Advanced Robotics

ISSN: (Print) (Online) Journal homepage: https://www.tandfonline.com/loi/tadr20

World robot challenge 2020 – partner robot: a data-driven approach for room tidying with mobile manipulator

Tatsuya Matsushima, Yuki Noguchi, Jumpei Arima, Toshiki Aoki, Yuki Okita, Yuya Ikeda, Koki Ishimoto, Shohei Taniguchi, Yuki Yamashita, Shoichi Seto, Shixiang Shane Gu, Yusuke Iwasawa & Yutaka Matsuo

To cite this article: Tatsuya Matsushima, Yuki Noguchi, Jumpei Arima, Toshiki Aoki, Yuki Okita, Yuya Ikeda, Koki Ishimoto, Shohei Taniguchi, Yuki Yamashita, Shoichi Seto, Shixiang Shane Gu, Yusuke Iwasawa & Yutaka Matsuo (2022) World robot challenge 2020 – partner robot: a data-driven approach for room tidying with mobile manipulator, Advanced Robotics, 36:17-18, 850-869, DOI: 10.1080/01691864.2022.2114297

To link to this article: https://doi.org/10.1080/01691864.2022.2114297

View supplementary material

Published online: 30 Aug 2022.

Submit your article to this journal

Article views: 85

View related articles

View Crossmark data

Full Terms & Conditions of access and use can be found at
https://www.tandfonline.com/action/journalInformation?journalCode=tadr20

ADVANCED ROBOTICS
2022, VOL. 36, NOS. 17–18, 850–869
https://doi.org/10.1080/01691864.2022.2114297

FULL PAPER

World robot challenge 2020 – partner robot: a data-driven approach for room tidying with mobile manipulator

Tatsuya Matsushima[a], Yuki Noguchi[a], Jumpei Arima[b], Toshiki Aoki[c], Yuki Okita[d], Yuya Ikeda[c], Koki Ishimoto[d], Shohei Taniguchi[a], Yuki Yamashita[d], Shoichi Seto[c], Shixiang Shane Gu[a], Yusuke Iwasawa[a] and Yutaka Matsuo[a]

[a]Graduate School of Engineering, The University of Tokyo, Tokyo, Japan; [b]Matsuo Institute, Tokyo, Japan; [c]Faculty of Engineering, The University of Tokyo, Tokyo, Japan; [d]Graduate School of Information Science and Technology, The University of Tokyo, Tokyo, Japan

ABSTRACT
Tidying up a household environment using a mobile manipulator poses various challenges in robotics, such as adaptation to large real-world environmental variations, and safe and robust deployment in the presence of humans. The Partner Robot Challenge in World Robot Challenge (WRC) 2020, a global competition held in September 2021, benchmarked tidying tasks in real home environments, and, importantly, tested for *full* system performances. For this challenge, we developed an entire household service robot system, which leverages a data-driven approach to adapt to numerous edge cases that occur during the execution, instead of classical manual pre-programmed solutions. In this paper, we describe the core ingredients of the proposed robot system, including visual recognition, object manipulation, and motion planning. Our robot system won the second prize, verifying the effectiveness and potential of data-driven robot systems for mobile manipulation in home environments.

ARTICLE HISTORY
Received 28 February 2022
Revised 2 June 2022 and 20 July 2022
Accepted 30 July 2022

KEYWORDS
Service robot; partner robot; mobile manipulation; robot system; deep learning

1. Introduction

The World Robot Challenge 2020 (WRC2020) is a global robot competition that was held in Aichi, Japan, September 2021. Among the challenges in the competition, the Partner Robot Challenge (Real Space) aimed to develop a robot that can support daily human activities and approximate tidying up in real-world household environments. Tidying up a room may seem easy for most humans (though boring for many people); however, for robots, it is a complicated task filled with difficulties. For example, the robot needs to generalize the rich diversity of objects in real environments, adapting quickly to different types, poses, and even unseen scenarios. Moreover, the robot needs to avoid obstacles for safe deployment while moving both quickly and smoothly to not disrupt daily human life. Furthermore, unlike in a simulator, the robot cannot get access to ground-truth information about the environment and needs to decide actions promptly between each real-time control commands.

This paper describes an entire robot system that won second prize in the WRC2020 competition. In particular, we describe how we leveraged a data-driven approach to handle a variety of household environments. In real environments, any two homes do not have the same objects, furniture, or layouts. Therefore, it is almost impossible for the developer to enumerate all possible situations and manually implement each desired robot behavior. In other words, the robot should be able to generalize or adapt to the current environment rather than strictly following the pre-defined program. Accordingly, we utilized a data-driven approach to both perception and control of the robot system so that the robot could determine its behavior in new environments. For instance, we utilize and evaluate the following data-driven modules on the household environment:

- Object detection and recognition module based on deep neural networks. They comprise several object detection modules (Mask R-CNN [1] and UOIS [2]) and prompt-based classification using pre-trained CLIP [3]. We also designed special prompts to enhance the performance of pre-trained CLIP for WRC2020 tidy-up task.
- An image segmentation module was trained in a sim-to-real manner. We randomized several aspects of the environment in the simulator, such as robot configurations, drawer knob positions, and object sizes.

CONTACT Tatsuya Matsushima matsushima@weblab.t.u-tokyo.ac.jp

Supplemental data for this article can be accessed here. https://doi.org/10.1080/01691864.2022.2114297

© 2022 Informa UK Limited, trading as Taylor & Francis Group and The Robotics Society of Japan

- Human pose recognition module to fulfill the human-in-the-loop task ('pass an object to the person shaking their hands').
- Predicting grasp pose combining classic principal components analysis (PCA) and reinforcement learning in the simulators.
- Grasp detection using a vision-based tactile sensor.

It is worth mentioning that various data-driven methods have been proposed in various robot learning tasks, such as locomotion [4, 5], object grasping [6], and pick-and-place [7]. While these components are also crucial in the development of the tidying-up robot, the development of each components alone is insufficient to develop a complete system that can operate well in real world. For example, the robot system should be able to handle various inputs (such as robot internal states, odometry, and camera inputs) that usually have drastically different frequencies and be optimized with respect to the computational workload to speed up the entire system. The notable difference between WRC and many existing robot learning tasks is that WRC needs to build the entire system rather than solve a specific task.

In the context of general-purpose household robots, tasks involved in household robot systems are standardized as benchmarks and some robot competitions are held. For example, ERL Consumer Service Robots Challenge [8] standardize functionalities and tasks on environment recognition, human-robot interaction, and object manipulation in house environments. Recently, RoboCup@Home Domestic Standard Platform League (DSPL) adopts a similar rule to the WRC2020 tidy-up task [9–11]. However, many studies based on these competitions on household service robot systems are rather biased towards the system integration for certain benchmarks than discussing the generalization and have explored separately from the robot learning studies, while some recent solutions of these competitions partly leverage learning modules (e.g. off-the-shelf object recognition modules) [12–14]. In the WRC2020 competition, we showcased a service robot system that extensively utilized data-driven modules for generalization and adaptation in household environments durable for real-world deployment.

To utilize data-driven approach in WRC2020 task, it is important to consider how to design the entire system and how to separate the complex system into trainable modules as well as how to train each module. However, many data-driven approaches are only tested on benchmarks that do not require an entire system design, or a detailed design of the entire system are not comprehensively discussed [15–17]. This paper describes an entire system for tidying up robot that works in real space using a data-driven approach (e.g. data collection, model selection, learning, and communication among modules), which might help practitioners to develop other data-driven robots.

The remainder of this paper is organized as follows. Section 2 describes the principle in system design of the developed robot system and the overview of its software and hardware. In Section 3, we present the modules used in the system for object and environment recognition, aiming to ensure robustness for deviation in the environment easy adaptation. Section 4 describes motion planning for navigation and manipulation reflecting the result of aforementioned recognition modules. Subsequently in Section 5, we present the result of integrated experiment that took place in WRC2020 competition and discuss the current limitation and the plan to extend our robot system.

2. System overview

This section describes the entire system for tidying up the room task in the WRC2020. Section 2.1 describes the tidy up task in this competition in detail. A major difficulty for service robots to work in the household environment is the significant variety in the real environments; there are numerous edge cases to consider because of the unknown environment and the high degree of freedom in behavior. In this case, we solely rely on data-driven approach, particularly deep learning, to handle the issue. Section 2.2 describes how we used the data-driven approach, especially deep learning, in our systems. Furthermore, we discuss the importance of the adaptation strategy in the data-driven approach. While the deep learning has exhibited great performance in various applications, its utilization is still limited in robot systems. Difficulty in the application of deep learning in complex robot system attributes the huge computation that the deep learning-based modules typically require, and without proper system design, the system throughput is significantly impaired. In Section 2.3, we describe object-centric statefull system (implemented with 'object manager') where the system stores and updates the result of perception about the world at the object level. The object manager asynchronously distributes the most up-to-date information to each module and prevents the deep learning modules from bottlenecking the computation. Finally, we describe the software and hardware architecture at the end of this section.

2.1. Task description of WRC2020 partner robot challenge

In task1 of the competition (15 min), the robot picks up the object scattered on the floor and tables and place them in the target area specified by their object category

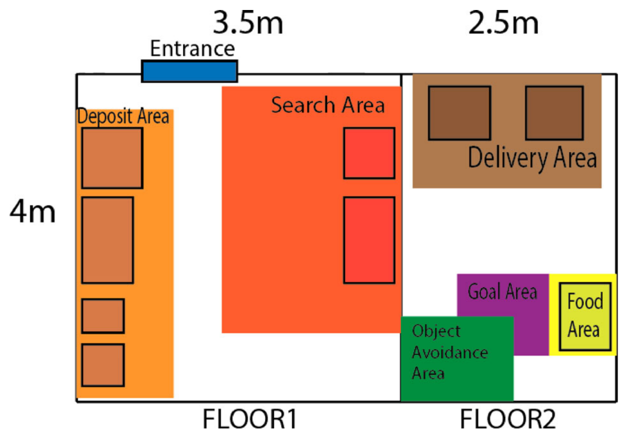


Figure 1. Floor map. In task 1, there are approximately 20 items on the search area in Floor1. The robot attempts to pick up these objects and place them into the designated deposit area. After finishing task 1, the HSR moves to Floor2 through object avoidance area to goal area. Subsequently, the robot picks up the specified food object from the shelf. Lastly, the robot is asked to deliver the object a person waving a hand in the delivery area.

in Floor1. In the task 2 (5 min), the robot moves to the neighboring room (Floor2) while avoiding small objects scattered on the floor, where it is asked to pick up the specified objects from the shelf and deliver it to a person waving the hand to the robot in the delivery area. Figure 1 illustrates the layout used in WRC2020.

In the task 1, the robots are required to quickly store objects scattered on the floor and the desk into the designated places. Regarding tidying up in Floor 1, each item is categorized as shown in Figure 2, and each category has a designated place to be stored. The scores are added regardless of where the items are placed, but the score is higher if the items are stored in the designated place. If the robot bumps into an object on the floor while moving, the score of the object it tries to store away afterwards will be minus. Some items, such as tableware and pens, have a designated storage direction, and a 'boss item' has higher points than other objects (in the WRC2020 tournament, the boss item was a golden spear). Additionally, if the robot opens the three shelves in the deposit area, the bonus points are added to the total score.

In task 2, the robots are expected to be able to avoid obstacles in tight spaces, plan grasps in tight spaces, and interact with humans. In concrete, after finishing task1, the robot has to avoid hitting the small items and navigate to the goal area in front of the shelf in Floor2 (task2a). Subsequently, the robot delivers the target objects on the shelf in food area to a person in the delivery area. There are two judges in Floor2, one is waving the hand, and the robot is asked to deliver the item to the waving person (task2b). WRC2020 rule book [19] explains the rules and scoring system in detail.

