

# Demonstrating Mobile Manipulation in the Wild: A Metrics-Driven Approach

Max Bajracharya, James Borders, Richard Cheng, Dan Helmick, Lukas Kaul, Dan Kruse,  
John Leichty, Jeremy Ma, Carolyn Matl, Frank Michel, Chavdar Papazov,  
Josh Petersen, Krishna Shankar, Mark Tjersland\*  
Toyota Research Institute (TRI)  
Los Altos, California 94022  
Email: firstname.lastname@tri.global

**Abstract**—We present our general-purpose mobile manipulation system consisting of a custom robot platform and key algorithms spanning perception and planning. To extensively test the system in the wild and benchmark its performance, we choose a grocery shopping scenario in an actual, unmodified grocery store. We derive key performance metrics from detailed robot log data collected during six week-long field tests, spread across 18 months. These objective metrics, gained from complex yet repeatable tests, drive the direction of our research efforts and let us continuously improve our system’s performance. We find that thorough end-to-end system-level testing of a complex mobile manipulation system can serve as a reality-check for state-of-the-art methods in robotics. This effectively grounds robotics research efforts in real world needs and challenges, which we deem highly useful for the advancement of the field. To this end, we share our key insights and takeaways to inspire and accelerate similar system-level research projects.

## I. INTRODUCTION

The ultimate goal of mobile manipulation research is to enable capable mobile manipulation robots to perform human-level tasks in diverse and complex environments. However, despite the vast robotics research in the past decades, autonomous mobile manipulators still struggle to operate in the real world, outside of specifically designed or modified environments. This should come as no surprise - mobile manipulation in human environments is an extremely challenging endeavor, requiring that robots:

- Accurately perceive their environment to estimate their own location, detect obstacles, landmarks, and objects of interest.
- Quickly generate collision-free motion plans to/from objects and locations of interest while avoiding obstacles.
- Manipulate a wide variety of unseen objects in ever-changing environments.

Each of the aforementioned tasks constitutes a difficult research problem on its own. This is why much of today’s robotics research is focused on individual challenges in the robotic pipeline. While this has led to significant progress in developing individual aspects of the mobile manipulation problem (e.g., motion planning, scene segmentation, grasping, etc.), relatively little work has been done that combines these capabilities and evaluates them in real-world environments.



Fig. 1. TTT operating in a real, unmodified grocery store.

This bears the risk of leaving important challenges—crucial for real-world deployment—unnoticed, while arguably less critical aspects are overemphasized. For example, motion planning research continues to push for more optimal trajectories in minimal time, while often settling for 90-95% success rates. However, at a systems level, the effect of a single percentage drop in success rate far outweighs even a significant drop in path optimality or computation time. Therefore, we believe that identifying and tackling fundamental problems that stand in the way of more widespread, real-world robot deployments by studying system-level performance in the field is an important contribution towards advancing the field of robotics.

In order to tackle the aforementioned difficulties and to push the development of capable mobile manipulation robots, we created a challenge task: a robot should go into a real grocery store, pick 20 unique items off the shelves from a randomly generated “shopping list” (out of  $\sim 1000$  potential items), place them in a basket, and bring them to its starting position. Figure 1 shows our robot in the grocery store we use for testing. Note that it is a real store rather than a lab replica, and that we do not modify it in any way for our tests. In addition to forcing us to address all the previously mentioned challenges (e.g., perception, planning, grasping), task performance is easily quantifiable, allowing objective measurement of performance and progress.

\*All authors contributed equally and are listed in alphabetical order.

This paper describes our robotic platform TTT, its custom hardware design, and algorithms that we developed to tackle the shopping challenge task. We focus on the entire mobile manipulation system rather than discussing any single subsystem or algorithm in detail. Most importantly, by developing and testing the system end-to-end over an extended period of time without abstracting away any components, we glean valuable insights into what important, unsolved challenges are for deploying mobile manipulators in real, semi-structured environments.

Section II briefly discusses related work. Section III provides a system overview and describes the system architecture. Sections IV, V and VI describe the hardware, perception and behavior systems, respectively. Section VII evaluates the robot’s performance on the grocery shopping challenge task. Section VIII concludes by discussing important lessons learned and avenues for future research. This paper is accompanied by a video that shows our robot in operation during the most recent (sixth) field test at the grocery store.

## II. RELATED WORK

Robotic mobile manipulation is a highly active, multi-faceted area of research. Interest in the field is driven by numerous potential applications and amplified by international competitions such as DARPA’s robotics challenge [15], RoboCup@Home [37], the Amazon Picking Challenge [11] and RoboCup@Work [21], each focusing on different challenges and performance criteria.

A substantial number of capable mobile manipulation platforms have resulted from past and current research projects. Systems that were influential in the class of wheeled dual-arm mobile manipulators like the one we are presenting include Willow Garage’s PR2 [25], CMU’s Herb2.0 [33], DLR’s Rollin’ Justin [7], JPL’s RoboSimian [17], KIT’s ARMAR-6 [4] and IIT’s CENTAURO [27]. An overview over wheeled mobile manipulation systems and the challenges involved is provided in [35] and [29]. Particularly related to the work we present are studies that involve the testing of mobile manipulation systems in semi-structured environments outside the lab. Dömel et al. [10] used a wheeled, single arm mobile manipulation system to fulfill fetch and carry tasks in a factory environment, similar to our shopping scenario. They conducted a daylong evaluation test to assess their system’s performance. Štibinger et al. [34] used a morphologically similar system in an outdoor competition to pick up and place simulated construction materials.

To the best of our knowledge, our study is the first to describe an approach for improving a fully autonomous dual-arm mobile manipulation system based on rigorous, repeated testing in an unmodified, real-world environment over an extended period of time.

## III. SYSTEM OVERVIEW

Figure 2 provides an overview of the software system. The main system capabilities are encapsulated within *modules*

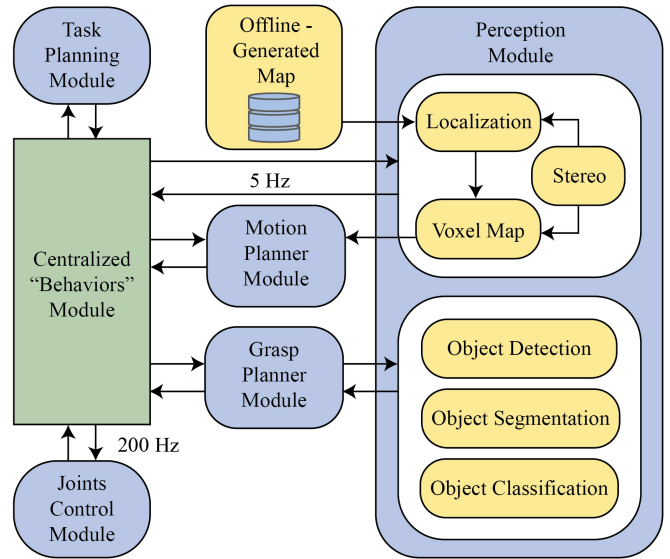


Fig. 2. Overview of our interconnected, modular software system designed for mobile manipulation tasks.

running on board at different rates, communicating through a custom interprocess communication (IPC) framework.

