. Manipulation tasks, including item retrieval, can also be approached from data-driven perspectives. Deep reinforcement learning was used to achieve object picking with a mobile manipulator in non-cluttered environments [18]. In contrast, learning from teleoperation data showed impressive results for dexterous manipulation tasks [19]. The work was extended to mobile manipulation in [20]. While the results are impressive, each task is trained individually, and no safety statements can be made. In contrast, our work relies on engineered components enhanced with learning-from-demonstration techniques to achieve safe and robust mobile manipulation in a supermarket environment.

III. System Overview

We briefly introduce the considered supermarket setting and detail our order-picking pipeline and system components.

A. Considered supermarket setting

Modern supermarkets are characterized by a large range of products, around 100,000 different products per store. Operators usually have access to detailed information of all those products, including mass, geometry, and shelf location in the store. In our demonstration, we assume that the robot can access this database to inform its decision. The large variety of products usually requires specialized grasping strategies per category, e.g., grasping tomatoes is different from grasping a large soft-drink bottle. We focus on a subset of products that can be picked with the suction gripper of our robot (see

Fig. 2: Examples of different products considered in this work.

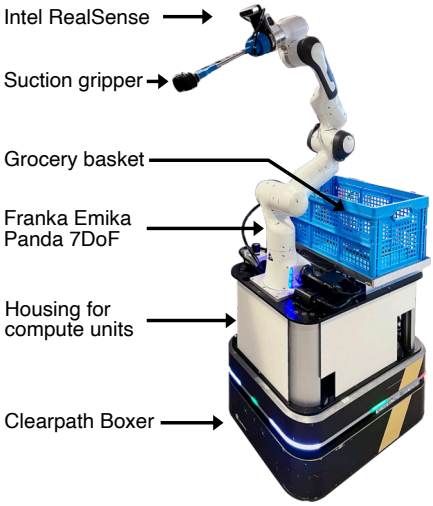


Fig. 3: Overview of our robot's hardware.

hardware design in Sec. III-B), e.g., cans, milk boxes, bottles, or bags of crisps (see Fig. 2). The masses of products range from 100g (instant food mixes) to 1.5 kg (soft-drink bottle) and the size are between 10 cm (cans) and 30 cm (soft-drink bottle). We assume that products to pick are visible, accessible from the shelf's front and in the robot's workspace.

For in-store picking, we focus on picking products during opening-hours and favor reliability over execution speed.

B. Hardware

Our mobile manipulator platform is comprised of various hardware components.

a) Robot: The mobile manipulator is composed of two robots, see Fig. 3. The moving base is a Clearpath Boxer, differential drive wheeled-robot that can achieve similar speeds to humans while having a relatively small footprint. The robotic arm is a Franka Emika Panda, a serial manipulator with seven degrees of freedom, equipped with torque sensors in every joint that can achieve high accuarcy while being safe to work around, see Fig. 1. The attached gripper is a custom 3D-printed suction gripper with two suction cups powered by a industrial vacuum pump.

b) Perception: The base uses a 270 degree Lidar sensor to localize itself and detect dynamic obstacles and humans in the environment. We mounted a Realsense D435 RGBD camera on the wrist link of the arm and use it to detect products and perform visual servoing during picking.

c) Compute Units: We use a total of four compute units to distribute the computational load of individual software

components. The first compute unit is the Franka Control Interface controlling the arm. The base's compute unit performs self-localization and collision avoidance for the base. The central compute unit is an Intel NUC with an Intel i7 10th generation CPU, running all planning components and the user interface for placing orders. A Dell Alienware laptop mounted on the robot with an RTX 3070 TI GPU runs the perception components. Our two computers are running the Robot Operating System (ROS) and communicate via a network switch.

C. Order-picking overview

The high-level overview of our order-picking system is illustrated in Fig. 4. Customers first place an order via the order placement website. The robot processes the order into a task assignment. For each item, it navigates to the item's shelf, locates it, picks and places it in the basket. When the order is completed, the customer can pick up the order from the robot.

D. System components

We used the order-picking sequence in Sec. III-C to guide our system development, while focusing on adaptiveness to recover from failure and inaccuracies in perception. In the following, we outline the main system components that are visualized in Fig. 5 and can be grouped in: (a) task planner, (b) motion planners, (c) low-level controllers, and (d) perception.

After receiving the customer order, the task planner (see Sec. IV) determines the order of picking products by minimizing the robot's travelled distance. The task planner uses a combination of a behavior tree and symbolic state information, such as the robot is holding a product or has arrived at the desired position, with an adaptive inference method to determine the best next symbolic action to execute. We define a *symbolic action* as an elementary robot behavior. Symbolic actions can be as simple as greeting the customer or as complex as picking an item. We use a set of five robot symbolic actions: picking items, placing items, looking for items, localizing the robot, and navigating the robot. Each symbolic action is realized by the motion planning and control components (see Sec. V). Motion planning is decomposed into path planning and online trajectory optimization for the base and reactive trajectory generation for the arm, augmented with a pseudo-prismatic joint on the base, see Sec. V. Lastly, the perception component (see Sec. VI) takes care of item detection and classification and provides item poses to the planning components.

IV. Task Planning and Execution

Once the customer submits an order, the task planner creates a plan to collect the items in the order throughout the store and return the shopping basket to the delivery location.

Our decision-making approach to creating these plans and executing them is designed explicitly with failure recovery in mind. It consists of 1) offline plans that leverage the known task structure, and 2) online planning to adapt the action sequence to unforeseen disturbances, following the Active Inference approach in [21].

Active Inference, a neuroscientific paradigm [22], formulates all perception and decision-making in the brain as Bayesian inference, combining prior predictions with novel sensory data. For decision-making, the "prior predictions" are rather *prior preferences*, i.e., desired states, and the probabilistic Bayesian inference is used to determine which symbolic actions have the highest probability of reaching that desired state. In our solution, a sequence of desired states for a task is planned offline and encoded in a Behavior Tree (BT). At runtime, the current desired state (or symbolic goal) is sent to the online active inference planner that computes a symbolic action sequence to transition from the current state to the desired one.

A. Offline planning

The structure of the task is modelled offline as a plan to be executed using the template BT shown in Fig. 6 and expanded by Fig. 7, making the robot try to pick every product up to N times, and then deliver the groceries to the delivery location. It specifies an initial welcome to the customer, a "Product Subtree" slot, an active inference node that sets the final task sub-goal of being at the delivery location for the online active inference planner, and a closing message to the customer. For each product, a sub-tree as in Fig. 7 is created automatically. Following the order list, every sub-tree for each product is inserted in the overall Behavior Tree (BT) structure from Fig. 6, as part of the sequence.