2.2. Deep neural networks based system

One of the difficulties in introducing service robots into the home is that there are many possible edge cases because of the unknown environment and the high degree of freedom in behavior. Traditionally, the developer needs to create program one-by-one to handle each edge case that they encounter. Recently, the advances of deep learning in various application areas (such as computer vision and natural language processing) has led to the introduction of various deep learning models for various robotics tasks. Unlike in a manual programming approach, a deep learning-based approach learns from the data, and can adapt to new edge cases by adding appropriate data.

The proposed robot system makes use of various deep learning-based recognition modules, and by randomizing the environment during training, we successfully developed a robust system that can respond to changes in the environment without any hard coding. Table 1 lists the correspondence between the model and the input data used in the system. However, such powerful recognition models require a long time for inference and can be a bottleneck for real-time execution. To solve this problem, we partitioned each deep learning module (e.g. object detection, classification, grasp proposal, motion planning, etc.) into ROS nodes as shown in Figure 3. Each of these nodes operates in parallel and communicates asynchronously, which guarantees the speed and performance of the system.

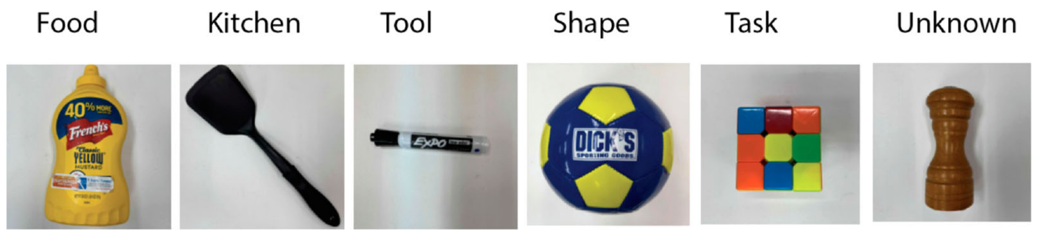


Figure 2. Objects' categories (examples) [18]. All objects are categorized under 'Food', 'Kitchen', 'Shape', 'Tool', 'Task', and 'Unknown', except for a boss item. 'Unknown' is also categorized to other categories. 'Unknown' object is not revealed just before the trials.

Table 1. Type and purpose of the model used and corresponding inputs.

Model Name	Input Data	Purpose
CLIP [3]	RGB image from RealSense and Xtion	Classification
Keypoint R-CNN [1]	RGB image from Xtion	Human Detection
Mask R-CNN [1]	RGB image from RealSense and Xtion	Object Detection
UOIS [2]	RGB-D image from RealSense and Xtion	Object Detection
Segmentation Model (Section 3.2)	Depth image from Xtion	Segmentation
Grasp Model (Section 4.1.1)	RGB-D image from RealSense and Xtion	Grasp Point Calculation

One of the challenges in operating a human support robot in a home is to adapt itself to different environments in each home. However, it is difficult to develop a robot system that learns how to operate in all possible environments in advance. The proposed robot system has a particular advantage in terms of adaptability to the environment for future operation in real life. More specifically, in the proposed system, data is collected each time the robot is operated, and the data collected is used to fine-tune each module, which creates a model that can adapt to each house environment during deployment. For example in the competition of WRC2020, because the lighting conditions are much different from those of our laboratory, the performance of recognition module was slightly impaired. However, by simultaneously collecting data during the four rounds of round-robin competition and fine-tuning the classifier, we succeeded in surpassing the performance of any round-robin competition in the semifinals.

2.3. Object-centric state-full system

Creating stateless modules is a straightforward implementation to utilize DNN-based modules in robot system. However, this design choice may significantly decrease the throughput of the entire system in the presence of DNN-based modules (such as object detection, key-point detection, and classification), because they require far more computational time than other modules, such as motion planning even with appropriate GPU accelerations and become the main bottleneck of the system. This is a problematic especially when some module depends on the output of the DNN-based module.

To avoid the reduction of entire throughput, we propose to use a stateful, object-centric design. The design is based on the idea of using parallelization and distributed processing to enhance computational efficiency. In summary, a module in the proposed system (called

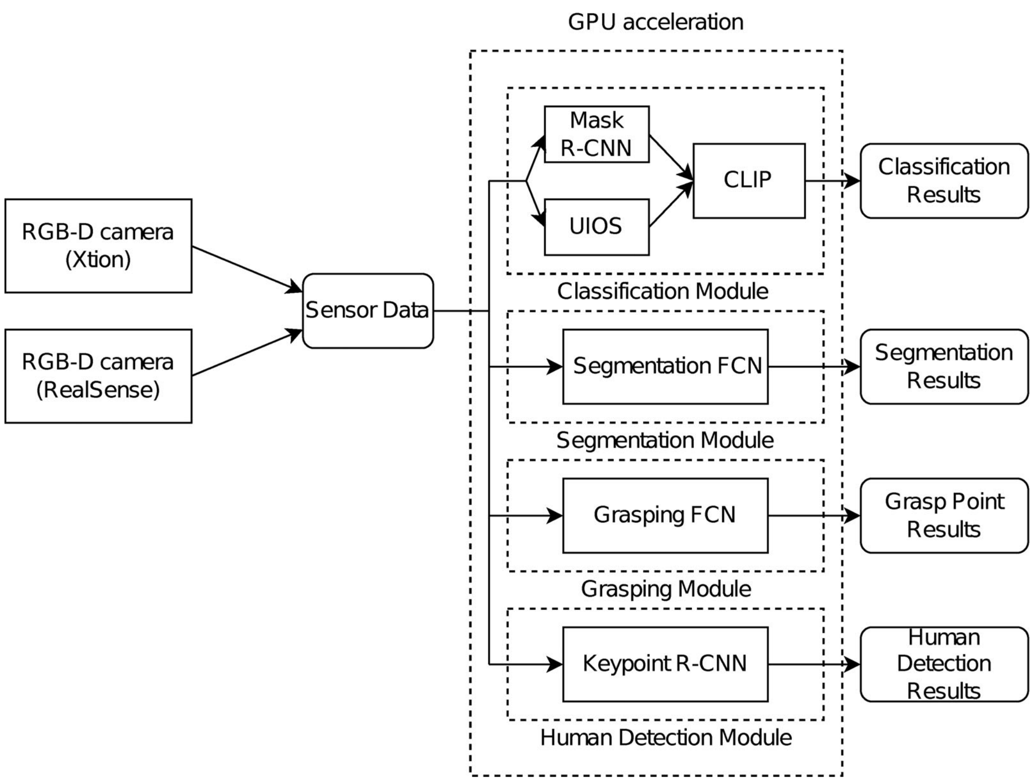


Figure 3. The proposed DNN-based module configuration. The module receives RGB-D image from Xtion and RealSense mounted on HSR. The corresponding sensor data is transmitted to each DNN-based module, as shown in Table 1. The process of each module operates at a sufficient frequency owing to parallelization and GPU acceleration.

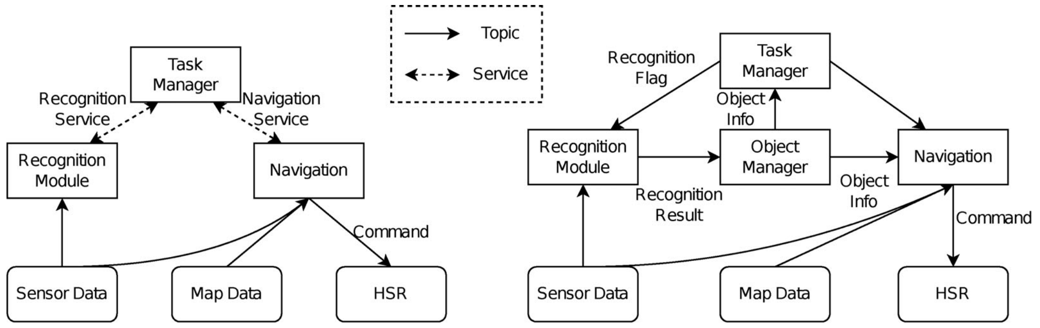


Figure 4. Brief overview of the proposed robot system. The left side of the figure is our initial stateless system design, and the right side of the figure is the final system design, the stateful and object-centric one. Owing to asynchronous processing and the object manager with the ability to store recognition results, the right system could reduce throughput degradation even when using computationally demanding DNN modules.

'Object manager') caches the outputs of the recognition module in the object-level, and each module communicates through the object manager. To be more specific, the object manager records information about each object (e.g. the detected location of the object, classification results) obtained from the deep learning models. By recording the information of each object, such as location and classification results in the object manager in advance and utilizing the stored results, we can reduce computation and achieve efficient operations. For instance, the robot can move directly to the next target object after placing the object, without having to perform object recognition again.

Figure 4 presents a brief overview of the designed robot system. The left side of the figure shows the initial stateless system design, and the right side of the figure shows the final system design, which is stateful and object-centric. In the stateless system design on the left, the task manager calls recognition module requiring huge synchronous computations, resulting in a significant drop in throughput. Meanwhile, in the stateful design on the right, the recognition module operates asynchronously with the task manager, and the recognition results are stored in the object manager accordingly, so the task manager does not have to call the recognition module when using the recognition results. Furthermore, in the right system, if necessary, the task manager can stop the recognition module through the recognition flag. This enables efficient processing even when the GPU resources are low.

2.4. Software

We adopted the method proposed by L. El Hafi et al. [20, 21] that configures a software development environment in a Docker container. Using Docker has several advantages, such as easy management of module versions and the ability to rapidly develop the same environment on different machines. In addition, unlike VirtualBox and other virtual machines, Docker operates on the kernel of the host machine, so it is as fast as running on it. It is crucial when executing systems that require computation, such as deep learning inference. The proposed software development environment is based on Robot Operating System (ROS) [22] on Ubuntu. Owing to the asynchronous communication feature provided by ROS, multiple modules can be processed efficiently in parallel. ROS also makes it easier to develop distributed systems, thus enhancing the stability of the system.

2.5. Hardware

Toyota's HSR (Human Support Robot) [23], a mobile manipulator, was used for the robot, which has a motor, encoder, a 6-axis force sensor in the wrist, a Lidar in the foot, an Xtion depth camera[1] in the head, a stereo camera, and a microphone as external sensors. As an additional sensor, a monocular wide-range RGB camera at the tip of the hand was replaced with a Realsense D435[2] depth camera (Figure 5). In addition to the internal processor, we used ROS [22] to communicate with a back-mounted laptop. For the laptop, we used an msi GS66 STEALTH with RTX3080 mobile 16GB.[3]

For this competition, we created two modes of operation: wireless mode and standard mode. In the wireless mode, the HSR is configured to operate in a high-speed WiFi environment ($\sim$ 1 Gbps), and the GPU-intensive calculations are executed on an external PC and the results are transmitted to the HSR. In addition, the most important communications were the depth camera images of the hand and head. The standalone mode is useful when WiFi connection is weak because all of the computation is performed on the backpack PC only and does not require wireless connection. In the standalone mode, multiple machine learning models were computed on a 16GB GPU (Figure 6).