At the highest level, a task planning module implements a hierarchical finite state machine (FSM), which keeps track of the robot’s state and determines what high-level action it should take next (e.g., look for item, grasp item, localize, etc.). At the core of the system is a centralized *behavior* module. Once the task planning module decides which action to take, the behavior module queries other perception/planning modules to aggregate all necessary information and send low-level commands to the hardware at 200Hz in order to achieve desired objectives. A centralized behavior module that has exclusive control over joints simplifies the system architecture and reduces potential for conflicts between modules.

A visual perception module, running at 5 Hz, keeps the robot localized and aware of obstacles by generating a stereo point cloud and a voxel map for spatial awareness. Once the robot needs to grasp an item, it queries an event-triggered grasp planning module, and whenever the robot needs to plan a motion, it queries an event-triggered motion planning module. In the following sections, we describe these modules (and sub-modules) in more detail.

While many of these modules are based on pre-existing methods, there was significant benefit for us to implement them from the ground up. This way we can ensure to have complete insight into and control over the implementation, allowing us to (1) integrate tooling/visualizations to debug issues with minimal overhead, and (2) easily and continuously modify/customize/improve our algorithms and implementations based on the results of our testing.

## IV. HARDWARE SYSTEM

Our mobile manipulator robot TTT consists of four distinct parts: (1) a pseudo-holonomic 4-wheeled chassis; (2) a 5-DoF torso; (3) two 7-DoF arms; and (4) a perception head

on a 2-DoF neck. Nearly all components have been custom-designed from a low level, creating a homogeneous system that minimizes the footprint and upper body mass of the robot while maximizing its manipulation capabilities, particularly its range of motion and payload capacity.

**Actuation:** To maximize the payload capability of the robot, we developed three sizes of rotary actuators that are tightly integrated and have very high specific torques and torque densities. Each actuator has an integrated custom motor controller, torque sensor, and output position encoder. The controllers are capable of communicating via CAN and EtherCAT. We currently prefer using CAN.

**Tools:** At the end of each arm is a tool interface that allows tools to be manually installed and removed. For the grocery shopping task, we use a Robotiq HandE gripper on the right arm and a custom suction gripper on the left arm. In contrast to most industrial applications of suction grippers that primarily perform top grasps, we exclusively use lateral side grasps for the grocery task due to the tight constraints of the shelf. This is a more challenging application for suction grasping due to the gravity-induced moment and shear force acting on the suction cup.

Our custom suction gripper integrates two vacuum generators, a Robotiq EPick with an integrated diaphragm pump capable of producing an 82% vacuum, and a high-speed axial pump able to generate much larger flow rates, albeit at a lower vacuum level. We use the high flow rate when approaching objects to help create initial contact and to hold onto bags that are inherently challenging for suction grasps. If the tool's internal pressure signals a sufficient seal, it switches over to the diaphragm pump, taking advantage of its much stronger vacuum.

**Compute:** The robot has a single central compute system consisting of a standard ATX motherboard with an Intel Core i9-12900K CPU and an NVIDIA A6000 GPU. It has 15 TB of removable U.2 NVMe storage for logging. The use of modular consumer components as opposed to embedded single board computers allows us to take advantage of new hardware as soon as it is released. The robot communicates over Wi-Fi 6 to a consumer mesh network system.

**Sensing:** We exclusively use stereo cameras for visual perception, with a pair of Basler acA2500-60uc color cameras with wide-angle lenses mounted on the pan/tilt head and another front-facing pair mounted to the chassis. Each arm has a Sunrise Instruments M3553E 6-axis force/torque sensor mounted between the wrist and the tool interface.

**Power:** To allow several hours of uninterrupted field testing, the robot has a large amount of energy storage contained in four hot-swappable battery packs. Each pack is made up of three BB-2590 lithium-ion batteries and a backplane adapter, for a total of 3.6 kWh of capacity. The packs plug directly into a custom power board stack in the center of the chassis which monitors and controls the robot power system.

## V. PERCEPTION MODULES

### A. Learned Stereo Depth and Voxel Mapper

The input to the visual perception system consists of color stereo image pairs. A learned stereo algorithm [31] is used to produce dense and accurate depth from camera pairs in the robot head and chassis. The resulting RGB-D frames are fused into a dynamic 3D voxel map [5, 6]. Each voxel accumulates color as well as first and second order position statistics.

### B. Object Detection, Segmentation, and Classification

Object perception is crucial for both grasping and creating a map that contains the items in the store. It is done in the following way:

- **Object Detection:** We run a YOLOv5 detector [13] trained on SKU-110k [14] to obtain 2D bounding boxes around all items in an input image.
- **Object Segmentation:** We utilize a UNet-based segmentation network [28], trained on a mix of synthetic and real data, in order to obtain a segmentation mask of the item within the bounding box. The item pointcloud is then obtained by overlaying this segmentation mask with the RGB-D image obtained with the learned stereo module described in Section V-A and back-projection to 3D.
- **Object Classification:** Given a query image crop using the bounding boxes from the detector, a metric-learning-based approach [20] is used to find the most similar items in a database of images that are scraped from the internet. A second stage Prototypical Network [32] gives a fine-grained match and is trained with open-set loss [24] to handle the case when the query item does not exist in the database.

This provides us with segmented pointclouds of the items in the scene, along with their classification. The pipeline is shown in Figure 3, and it is utilized to map items within the store prior to robot deployment, and also by the robot when picking items from the shelves and placing them in its basket.

### C. Mapping and Localization

To enable our mobile manipulation robot to shop for groceries, it has to know its own pose as well as where items are located in the store. Since the venue is known in advance, there is no need to pursue a SLAM approach. Instead, we treat the two tasks separately: Mapping is done prior to robot deployment, and localization within the generated map is performed by the robot during task executions.

**Mapping:** The input to the mapping pipeline is a time-ordered stream of stereo images  $(L_0, R_0), \dots, (L_n, R_n)$ , acquired with stereo cameras that are moved through the store. We run our learned stereo network on each image pair to compute per-pixel depth. We detect image keypoints in each image with a DoG detector [23] and store a RootSIFT [2] descriptor for every keypoint that has valid depth. The map is then generated via the following four steps:

- **Visual Odometry (VO):** We estimate camera pose deltas between each pair of subsequent left images  $(L_{k-1}, L_k)$ .