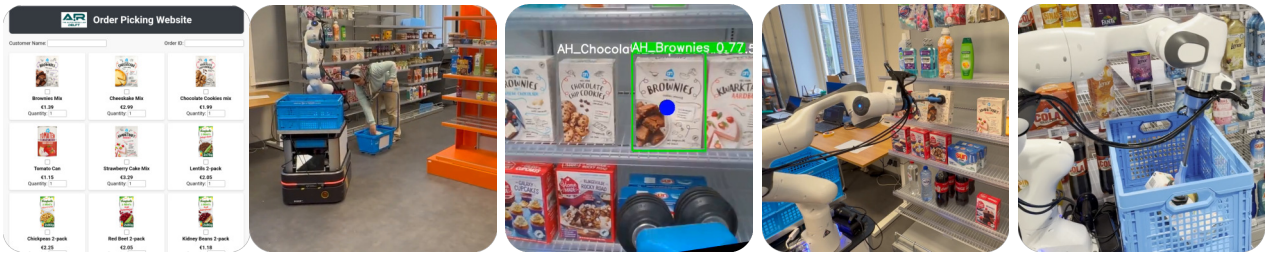
B. Online planning with Active Inference

The active inference planner (AIP) takes the task sub-goals `isPlaced` and `isAt` as desired item states being placed in the robot's basket and the robot being at the delivery location, and computes a symbolic action plan based on the robot's symbolic actions to achieves those states. Our planner uses discrete active inference, which relies on a generative model that contains beliefs about future states and symbolic action plans, where plans that lead to preferred states are more likely. The preferred sequence of symbolic actions is the one with the highest probability of achieving desired states.

Our active inference planner rests on the tuple $(\mathcal{O}, \mathcal{S}, \mathcal{A})$. This is composed of a finite set of observations $\mathcal{O}$, a finite set of symbolic states $\mathcal{S}$, and a finite set of symbolic actions $\mathcal{A}$ that correspond to the robot's symbolic actions.

The continuous state of the world $x \in \mathcal{X}$ is discretized through a symbolic observer into boolean variables about the relevant states of the world, e.g., item held by the gripper. These discrete observations o are used to build a probabilistic belief about the symbolic current state, described in Table I.

The AIP computes the posterior distribution over p plans $\boldsymbol{\pi}$ through free-energy minimization [21]. The symbolic action to be executed by a robot in the next time step is the first



(a) Customer order (b) Navigate to shelf (c) Locate item (d) Pick item (e) Place item

Fig. 4: Overview of ideal flow of symbolic actions to complete an order.

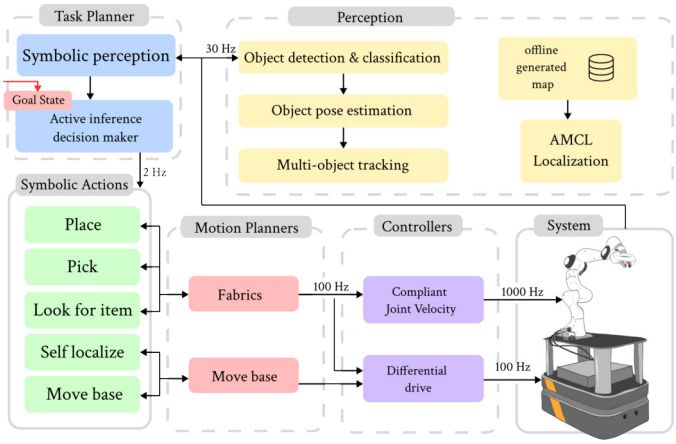


Fig. 5: Overview of system components.

symbolic action of the most likely plan, denoted with $\pi_{\zeta,0}$:

$$\zeta = \max(\underbrace{[\boldsymbol{\pi}_1, \boldsymbol{\pi}_2, ..., \boldsymbol{\pi}_p]}_{\boldsymbol{\pi}^\top}), \; a_{\tau=0} = \pi_{\zeta,0}. \tag{1}$$

The combination of offline plans modelled as BT's and online planning using active inference facilitates responsive symbolic action selection for long-term tasks within partially observable and dynamic environments, which is particularly crucial in addressing disturbances in retail settings.

TABLE I: Notation for belief states. s is the probabilistic belief state and l is the corresponding one-hot encoding

Belief State $\in (0, 1)$	**Description**
$s^{(at)}$	Belief about being at the goal location
$s^{(loc)}$	Belief about being self-localized
$s^{(reach)}$	Belief about reachability of an object
$s^{(hold)}$	Belief about holding an object
$s^{(vis)}$	Belief about an object being in sight
$s^{(place)}$	Belief about an object being placed at a location
Boolean State $\in [0, 1]$	**Common Name**
$l^{(at)}$	`isAt(goal/obj)`
$l^{(loc)}$	`isLocalized`
$l^{(reach)}$	`isReachable(obj)`
$l^{(hold)}$	`isHolding(obj)`
$l^{(vis)}$	`isVisible(obj)`
$l^{(place)}$	`isPlaced(obj)`

This approach offers the advantage of not having to account for every conceivable contingency and recovery behavior within a BT, and at the same time allows for continuous online planning. This effectively minimizes computational complexity, enabling the development of a robot capable of adhering to predefined routines while also adapting locally to unforeseen events through real-time online planning with active inference.

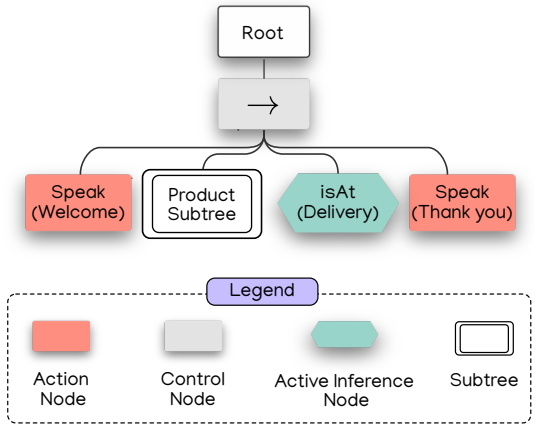


Fig. 6: Overall BT structure. The symbolic action `Speak` interfaces with the voice module to produce a suitable message for the customers (see Sec. VII).

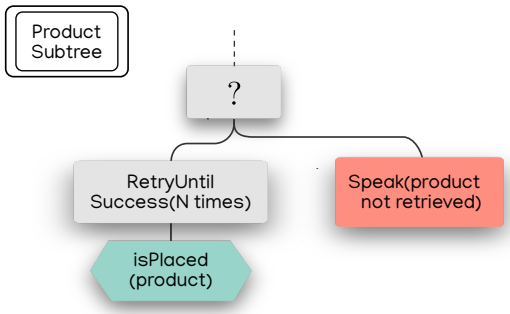


Fig. 7: Sub-tree structure for placing an item in the basket. The active inference node sets a prior over the state `isPlaced` for a product, triggering the online decision-making. The symbolic action `Speak` produces a voice message to explain the failure in case one happens (see Sec. VII).

TABLE II: Notation for symbolic actions

Symbolic Actions	Preconditions	Postconditions
`selfLoc()`	-	`isLocalized`
`moveTo(goal/obj)`	`isLocalized`	`isAt(goal)/ isReachable(obj)`
`pick(obj)`	`isReachable(obj)` `!isHolding` `isVisible(obj)`	`isHolding(obj)`
`place(obj)`	`isHolding`	`isPlaced(obj)` `!isHolding(obj)`
`look(obj)`	-	`isVisible(obj)`

V. Trajectory generation

This section outlines the various approaches we employ for real time trajectory generation (Sec. V-A, Sec. V-B). Furthermore, we explain how the symbolic actions introduced in Fig. 5, looking for products (Sec. V-C), pick (Sec. V-D) and place (Sec. V-E) are realized and we explain how they use the trajectory generation approaches.

A. Navigation of the base

To navigate the mobile base in the store, we employ the ROS MoveBase framework, configured with A* as the global and Timed-Elastic-Bands as the local planner [23]. We record an environment map including static obstacles prior to deployment. Additionally, we manually define keep-out areas to prevent the robot from going into unsafe or crowded areas, such as checkout zones. The local planner uses the environment map and online lidar sensor information for avoiding collisions with static and moving obstacles, such as humans.