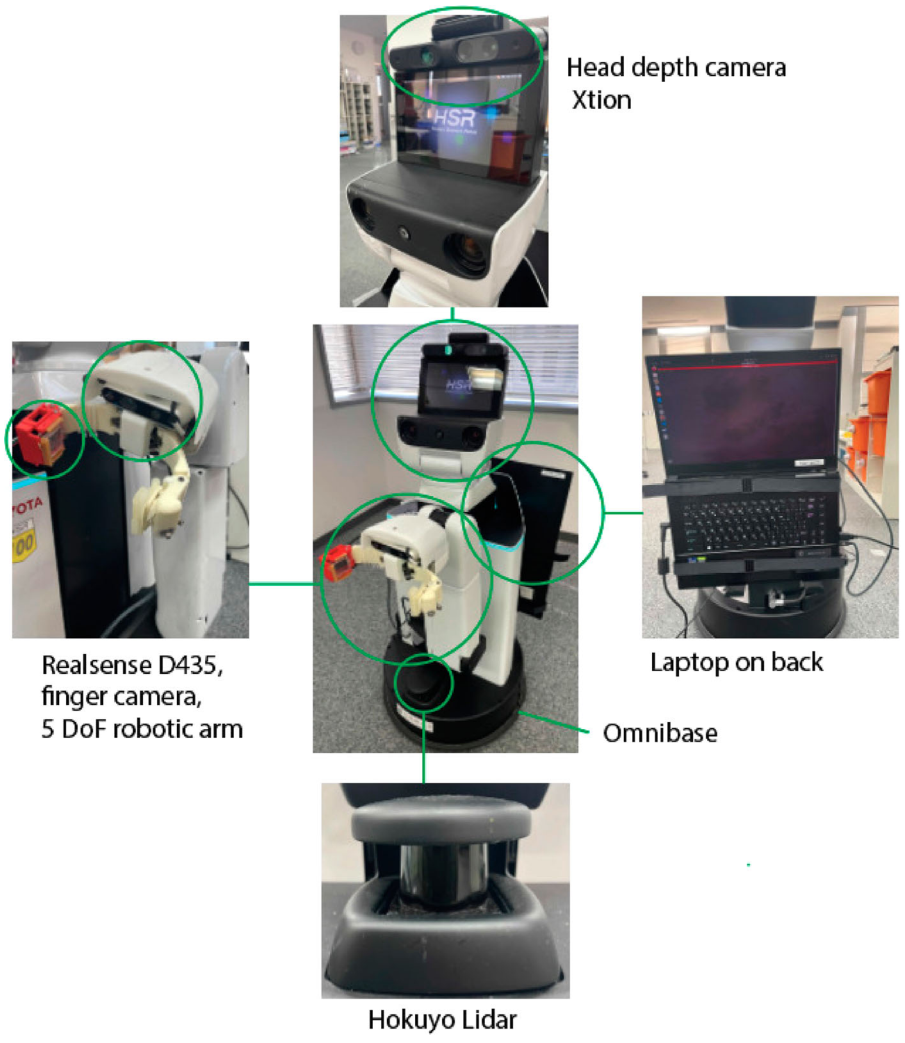


Figure 5. Customized HSR (Human Support Robot). HSR is a mobile manipulator designed to support human activities in daily life, such as nursing care and housework. It can pick objects under 500g and is designed for safety in human interaction. In the competition, we focused on clean up tasks with HSR. The team replaced a hand RGB camera for the Realsense D435 and one of the fingertips for the tactile sensor module.

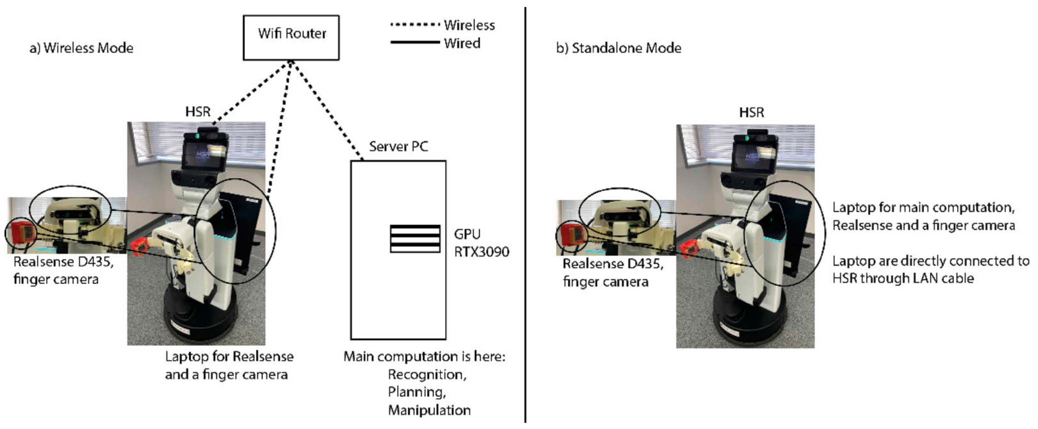


Figure 6. Wireless and Standalone mode. The wireless mode can use external powerful computational resources via a wireless network. The standalone mode computes all processes in the laptop installed on the back of HSR. This mode is much useful when the WiFi connection is weak. Our team picked standalone mode in the competition because WiFi bandwidth was limited to about 30 Mbps.

The mode selection is a trade-off between network speed and computation speed. In the competition, the participants were allowed to use WiFi connection prepared by the organizer only, whose bandwidth was 30 Mbps at the maximum. We chose the standalone mode because we found that the bandwidth was too narrow for transmitting massive data (e.g. point cloud from two RGB-D cameras and LiDAR sensor), which caused a severe delay in sensor data and critical errors in localization and navigation.

3. Recognition

We aggressively utilized multiple DNN models aiming at generalization in recognition of objects (Section 3.1), environment (Section 3.2), and human (Appendix 1) in the competition arena.

3.1. Object recognition

In the WRC2020 task, the technology for fast and accurate recognition of objects is the fundamental technique for the overall task operation. For example, if an object cannot be detected, or if detection is a slow process, it might be dangerous for collision with the object during navigation. Moreover, if the object's exact shape cannot be obtained, the robot fails to calculate the correct grasping pose, resulting in grasping failure. Furthermore, if the category of the object cannot be classified correctly, the robot cannot place the object in a defined location. Thus, as object recognition is applied to various tasks such as grasping and navigation, it is crucial to estimate the pose, shape, and category of the object fast and accurately.

Object recognition system works in two situations, such as searching the objects and confirming the object to pick and place in tidying up in Room1 (task1). To navigate the robot, it has to acquire the object's information about the pose and the category of the objects. In the proposed system, before the robots start picking the objects, it looks around the searching area using the head camera and recognizes the objects for searching. The robot stands outside of the search area near the drawer, and divides the search area into three parts (floor, lower table, and upper table), and images of each place are captured by the head camera. After deciding the object to pick, the robot moves in front of the object and investigates it using hand camera from an angle above it for confirmation. At this instant, the robot recognizes the object and plans how to grasp the object. In picking the target object from the food area's shelf in task 2b, the robot photographs each row of the shelf. The hand camera moves in front of the center of each row of the shelf, and captures an image of the left side and the right side of each row.

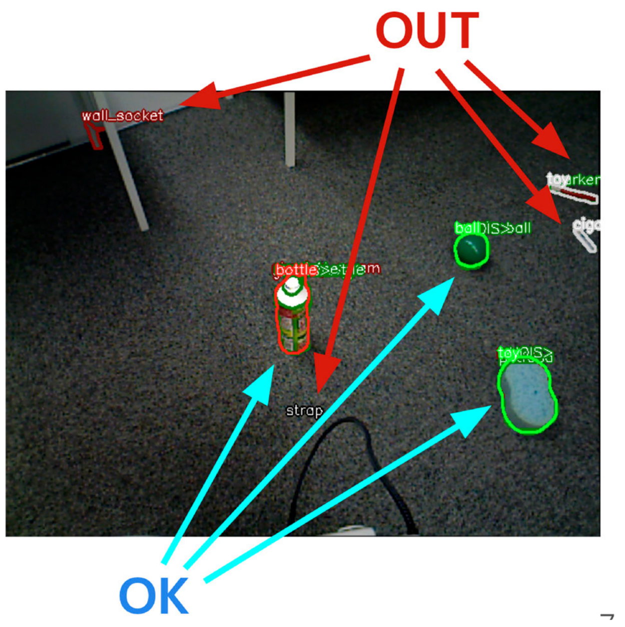


Figure 7. Object recognition flow. Color images are captured with head-mounted or hand-mounted RGB-D camera and the masked images for each object in the captured images are created using object detection module. Subsequently, the masked images are classified using the CLIP classifier. In the case of the figure, the images of a mustard bottle and a screw driver on the floor are captured with the cameras. The object detection module segments object area (namely, the area of the mustard bottle and screw driver) on the images and masks out other regions of images than those of the objects (e.g. the floors, toy airplane in the right image).

Recognition system is divided into the detection part and classification part. Owing to the two stages of object recognition module, during picking objects, the robot does not need to wait for the output from the object classification model, which usually involves extensive computations.

3.1.1. Object detection

In the object detection module, we used two object detection models, Mask R-CNN [1] in Detectron2 [24] and UOIS [2], as a combination. Mask R-CNN in Detectron2 is a widely adopted object detection model and UOIS is robust for detecting unseen objects. In the WRC2020 tidy-up task, because a lot of objects are placed in the nearest area, the system has to detect overlapping objects. Even if assembling these two models, some furniture or floors are erroneously detected as objects. The detected results were filtered to avoid misidentification using hand-designed rules: excluding objects with too small or too big pixel areas, excluding objects whose labels are not in the valid list, excluding overlapped objects, excluding objects that are not in the task1 area, and excluding objects at the edge of the image, as shown in Figure 7. Although we can obtain the object label from these two detection models, the accuracy of classification is insufficient, which motivates combining another classification module as in Section 3.1.2. Figure 8 illustrates the object recognition workflow. Mask R-CNN and UOIS are primarily used for segmentation images.

3.1.2. Object classification

After applying the segmentation mask from the detection module mentioned above to the original RGB images, the image is classified by CLIP [3], which is a recently proposed multi-modal deep learning model. CLIP comprises three parts, text encoder, image encoder, and fully connected layers. CLIP inputs the prompt, the text description of the object, and the picture, then outputs the probability of identification between the name and the picture. The advantage of CLIP over other classification models is that it is easy to tune the model to the specific situation by only editing the prompt. This technique to improve performance using text prompts for clarifying the task has been proposed in a line of studies on large language models (LLMs) and is called 'prompt tuning' [25–28]. For task1 and task2, we prepared prompts for each object that does not directly use the label name, but the handmade descriptions, such as 'red strawberry with green stem, a type of fruit' as shown in Table 2. To adapt to the task situation, we used pretrained CLIP model (ViT-B/32)[4] and only the fully connected layers after pretrained CLIP module were fine-tuned with cropped YCB image datasets [18]. The fully connected layers are trained separately for task1 (the model is trained with objects of all categories) and task2b (the model is trained with only 'food' objects), while the text prompts are shared between task1 and task2b. Besides, considering the change in the light condition of the floor pattern, we captured some images of objects which were placed on the same as the competition during preparation day. Those images were used to fine-tune the fully connected layers, and the process is very computationally light. Data augmentation is also performed in each training to make the model robust to input noise.

3.1.3. Evaluation

In addition to the CLIP model, we prepared the pretrained ResNet18 model[5] [29] as a baseline model. For the baseline ResNet model, the weight of ResNet module was fixed, and we classified the object class of given images according to k-nearest neighbors of L2 distance of the output of ResNet modules. For the CLIP model, we used inner product as a metric for object classification. Compared to the baseline ResNet [29] model, CLIP exhibits a more accurate classification ability as shown in Figure 9. Before prompt tuning, CLIP classifier made mistakes the same way as the pretrained ResNet model; for instance, lemon was classified as strawberry. However, the accuracy of classification increased after prompt tuning, as shown in Figure 10. In detail, we made prompts including color information, such as 'blue coffee can'. Besides, the phrase 'the type of fruits' improves

Table 2. Known objects appeared in the competition and the corresponding prompts, which we tuned.