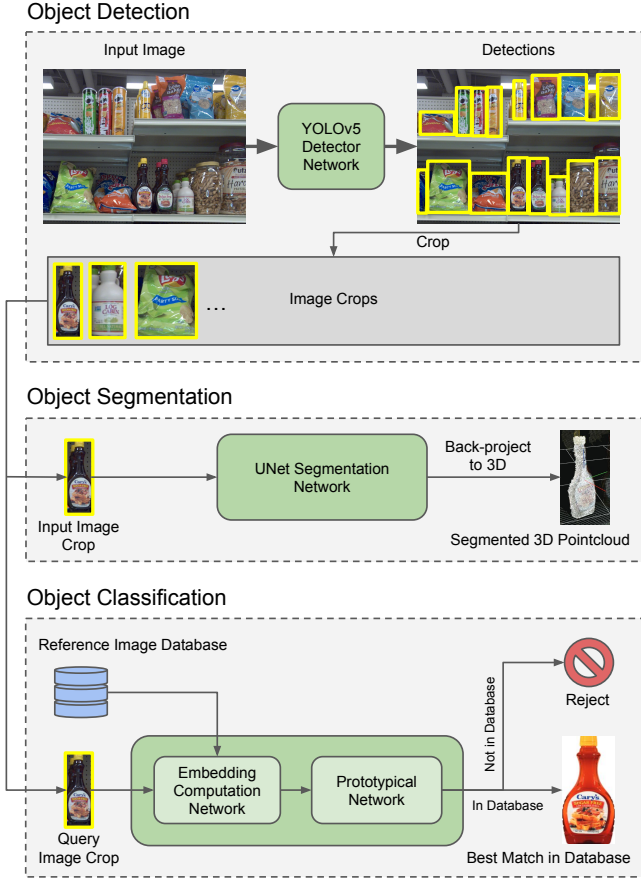


Fig. 3. The object perception pipeline as outlined in Section V-B.

First, we match the descriptors in  $L_{k-1}$  to their nearest neighbors in  $L_k$  and vice versa. We only keep mutual nearest neighbors as valid correspondences. Next, we compute the pose of  $L_k$  relative to  $L_{k-1}$  by running a PnP solver [22] within a RANSAC loop [12].

- **Loop detection and closure:** Since VO leads to pose drift, we perform loop detection and correct for the accumulated VO pose error. We train a NetVLAD-based network [3] to output a global image descriptor  $\mathbf{g}_k$  for each image  $L_k$ . To detect a loop closure, we find the 5 nearest neighbors to  $\mathbf{g}_k$  and compute pose deltas between the corresponding images and  $L_k$  in the same way as in the VO section above. We accept the pose delta with the highest number of inliers (if that number exceeds 200). Then, we correct for the pose drift using a deformation modeling technique, similar to [36].
- **Bundle Adjustment (BA):** We pass the following input to a state-of-the-art BA algorithm [1]: (i) the 2D locations of the inlier keypoints, (ii) initial estimates of the 3D coordinates of the physical points we want to reconstruct (computed by averaging the 3D coordinates of corresponding keypoints) and (iii) the camera poses computed by VO and loop closures. The output is a set of refined 3D point coordinates and camera poses. However, the resulting 3D points are sparse and noisy (see Figure 4(a))

which makes it difficult to use them for collision-free navigation. This is why we only use the refined poses to position and orient the 3D point clouds computed using the learned stereo module. The result is a dense and clean 3D geometric reconstruction (see Figure 4(b)).

- **Mapping Store Items:** Our maps have to contain more than just a geometric reconstruction of the environment. They have to be enriched with the locations of grocery items in the store. To achieve this, we employ the object perception module described in Section V-B. Since all camera poses are registered within a single coherent map coordinate system, we get the object location within the map by transforming its pointcloud according to the corresponding camera pose and computing the centroid.

**Localization:** The robot uses the map generated offline to localize itself within the store. At the beginning of a shopping task execution, the robot estimates its pose relative to the map using the same procedure outlined in the above paragraph on loop detection and closure, where  $L_k$  is the current image from the robot's chassis camera. Once localized, it relies on VO to navigate to each item in the shopping list. To correct for the VO drift, the robot re-localizes each time it arrives at an item.

## VI. PLANNING

### A. Task Planning

At the highest level, our task planner is implemented as a hierarchical FSM, which guides the high level actions of the robot. For example, when the localization/navigation modules indicate that the robot is near the item to be grasped, a transition into a *detection* state is triggered, where the robot runs object detection, segmentation, and classification (as detailed in Section V-B) to find the requested item. A simplified illustration of the highest hierarchy of our FSM is shown in Figure 5.

### B. Navigation

The first step of the navigation pipeline is to generate a 2.5D elevation map from the geometric reconstruction described in Section V-C. At the beginning of a shopping run, using that same geometric reconstruction with the (offline) mapped items, navigation goals (2D locations on the ground plane) are generated for each item in the randomly-generated shopping list by searching for a collision-free pose of a planar robot collision body along the outward-facing-axis of the item co-ordinate frame. Obstacle-free paths (a sequence of 2D points) are then generated between all permutations of the items in the shopping list using an A\* search within an inflated obstacle map generated from the 2.5D elevation map. The Christofides algorithm [9] is then applied to a graph of these obstacle-free paths to compute the approximate shortest path that visits all items for the generated shopping list. This gives us a full path (set of 2D waypoints) that takes the robot to all the items in our shopping list. To track this path, a path follower algorithm [18] is used to generate collision-free trajectories between consecutive waypoints.



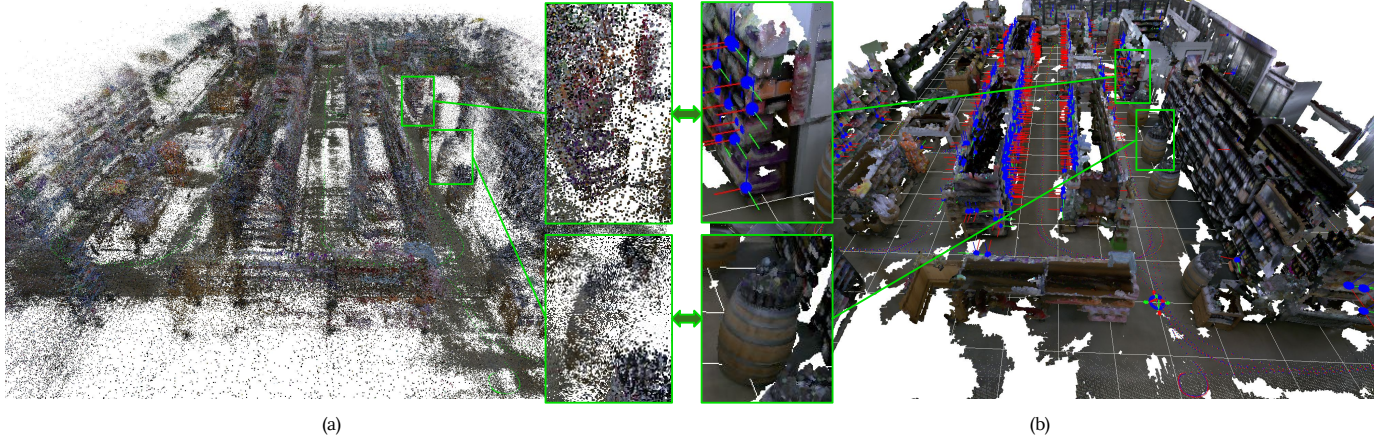


Fig. 4. (a) Output of a state-of-the-art BA algorithm [1]. Notice the noise and sparsity of the reconstruction. (b) Our map generated by registering the stereo-based 3D pointclouds using the poses computed by the same BA algorithm. The green rectangles show a side-to-side comparison between the same two regions in the maps and highlight the difference in reconstruction quality. The positions of the mapped store items are rendered as blue spheres and their orientations are indicated by red, blue and green coordinate frames.