B. Motion of the arm

To generate the arm motions (and base during picking), we employ Optimization Fabrics (fabrics). Fabrics is based on behavior composition, defined as differential equations in manifolds, which enables safe real-time planning at high frequencies [15], [24], [25]. Since fabrics is a local, reactive trajectory generation method, we require global guidance to perform complex symbolic actions, such as product picking or placing. The global guidance for fabrics consists of a sequence of local goals, where we only continue to the next goal if the previous goal has been reached. In Sec. VII-C we show how this sequence of goals, i.e., reference trajectory, can be obtained by human teaching. In the following, we briefly explain how fabrics works and how to use it.

1) Method: Requirements such as collision avoidance, joint-limit avoidance or self-collision avoidance are referred to in fabrics as behavioral components. Each component is represented as a differential equation of the form $\boldsymbol{M}(\boldsymbol{x},\dot{\boldsymbol{x}})\ddot{\boldsymbol{x}} + \boldsymbol{f}(\boldsymbol{x},\dot{\boldsymbol{x}}) = \boldsymbol{0}$ on an appropriate manifold $\mathcal{X}$ of the configuration space $\mathcal{Q}$, where $\boldsymbol{M}$ and $\boldsymbol{f}$ are the importance metric and the forcing term respectively, and $\boldsymbol{x}$, $\dot{\boldsymbol{x}}$ and $\ddot{\boldsymbol{x}}$ are the state, e.g., full configuration of the robot or end effector pose, and its derivatives in the manifold $\mathcal{X}$. By respecting simple construction rules, each behavioral component can be ensured to be an optimization fabric, i.e., a dynamic system that is stable by construction. All components can then be combined in the robot's configuration space by applying the operations of *pull-back* and *summation*. In the configuration space, we obtain one optimization fabrics of the form $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f} = \boldsymbol{0}$, where $\ddot{\boldsymbol{q}}$ is the second derivative of the configuration and $\boldsymbol{M}$ and $\boldsymbol{f}$ the resulting importance metric and forcing term, respectively. We define goal states of the robotic arm as a set of constraints $\mathcal{S}_c = \{\mathcal{C}_1, \mathcal{C}_i, \mathcal{C}_n\}$ where each constraint $\mathcal{C}$ is defined by the tuple $\mathcal{C} = (\text{fk}_\text{p}, \text{fk}_\text{c}, \boldsymbol{x})$. Here, fk_p is the forward kinematics to the parent link, fk_c the forward kinematics to the child link, and $\boldsymbol{x}$ is the desired position vector. These constraints are implemented in fabrics as a forcing term with the differential map $\phi_\text{goal} = (\text{fk}_\text{c} - \text{fk}_\text{p}) - \boldsymbol{x}$. On the manifold defined by this differential map, we define the forcing potential ψ that can be pulled at forcing our optimization fabric as $\boldsymbol{M}\ddot{\boldsymbol{q}} + \boldsymbol{f} + \partial_{\boldsymbol{q}}\psi = 0$. The final policy is then defined as the solution to the damped differential equation as

$$\ddot{\boldsymbol{q}} = -\boldsymbol{M}^{-1}(\boldsymbol{f} + + \boldsymbol{B}\dot{\boldsymbol{q}} + \partial_{\boldsymbol{q}}\psi), \tag{2}$$

where $\boldsymbol{B}$ is a positive definite damping matrix. For further details, we refer readers to [25].

The key advantages of fabrics are their fast computation and flexibility to compose behaviours and define the desired goal in a manifold, rather than being fixed to defining a target configuration or end-effector pose. For example, some tasks may require aligning the end-effector to face a specific point, while the actual position along the line is of little importance, see Fig. 9. Similarly, some products, like a can, can be grasped from many directions and one may only need to specify a subset of desired grasping poses, e.g., the grasp height and that the grasp should be perpendicular to the vertical axis, but not the specific approach direction.

2) Safety: An important aspect of applications of robotics solutions to human-shared environments is *safety* and failure-free operation during an extended amount of time. For the latter, the most important aspect is that joint-limits and self-collision is avoided at all times, because these can lead to hardware shutdowns. In fabrics, these constraints are achieved by defining a joint-limit avoidance and self-collision avoidance components as described above, making these failures virtually impossible. Safety, however, is more complex to achieve, as it requires an accurate environment model and a reliable prediction of humans and their individual joints. In this work,

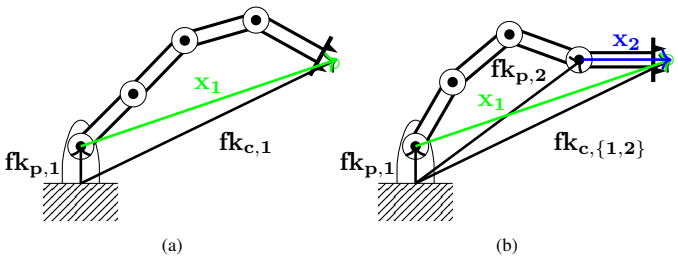


Fig. 8: Goal constraints for fabrics. In (a), the only constraint is defined by $\mathcal{C}_1 = (\text{fk}_{\text{p},1}, \text{fk}_{\text{c},1}, \boldsymbol{x}_1)$. In (b), a second constraint is added as $\mathcal{C}_2 = (\text{fk}_{\text{p},2}, \text{fk}_{\text{c},2}, \boldsymbol{x}_2)$ to align the end-effector horizontally.

Fig. 9: Illustration of the orientation constraints for trajectory generation for look-for-product.

we instead opted for safety through compliance during arm motion, i.e., the robot is compliant to external forces. This is achieved by tracking the desired acceleration output from Eq. (2) with a low-level controller that outputs the torques for the individual joints. Specifically, we use a PI controller that tracks the velocity that is obtained by integrating the desired acceleration. This approach allows ensuring that collisions are non-harming to the robot and its environments.

3) Usage: As an example, a reaching problem, where the end-effector should be at a certain position is defined by $\mathcal{C}_{\text{ee}} = (\text{fk}_0, \text{fk}_{\text{ee}}, \boldsymbol{x}_{\text{ee}})$, where fk_0 is the forward kinematics to the base link of the robot and $\boldsymbol{x}_{\text{ee}}$ the desired position of the end-effector in the base link frame.Fig. 8 shows two examples of composing goal states by constraints.

A problem we encountered when picking products with only the arm, is that the arm's workspaceis limited w.r.t. the shelf's size. This in combination with variability in base position or product location, often resulted in the product being out of reach or requiring difficult arm configurations close to joint limits. Therefore, during picking, we augment arm motion by integrating the forward motion of the base as a pseudo-prismatic joint to the kinematic chain. This can be easily done in fabrics by appending the base motion to the state vectors $\boldsymbol{q}$, $\dot{\boldsymbol{q}}$, and $\ddot{\boldsymbol{q}}$, see Fig. 13.