Object Name	Prompt	Used in Task 2b
chips can	a photo of a pringles can	√
master chef can	a photo of a blue coffee can	√
cracker box	a photo of a box of cheez-it	√
sugar box	a photo of a yellow box of Domino sugar	√
tomato soup can	a photo of a can of campbells tomato soup	√
mustard bottle	a photo of a yellow mustard	√
tuna fish can	a photo of a StarKist can tuna	√
pudding box	a photo of a box of brown chocolate pudding jello	√
gelatin box	a photo of a box of red jello	√
potted meat can	a photo of a can of spam	√
banana	a photo of a banana, a type of fruit	√
strawberry	a photo of a red strawberry with green stem, a type of fruit	√
apple	a photo of a red apple, a type of fruit	√
lemon	a photo of a lemon, a type of fruit	√
peach	a photo of a yellowish peach, a type of fruit	√
pear	a photo of a green pear, a type of fruit	√
orange	a photo of a orange, a type of fruit	√
plum	a photo of a purple plum, a type of fruit	√
pitcher base	a photo of a blue pitcher	
pitcher lid	a photo of a pitcher lid	
bleach cleanser	a photo of a soft scrub bleach bottle	
windex bottle	a photo of a windex spray bottle	
wine glass	a photo of a wine glass	
bowl	a photo of a red bowl	
mug	a photo of a red mug	
sponge	a photo of a sponge	
plate	a photo of a plate	
fork	a photo of a red fork	
spoon	a photo of a red spoon	
spatula	a photo of a spatula	
key	a photo of a key	
large marker	a photo of a expo marker	
small marker	a photo of a expo marker	
plastic bolt	a photo of a bolt	
medium clamp	a photo of a clamp	
card	a photo of a card	
ball	a photo of a sports ball	
rope	a photo of a rope	
chain	a photo of a chain	
foam brick	a photo of a brick	
dice	a photo of a die	
marbles	a photo of a marble	
cups	a photo of a toy cup	
peg	a photo of a wooden puzzle	
toy airplane	a photo of a toy airplane	
magazine	a photo of a time magazine	
shirt	a photo of a black t-shirt	
lego duplo	a photo of a lego	
timer	a photo of a timer	
rubiks cube	a photo of a rubiks cube	

Note: For the task 2b, we trained object classifier with food objects only (denoted by √ in the table).

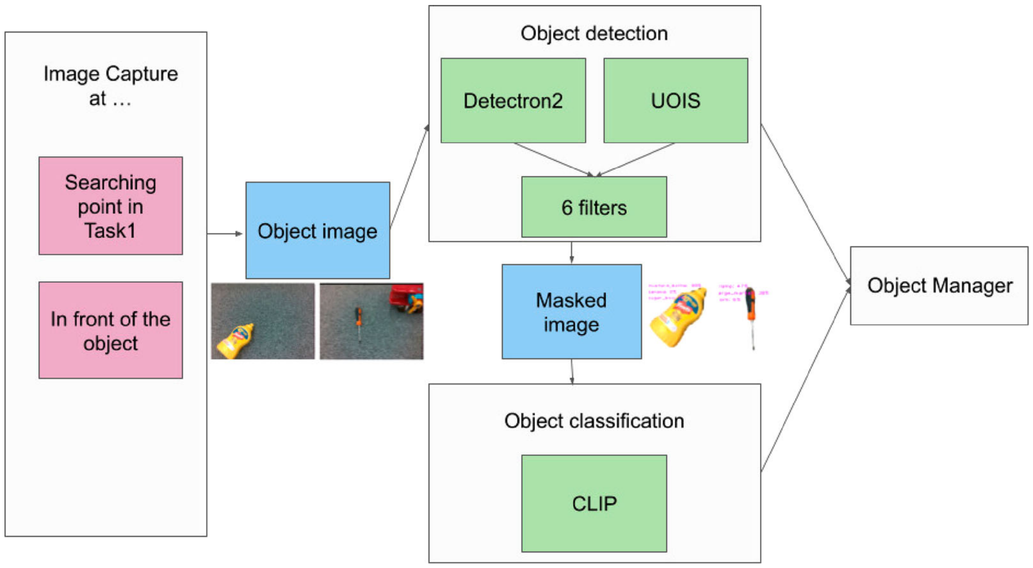


Figure 8. Example of detected objects, each visualized with a colored outline based on results by hand-crafted filters. Green objects are pickable candidates. The far red object is out-of-bounds, the red bottle overlaps with a near-duplicate detection, and the gray objects are too close to the edge of the camera view.

the classification accuracy by recognizing strawberry as a strawberry, not an apple. In addition to prompt tuning, background subtraction improved classification accuracy compared to only cropping. In each training, the five types of data augmentations were applied to the training image, translating, scaling and rotating, dropping out the rectangular regions, changing hue, saturation and value, changing brightness, and contrast and blurring [30]. Figure 11(a) illustrates successful classification among similar colored objects with the proposed recognition module. Despite these improvements, some objects were still identified incorrectly; for example, ‘Lego’ as ‘nine peg hole test’, or ‘colored wood block’ as ‘foam brick’(Figure 11 (b)). The performance gain can be attributed to the pretraining of CLIP, which is trained with massive and unrestricted paired (image and text) datasets collected from the internet. The dataset could contain some hints on the textural information on objects appearing in the competition.

3.2. Recognition of world

3.2.1. Sim-To-Real image segmentation

Since the layout of the room and positions of furniture are known beforehand, one may think of using this information as constants. For example, one may carefully measure and calculate the (3D) position of the drawer knobs beforehand and use those positions to grasp at that location in a completely open-loop fashion. However, this is a brittle and risky approach, primarily because numerous factors can cause a failure to pull the drawer, such as localization error, positions of knobs being slightly shifted, or drawers being too far in or out, causing missed grasps or unwanted collisions.

Instead of adopting this simple but risky approach, we implemented a perception module that can consistently estimate the state of the environment within view. A crucial component of this module is a single image segmentation model trained to recognize various objects and furniture in the room. Specifically, a Fully Convolutional Network (FCN) [31] based on the DeepLab-v3 architecture [32] was used to predict the class of each pixel of a given depth image.

The depth image, which was obtained from the head camera of the robot, had a resolution of 640 × 480 pixels. Units are in meters and missing values were assigned a value of zero. We deliberately decided to avoid using color (RGB) for this model to circumvent the challenge of generalization over different lighting, textures, and other visual phenomena.

The following lists the classes that the model is trained to predict.

- Background
- Wall
- Pickable object
- Shelf
- Left bin
- Right bin
- Drawer frame
- Bottom drawer
- Bottom drawer knob
- Left drawer
- Left drawer knob

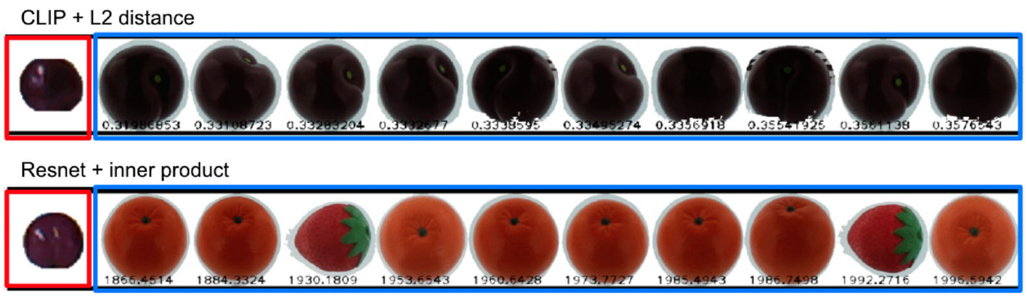


Figure 9. Classification results images for comparison between ResNet and CLIP. The upper image shows the CLIP result, and the lower image shows the ResNet result. The image surrounded by a red frame (a plum) is the recognized image and the images surrounded by a blue frame are images classified to be similar.

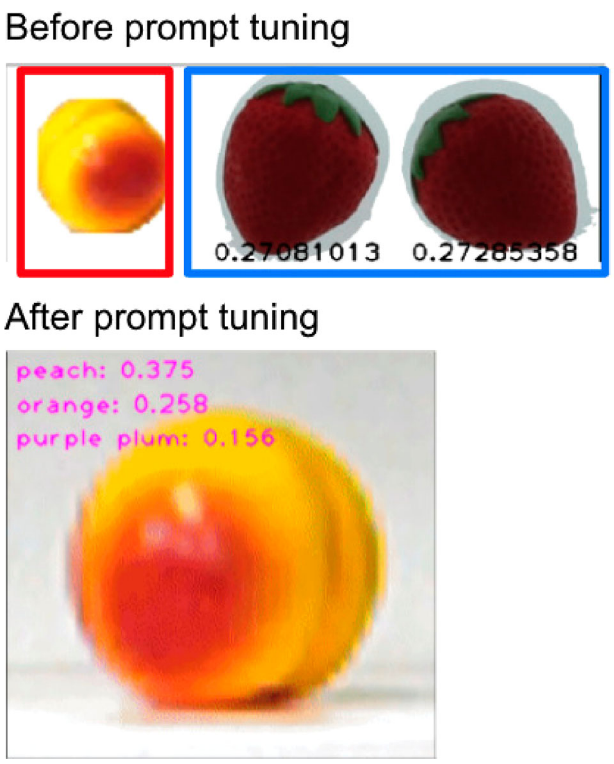


Figure 10. Comparison between with and without prompt tuning. The upper image illustrates a result of recognition without prompt tuning; the object to classify is a plum (surrounded with red frame), but the model erroneously outputs the nearest images (surrounded with blue frame) of strawberry and recognizes the target object as a strawberry. On the other hand, the lower image illustrates the result with prompt tuning; the model correctly recognizes the image of a peach.

- Top drawer
- Top drawer knob
- Miscellaneous drawer
- Tall table
- Long table A
- Long table B
- Left tray
- Right tray
- Left container
- Right container

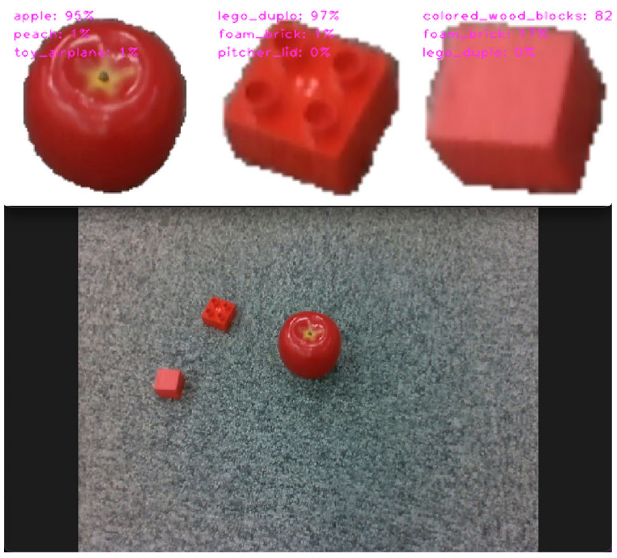
Obtaining a sufficient amount of human-annotated training data can become expensive in terms of effort, time, and monetary cost. Instead, we adopted a sim-to-real technique by generating synthetic image data and annotations from a simulation. The developed simulation uses PyBullet [33] for both physics and rendering but many of the assets, including the URDF of the HSR and furniture models, were ported from an existing Gazebo [34] simulation. The resulting simulation is depicted in Figure 12. To train a robust model, the synthetic depth images were rendered and applied with noise during training to mimic the noisiness of the real depth camera. Scenes were also generated with randomization. The following lists what was randomized for data generation.