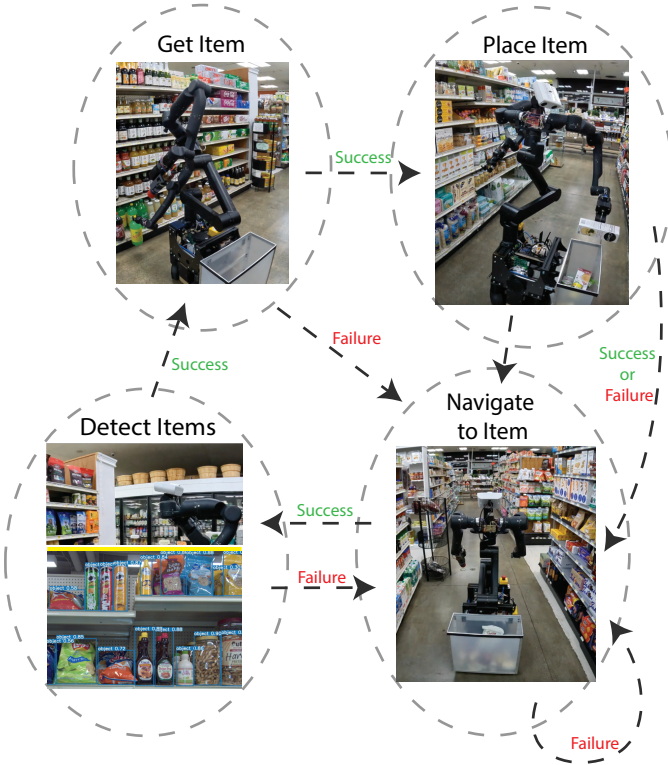


Fig. 5. Highly simplified task summary. Note that our hierarchical finite state machine includes many more states/transitions, but this provides a pictorial description of the main components of the grocery task.

The path follower uses high-rate, smooth pose estimates from combining wheel odometry (200 Hz) and visual odometry (5 Hz). Only at the end of each path (when it arrives at the item) does it re-localize within the map (as described in Section V-C). This is to prevent pose noise caused by the localization updates from affecting the performance of the path follower during execution.

### C. Motion Planning

To accomplish different tasks, we must be able to command the robot to arbitrary target end-effector poses (e.g. for grasping objects and placing them) while avoiding collisions. For this purpose, we developed a motion planner that can achieve 100% reliability with sub-second average planning times in a changing environment for our highly redundant, 21 DoF robot (the longest kinematic chain being 12 DoF). To describe the collision environment, we utilize the 3D voxel map mentioned in Section V-A. Our robot leverages (1) a dynamic roadmap (DRM) [19] that is pre-checked for collisions against any potential voxel map combined with (2) an optimization-based inverse kinematics (IK) solver [30] to quickly compute optimal plans in configuration space towards goals specified in Cartesian space.

When a motion plan is requested, the voxel map is first processed and used to prune out all nodes in the DRM that are in collision. This is extremely fast, as these collision-checks are done offline based on the robot model and stored for efficient online queries. Since some collision-checks cannot be done offline (e.g., if the robot model changes because it grabbed an item), we leverage GPU-accelerated collision-checking for pruning out any nodes in collision with any newly added collision bodies. After this step, all remaining nodes in the DRM can be assumed to be collision-free, and a set of collision-free paths can be generated by searching for the shortest path to all nodes in the neighborhood of our target Cartesian pose (e.g., using A\*). Finally, we rely on a QP-based IK solver to connect from nodes in this neighborhood of the target pose to the exact target pose, and the shortest resulting path is chosen.<sup>0</sup>

The goals of this approach, outlined in Figure 6, are to front-load as much computation as possible to offline pre-computation (i.e., *training* a large DRM) and to parallelize expensive aspects of online planning (i.e., collision-checks).

<sup>0</sup>A shortcutting-step is added at the end in order to optimize the robot path.

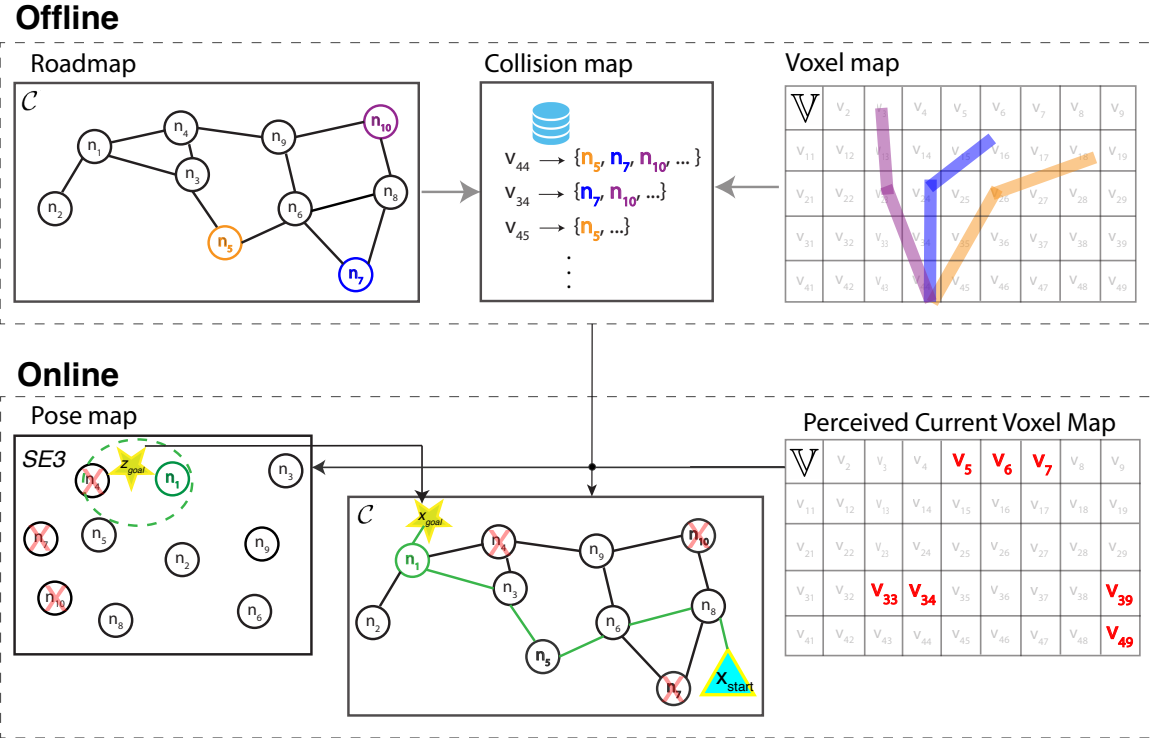


Fig. 6. Depiction of the motion planning pipeline. The top block denotes the offline process of building the roadmap and collision map of the DRM. The colored links in the voxel map (top right) depict the robot configurations corresponding to the same-colored nodes in the roadmap (top-left), such that the mapping from voxels to nodes-in-collision can be built. The bottom block denotes the online process of querying the DRM for a collision-free path, leveraging the collision map to avoid expensive collision checks.

