applied sciences

MDPI

Article

Unified Software Platform for Intelligent Home Service Robots

Jae-Bong Yi, Taewoong Kang, Dongwoon Song and Seung-Joon Yi *

Department of Electrical Engineering, Pusan National University, Busan 46241, Korea; niteofhunter@pusan.ac.kr (J.-B.Y.); touy@pusan.ac.kr (T.K.); vehddnsv@pusan.ac.kr (D.S.)
* Correspondence: seungjoon.yi@pusan.ac.kr; Tel: +850-51-510-7917

Received: 29 July 2020; Accepted: 20 August 2020; Published: 25 August 2020

Abstract: Although the mobile manipulation capability is crucial for a service robot to perform physical work without human support, the long-term autonomous operation of such a mobile manipulation robot in a real environment is still a tremendously difficult task. In this paper, we present a modular, general purpose software framework for intelligent mobile manipulation robots that can interact with humans using complex human speech commands; navigate smoothly in tight indoor spaces; and finally detect and manipulate various household objects and pieces of furniture autonomously. The suggested software framework is designed to be easily transferred to different home service robots, which include the Toyota Human Support Robot (HSR) and our Modular Service Robot-1 (MSR-1) platforms. It has successfully been used to solve various home service tasks at the RoboCup@Home and World Robot Summit international service robot competitions with promising results.

Keywords: home service robot; unified software platform; mobile manipulation

1. Introduction

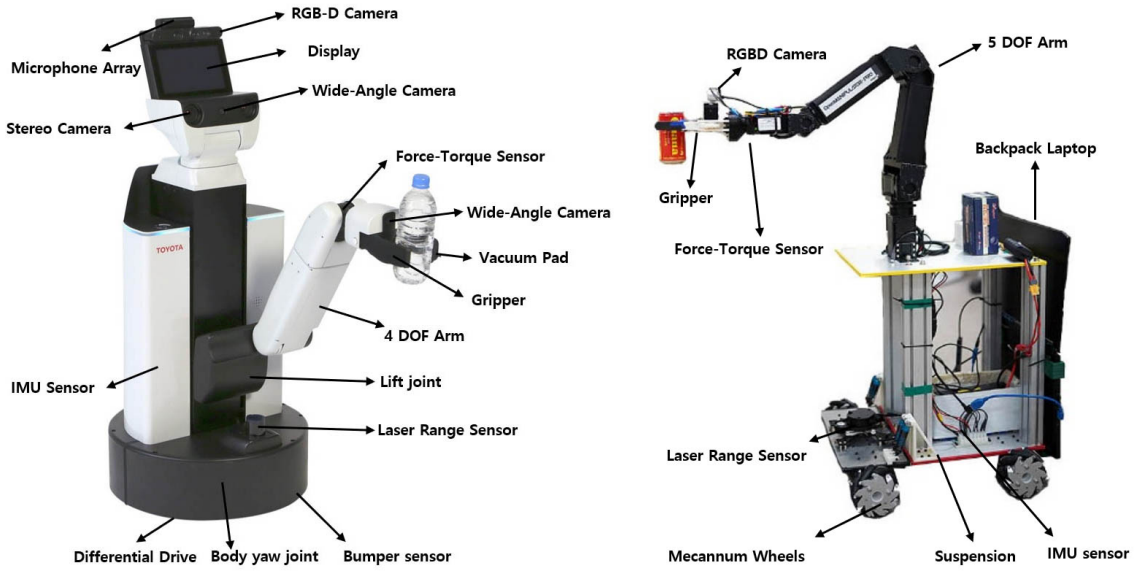
One of the main goals of robotic research is the commercial adoption of intelligent service robots—robots that can safely operate in various human environments without special infrastructure; communicate with untrained people using natural language and gestures; and handle various tasks that can help human lives. Thanks to the recent advancements of smartphones and motorized personal transportation devices, we have gained access to affordable processors, sensors and actuators, which has significantly lowered the hardware costs of intelligent mobile robots. In addition, the recent breakthroughs in the machine learning field have allowed the robots to robustly perceive the environment and communicate with people using natural language. As a result, we now have an increasing number of commercially available autonomous service robots that aim for home or enterprise markets [1–5].

However, mainly due to the relative high cost and weight of the robotic arm, most of the current service robots either lack an arm [1,2] or have less capable arms, mainly for gestures [3,4]. This omission greatly limits the uses of the service robot, as the robot cannot physically interact with the environment by itself and needs continuous human support during its operation. Although there are mobile robots equipped with high degree of freedom (DOF) manipulators with high payload and accuracy [6,7], they are not ideal for tight indoor environments, as their long, heavy and rigid manipulators increase the system size and weight significantly, require complex motion planning and pose possible safety risks.