- Robot configuration
- Wall height and thickness
- Shifted positions and rotations of furniture
- Drawer knob position, rotation, and shape
- Drawer position, including open/close state
- Shape and size of trays and containers
- Presence of miscellaneous drawers
- Number, poses, shapes, and sizes of pickable objects

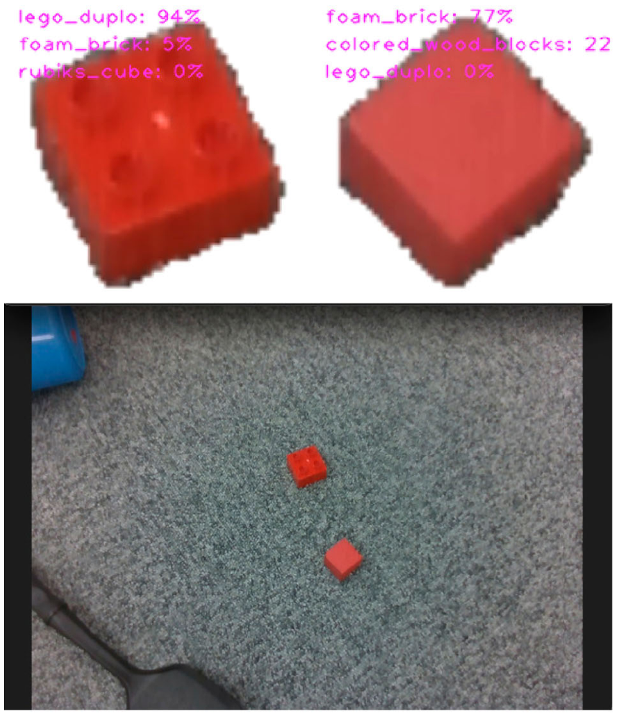
Variations in shapes were made by sampling meshes from ShapeNet [35].

3.2.2. Evaluation

Example predictions of the resulting model are presented in Figure 13. The figure demonstrates its robustness, as it can recognize objects even when the camera viewpoint or shapes and drawer knobs change. These predictions are used with depth information to obtain segmented point clouds which can then be used to estimate information, such as object positions, as shown for drawer knobs in Figure 14.



(a) Successful case



(b) Failure case

Figure 11. Examples of success and failure classification among similar objects: apple, Lego and wood block.In (a) our classifier can recognize similar color objects with subtle texture difference, but in (b), the colored wood block is recognized as a form brick (while Lego is classified correctly). (a) Successful case; (b) Failure case.

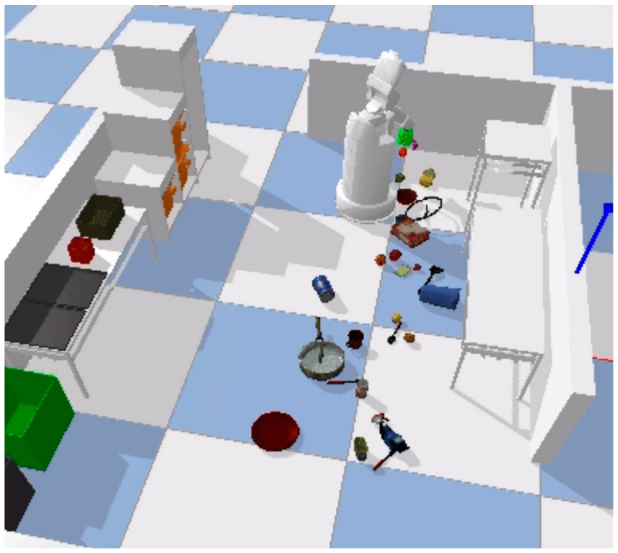


Figure 12. Simulation environment used for image data generation. The room layout and most of the furniture reflect the real competition environment.

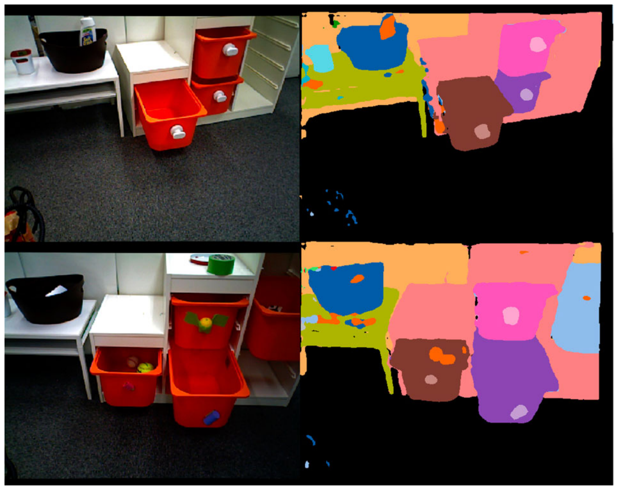


Figure 13. Two samples of model predictions on real data. The model can generalize to varying viewpoints, positions, and shapes.

4. Object manipulation and motion planning

4.1. Object manipulation

4.1.1. Grasp pose prediction

Grasping becomes an important component of the robotic system to reliably place the objects scattered around the room at their assigned locations. At the same time, grasping is challenging because different objects in different configurations can afford different stable grasp poses. We attempted various grasp planning methods as illustrated in Figure 15. In the main text, we describe the methods that we mainly used in the competition. See Appendix 2 for the rest.

PCA

The first and most simple method we attempted was using principal components analysis (PCA). Given the point cloud of a target object, the points were projected to the XY plane and PCA with 2 components used to fit this data. A grasp was calculated so that the gripper is positioned above the center of the object and the

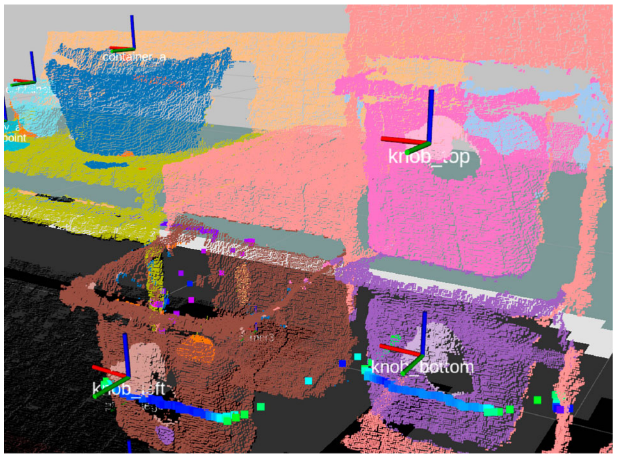


Figure 14. The perception system processes the model predictions with depth information to accurately estimate the state of the environment, such as the positions of drawer knobs.

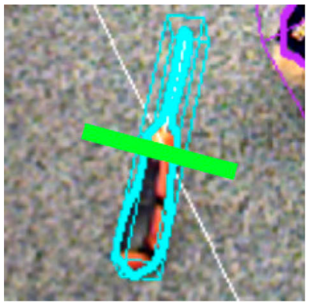
fingers are aligned along the 2nd component of PCA. Intuitively, this results in grasps that are perpendicular to the longer axis, which is useful for estimating stable grasps of long objects, such as a banana. One assumption that this approach makes is that objects can be grasped in a top-down fashion, which is true in most cases in Task1. In fact, we found this method to be the most reliable over the course of development.

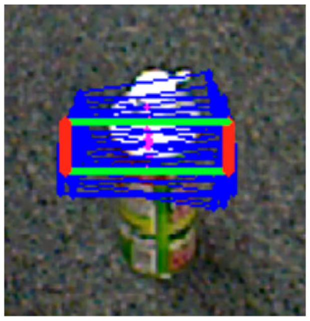
Sim-to-Real Transfer We attempted a custom DNN-based model which was trained in the simulator described in Section 3.2. The model architecture and training method is largely based on [39], a reinforcement learning method. The model takes a heightmap as input which can be generated by projecting a point cloud onto the XY plane and outputs a map of Q-values, which corresponds to the likelihood of success for a grasp at the pixel location. Possibly owing to the resolution of the input heightmap, we found this method to lead to grasps that were occasionally off by a few centimeters.

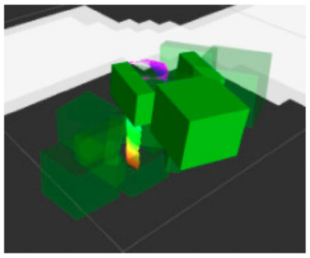
Ultimately, we opted to use the last method for generating grasp candidates (Figure 16) and combined it with the PCA method. The grasp pose was selected depending on the Q-value of the model (Figure 17). If the Q-value was above a set threshold, meaning that it was very confident, the model was used; else, PCA was used. Since we found PCA to be sufficient for most objects as they are simple and convex, this threshold was set very high.

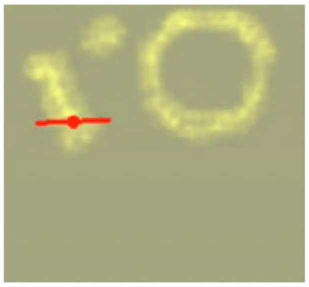
4.1.2. Grasp detection with hand-made tactile sensor

The judgment of the success or failure of the grasp is crucial for the robot to perform the pick and place operation


(a) PCA


(b) Multi-Object Multi-Grasp [36]


(c) GraspNet [37]


(d) GGCNN [38]

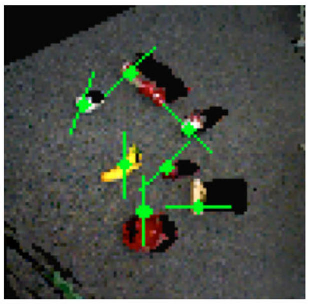

(e) Grasp FCN

Figure 15. Visualizations of the attempted grasp estimation methods. Both PCA and GraspNet take as input a point cloud of a single object, but PCA calculates a 4 degrees of freedom (DoF) grasp while GraspNet can produce 6DoF grasps. Both the grasp detector and GGCNN take as input a depth image but the detector outputs bounding boxes while GGCNN outputs a grasp quality heatmap. The Grasp FCN also outputs a heatmap but takes as input a more view-invariant heightmap. See Appendix 2 for the method illustrated in (b)–(d). (a) PCA; (b) Multi-Object Multi-Grasp [36]; (c) GraspNet [37]; (d) GGCNN [38]; (e) Grasp FCN.

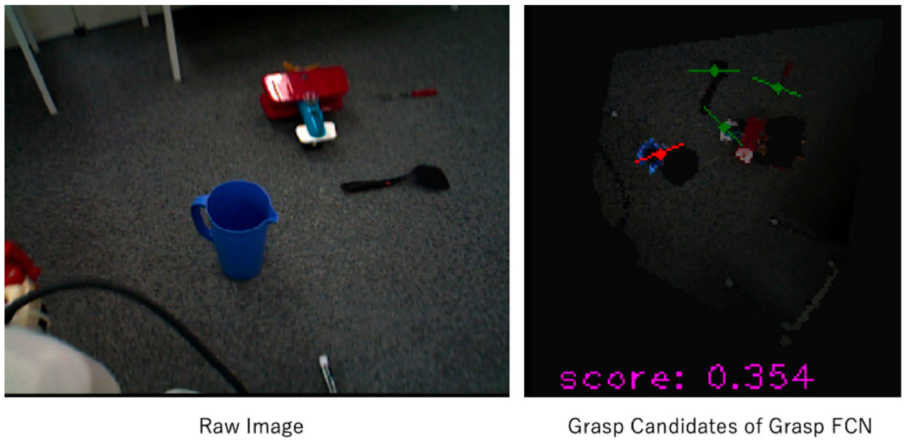


Figure 16. Left: RGB image from a head camera. Grasp FCN generates grasp candidates with the quality score. Right: The red line illustrates the pose with the highest score, and the green illustrates other candidates.