#### D. Grasp Planning

Upon successfully navigating to a target item’s location as specified in the map and detecting the actual object instance in the live camera images, the robot plans how to pick this item off the grocery shelf and place it into its shopping basket. This requires the robot to take the following steps:

- Plan a collision-free path to the front of the desired item.
- Reach into a cluttered shelf with either the parallel-jaw gripper or the suction gripper.
- Grasp the item via a pinch or suction grasp.
- Extract the item from the shelf.
- Plan a collision-free path to place the item in its basket.
- Move the gripper over the basket and release the item.

Before the robot extends its end-effector tool into the shelf, it corrects the pose of the tool, compensating for any kinematic errors, using a custom, GPU-accelerated point cloud registration method. After extraction, the robot evaluates whether the grasp was successful or not by examining the wrench at the tip and (a) the position of the gripper fingers or (b) pressure signals from the suction gripper.

In order to tackle grasping a variety of items in different environmental contexts (e.g., sitting on a shelf or hanging from a hook), we use a learned mapping from the perceived item to a discrete, comprehensive set of grasp/extract strategies for each item (see Figure 7). Through experimentation with the robot, we found five different categories of grasps (e.g., grasp by handle, grasp by cap, suction at flattest region), and four categories of extraction (e.g., extract from hook) that

handle most items represented in our chosen challenge task. We trained a PointNet-based [26] and a ResNet-based [16] network for grasp and extraction classification, respectively, with the grasp classifier using segmented point clouds and the extraction classifier using RGB crops of item instances as input. Once the type of grasp is inferred, a downstream process computes the grasp pose that is optimal for that grasp type. For example, for handle grasps, a keypoint detector was trained to find the center of handles in RGB images in order to anchor the grasp about that point. For convenience, classification of the grasp/extract type is done offline for every mapped item in the grocery store, while the grasp pose is computed online. We use an engineered, geometry-based heuristic to select the object instance to pick if the robot detects multiple instances of a desired item type on the shelf.

## VII. EVALUATION

### A. Task Description

The challenge task requires the robot to fulfill a randomly generated shopping list in a real-world, unmodified<sup>1</sup> grocery store. The items in the store are mapped prior to task execution using the procedure described in Section V-C. We conduct our tests at nighttime outside of the store’s opening hours. This

<sup>1</sup>We install a mesh Wi-Fi network with Ethernet backhaul connections for the duration of our tests to ensure a stable connection between the operator station and the robot. However, this is not required since our robot is fully autonomous, and we do not consider it a modification of the environment.





Fig. 7. Object Grasp Examples (top, left to right): (1.i) flat, cylindrical object grasp; (1.ii) cap grasp; (1.iii) handle grasp; (1.iv) heavy, deformable object grasp; (1.v) suction grasp. (bottom, left to right): (2.i) extraction from a lip-free shelf; (2.ii) extraction of a jug; (2.iii) extraction from a box; (2.iv) extraction from a hook.

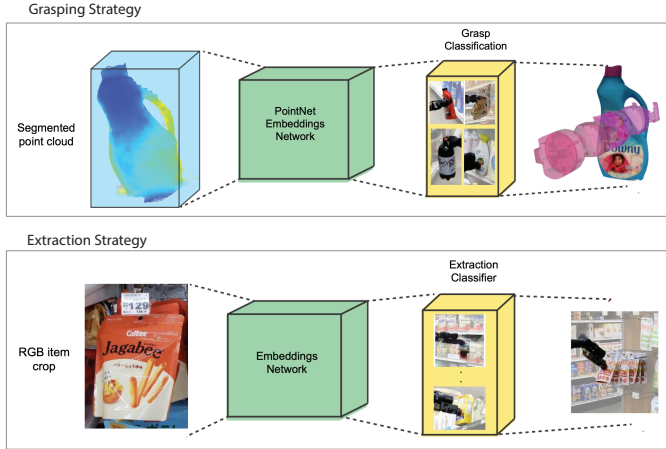


Fig. 8. **Top:** Network diagram for classifying grasp type based on segmented point cloud of item. This is combined with the pose of the object (computed downstream) to obtain a 6D grasp pose. **Bottom:** Network diagram for classifying extraction type based on RGB crop (e.g. extracting from hook vs. from box).

means no other shoppers are present and lighting conditions in the store are not affected by sunlight.

At the start of the task, the robot localizes itself relative to the map. Next, a shopping list of 20 unique items is generated randomly, where up to two instances of each item may be requested. The task is for the robot to autonomously collect all items in a basket and bring them to the starting point.

During the task, no human intervention is allowed, unless the robot is about to do something dangerous to itself or the environment. For those occasions, the human operator has a remote E-stop to end the task immediately. For liability and safety reasons, the robot operates outside of opening hours. Moreover, while no modifications are made to the grocery store, we narrow the scope of items we attempt to grasp primarily to avoid damaging products or the robot itself.

Namely, we remove produce, items inside a refrigerator, items that are heavier than  $4.5 \text{ kg}^2$ , and glass items. We will consider those in future work.

### B. Task Metrics

Using the grocery shopping task as a way to continuously evaluate the performance of our system in an end-to-end fashion allows us to make data-driven decisions about where to focus our development efforts in order to efficiently improve the most important robot capabilities. To gather the required data, we conduct field tests every three months in an actual grocery store, during which the shopping task is executed as often as possible, over the course of five consecutive nights for four hours per night. We do not change the hardware or software of the robot system for the duration of this week-long test.

Based on detailed data logs, videos and records of all attempted picks, we use several metrics for performance assessment. In the following, we define the most important ones.

**Reliability:** A task is considered completed if the robot navigates to all 20 items, attempts to pick the requested number of instances, and drives back to its starting point. Note that this definition does not take into account how many items were successfully picked and placed in the basket. Tasks are considered unsuccessful when they were prematurely stopped by an unrecoverable hardware or software fault, or by an operator-issued E-stop.

We define the *task success rate* as the ratio of the number of completed tasks to the number of started tasks. We use this as our highest-level indicator of system reliability. Since

<sup>2</sup>Note that TTT's arms are capable of handling up to 10 kg of payload, but most items in the grocery store that exceed 4.5 kg are too heavy for our suction tool and not graspable with a parallel-jaw gripper (e.g., heavy boxes of soda cans).

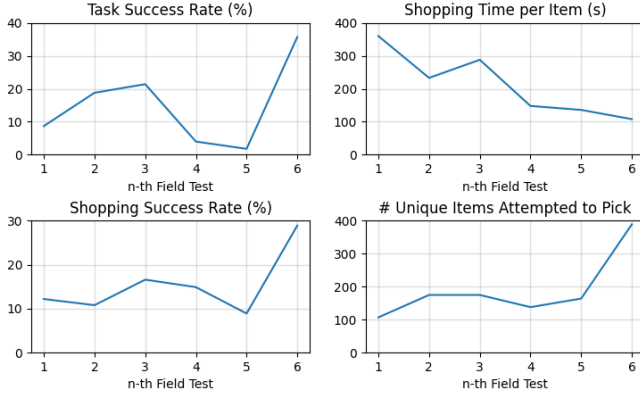


Fig. 9. Metrics that we use to drive our development. **Top Left:** Overall system reliability measured by the task success rate. **Top Right:** Speed measured by the shopping time per item. **Bottom Left:** Shopping performance measured by the shopping success rate. **Bottom Right:** Number of unique items attempted to grasp.

any of the robot’s subsystems, from hardware to motion planning, can cause a task-ending failure, we use finer-grained indicators to give us more insight into what caused a specific failure. The unforgiving, multiplicative nature of the overall system reliability, measured by the task success rate, strongly motivates the development of fault recovery strategies for all subsystems.