As a result, there came a new generation of home service robots designed with the goal of autonomous mobile manipulation in home environments. Toyota's Human Support Robot [5] is a relatively small and lightweight home service robot with a compliant low-DOF manipulator that

simplifies the motion planning and lowers safety risks. The HSR has been adopted as a standard platform for two major international robotic competitions, the RoboCup@Home Domestic Standard Platform League (DSPL) and the World Robot Summit (WRS). By 2018, 33 universities in eight countries were using the HSR, which was on offer to the public since 2017 [8]. Stretch RE1 [9] is another commercial home service robot with a very small footprint and a compliant Cartesian manipulator that can handle various home service tasks in a tight home environment.

In this paper, we present a unified, modular software framework that can be quickly adopted for different intelligent mobile manipulation robots. It consists of navigation, visual perception, manipulation, human–robot interaction and high level autonomy modules, and can handle general service tasks given by complex voice commands. The remainder of the paper proceeds as follows. Section 2 presents details of two robotic platforms, which are shown in Figure 1. we have used to test the software framework. Section 3 explains how the robot detects household objects, furniture and humans from sensory inputs. Section 4 describes motion generation and control methods for object pickup and furniture operation. Section 5 presents how the robot governs its autonomous behavior and communicates with human operators. Section 6 shows the experimental results from two international service robot competitions. Finally, we conclude with a discussion of potential future directions arising from this work.



(**a**) (**b**)

Figure 1. Two home service robot platforms used for this work. (**a**) Toyota HSR (Human Support Robot), (**b**) MSR-1 (Modular Service Robot-1).

2. Related Work

Building and deploying a fully autonomous mobile manipulation system requires reliable integration of multiple software components, which includes mapping and localization; navigation; 3D perception of manipulation target objects; motion planning and control; natural language-based human–robot interface; and finally, intelligent decision making. This heavy burden, in addition to the lack of standardized middleware, readily available open source software modules and a common hardware platform, made the development of a service robot system rather difficult before the introduction of ROS [10–13]. Thanks to the widespread adoption of ROS middleware and breakthroughs in the machine learning field, there has been noticeable progress in many of those areas. However, most of academic research has focused on a small set of of topics under well controlled lab settings, rather than fully integrating a deployable system for real world environments. In [14],

the authors suggest a service robot system which integrates multimodal perception, a natural language interface and mobile manipulation for elderly care application. The system was implemented on a custom hardware platform and tested in a large, long-term experimental test with elderly users for usability and acceptability. Being one of the earlier service robot systems with ROS middleware, it relies upon classic machine vision techniques for perception which have been found to be unreliable, and have not been adopted in other hardware platforms. In [15], the authors presented a software framework for a service robot utilizing deep learning-based perception algorithms, but it was limited by some rather basic custom hardware used for the experiments and is unlikely to be capable of handling general home service tasks.

Recently in [16], the authors proposed a software architecture aimed at home service tasks similar to our work. The authors presented a fully integrated software system including perception, navigation, manipulation, a human–robot interface and decision making, and they incorporated modern deep learning-based perception pipeline and ROS middleware as well. The suggested software was implemented on two custom robots to show its the compatibility and modularity, and tested in the real world by one day of public demonstration. However, the software architecture suggested in [16] has a number of key differences from this work. The human input is provided from a screen, not by verbal communication including complex sentences. It has been adopted in only one mobile robot; the other platform was stationary, so it is not clear how easily it can be adopted to other home service robots, especially commercial ones. Finally, although the system has been real-world tested by a day of public demonstration, such a demonstration is quite different from robotic competitions where the robot has to reliably accomplish far more complex tasks with previously unknown objects, in a new, more complex and dynamic environment, and with human operators with no engineering knowledge.

There have been robotic competitions that test the performances and reliability of the integrated robotic systems in realistic, competitive environments. The DARPA Robotics Challenge (DRC) was one such competition including navigation and manipulation in human environments, which thereby greatly accelerated academic research in humanoid robotics. Unfortunately, due to the inherent instability of upright humanoid robot platforms and the high risk of hardware damage in cases of falling, most teams have decided to take a very conservative approach of minimizing autonomy and mostly focused on motion planning and control during the development [17,18].