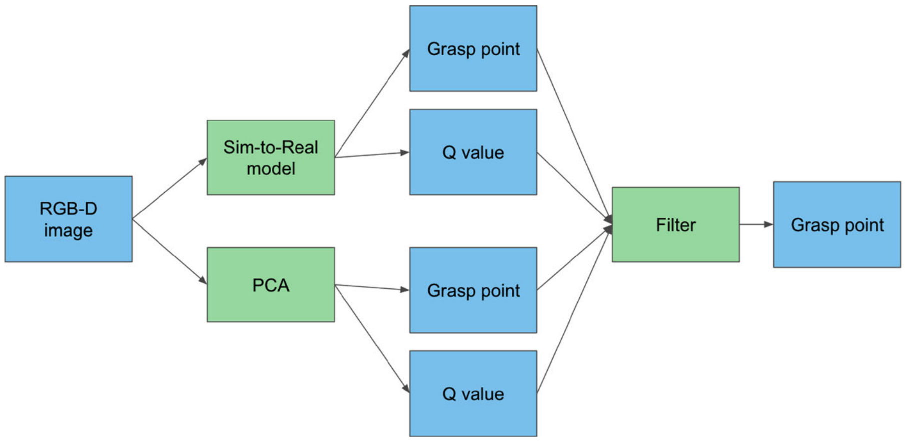


Figure 17. Grasp pose prediction flow.

correctly. In HSR, it is relatively easy to implement the grasp detection according to the angle of the gripper and the 6-axis force sensor on the wrist. However, it is difficult to detect a successful grasp of a light object, such as a pen or a key that appeared in the WRC2020 competition.

We created a tactile sensor based on Ref. [40], which is composed of black painted lead balls, transparent gels, acrylic board, a fish-eye camera, and 3D printed parts. When force is applied to the sensor, the gel deforms and the lead balls move. The camera could capture the markers' movement and use it as the sensor value. The advantages of the proposed tactile sensor is the ease of fabrication and the affordability of the amount of material. The fabrication method of the sensor and the 3D printer parts are available as open-source, and the materials are relatively cheap (about 60 USD) and easy to purchase. For simplification of the fabrication process, we replaced a hand-made gel with a pair of non-slip sheets of furniture and place markers between the two sheets.

By using this sensor, we detected the grasping state of HSR. There are three states, 'open', 'objects in hand', and 'nothing in hand'. These three states are categorized by marker positions.

For the calibration of marker positions and grasping state, we registered the initial markers positions in open and closed state. The marker positions in the open and closed states were registered using the Blob detector[6] from a binalized camera image, as in Figure 18(c).

Figure 18 illustrates how the marker positions have been registered and processed. In Figure 18(e–g), the detected markers positions by the Blob detector are illustrated with white dots. For comparison, the marker positions of registered 'nothing in hand' is illustrated by red points in image e–g. In Figure 18(f), white dots and red points are almost in the same positions. In the case of 'open' state (Figure 18(e)), the number and arrangement of white dots are different from the registered red dots, which means the Blob detector cannot detect the lead

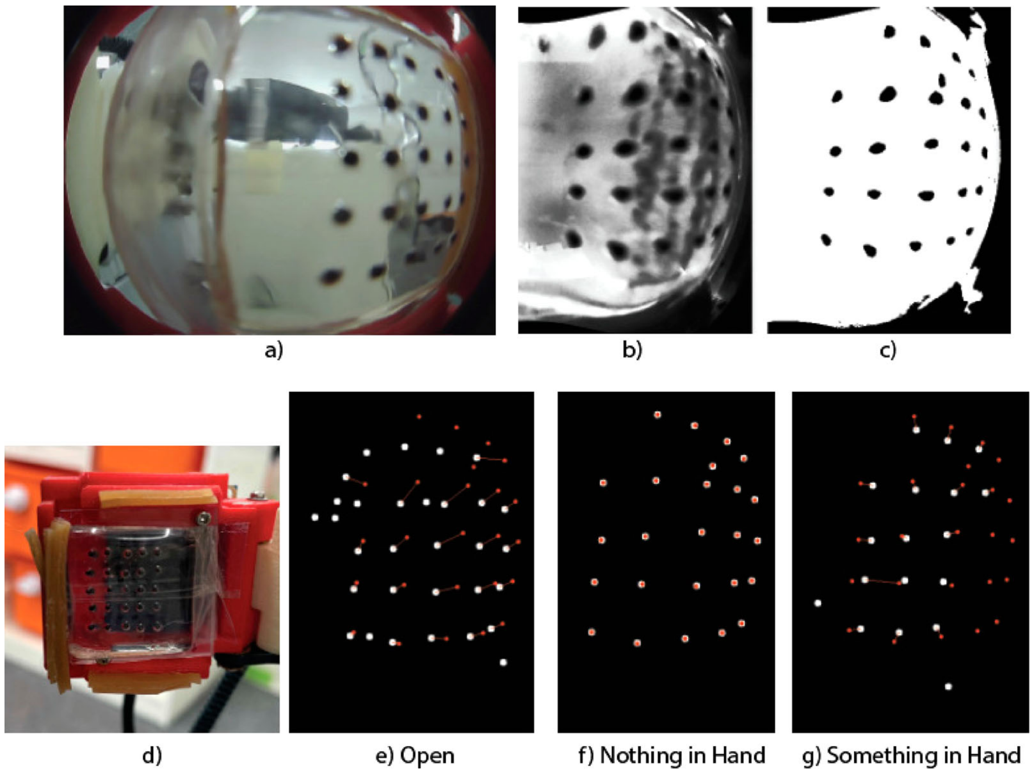


Figure 18. Grasp detection mechanism. (a) Raw image taken by the finger camera. (b) Gray image of the image (a). (c) Binarized image by OTSU's method. (d) Fingertip attached to HSR hand. The nails are made of rubber and lead balls are embedded into the gel. (e–g) Examples of marker positions. 'Open' and 'Nothing in Hand' images are registered, and when the markers positions deviate from both registered positions, it is detected as 'Something in Hand'. (`https://docs.opencv.org/4.x/d7/d4d/tutorial_py_thresholding.html`).

points correctly since the light comes into the finger camera from the open gripper. In the case of 'something in hand' state (Figure 18(g)), the white points are translated from red dots, which means that some object in the hand pushes the gel and the lead balls move accordingly.

Figure 19(a) is a successful example of grasp detection with a spoon. On the right side of the image, the detected marker dots and red points are deviated and grasping can be detected. This object could not be detected by the wrist sensor or the open-angle of the gripper. Figure 19(b) represents a failure case. The key was caught in the rubber fingernail (Figure 19(d)) and almost no displacement occurred on the marker, as shown in Figure 19(b).

Figure 20 shows relationships between deviations of marker positions and thickness of flat objects in hand. Although the proposed tactile sensor is insensitive to a flat and extremely thin object like paper ($\sim$0.1 mm) because the marker does not move a lot, it can detect the grasp of considerably thin and flat object like a card ($\sim$1 mm) or styrofoam board ($\sim$1 cm) since the markers deviate to some extent.

4.2. Navigation

Moving from the current location to the destination while avoiding obstacles, a task termed navigation, is a basic operation for operating service robots at home. For the navigation of household service robots, it is particularly important to accurately recognize objects that are difficult to recognize with sensors, such as small, transparent, or flexible objects. Additionally, moving in response to the domestic environment, which changes constantly, is also important.

To navigate while avoiding collision with an object, it is necessary to have a positional relationship with the object. Therefore, scan data and point cloud data that can directly acquire distance information are often used as sensors for navigation. However, among various objects in a home, there are many objects that are difficult to recognize only from the shape information of the object (key, pen, clothes, etc.).

To avoid these objects, we use the following four data for obstacle information to be reflected in the costmap for path planning.

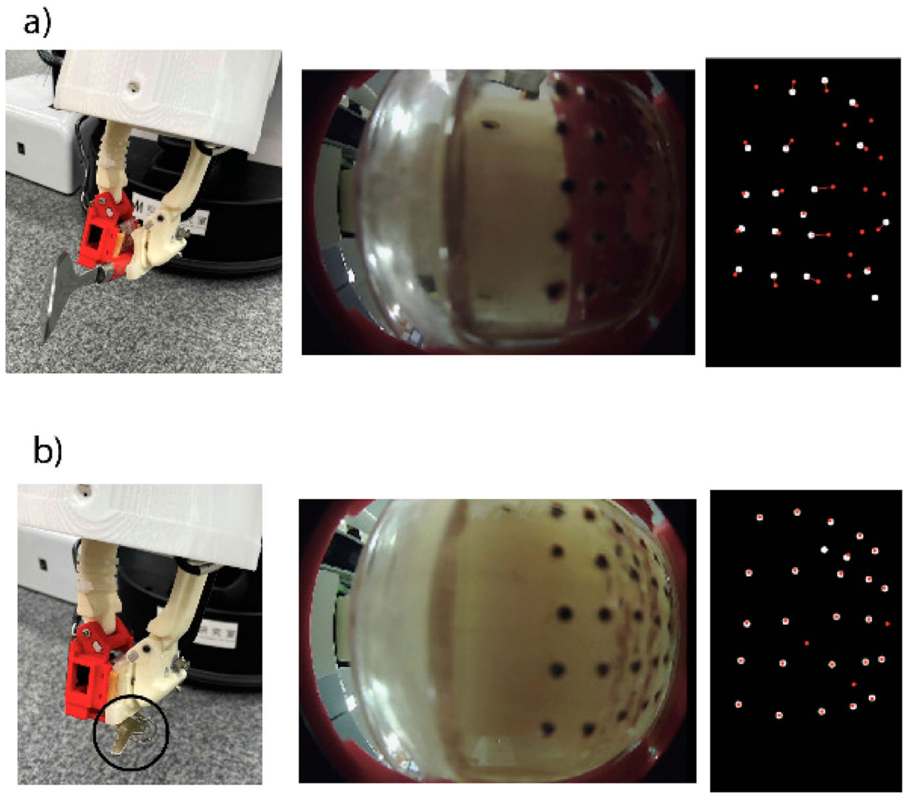


Figure 19. Successful and failure cases of grasping. (a) Successful case. Spoon can be detected using the tactile sensor. (b) Failure case. The key hangs on between the edge parts of the tactile sensor module and the opposite gripper, which does not change the position of the markers over a threshold.

(1) Map created in advance (information on furniture and walls excluding objects).
(2) 2D-LiDAR scan.
(3) Point cloud filtered from the point cloud that can be taken with the depth camera.
(4) Point cloud of the object recognized from an image captured by the head RGB-D camera.

The information in (1), (2), and (3) enables navigation by avoiding collisions with objects that can be recognized as shapes obtained from scan data and point clouds.

In (3), the following filtering process is implemented. The point cloud obtained from the depth camera is first downsampled and outliers removed. Subsequently, the maximum height and minimum height of each voxel in the robot's base link coordinate are calculated, and if the value is below a threshold value, it is judged as the ground and removed. And while the robot is carrying an object, it often determines its arm as an obstacle, so it removes the robot's own point cloud from link information and 3D model of the robot.