**Shopping performance:** We measure the shopping performance as the ratio of the number of successfully retrieved items to the total number of items in the shopping list. Even though this metric can be negatively affected by items that are in the shopping list but out of stock, it gives us valuable insights into the system’s performance.

**Speed:** We quantify speed as the overall time it takes the robot to complete one shopping list, divided by the number of items it retrieved. This overall speed metric incentivizes us to optimize speed on all levels, primarily for navigation, motion planning and motion execution.

### C. Results

Guided by the metrics described above, we have worked on improving our system’s performance in the grocery store shopping task for 18 months. In this time period, we conducted six field tests at a local grocery store at three months intervals. From the detailed log data collected during these tests, we track the above metrics that allow us to assess the overall performance and identify the most meaningful areas of improvement.

Besides these high-level metrics, data from the field tests enables us to perform fine-grained analysis of all system failures. A visualization of such analyses for two consecutive quarters is shown in Figure 10, where the main focus areas were reducing hardware and low-level software reliability problems with the new TTT platform (“joint control errors”) and avoiding collisions with the environment.

**Reliability:** Reliability, measured by the task success rate, is crucial for system up-time, a necessary requirement to collect

the data for our data-driven development approach.

We initially started this project with a different robot platform, FMT. While kinematically and morphologically almost identical, FMT is more heterogeneous in terms of hardware components, and its arms have a lower payload capacity than our current TTT platform. With FMT, we achieved a 21.4% task success rate by the third field test. For the following field tests, we switched to the newly developed TTT platform. Naturally, this led to a decrease in system reliability as we worked through the challenges of bringing up a new platform. After the associated drop, system reliability recovered and got better than ever: the success rate reached 35.7% by the sixth field test (see Figure 9, top left).

This number is low compared to human performance, giving us lots of room to improve. However, we argue that it is high for a complex robot system operating in a real-world setting, considering that a shopping run can last up to 30 minutes and consists of hundreds of consecutive actions (such as planning, moving, driving, grasping). Even if each individual action was 99% reliable, it only takes 103 consecutive actions for the task success rate to drop below the 35.7% achieved by our robot.

**Shopping performance:** We regard the shopping success rate (the ratio of the successfully retrieved items to the number of items in the shopping list) as the best overall indicator of shopping performance. It is sensitive to almost every aspect of the system’s performance, except for speed. Largely driven by the significant increase in our system’s reliability between the last two field tests, the overall performance has started a very exciting upward trend from 8.9% in the fifth to 28.9% in the sixth field test (see Figure 9, bottom left). This increased performance and the resulting higher number of grasp executions per field test unlock much more effective data collection, which will further accelerate our development.

**Speed:** We made continuous overall progress on the shopping speed, which is measured by the *shopping time per item*. This time includes navigating to an item as well as picking it. We were able to reduce this metric by approximately 70% over the course of six field tests, from six minutes per item to below two minutes (see Figure 9, top right). The largest increase in speed occurred when we switched from the FMT to the more capable TTT robot platform. However, switching to a faster robot was only one aspect of increasing the speed. More importantly, we developed tools to investigate which parts of the shopping task take up significant time and were able to focus our speed-up efforts there. This led to progress in widely applicable robot capabilities such as faster motion planning for redundant systems, faster collision-free navigation in confined spaces, and faster grasping.

**Unique items attempted to pick:** Of the many more detailed metrics we track, one that is particularly intuitive and valuable to us is the *number of unique items attempted to pick* during a field test. We aim to steadily increase this number, which makes the overall problem harder, without regressing on the other metrics. It is affected by our mapping capabilities and the robot’s ability to grasp a wide variety of items with its two complementary tools, but also by the sheer time the robot is



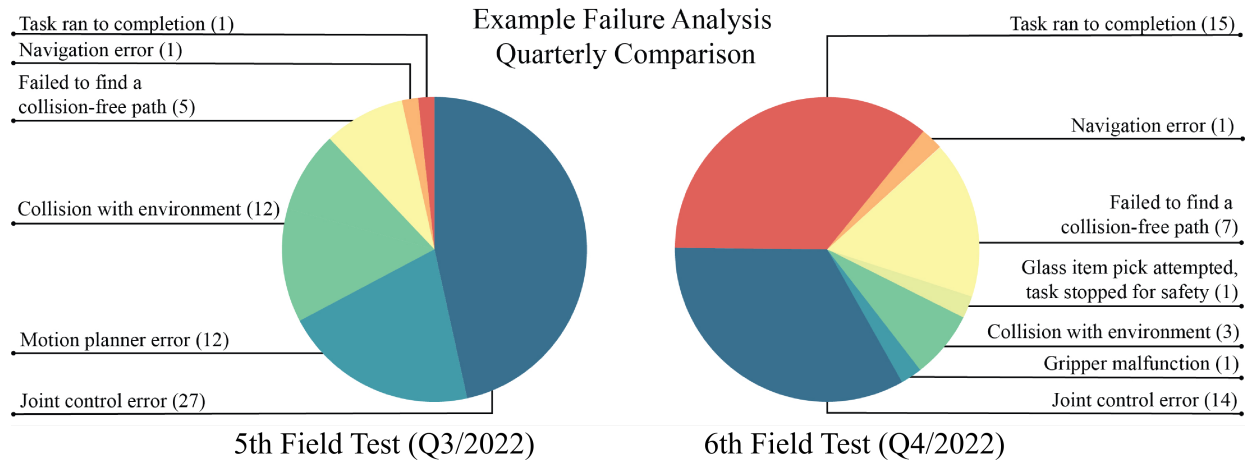


Fig. 10. Example of a comparison of quarterly field test failure analysis results. The pie charts break down causes of task failures, with the left and right charts corresponding to the fifth and the sixth field test, respectively.

actually executing shopping runs, which is in turn affected by its reliability. Thanks to a map containing close to 1000 items, an ever more capable grasping pipeline and the significant uptick in system reliability, this metric has entered a steep upward trend from 164 unique items in the fifth field test to 389 in the sixth (see Figure 9, bottom right). Each attempted pick, successful or not, creates invaluable data from which we can learn and improve. For this reason we expect this increase to benefit many aspects of our development pipeline.

#### VIII. LESSONS LEARNED AND KEY TAKEAWAYS

Based on our extensive experiments in the grocery store and analysis of the resulting data (including successes and failures), we have compiled key takeaways that we believe could accelerate deployment of robots in the wild.