C. Look for a product

Looking for the product is triggered as soon as the robot base is in front of the shelf that is expected to have the desired product. The camera frame is then located at a position $\boldsymbol{x}_{\text{camera}}$. The camera must be pointed towards the expected product location, defined as $\boldsymbol{x}_{\text{item}}$. We can model this goal as two constraints. First, the camera link should not move from its current location, thus $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{camera}}, \boldsymbol{x}_{\text{camera}})$. Secondly, the camera should face the product location. We compute the ray connecting the camera and product location $\boldsymbol{x}_{\text{ray}} = \boldsymbol{x}_{\text{item}} - \boldsymbol{x}_{\text{camera}}$ to define $\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{\text{ray}})$ that aligns the end-effector with the defined ray $\boldsymbol{x}_{\text{ray}}$, see Fig. 9.

D. Picking of a product

To define a goal for picking products we use a combination of three constraints. First, the position of the vacuum cup is determined by the position of the product, thus $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{suction-cup}}, \boldsymbol{x}_{\text{suction}}, w_{\text{suction}})$. Secondly, we limit ourselves to picking products from the shelf, thus constraining the end-effector to be perpendicular to the shelf by defining $\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{\text{flange,end-effector}})$. Lastly, our gripper is composed of two suction cups, which, depending on the product should align with a specific angle for executing the most reliable grasp. The desired alignment defines our third constraint for the picking as $\mathcal{C}_3 = (\text{fk}_{\text{suction1}}, \text{fk}_{\text{suction2}}, \boldsymbol{x}_{\text{alignment}})$. Although it is possible to program picking, including approach and retreat, as a sequence of this set of constraints, it is difficult to capture all the nuances of picking in the code. Therefore, we make use of a human operator to teach the robot the best trajectory to reliably pick specific products, following the learning-from-demonstration paradigm [26]. Teaching has proven an effective way to generate sequences of the previously mentioned constraints thus encoding the human understanding of the picking problem into recorded trajectories. We explain the process of recording and playing back trajectories in detail in Sec. VII-C.

E. Placing of a product

Placing the product consists of four phases. The robot navigates to a configuration to its right or to its left depending on whether it was a right-sided pick or a left-sided pick. Then, it moves above the crate facing downwards. This is defined by two constraints, $\mathcal{C}_1 = (\text{fk}_0, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{0,end-effector}, 1)$ and $\mathcal{C}_2 = (\text{fk}_{\text{flange}}, \text{fk}_{\text{end-effector}}, \boldsymbol{x}_{flange,end-effector})$. The product is placed by moving the arm downwards until an external force, from touching the crate's bottom or an already placed product, is detected. This triggers the gripper release and goal change to the homing position ready for the next product.

VI. Perception

The three perception components are shown in Fig. 5.

- **Object detection and classification:** Generates 2D object proposals and classifies them in a binary way based on a provided target class, resulting in the proposals being classified as either target class or not. Object detection and classification are realized using two different models. Both models are fine-tuned on a supermarket dataset, but do not require retraining to add new products, as we will explain in the following.
- **Object pose estimation:** Uses 2D object proposals in combination with a depth image to convert them to 3D. To estimate the orientation around the z-axis, we use a plane-fit of the pointcloud frustum. That first order approximation of the surface of the products front has proven to be a reliable approach, even for non-planar surfaces, see Fig. 2.
- **Multi-object tracking:** To track the objects over time, we use a set of Kalman Filters, one per object. The object proposals are assigned to Kalman Filter tracks using the Hungarian Algorithm [27], [28].

Because a supermarket has a large and often changing set of products, the main requirement for our perception pipeline is

that it should be easily adaptable. This observation from retail environments led us to the constraint on the perception pipeline that it should not require retraining when new products are added. Such a constraint can be addressed using, so-called, *few-shot* models. In the following, we describe the details of our object detection and classification method.

The method utilizes YOLO for object detection, relying on product details for object classification. Additionally, our system allows adding products using a single or few images.

This dynamic addition of new products is possible trough a few-shot model we dub ProtoProductNet. This model is based on ProtoNet [29]. ProtoNet matches query images to target classes by their distance in feature space. For each target class a prototype is constructed that is essentially the mean of the features of a number of example images of this class. By matching query images to target prototypes, ProtoNet essentially learns to encode features that classify similarity between query images and target classes. Because this model picks random query- and target classes from a dataset for every iteration, ProtoNet learns a general feature extraction strategy that is invariant of the actual class. This is important for adding new products, as classifying new products is as easy as providing the model with new target images.

The exact implementation of ProtoNet we use is based on P>M>F [30]. P>M>F shows that in few-shot learning pre-training (P) is more important than meta-training (M), which is in turn more important than fine-tuning (F). For the best results the authors of P>M>F suggest using ProtoNet with a Vision Transformer pre-trained with DINO [31] as the feature extractor and meta-training it with a small learning rate.

However like most few-shot classification models, ProtoNet assumes that query images can only be one target class. Not only would comparing a query image to all supermarket products increase inference time, attributing it to a likeliest product is unsafe. If our product detector misidentifies a human as a product, the classifier must correctly recognize that and not classify it as the most likely product.

ProtoProductNet makes exactly this possible. It uses a ViT pre-trained with DINO to extract image features, and predicts if those features are part of a target prototype based on their cosine distance. ProtoProductNet then passes this cosine distance through a linear layer combined with a sigmoid function to translate it to a confidence score. If query images have a confidence < 0.5, they are considered not the target class. Using a sigmoid function however leads to a loss of relational information between classes. In contrast to ProtoNet, that predicts only the most likely target class among a set of target classes with a softmax function, a sigmoid predictor only uses the cosine distance per class to make predictions. As this mechanic is an important reason why ProtoNet works so well, ProtoProductNet will also be allowed to choose the likeliest from a number of prototypes. This means that next to a target prototype, a number of helper prototypes will be chosen. When classes are likelier to be a helper class then the target class, they are considered to be not the target class. As classes that are close together in feature space are harder to distinguish, it makes only sense to choose helper prototypes that are close to the target prototype.

VII. Interaction and Teaching Capabilities

To quickly adapt our robot to new store environments and products, we created four interfaces for operators: a digital twin for remote monitoring and control, adding product classes to the perception, trajectory teaching mode, and audio explanations of the robot's symbolic actions.

A. Digital Twin for Remote Monitoring and Control

Herein, we introduce a digital twin mechanism to support remote monitoring and control of a mobile manipulator in a retail setting, as shown in Fig. 10. By scanning the environment in three dimensions, we construct a virtual model that accurately represents the workspace. The robot, when operational in a supermarket, is connected to this digital twin through Wi-Fi or 4G, enabling operators to monitor its status and issue commands remotely. The addition of a tablet interface allows for flexible monitoring and control by on-site staff, who can easily adjust the robot's course or teach it new tasks as needed.

B. Interactively adding product classes to perception

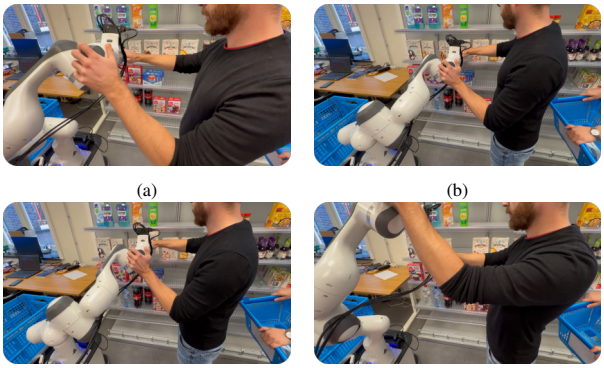
Sec. VI explains ProtoProductNet, our adaptation to the state-of-the-art ProtoNet, to make the few-shot learning approach scalable for the supermarket environment. To add new unseen product classes to ProtoProductNet, we developed a custom user interaction for human operators. The interaction contains the following steps 1) the operator uses a barcode scanner, attached to the robot, to scan the new product 2) the operator puts the product in view in front of the robot's camera 3) a GUI with the view of the camera pops up on the screen and the operator interactively drags a box around the new product. The robot should already start detecting the product, as can be verified by a bounding box appearing on screen. To further enhance the detection the operator can add images from different angles of the product. The cropped images selected by the operator are saved locally and combined with the product's barcode. If the same barcode is encountered in future orders, our perception pipeline will now know how to classify the product accurately, without retraining the network.