It is possible to avoid objects that cannot be seen from the 2D-LiDAR mounting position and objects that are indistinguishable from the ground by the point cloud height filtering process, by treating the object recognized by the camera in (4) as an obstacle.

To reduce the impact of cumulative error in each estimation module, we implement position-based navigation relative to real-time recognized objects rather than absolute position-based navigation on pre-made maps.

Table 3 lists the source of information on the target position for navigation. Except for the start position of each task, when moving to an object or furniture, the goal is calculated from the real-time recognition result. This enables robust navigation to the cumulative error of self-position estimation by making heavy use of relative position-based navigation with recognition results, instead of absolute position-based navigation of pre-created maps.

However, it may occur several times while the robot is running that the route to the target cannot be calculated due to an estimation error in the object recognition position.

If the robot cannot calculate the route to the goal for a certain period of time, the distance that the robot takes as a margin from the object used for path calculation is gradually narrowed from the point cloud of (4). At this

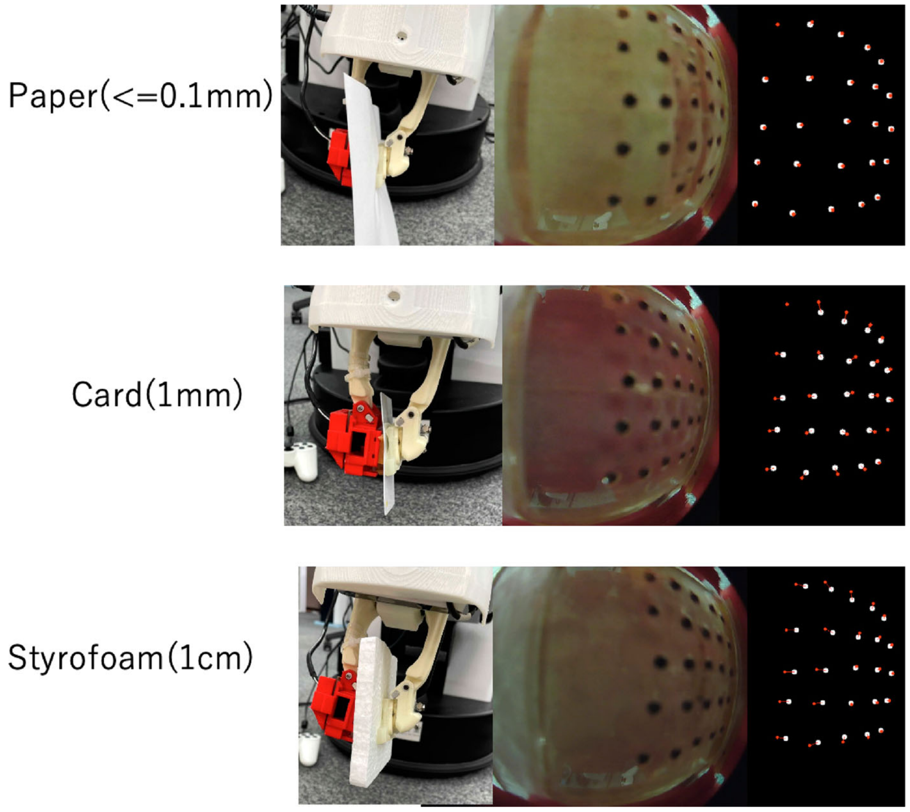


Figure 20. Marker deviations when flat and objects are in hand; paper(∼ 0.1 mm), a card(∼ 1 mm), and a styrofoam board (∼ 1 cm). While the paper is too thin to detect marker deviation with the proposed tactile sensor, it can detect the deviation of marker with considerably thin objects.

Table 3. Recognition information used for the target position of the navigation.

Target Position	Recognition Information
Object	Object Recognition (3.1)
Furniture	Sim-To-Real Image Segmentation (3.2)
Person	Human Recognition (??)
Task Start Point	Map

time, instead of completely eliminating the cost of the narrowed part, we leave some cost, allowing the robot to move as far away from the object as possible after recalculating the route.

In the case of Figure 21(a), small objects with a height that cannot be detected from 2D-LiDAR, namely, a pitcher lid, a small ball, and a T-shirt, are placed on the floor. We detect these small objects from head-mounted RGB-D camera using object recognition modules described in Section 3 and cut out the corresponding point clouds of the objects as shown in Figure 21(b). The costmap for navigation is updated using both LiDAR point cloud and detected object point clouds from head camera, and the path is planned as in Figure 21(c). Since this procedure of object detection, merging point clouds, and updating costmap and path plan is done in a closed-loop manner, the robot can avoid collisions dynamically.

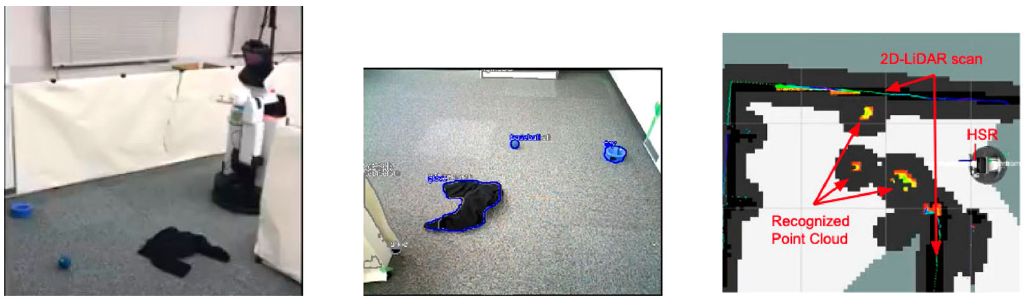
5. Integrated experiment

5.1. Preparation time at the site of the competition

At the WRC, each team brought their robot to the venue, and after four days of preparation, the competition was held over three days. Two competition venues and three practice areas were set up, and the practice areas were made similar to the competition venue.

At the first step of preparation, our team created a map of the field using LiDAR mounted on HSR. Subsequently, we captured pictures of objects at the venue and incorporated the data into the training model to tune the CLIP and other object recognition, primarily because the lighting at the venue was different from the environment in the laboratory, which may impair the recognition performance.

In particular, in the competition, we used the object classifier fine-tuned with a dataset that we collected in the laboratory (748 images) and the competition venue (556 images). We did not use the original YCB dataset because the masking strategy is different from ours (explained in Section 3.1.1), which may cause deterioration of the recognition performance. At the venue, we captured the images for the entire YCB objects by placing them on the floor of the arena for several times, and created a



(a) Third Person View (b) Recognition Result (c) Costmap reflecting recognition result

Figure 21. An example of obstacle avoidance during navigation. In an environment like (a), the robot recognizes obstacles in two ways. The first one is to use 2D-LiDAR scan information to recognize obstacles above of a LiDAR sensor height e.g. large objects, furniture, and walls. The second is to use RGB-D camera images to recognize low-height objects, such as clothes and balls that 2D-LiDAR cannot recognize. The object surrounded by blue in (b) is the object recognized from the camera image. Objects recognized in these two ways are reflected in the cost map in (c) used for path planning. (a) Third Person View; (b) Recognition Result; (c) Costmap reflecting recognition result.

dataset of masked images using the object detection module (described in Section 3.1.1). The class of corrected images was annotated manually, and the fully-connected layers of classifier were fine-tuned with the combined dataset.

5.2. *Result*

Five teams competed at the real venue. Two competition rooms had the same setup were set up at the venue, and the two teams started discussing at the same time, and the scores were used to determine the winner. First, a round-robin tournament was held as a preliminary round, and then the top four teams competed in a tournament format. Our team played six trials in total. Table 4 lists the summarized scores changes (see Appendix 3 for the full score sheets).

Our team got the best score in the semifinals (the fifth game). In this case, we cleaned up 11 items in task1 and managed to complete task2 correctly.

For the task 1, the statistics for all six trials areas are noted in Table 5. For object recognition (category classification), the overall success rate across categories was 92% and 'Food', 'Shape', and 'Tool' objects are perfectly categorized, while 'task' category had a lower rate of 78%. In the competition, the some 'unknown' objects outside YCB dataset (not listed in the competition rule book) appeared, which were required to be stored in the place of either the corresponding category (announced in the trial) or 'unknown' category. We treated these unknown objects in the same way as the pre-announced objects in the rulebook by assuming that the recognition module generalized well even for the unknown objects, that is, we did not explicitly classify the 'known' and 'unknown'.

Table 5. Success ratio of object recognition and tidy up in the competition.

Object Category	Recognition	Tidyup	Both Recognition and Tidyup
Food(Listed in the rule book)	1.0 (11/11)	0.82 (9/11)	0.72 (8/11)
Kitchen(Listed in the rule book)	0.88 (14/16)	0.69 (11/16)	0.5 (8/16)
Shape(Listed in the rule book)	1.0 (11/11)	1.0 (11/11)	1.0 (11/11)
Tool(Listed in the rule book)	1.0 (20/20)	0.45 (9/20)	0.45 (9/20)
Task(Listed in the rule book)	0.78 (7/9)	0.56 (5/9)	0.33 (3/9)
Unknown Objects (**Not** Listed in the rule book)	0.5 (3/6)	0.5 (3/6)	0.33 (2/6)
Boss Item (Shachihoko)	N/A	0.5 (2/4)	N/A
Total	0.92 (66/73)	0.65 (50/77)	0.56 (41/73)

Note: Since the boss item was placed at the same location in each trials, the recognition was not required.

Table 4. Competition results (6 trials).

Trial	1st	2nd	3rd	4th	5th	6th
Task1 tidy up counts	5	8	9	7	11	10
Task1 score	145	185	240	145	320	250
Task2a success	True	False	True	False	True	True
Task2b target	tomato soup can	sugar box	peer	apple	spam	chips can
Task2 score	390	200	390	100	410	150
Restart Robot	1	0	1	1	0	2
Total Score	535	385	630	245	730	400

The success rate of recognition of unknown objects was 50% (2 out of 4).

The overall probability of successful grasping was 65% in all six trials. While 'Shape' category had the highest success rate of 100%, 'tool' category had the lowest success rate of 45%. The main reason was the difference in difficulty. 'Shape' objects are mostly sphere shaped and easy to get a good grasping position. On the other hand, 'tool' objects are relatively small and difficult items. In terms of task speed, the average time per object in task1, including failure and restart times, was 98.7 s. The average time per one successful object was 56.2 s. It is quite slow compared to human cleanup speed but is deemed as a good result in the WRC2020 competition.

6. Conclusion

In the paper, we present the entire robot system developed for tidying up tasks that achieved the second prize in World Robot Challenge, a world-wide robot competition held in September 2021. Our solution leveraged the data-driven approach for managing variations in home environments rather than directly pre-programming for edge cases. We demonstrate the generalization ability for deviations in the environment and flexibility to update modules using corrected data during trials. Moreover, we also demonstrate the system design for ensuring high throughput even if it contains modules deep neural networks, which requires high-computational loads.

While the tidy-up task in WRC and our corresponding solution involves some essential aspects of service robots in home environments, for example, precise object recognition, object manipulation, and navigation, they are still insufficient for realizing generalist household robots. For instance, safe manipulation and navigation in the environment with a dynamic environment (e.g. moving objects and people), and social interaction between robots and human are the lacking aspects in the WRC task. Further development and standardized benchmarks will bring generalist home robots into reality.

As future work, we plan to extend our robot system to accept massively corrected datasets by deployments [41] for adapting to more diversified environments than those used in the WRC2020 competition and establish a methodology for developments and operations of data-driven service robot systems.