One the other hand, the RoboCup [19] is another league of robotic competitions using fully autonomous robots. More importantly, some of the RoboCup leagues have been using standard platform hardware, such as Softbank Nao robot [20] which has been used for the humanoid soccer league since 2008. The use of a standard hardware platform and the annual open source release of each team's software [21,22] helped those leagues to improve much faster than other leagues using custom platforms. Recently, the service robot league, RoboCup@Home, has started to use Toyota HSR as the standard platform hardware since 2018. A number of software systems have been proposed [23–25] to be used with the HSR platform, which integrates a natural language interface, navigation, deep learning-based 3D perception and manipulation. Compared to this work, the software suggested in [23] lacks the task-specific sentence parser we used to analyze complex voice commands; has more limited manipulation motion planning that assumes a stationary base position; and finally, has not yet generalized to be adopted to other service robots. It show3e very good autonomous mobile manipulation performance at the RoboCup 2018 competition, but the task setting lacked navigation and a human–robot interface. The software system suggested in [24,25] is a more generalized and comprehensive one, which is based on a common layered software framework that has been used for multiple mobile robots, including the HSR. There still exist a couple of differences compared to this work. The software framework suggested in this work is composed of platform-independent function modules, such as a navigation module, and platform-dependent libraries such as kinematics. This allows the software to be quickly adopted to different hardware platforms. On the other hand, the LAAIR framework in [24] uses a layered structure of abstracted control elements, such as control and skills, which can make it harder to share functional modules

between different hardware platforms. Additionally, due to technical issues, most capabilities presented in [25] have not been publicly validated yet.

3. Hardware Platform

Although we first developed the software framework mainly to use on the Toyota HSR platform, it can be quickly adapted on any home service robot with similar capabilities thanks to its modular nature. A new robot requires hardware drivers for its actuators and sensors; forward and inverse kinematics for its manipulator and drivetrain; and some changes in Cartesian motion parameters to adopt the suggested software framework. At the time of writing, we have adopted our software framework to two robotic platforms, the commercial HSR platform and the Modular Service Robot-1 (MSR-1) platform we have been developing. In this section, we describe the hardware details of both robots and how they are utilized by the suggested software framework.

3.1. Mobile Base

The mobile base moves the robot around in the indoor environment, and has an inertial measurement unit (IMU) and LIDAR sensor to help the robot map the environment and localize itself. We assume the base to have omnidirectional mobility, as additional base mobility allows the robot to handle most manipulation tasks with a relatively short and low-DOF manipulator. Toyota HSR utilizes two differential wheels driven by geared motors and a central yaw joint rotating robot body to provide pseudo-omnidirectional mobility. On the other hand, the MSR-1 is equipped with four Mecannum wheels directly driven by high torque BLDC motors, which allows the robot to move at a much higher velocity than HSR in truly a omnidirectional way.

3.2. Manipulator and Gripper

For autonomous manipulation tasks in tight spaces, a short, low-DOF manipulator is often more suitable than long high-DOF manipulators. The Toyota HSR is equipped with such a manipulator, which consists of a shoulder pitch joint and roll-pitch-roll wrist joints, and is mounted on the torso assembly which can be lifted by a prismatic waist joint. To prevent possible damage from collision, arm joints use series elastic actuators utilizing elastic timing belts. On the other hand, MSR-1 is equipped with a manipulator with 5 traditional revolute joints at the current stage. We are currently working on a new low-DOF manipulator with a compliant parallel drive mechanism for the robot. Both robots have a two-finger parallel gripper, a wrist force-torque sensor and a hand camera for additional sensory feedback. The HSR also has a vacuum pad for picking up small items such as postcards.

3.3. Sensory Head

HSR has a sensory head connected to the torso assembly by a pan and q tilt joint. It is equipped with a wide range of sensors, including a ASUS Xtion RGBD camera, two industrial RGB cameras arranged in a stereo configuration, one wide angle RGB camera and an omnidirectional microphone array. It also has a small display that can be used for status monitoring and human–robot interactions. As we have found that a single RGBD camera is sufficient for most indoor tasks, we simplified the sensor setup of MSR-1 as a single RGBD camera mounted at the wrist of the manipulator.

3.4. Onboard Computers

HSR is equipped with two onboard computers. The main one has an Intel i7 quad-core CPU with 16 GB of RAM, which runs Toyota's proprietary code that handles sensor and motor interfaces, motion planning and control and the web-based remote user interface. An NVidia Jetson TX1 embedded computer is also provided for light GPU-based computing tasks, as both onboard computers do not have enough computing power for advanced software algorithms. Toyota provides an official

laptop mount which can house a powerful laptop tethered by Ethernet that can handle CPU and GPU intensive computations. For MSR-1, we have simplified the computer setup so that a single laptop mounted at the back of the robot handles all the CPU and GPU load by itself. In addition to handing the computation, the back mounted laptop is also used for monitoring the current status of the robot using its display.