C. Teaching grasping trajectories to the robot

As outlined in Sec. V, fabrics require global guidance to effectively execute complex symbolic actions that are essential for some products, see Fig. 2. To simplify the process of obtaining this guidance in the form of trajectories, we leverage human expert demonstrations, effectively teaching the robot. We distinguish between two phases for teaching, the *recording* and the *playback*. For both phases, we assume that the product is visible and detected by the robot, such that we can compute a transformation between the root link and the product.

(a) Remote Monitoring & Control (b) Retail Environments (c) Mobile Monitoring & Control

Fig. 10: Overview of the remote monitoring and control system. a) Laptop-based remote interface for monitoring and control system; b) Visualization of the robot within the actual retail environment; c) Tablet interface for on-the-go monitoring and task programming.



(a) (b) (c) (d)

Fig. 11: The human operator can easily 'teach' the robot a new picking strategy by moving the arm, thus passing implicit knowledge to the robot.

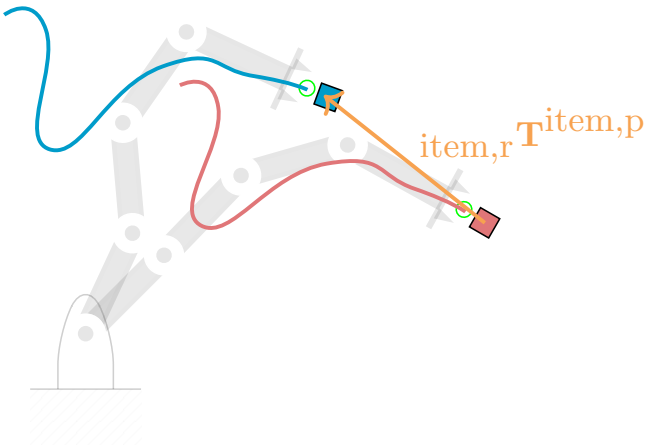


Fig. 12: To generalize to different item poses, recorded trajectories (red) are transformed based on the the transformation between the item's pose during recording and during playback (orange). The new trajectory (blue) is then tracked using our trajectory generation method.

1) Recording: When recording a trajectory, we first reduce the stiffness of the robot to the bare minimum, such that it can easily be pushed around by the human operator. Then, the human operator can activate recording by pressing a button on our tablet interface. From that moment onwards, the state values $\boldsymbol{x}$ for the constraints defined for picking in Sec. V-D are recorded, see Fig. 11. The state of the gripper, active or non-active, and whether a product is attached to the gripper are also recorded. The generated sequence of constraint values and gripper states is stored as a reference trajectory.

We additionally record the transformation matrix between the root link of our kinematic chain and the product to be picked. That allows us to later generalize the recording to different product poses by applying a rigid body transformation to the trajectory.

2) Playback: During trajectory playback, we loop through the recorded goals sequentially, continuing when a desired goal accuracy has been reached. In contrast to the recording part, where we exclude motion of the base, during playback the base motion is activated, see Fig. 13. To account for different product poses between recording and playback, we transform the goals on the fly based on the product pose estimate, see Sec. VI, using the following transform:

$$ {}_{\text{item},r}\boldsymbol{T}^{\text{item},p} = \left({}_{\text{base}}\boldsymbol{T}^{\text{item},r}\right)^{-1}_{\text{base}} \boldsymbol{T}^{\text{item},p}, $$

where ${}_{\text{base}}\boldsymbol{T}^{\text{item},r}$ is the transformation matrix between the manipulator's base link and the product during recording and ${}_{\text{base}}\boldsymbol{T}^{\text{item},p}$ is the transformation matrix between the manipulator's base link and the product during playback, see Fig. 12. Using this continual feedback we effectively employ a visual servoing [32] approach and are robust against changes in product location during the pick. In addition to fabrics goals, the recording also contains information about the state of the vacuum pump. This state information is replicated during playback, and used to know when a product should have been attached. In the playback routine for picking we then modify the fabrics goal if a product is not yet attached where it is expected. The goal is modified to effectively push further into the shelf along the z-axis of the nozzle head, until a product is attached, or a maximum threshold is reached.

D. Generalizability

To reduce the number of taught trajectories, we rely on generic trajectories. Our gripper design led to a *horizontal* and a *vertical* pick trajectory, where the two suction cups are either aligned horizontally or vertically. As most considered products have a planar surface, we can use these two trajectories for most products. These trajectories are replaced by product specific trajectories in case of repeated failures. For example, unconventional bottle shapes require modified trajectories. Similar to existing works on learning-from-demonstration [33], we argue that non-experts can, over time, create an increasingly complete trajectory database for all products to further improve performance. Importantly, general trajectories

(a) (b) (c) (d)

Fig. 13: During playback, fabrics actively use the base's forward motion as a prismatic joint to compensate for misplacement during navigation.

have proven to be sufficient for most of our products. Note that all trajectories are robot agnostic and only gripper specific, so we expect them to be transferable to other robots with similar gripper designs. The generalizability of our approach relies on the transformation of trajectories according to the item's pose. The robot's workspace is a natural limitation, as items placed outside the workspace (i.e. on the lowest or highest shelf or at the back on the shelf) are kinematically unreachable and thus, not resolvable by our teaching approach.

E. Audio feedback

During the robot's operation, we are interested in providing audio explanations of the robot's actions as feedback to operators, e.g., to monitor the robot and to be notified about failures. We do this by 1) generating compact prompts of the robot's action and state and 2) using Large Language Models (LLMs) to generate short informative explanations that are played via the robot's speaker.

a) Prompt generation: We leverage the structure of the generated BT's and symbolic state information (see Sec. IV) to generate prompts for an LLM. For every sub-tree in the generated BT, we automatically add explanation nodes that generate prompts for symbolic actions and items. An explanation is described by its name a formulated as a verb in the present continuous form, e.g., $a = \texttt{placing}$. The item's name i is taken from the product database, e.g., $i =$ *Whole Milk*. We generate string prompts of the form pr:

$$pr := action \quad a \; i \; c,$$

where $c \in \{\texttt{running}, \texttt{failed}, \texttt{completed}, \texttt{retry}\}$ is the status returned from the sub-tree. An example for a generated prompt is "*action* `picking Whole Milk retry`".

b) Explanation generation: We generate explanations by prompting an LLM on the fly with our generated prompts. To this end, we provide the LLM with a context describing that the robot is deployed as an order picking robot in a supermarket with five symbolic actions and that the task is to generate a concise explanation of the prompt to operators. During operation of the robot, we simultaneously generate the explanations and play them back via the robot's speaker. For instance, for the prompt "*action* `picking Whole Milk retry`", we generate the explanation: "Oops! It seems like I had a little trouble placing the Whole Milk into my basket. No worries, I'll give it another try and make sure it goes in smoothly this time".