##### **System/component reliability is the most important metric for real-world deployment:**

- Working on end-to-end systems with many actions chained in series emphasizes how important reliability and accuracy of the individual components/subsystems are. If each individual action is 99% reliable, and we take 10 actions per minute, overall reliability would be 0.2% for a task that takes 1 hour.
- Resetting the robot is not a viable option in the real-world. When the robot gets stuck or hits an object or goes down unexpectedly, human intervention is required, at least partially defeating the purpose of an autonomous robot.
- Homogeneous hardware enables more reliable systems. TTT, using our in-house actuators in all of its joints, is more reliable than our previous, more heterogeneous robot architectures. Our observation is that more homogeneous systems require fewer unique hardware and software components. These components are therefore easier to test, improve and harden.
- In order to see research results transition to real-world applications, there needs to be a viable path from proof-of-concept towards making methods reliable. While ‘hardening’ certain methods is often deemed an engineering

rather than a research problem, we want to challenge this notion. Making methods that are powerful and promising in a laboratory setting more robust or fault-tolerant to real-world challenges can be seen as a research problem in its own right, requiring creative ideas and sometimes entirely new approaches. Switching robot platforms in the middle of the project highlighted the need for methods that generalize beyond a highly specific hardware or system architecture.

##### **Machine learning has revolutionized many aspects of robotics and enabled several new capabilities, but we are still far from truly intelligent robots:**

- While we use learning methods for narrow tasks such as stereo reconstruction and gripper selection, a *reliable* learning-based method to synthesize broader robot behaviors (including fault responses) is still missing, but it is exciting to see recent progress towards this vision (e.g., [8]).
- Explainable AI will be crucial for deployment of powerful learned models in mobile manipulation. The lack of interpretability of current learned methods make fault recovery and failure analysis extremely difficult, with the only choice often being to add more/better data. Therefore, we are very excited to see advancements in this area leading to more reliable deployment in mobile manipulation robots.

##### **Manipulation with a mobile robot requires constant hardware-software co-design and co-evolution:**

- A robot that can navigate and manipulate objects within a human-centric environment benefits from a smaller footprint. In a sense, this places hardware and software development at tension with each other – smaller actuators are more difficult to design and manufacture, yet larger robots face more challenging collision constraints, affecting all areas of planning.
- Constant trade-offs are necessary when considering end-effector designs for mobile manipulators, including balancing safety vs. strength, simplicity vs. versatility, and

power vs. form-factor. For instance, grasps for suction grippers are easier to generate compared to grasps for mechanical grippers. However, designing suction grippers that can successfully grasp a wide variety of materials and weights while maintaining a small form-factor poses new challenges in hardware design.

## IX. DISCUSSION

We hope we have impressed upon the reader that mobile manipulation is a hard problem, with the need and many exciting opportunities for future research. Two things we deem critical for continued progress are: (1) end-to-end system testing, and (2) field testing in real-world (non-lab) environments.

End-to-end system testing (in addition to unit tests and integration tests) is necessary because algorithms as well as hardware subsystems that perform well in isolation may not perform well on a complex robotic system. Concrete examples that we encountered include:

- Our voxel mapper reconstructs and tracks obstacles. It uses the robot kinematics to distinguish the robot itself from the environment by masking out robot pixels, preventing them from entering the voxel mapper pipeline. Our motion planner relies on an accurate voxel map for collision free planning. While both these components functioned well (passing all unit tests), end-to-end testing revealed rare instances in which kinematic uncertainty leads to parts of the robot erroneously being labeled as environment, preventing successful motion planning. This led us to (1) develop a purely vision-based robot mask, independent from the robot kinematics, and (2) make our motion planner robust to spurious voxels arising from an imperfect robot mask.
- If an item to grasp is far enough from the current robot position, requiring our gripper to make a difficult reach, the robot first does a lateral drive to make the grasp easier. To keep the item in view, the robot needs to rotate its head, resulting in skewed images. We learned that this leads to a much higher likelihood of failed or inaccurate item perception. This observation led us to re-think this aspect of the overall item retrieval pipeline.

There are many more examples of such subtle interplay that are difficult to produce when testing modules in isolation, but that arise during long periods of end-to-end testing. Such instances are important for highlighting inaccurate assumptions made by different algorithms so that we can push towards addressing them.

In addition to end-to-end testing, periodic testing in real-world environments is crucial as it will highlight issues that do not arise in the lab, and expose implicit assumptions that hold true in the lab but not in the field. As an analogy, consider how in the machine learning field, it is standard practice to divide a dataset into a training set, validation set, and test set. The validation set can be used to tune hyperparameters of the learning algorithm, whereas the test set should only ever be used for evaluation. The reasoning is that having access to the validation set, even just to tune hyperparameters, introduces

the risk of overfitting to it. *A lab environment is analogous to a validation set* - even if it is designed to approximate the real world, we risk overfitting to its specific characteristics. *Testing in the field is analogous to testing on a test set.*

We have found that our metrics are significantly better in the lab (i.e. our mock grocery store) versus the real-world grocery store. Part of that is because the real environment is more challenging (e.g. customers place items in odd configurations, the store layout and placement of items is constantly changing), but a large part is also that we have constant access to the lab when designing/tuning our algorithms, and so we can unconsciously overfit to the challenges we encounter there.

## ACKNOWLEDGMENTS

We would like to thank JC Hancock, Jordan Skerbetz, Andrew Custer and Jonathan Yao for their support with map data collection and robot testing.

## REFERENCES

- [1] Sameer Agarwal, Keir Mierle, and The Ceres Solver Team. Ceres Solver, 2022. URL <https://github.com/ceres-solver/ceres-solver>.
- [2] Relja Arandjelovic. Three Things Everyone Should Know to Improve Object Retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.
- [3] Relja Arandjelovic, Petr Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [4] Tamim Asfour, Mirko Waechter, Lukas Kaul, Samuel Rader, Pascal Weiner, Simon Ottenhaus, Raphael Grimm, You Zhou, Markus Grotz, and Fabian Paus. ARMAR-6: A High-Performance Humanoid for Human-Robot Collaboration in Real-World Scenarios. *IEEE Robotics & Automation Magazine*, 26(4), 2019.
- [5] Max Bajracharya, Jeremy Ma, Andrew Howard, and Larry Matthies. Real-Time 3D Stereo Mapping in Complex Dynamic Environments. In *Proceedings of the International Conference on Robotics and Automation-Semantic Mapping, Perception, and Exploration (SPME) Workshop*, volume 15, 2012.
- [6] Max Bajracharya, James Borders, Dan Helmick, Thomas Kollar, Michael Laskey, John Leichty, Jeremy Ma, Umashankar Nagarajan, Akiyoshi Ochiai, Josh Petersen, Kevin Stone, and Yutaka Takaoka. A Mobile Manipulation System for One-Shot Teaching of Complex Tasks in Homes. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.
- [7] Christoph Borst, Thomas Wimbock, Florian Schmidt, Matthias Fuchs, Bernhard Brunner, Franziska Zacharias, Paolo Robuffo Giordano, Rainer Konietzschke, Wolfgang Sepp, and Stefan Fuchs. Rollin’Justin-Mobile Platform with Variable Base. In *Proceedings of the IEEE Interna-*

*tional Conference on Robotics and Automation (ICRA)*, 2009.