Notes

1. http://xtionprolive.com/asus-xtion-pro-live
2. https://www.intelrealsense.com/depth-camera-d435/
3. https://us.msi.com/Laptop/GS66-Stealth-10UX/Overview
4. Obtained from https://github.com/openai/CLIP.
5. Obtained from https://github.com/pytorch/vision
6. https://docs.opencv.org/3.4/d0/d7a/classcv_1_1SimpleBlobDetector.html

Notes on contributors

Tatsuya Matsushima received B.S and M.S degrees from the University of Tokyo in 2018 and 2020, respectively. He is currently pursuing Ph.D. degree at the Graduate School of Engineering, the University of Tokyo. He is working on deep learning and its applications to robotics.

Yuki Noguchi is a Master's student at the Graduate School of Engineering at the University of Tokyo.

Jumpei Arima is an engineer at Matsuo Institute. He received B.S and M.S degrees from Meiji University in 2019 and 2021.

Toshiki Aoki is a senior student at the University of Tokyo. His area of research lies in applying machine learning techniques to Human-Computer Interaction and Computer Graphics.

Yuki Okita is a Master's student at the University of Tokyo. He received his BS degree from the University of Tokyo in 2021. His research area includes mathematical optimization, especially continuous optimization.

Yuya Ikeda is a senior student at the Faculty of Engineering, the University of Tokyo.

Koki Ishimoto is a researcher at the Tokyo Institute of Technology. He received his BS and MS degrees from the University of Tokyo in 2019 and 2022.

Shohei Taniguchi is a Ph.D. student at the University of Tokyo. His research interests are in deep generative models, reinforcement learning, and their applications in robotics.

Yuki Yamashita received his B.E. degree from the University of Tokyo in 2021. He is currently a Master's student at the Graduate School of Information Science and Technology, the University of Tokyo, researching natural language processing and computer vision. He is also a researcher at the University of Tokyo Hospital, working on deep learning applications in medical images.

Shoichi Seto is a senior student at the Faculty of Engineering, the University of Tokyo. His research area is the application of artificial intelligence in space projects.

Shixiang Shane Gu is a Senior Research Scientist at Google AI, Brain Team and a Visiting Associate Professor (Adjunct Professor) at the University of Tokyo, researching deep learning, reinforcement learning, probabilistic machine learning, and robotics. Shane holds PhD in Machine Learning from the University of Cambridge and the Max Planck Institute for Intelligent Systems, supervised by Richard E. Turner, Zoubin Ghahramani, and Bernhard Schölkopf. Shane holds B.ASc. in Engineering Science from the University of Toronto, supervised by the thesis advisor Geoffrey E. Hinton. Shane previously was also a visiting scholar at the Department of Computer Science at Stanford University hosted by Emma Brunskill. Shane's academic work received Best Paper Award at CoRL 2019, Google Focused Research Award, Cambridge-Tübingen PhD Fellowship, and NSERC Scholarship, and was featured in Google Research Blogpost and MIT Technology Review.

Yusuke Iwasawa is an assistant professor at the Graduate School of Engineering, the University of Tokyo. He received his BS and MS degrees from Sophia University in 2012 and 2014, and he received his Ph.D degree from the University of Tokyo in 2017. He is working on deep learning, especially transfer learning and its applications.

Yutaka Matsuo is a professor at the Graduate School of Engineering, the University of Tokyo. He received his BS, MS, and Ph.D. degrees from the University of Tokyo in 1997, 1999, and 2002. After working at the National Institute of Advanced Industrial Science and Technology (AIST) and Stanford University, he joined the faculty of University of Tokyo in 2007. At Japan Society for Artificial Intelligence (JSAI), he has served as Editor-in-chief and the chair of the ELSI committee and has been a board member since 2020. He is the president of Japan Deep Learning Association (JDLA), and a member of the board of directors at SoftBank Group Corp. He is working on artificial intelligence, especially on deep learning and web mining.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Tatsuya Matsushima http://orcid.org/0000-0002-1537-7770

References

[1] He K, Gkioxari G, Dollár P, et al. *Mask R-CNN*. Proceedings of the IEEE International Conference on Computer Vision (ICCV); 2017.
[2] Xie C, Xiang Y, Mousavian A, et al. Unseen object instance segmentation for robotic environments. IEEE Trans Robot. 2021;37(5):1343–1359.
[3] Radford A, Kim JW, Hallacy C, et al. Learning transferable visual models from natural language supervision. In: International Conference on Machine Learning. PMLR. 2021. p. 8748–8763.
[4] Lee J, Hwangbo J, Wellhausen L, et al. Learning quadrupedal locomotion over challenging terrain. Sci Robot. 2020;5(47):Article ID eabc5986.
[5] Yang Y, Caluwaerts K, Iscen A, et al. Data efficient reinforcement learning for legged robots. In: Conference on Robot Learning. Osaka: PMLR. 2020. p. 1–10.
[6] Joshi S, Kumra S, Sahin FRobotic grasping using deep reinforcement learning. In: 2020 IEEE 16th International Conference on Automation Science and Engineering (case). 2020. p. 1461–1466.
[7] Berscheid L, Meißner P, Kröger T. Self-supervised learning for precise pick-and-place without object model. IEEE Robot Autom Lett. 2020;5(3):4828–4835.
[8] Basiri M, Piazza E, Matteucci M, et al. Benchmarking functionalities of domestic service robots through scientific competitions. KI – Kunstl Intell. 2019;33(4):357–367.
[9] Yamamoto T, Takagi Y, Ochiai A, et al. Human support robot as research platform of domestic mobile manipulator. In: Robot World Cup. Sydney: Springer. 2019. p. 457–465.
[10] Matamoros M, Seib V, Memmesheimer R, et al. Robocup@ home: summarizing achievements in over eleven years of competition. In: 2018 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). Torres Vedras: IEEE. 2018. p. 186–191.
[11] Matamoros M, Viktor S, Paulus D. Trends, challenges and adopted strategies in robocup@ home. In: 2019 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC). Porto: IEEE. 2019. p. 1–6.
[12] Lima PU, Azevedo C, Brzozowska E, et al. Socrob@ home. KI – Kunstl Intell. 2019;33(4):343–356.
[13] van der Burgh M, Lunenburg J, Appeldoorn R, et al. Tech united eindhoven@ home 2019 champions paper. In: Robot World Cup. Sydney: Springer. 2019. p. 529–539.
[14] Song D, Kang T, Yi J, et al. Robocup@ home 2021 domestic standard platform league winner. In: Robot World Cup. Springer. 2021. p. 291–301.
[15] Haarnoja T, Ha S, Zhou A, et al. Learning to walk via deep reinforcement learning. In: Robotics: Science and Systems. Messe Freiburg. 2019.
[16] Mahler J, Matl M, Satish V, et al. Learning ambidextrous robot grasping policies. Sci Robot. 2019;4(26):Article ID eaau4984.
[17] Shridhar M, Manuelli L, Fox D. Cliport: What and where pathways for robotic manipulation. In: Conference on Robot Learning. London: PMLR. 2022. p. 894–906.
[18] Calli B, Walsman A, Singh A, et al. Benchmarking in manipulation research: using the yale-cmu-berkeley object and model set. IEEE Robot Autom Mag. 2015;22(3):36–52.
[19] World Robot Summit 2020 Partner Robot Challenge Real Space Rule & Regulations. https://wrs.nedo.go.jp/wrs2020/challenge/download/Rules/DetailedRules_Partner_EN.pdf. 2021.
[20] Hafi LE, Isobe S, Tabuchi Y, et al. System for augmented human–robot interaction through mixed reality and robot training by non-experts in customer service environments. Adv Robot. 2020;34(3–4):157–172.
[21] Hafi LE, Ricardez GAG, von Drigalski F, et al. Software development environment for collaborative research workflow in robotic system integration. Adv Robot. 2022;36(11):533–547.
[22] Quigley M, Conley K, Gerkey B, et al. Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software. Vol. 3. Kobe, Japan. 2009. p. 5.
[23] Yamamoto T, Terada K, Ochiai A, et al. Development of human support robot as the research platform of a domestic mobile manipulator. ROBOMECH J. 2019;6(1):1–15.
[24] Wu Y, Kirillov A, Massa F, et al. Detectron2. https://github.com/facebookresearch/detectron2. 2019.
[25] Brown T, Mann B, Ryder N, et al. Language models are few-shot learners. Adv Neural Inf Process Syst. 2020;33:1877–1901.
[26] Lester B, Al-Rfou R, Constant N. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:210408691. 2021;.
[27] Zhang X, Iwasawa Y, Matsuo Y, et al. Amortized prompt: lightweight fine-tuning for clip in domain generalization. arXiv preprint arXiv:211112853. 2021.
[28] Kojima T, Gu SS, Reid M, et al. Large language models are zero-shot reasoners. arXiv preprint arXiv:220511916. 2022.
[29] He K, Zhang X, Ren S, et al. Deep residual learning for image recognition. In: Proceedings of the IEEE

Conference on Computer Vision and Pattern Recognition. Las Vegas, NV: IEEE. 2016. p. 770–778.

[30] Buslaev A, Iglovikov VI, Khvedchenya E, et al. Albumentations: fast and flexible image augmentations. Information. 2020;11(2):Article ID 125.

[31] Long J, Shelhamer E, Darrell T. Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. Boston, MA: IEEE. 2015. p. 3431–3440.

[32] Chen LC, Papandreou G, Schroff F, et al. Rethinking atrous convolution for semantic image segmentation. arXiv preprint arXiv:170605587. 2017.

[33] Coumans E, Bai YP. a python module for physics simulation for games, robotics and machine learning. http://pybullet.org. p. 2016–2021.

[34] Koenig N, Howard A. Design and use paradigms for gazebo, an open-source multi-robot simulator. In: 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE CAT. no.04ch37566). Vol. 3. Sendai: IEEE. 2004. p. 2149–2154.

[35] Chang AX, Funkhouser T, Guibas L, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:151203012. 2015.

[36] Chu FJ, Xu R, Vela PA. Real-world multiobject, multigrasp detection. IEEE Robot Autom Mag. 2018;3(4):3355–3362.

[37] Mousavian A, Eppner C, Fox D. 6-dof graspnet: Variational grasp generation for object manipulation. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. Seoul: IEEE. 2019. p. 2901–2910.

[38] Morrison D, Corke P, Leitner J. Learning robust, real-time, reactive robotic grasping. Int J Rob Res. 2020;39 (2–3):183–201.

[39] Zeng A, Song S, Welker S, et al. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). Madrid: IEEE. 2018. p. 4238–4245.

[40] Yamaguchi A, Atkeson CG. Combining finger vision and optical tactile sensing: Reducing and handling errors while cutting vegetables. In: 2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids). Cancun: IEEE. 2016. p. 1045–1051.

[41] Matsushima T, Furuta H, Matsuo Y, et al. Deployment-efficient reinforcement learning via model-based offline optimization. In: International Conference on Learning Representations. 2021. https://openreview.net/forum?id=3hGNqpI4WS.

[42] Jiang Y, Moseson S, Saxena A. Efficient grasping from rgbd images: Learning using a new rectangle representation. In: 2011 IEEE International Conference on Robotics and Automation. Shanghai: IEEE. 2011. p. 3304–3311.

[43] Sundermeyer M, Mousavian A, Triebel R, et al. Contact-graspnet: Efficient 6-dof grasp generation in cluttered scenes. In: 2021 IEEE International Conference on Robotics and Automation (ICRA). Xi'an: IEEE. 2021. p. 13438–13444.