VIII. Validation

To evaluate the performance and how well our robot adapts, we validated our robot in picking customer orders in two realistic environments. Our driving validation questions were:

1) What is the success rate of our robot?
2) What are causes for failures of the robot?
3) How well did the robot recover from disturbances?

We first describe the two validation environments, followed by summarizing the robot's performance and how it recovered from introduced disturbances such as misplaced items. A video showcasing our robot can be found in the paper's supplementary material.

A. Validation environments

We evaluated our robot in two different, realistic but controlled environments:

a) AIRLab (see Fig. 1): Our *AI for Retail* (AIRLab) environment is a university laboratory at TU Delft that resembles parts of real supermarkets of our Dutch retail partner. AIRLab has OEM shelves and products. We used this environment to develop and validate our system during development.

b) Realistic store: We also validated our robot in a realistic store layout of our Dutch retail partner, used by their development teams for testing before moving into their real stores. For confidentiality, we cannot show this store. The main differences to AIRLab are a larger number of products in the shelves that are also more densely packed, similar to the real stores of our partner. Moreover, the shelves also contained product information tags. We prepared one full day in this store to validate our system, including creating a map, scanning available products, and connecting to the product database.

B. Comparison to teleoperated systems

A direct comparison to a human picker seems of little use as the current stage of the system is not competitive in terms of capabalities and speed to a human picker. Instead, we compare the autonomy of the picking strategy to a teleoperated version of it. The focus on the picking for this lies in its important contribution to the overall success rate and execution time,

TABLE III: Number of attempted, succeeded and recovered symbolic action attempts for picking and placing (the more complex symbolic actions). Note, that a recovery is defined as a successful execution of a symbolic action after a failure.

	AIP-goal			picking			placing		
	attempted	succeeded	recovered	attempted	succeeded	recovered	attempted	succeeded	recovered
AIRLab	81	50	9	65	45	9	46	43	2
Realistic store	151	90	26	153	98	25	91	90	6

see Fig. 16a. For this study, the robot starts facing the shelf where the product is expected. The teleoperator has access to the camera image from the camera mounted on the end effector. Control is limited to Cartesian movements of the end effector, base forward motion and vacuum activation. Then, the teleoperator has access to the same controls available to the robot in autonomous mode. Therefore, the symbolic actions evaluated in this study are the *look-for-a-product* and *picking-of-a-product*. This limited study was performed for a subset of five different products from the set used in the demo stores. Teleoperation results in similar execution times than the autonomous mode, see Fig. 14. This indicates that the execution times is likely limited by the hardware setup, including sensors and actuators, and not by the modules responsible for the autonomous behavior. However, we acknowledge that the teleoperation study is limited in scope and that more capable teleoperation setups, see for example [34], might substantially outperform the autonomous mode.

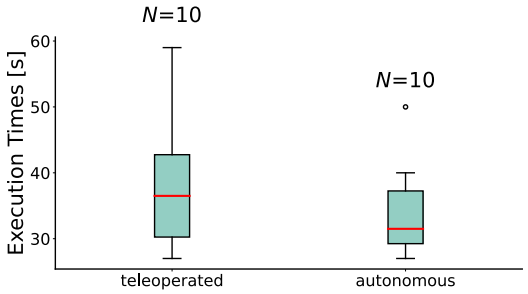
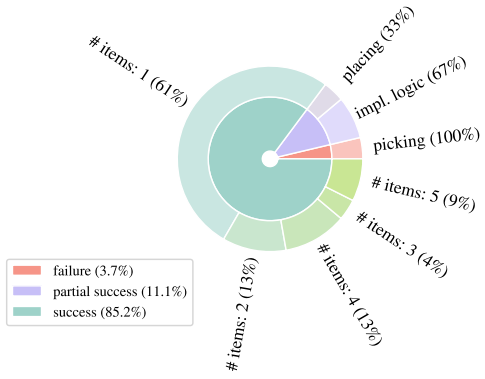
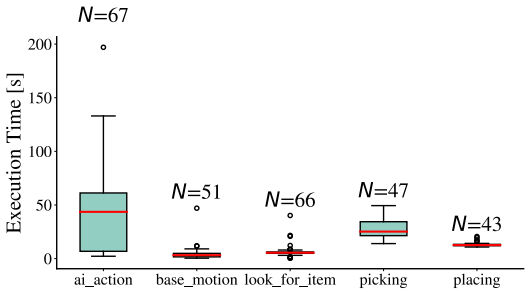


Fig. 14: Execution times for the combination of *look-for-a-product* and *picking-of-a-product* between teleoperation and autonomous mode.



(a) Success-rate across order sizes and failure causes.



(b) Execution times of symbolic actions in seconds. Note that symbolic actions remain active until the desired state, i.e., product in basket, is reached or a failure occurs. In that case, the symbolic action is re-attempted.

Fig. 15: Success-rate and action execution times in AIRLab environment. A total of $N = 27$ were performed.

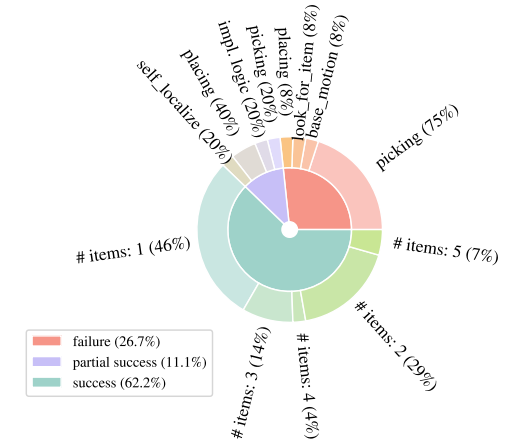
C. Performance

Performance is evaluated by success rate and execution times. Orders are divided into *success*, i.e. the entire order was successfully collected and returned to the client, *partial success*, i.e., at least one product was not collected and at least one was collected, and *failure*, i.e., no product was collected. This is visualized by the inner ring in Figs. 15a and 16a. We also inform about the failure reasons and the number of products a successful order contained (outer ring in Figs. 15a and 16a). For each symbolic action, we report execution times and how many failures were recovered by the adaptive task assignment method, see Table III. As the symbolic action remains active throughout the entire treatment of one product, there lower bound is the sum of the execution times of the other symbolic actions.

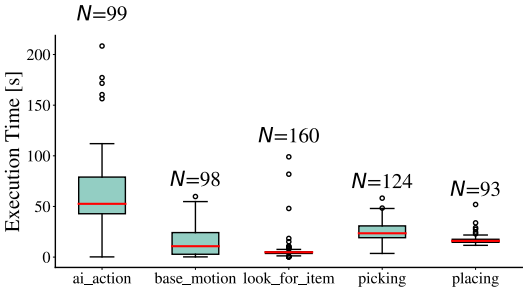
D. Success rate and recoveries

Our evaluation in a lab-like environment reveals that we can achieve a success rate of about 60% for few-items orders, see Fig. 15a. Additionally, we observe that most failures are caused during 'picking'. In Table III, we see that recoveries, i.e., a symbolic action failed at least ones before it succeeded, were common. Thus, it shows that decision-making that is able to deal with disturbances is essential in this sort of application. Investigating the execution times, it can be seen that 'picking' is also the symbolic action that takes most time in collecting an item, roughly 50s on average between starting the grasp at its completion, see Fig. 15b.