- [8] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jor-nell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. RT-1: Robotics Transformer for Real-World Control at Scale. In *arXiv preprint arXiv:2212.06817*, 2022.
- [9] Nicos Christofides. Worst-Case Analysis of a New Heuristic for the Travelling Salesman Problem. Technical Report 388, Graduate School of Industrial Administration, Carnegie Mellon University, 1976.
- [10] Andreas Dömel, Simon Kriegel, Michael Kaßecker, Manuel Brucker, Tim Bodenmüller, and Michael Suppa. Toward Fully Autonomous Mobile Manipulation for Industrial Environments. *International Journal of Advanced Robotic Systems*, 14(4), 2017.
- [11] Clemens Eppner, Sebastian Höfer, Rico Jonschkowski, Roberto Martín-Martín, Arne Sieverling, Vincent Wall, and Oliver Brock. Lessons from the Amazon picking challenge: Four aspects of building robotic systems. In *Robotics: science and systems*, pages 4831–4835, 2016.
- [12] Martin A. Fischler and Robert C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6), 1981.
- [13] Jocher Glenn. YOLOv5 by Ultralytics, 2020. URL <https://github.com/ultralytics/yolov5>.
- [14] Eran Goldman, Roei Herzig, Aviv Eisenschtat, Jacob Goldberger, and Tal Hassner. Precise Detection in Densely Packed Scenes. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.
- [15] Erico Guizzo and Evan Ackerman. The Hard Lessons of DARPA’s Robotics Challenge [News]. *IEEE Spectrum*, 52(8), 2015.
- [16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [17] Paul Hebert, Max Bajracharya, Jeremy Ma, Nicolas Hudson, Alper Aydemir, Jason Reid, Charles Bergh, James Borders, Matthew Frost, Michael Hagman, John Leichty, Paul Backes, Brett Kennedy, Paul Karplus, Brian Satzinger, Katie Byl, Krishna Shankar, and Joel Burdick. Mobile Manipulation and Mobility as Manipulation—Design and Algorithms of RoboSimian. *Journal of Field Robotics*, 32(2), 2015.
- [18] Daniel M Helmick, Stergios I Roumeliotis, Yang Cheng, Daniel S Clouse, Max Bajracharya, and Larry H Matthies. Slip-Compensated Path Following for Planetary Exploration Rovers. *Advanced Robotics*, 20(11), 2006.
- [19] Marcelo Kallmann and Maja Mataric. Motion Planning using Dynamic Roadmaps. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2004.
- [20] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese Neural Networks for One-Shot Image Recognition. In *ICML deep learning workshop*, volume 2, 2015.
- [21] Gerhard K. Kraetzschmar, Nico Hochgeschwender, Walter Nowak, Frederik Hegger, Sven Schneider, Rhama Dwiputra, Jakob Berghofer, and Rainer Bischoff. RoboCup@Work: Competing for the Factory of the Future. In *RoboCup 2014: Robot World Cup XVIII*, 2015.
- [22] Vincent Lepetit, Francesc Moreno-Noguer, and Pascal Fua. EPnP: An Accurate O(n) Solution to the PnP Problem. *International Journal Computer Vision*, 81(2), 2009.
- [23] Tony Lindeberg. Detecting Salient Blob-Like Image Structures and Their Scales with a Scale-Space Primal Sketch: A method for Focus-of-Attention. *International Journal Computer Vision*, 11(3), 1993.
- [24] Bo Liu, Hao Kang, Haoxiang Li, Gang Hua, and Nuno Vasconcelos. Few-Shot Open-Set Recognition Using Meta-Learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.
- [25] Wim Meeussen, Melonee Wise, Stuart Glaser, Sachin Chitta, Conor McGann, Patrick Mihelich, Eitan Marder-Eppstein, Marius Muja, Victor Eruhimov, Tully Foote, John Hsu, Radu Bogdan Rusu, Bhaskara Marthi, Gary Bradski, Kurt Konolige, Brian Gerkey, and Eric Berger. Autonomous Door Opening and Plugging in with a Personal Robot. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.
- [26] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep Learning on Point Sets for 3D Classification and Segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [27] Vignesh Sushrutha Raghavan, Dimitrios Kanoulas, Darwin G. Caldwell, and Nikos G. Tsagarakis. Reconfigurable and Agile Legged-Wheeled Robot Navigation in Cluttered Environments With Movable Obstacles. *IEEE Access*, 10, 2022.
- [28] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional Networks for Biomedical Image Segmentation. In *Proceedings of the 18th International Conference on Medical Image Computing and Computer-*

*Assisted Intervention (MICCAI)*, 2015.

- [29] Martin Sereinig, Wolfgang Werth, and Lisa-Marie Faller. A Review of the Challenges in Mobile Manipulation: Systems Design and RoboCup Challenges. *e & i Elektrotechnik und Informationstechnik*, 137(6), Oct 2020.
- [30] Krishna Shankar, Joel W. Burdick, and Nicolas H. Hudson. A Quadratic Programming Approach to Quasi-Static Whole-Body Manipulation. In *Algorithmic Foundations of Robotics XI: Selected Contributions of the Eleventh International Workshop on the Algorithmic Foundations of Robotics*, 2015.
- [31] Krishna Shankar, Mark Tjersland, Jeremy Ma, Kevin Stone, and Max Bajracharya. A Learned Stereo Depth System for Robotic Manipulation in Homes. *IEEE Robotics and Automation Letters*, 7(2), 2022.
- [32] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical Networks for Few-shot Learning. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, 2017.
- [33] Siddhartha S. Srinivasa, Dmitry Berenson, Maya Cakmak, Alvaro Collet, Mehmet R. Dogar, Anca D. Dragan, Ross A. Knepper, Tim Niemueller, Kyle Strabala, Mike Vande Weghe, and Julius Ziegler. Herb 2.0: Lessons Learned From Developing a Mobile Manipulator for the Home. *Proceedings of the IEEE*, 100(8), 2012.
- [34] Petr Štibinger, George Broughton, Filip Majer, Zdeněk Rozsypálek, Anthony Wang, Kshitij Jindal, Alex Zhou, Dinesh Thakur, Giuseppe Loianno, Tomáš Krajník, and Martin Saska. Mobile Manipulator for Autonomous Localization, Grasping and Precise Placement of Construction Material in a Semi-Structured Environment. *IEEE Robotics and Automation Letters*, 6(2), 2021.
- [35] Shantanu Thakar, Srivatsan Srinivasan, Sarah Al-Hussaini, Prahar M Bhatt, Pradeep Rajendran, Yeo Jung Yoon, Neel Dhanaraj, Rishi K Malhan, Matthias Schmid, Venkat N Krovi, and Satyandra K. Gupta. A Survey of Wheeled Mobile Manipulation: A Decision-Making Perspective. *Journal of Mechanisms and Robotics*, 15(2), 2023.
- [36] Thomas Whelan, Stefan Leutenegger, Renato Salas-Moreno, Ben Glocker, and Andrew Davison. ElasticFusion: Dense SLAM without a Pose Graph. In *Proceedings of Robotics Science and Systems (RSS)*, 2015.
- [37] Thomas Wisspeintner, Tijn Van Der Zant, Luca Iocchi, and Stefan Schiffer. RoboCup@ Home: Scientific Competition and Benchmarking for Domestic Service Robots. *Interaction Studies*, 10(3), 2009.