A remarkable property of our system is its reliability and fault tolerance at a very low computational cost. Specifically, apart from the perception, all the components, including the



(a) Success-rate across order sizes and failure causes.



(b) Execution times of symbolic actions in seconds. Note that symbolic actions remain active until the desired state, i.e., product in basket, is reached or a failure occurs. In that case, the symbolic action is re-attempted.

Fig. 16: Success-rate and action execution times in realistic store environment. A total of $N = 45$ were performed.

decision-making and the trajectory generation, are running on the Intel NUC with an Intel Core i7-10710U, a low-power CPU that is roughly 80% worse than the compute unit used in [11] according to www.cpubenchmark.net. This relies on our multi-level approach to adaptability and recovery from disturbances and runtime uncertainties:

a) Skill level: Our Skills are adaptive to disturbances such as sensor noise, to which our object recognition is robust, or physical disturbances. Examples of the latter are the ability of our compliant arm control to accommodate someone holding it —e.g., if an operator identifies an issue with the item being picked by the robot and wants to take it from the robot— or the visual serving enabled by our object detection and trajectory generation that continuously adapts the motions in case the position of the object changes in the field of view of the robot.

b) Task execution level: If the adaptability of the symbolic actions falls short of accounting for a disturbance, e.g., the operator took the item from the robot. It is now out of its field of view; failing to detect the item, our extremely reactive online planner would generate an alternative sequence of symbolic actions to achieve the desired intermediate subgoal belief state of the item being in sight, resulting in the trajectory generation component smoothly transitioning to a trajectory for the end effector to look for another instance of that item in the shelf. The formulation of the online planning problem in terms of desired states instead of symbolic actions results in a failure recovery behaviour that is easier to scale since there is no need to re-write an entire application-specific logic, which is the case in solutions based on state machines, but one can extend the definition of the planning problem with new states and eventually new symbolic actions if new symbolic actions are developed for the robot.

c) Task plan level: The BT structure with pre-defined recoveries retries a subgoal, e.g., getting an item into the basket, up to three times when it fails. This ensures a reasonable trade-off of reliability and performance, e.g., most of the time the second attempt to pick and item was enough. If the third attempt fails, most likely, the item can not be grasped. This heuristic is computationally simple and easily adjusts to new items, e.g., allowing more attempts for incredibly challenging items.

The evaluation in the realistic store environment shows that the system can be deployed to a human-shared environment without a major loss of performance, see Fig. 16a. This test environment also confirms that most reliability issues are caused by the picking action.

IX. Lessons Learned and Key Takeaways

Throughout our project, we have gained valuable insights that inform our approach to deploying robotic systems effectively. These lessons, drawn from hands-on experience, highlight key considerations and strategies we believe essential for successfully implementing robotic software solutions.

- **Human expert trajectories are an efficient way of encoding grasping strategies**: Through the recording of human expert picking trajectories, we could address a significant portion of collision avoidance challenges and grasping strategies for specific products. This approach effectively allows for the encoding of per product strategies regarding grasp approach, location, and retrieval in a far more streamlined manner compared to traditional hard-coded behaviors. We showed that default trajectories generalize for various similarly shaped products, so that the number of trajectories can be much lower than the number of different products. Although the adaptation of recorded trajectories proved successful, we require more complex methods to mitigate remaining failure cases.
- **Accurate product detection and continuous visual feedback are crucial**: Accurate product detection and pose estimation emerged as critical requirements within the confines of supermarket environments because of the small size of certain products and the lack of clearance. Continuous visual feedback, particularly through visual servoing techniques, played a pivotal role by enabling real-time tracking of products and refining pose estimations as the robotic arm approached the target object.

Failure case	Potential causes
Product knocked over during pick	Inaccuracy in product detection or trajectory following, resulting in insufficient vacuum seal
Collision with shelf	Changes in environment due to shelf railing, price tags or discount tags
Product outside reachable space	The arm cannot physically reach the bottom or top shelf
Collision with surrounding products on the shelf	Products are differently positioned than during taught behavior
Vacuum gripper fails to attach	Factors like product size, weight, shape and material can cause vacuum suction to be insufficient

TABLE IV: Qualitatively evaluated list of potential failure cases

- **Vacuum grippers may fail with light and small products**: We underestimated the inherent difficulty in effectively picking very light and small products. This challenge highlights the need for alternative gripping mechanisms or specialized approaches tailored to handling such delicate items.
- **Grasping angles heavily influence seal integrity and stability**: We put particular emphasis on selecting optimal grasping angles to ensure both seal integrity and product stability during suction grasping. Preferably, the grasp is positioned on a product's flat side to establish a secure seal while minimizing the risk of product displacement or toppling. A slight angle of approach that gently presses the product against the surface further enhances stability.
- **Compliant robots are key in dynamic environments**: Compliance is essential when safely operating rigid robotic arms in dynamic environments and alongside humans. To guarantee collision avoidance with humans requires accurate human detection and intent detection. As this is highly complex, we opted for safety through compliance during arm motion and rely on raw lidar data during base motion. Beyond safety, compliance offers inherent forgiveness in the event of grasping failures. Together with our adaptive online task planner, this combination enables our robot to recover or retry autonomously, minimizing human interventions.
- **Whole body control enhances efficiency**: Whole body (or semi-whole body) control simplifies the task of picking products from diverse shelves within a supermarket setting. The expanded configuration space with the additional degrees of freedom significantly enhances planning efficiency and reliability, thereby streamlining operations.
- **Rapid and iterative software development is imperative**: Rapidly iterating our software directly on the physical robot was a critical success factor for us. Swift iterations serve as a reliable indicator of the eventual outcome. Furthermore, our realistic rest lab environment with anticipated operational conditions enhanced overall system robustness.
- **Lack of high quality mobile manipulators hinders progress**: Contrary to prevailing notions, the development of mobile manipulators present substantial challenges, particularly in research. Existing solutions remain scarce, and researchers are often forced to deal with inaccurate and unreliable mobile bases and robotic arms with short battery lives. Similarly, versatile gripper design is an unsolved research topic. This underscores the need for continued exploration and innovation in this domain to bridge existing gaps and facilitate rapid advancements in robotic research.

X. Conclusion

In this DEMO-paper, we present our approach to order-picking in human-shared retail environments. In contrast to previous approaches for that application [11], we deploy a failure-robust task-planning method that can recover from failure of the individual sub-modules and propose a way to simplify the 'programming' of the robot by leveraging teaching. We show that a significantly simpler robot design than [11] can be successful at the task of in-store order-picking. The approach was evaluated in a realistic store environment and in a lab-like environments without much modifications. In the future, we must address the challenge of interacting with more complex-shaped items, such as fruits and vegetables. That requires the integration of either a gripper switching method or a more intricate gripper design, as suggested in [11].

References

[1] J. McGrath, *Report on labour shortages and surpluses*. Publications Office of the European Union, 2021.

[2] V. Astrov, S. Leitner, R. Grieveson, D. Hanzl-Weiss, I. Mara, and H. Weinberger-Vidovic, "How do economies in EU-CEE cope with labour shortages?" wiiw Research Report, Tech. Rep., 2021.

[3] D. Rodriguez-Guerra, G. Sorrosal, I. Cabanes, and C. Calleja, "Human-robot interaction review: Challenges and solutions for modern industrial environments," *IEEE Access*, vol. 9, pp. 108 557–108 578, 2021.

[4] M. Kronmüller, A. Fielbaum, and J. Alonso-Mora, "Online flash delivery from multiple depots," *Transportation Letters*, pp. 1–17, 2023.

[5] E. Guizzo and E. Ackerman, "The hard lessons of DARPA's robotics challenge [news]," *IEEE Spectrum*, vol. 52, no. 8, pp. 11–13, 2015.

[6] T. Wisspeintner, T. Van Der Zant, L. Iocchi, and S. Schiffer, "RoboCup@ Home: Scientific competition and benchmarking for domestic service robots," *Interaction Studies*, vol. 10, no. 3, pp. 392–426, 2009.

[7] C. H. Corbato, M. Bharatheesha, J. Van Egmond, J. Ju, and M. Wisse, "Integrating different levels of automation: Lessons from winning the amazon robotics challenge 2016," *IEEE Transactions on Industrial Informatics*, vol. 14, no. 11, pp. 4916–4926, 2018.

[8] G. K. Kraetzschmar, N. Hochgeschwender, W. Nowak, F. Hegger, S. Schneider, R. Dwiputra, J. Berghofer, and R. Bischoff, "RoboCup@ Work: Competing for the factory of the future," in *RoboCup 2014: Robot World Cup XVIII 18*. Springer, 2015, pp. 171–182.

[9] M. Sereinig, W. Werth, and L.-M. Faller, "A review of the challenges in mobile manipulation: Systems design and RoboCup challenges." *Elektrotech. Informationstechnik*, vol. 137, no. 6, pp. 297–308, 2020.

[10] S. Thakar, S. Srinivasan, S. Al-Hussaini, P. M. Bhatt, P. Rajendran, Y. Jung Yoon, N. Dhanaraj, R. K. Malhan, M. Schmid, V. N. Krovi *et al.*, "A survey of wheeled mobile manipulation: A decision-making perspective," *Journal of Mechanisms and Robotics*, vol. 15, no. 2, p. 020801, 2023.

[11] M. Bajracharya, J. Borders, R. Cheng, D. Helmick, L. Kaul, D. Kruse, J. Leichty, J. Ma, C. Matl, F. Michel, C. Papazov, J. Petersen, K. Shankar, and M. Tjersland, "Demonstrating mobile manipulation in the wild: A metrics-driven approach," in *Robotics: Science and Systems XIX*, ser. RSS2023. Robotics: Science and Systems Foundation, Jul. 2023. [Online]. Available: http://dx.doi.org/10.15607/RSS.2023.XIX.055

[12] A. Dömel, S. Kriegel, M. Kaßecker, M. Brucker, T. Bodenmüller, and M. Suppa, "Toward fully autonomous mobile manipulation for industrial environments," *International Journal of Advanced Robotic Systems*, vol. 14, no. 4, p. 1729881417718588, 2017.

[13] P. Štibinger, G. Broughton, F. Majer, Z. Rozsypálek, A. Wang, K. Jindal, A. Zhou, D. Thakur, G. Loianno, T. Krajník *et al.*, "Mobile manipulator for autonomous localization, grasping and precise placement of construction material in a semi-structured environment," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 2595–2602, 2021.

[14] M. V. Minniti, R. Grandia, K. Fäh, F. Farshidian, and M. Hutter, "Model predictive robot-environment interaction control for mobile manipulation tasks," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 1651–1657.

[15] N. Ratliff and K. Van Wyk, "Fabrics: A foundationally stable medium for encoding prior experience," *arXiv preprint arXiv:2309.07368*, 2023.

[16] M. Kalakrishnan, S. Chitta, E. Theodorou, P. Pastor, and S. Schaal, "Stomp: Stochastic trajectory optimization for motion planning," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 4569–4574.

[17] R. Bormann, B. F. de Brito, J. Lindermayr, M. Omainska, and M. Patel, "Towards automated order picking robots for warehouses and retail," in *Computer Vision Systems: 12th International Conference, ICVS 2019, Thessaloniki, Greece, September 23–25, 2019, Proceedings 12*. Springer, 2019, pp. 185–198.

[18] C. Wang, Q. Zhang, Q. Tian, S. Li, X. Wang, D. Lane, Y. Petillot, and S. Wang, "Learning mobile manipulation through deep reinforcement learning," *Sensors*, vol. 20, no. 3, p. 939, 2020.

[19] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," *arXiv preprint arXiv:2304.13705*, 2023.

[20] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation," *arXiv preprint arXiv:2401.02117*, 2024.

[21] C. Pezzato, C. H. Corbato, S. Bonhof, and M. Wisse, "Active inference and behavior trees for reactive action planning and execution in robotics," *IEEE Transactions on Robotics*, vol. 39, no. 2, pp. 1050–1069, 2023.

[22] K. Friston, T. FitzGerald, F. Rigoli, P. Schwartenbeck, and G. Pezzulo, "Active inference: a process theory," *Neural computation*, vol. 29, no. 1, pp. 1–49, 2017.

[23] C. Rösmann, F. Hoffmann, and T. Bertram, "Integrated online trajectory planning and optimization in distinctive topologies," *Robotics and Autonomous Systems*, vol. 88, pp. 142–153, 2017.

[24] K. Van Wyk, M. Xie, A. Li, M. A. Rana, B. Babich, B. Peele, Q. Wan, I. Akinola, B. Sundaralingam, D. Fox *et al.*, "Geometric fabrics: Generalizing classical mechanics to capture the physics of behavior," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3202–3209, 2022.

[25] M. Spahn, M. Wisse, and J. Alonso-Mora, "Dynamic optimization fabrics for motion generation," *IEEE Transactions on Robotics*, vol. 39, no. 4, pp. 2684–2699, 2023.

[26] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, and J. Kober, "Interactive imitation learning in robotics: A survey," *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.

[27] E. Hamuda, B. Mc Ginley, M. Glavin, and E. Jones, "Improved image processing-based crop detection using kalman filtering and the hungarian algorithm," *Computers and electronics in agriculture*, vol. 148, pp. 37–44, 2018.

[28] B. Sahbani and W. Adiprawita, "Kalman filter and iterative-hungarian algorithm implementation for low complexity point tracking as part of fast multiple object tracking system," in *2016 6th international conference on system engineering and technology (ICSET)*. IEEE, 2016, pp. 109–115.

[29] J. Snell, K. Swersky, and R. S. Zemel, "Prototypical networks for few-shot learning," *CoRR*, vol. abs/1703.05175, 2017. [Online]. Available: http://arxiv.org/abs/1703.05175

[30] S. X. Hu, D. Li, J. Stühmer, M. Kim, and T. M. Hospedales, "Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference," 2022.

[31] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin, "Emerging properties in self-supervised vision transformers," 2021.

[32] M. Kmich, H. Karmouni, I. Harrade, A. Daoui, and M. Sayyouri, "Image-based visual servoing techniques for robot control," in *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)*, 2022, pp. 1–6.

[33] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," *Robotics and autonomous systems*, vol. 57, no. 5, pp. 469–483, 2009.

[34] S. Behnke, J. A. Adams, and D. Locke, "The $10 million ana avatar xprize competition: How it advanced immersive telepresence systems," *IEEE Robotics & Automation Magazine*, vol. 30, no. 4, pp. 98–104, 2023.