4. Perception

4.1. Previously Known Manipulation Objects

For autonomous mobile manipulation, robust detection and pose estimation of target objects are crucially needed. Recently, there has been a major breakthrough in computer vision field thanks to the deep learning algorithms, which allows for reliable real-time object detection and segmentation from an RGB image stream [26,27], and estimating 6D poses of visible objects [28]. However, such RGB-based methods do not provide precise estimation of object poses required for the manipulation. On the other hand, there are depth-based object detection and pose estimation approaches using object geometries [29], but such methods are generally less reliable in cluttered environments compared to RGB-based detection methods, and they are not suitable for distinguishing multiple objects with similar geometries, for example, different types of canned drinks.

To overcome such shortcomings, out object perception pipeline uses a combination of both approaches. It first uses a 2D object detection algorithm to detect the target objects and their bounding boxes in the RGB image, and then uses the matching depth image and detected object bounding boxes to get the point clouds of each object. Finally, the object poses are estimated from the per-object point clouds. We will describe each step in more detail in following subsections.

4.1.1. Object Detection from RGB Image

We use the YOLOv3 [30] real-time object detection algorithm to detect known objects and their bounding boxes from the RGB image stream. Although the object detection performance relies heavily upon the quality and quantity of the training data, manual acquisition and labeling of training data is often not feasible due to the time constraint, especially at the robotic competitions. Thus, we utilize an automated process to quickly generate a large amount of training data, which is shown in Figure 2. Shadowless object images from all viewpoints are obtained utilizing a motorized turntable inside a small studio box, which are then augmented and superimposed over photographs of the actual test environment to generate synthesized training data. Finally, the classification network is trained using the training data until the average loss decreases below a preset threshold. Figure 3 shows the 2D object detection results after the first setup day of the World Robot Summit 2018 robotic competition, where we successfully generated the labeled training data and trained the classifier for 60 objects that were unveiled on the first day of the competition.

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

Figure 2. Automated generation of training data. (**a**) Turntable and studio setup, (**b**) Captured object images, (**c**) Background image and (**d**) Synthesized training image.

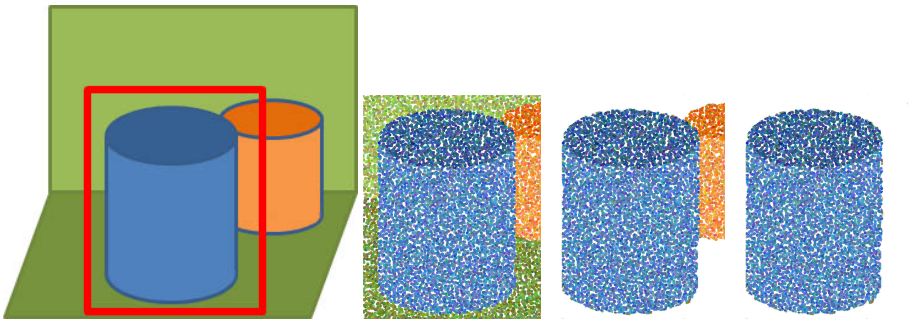
Figure 3. Detected objects and their bounding boxes from a RGB image.

4.1.2. Per-Object Point Cloud Generation

After detecting an object in the RGB image, we still need to estimate its pose to precisely manipulate the object. We first use the 2D object bounding box to segment the corresponding depth image, and project it in the 3D space using camera transform and parameters to generate the point cloud inside the 2D bounding box. As the bounding box is usually larger than the object, the resulting point cloud often contains a large number of non-object points, such as the surface below or behind the object, or even points from other objects.

For most manipulation tasks in the home environment, the target object is placed on a flat surface such as a table. For some harder cases, the target object can be placed inside a bookshelf or a cupboard, surrounded by a ceiling and side walls. To reject those non-object points, we periodically use the RANSAC algorithm on a full point cloud from the depth image to detect and track planar surfaces, and remove points near the known surfaces from the per-object point cloud described above.

Even after removing points from surrounding surfaces, the resulting point cloud may include points from other objects inside the bounding box. To handle this case, we apply the k-nearest neighbor clustering algorithm to cluster the points into multiple clusters, and pick the cluster which is closest to the centroid of the bounding box as the correct one. The point cloud generation process is visually shown in Figure 4.



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

Figure 4. Per-object point cloud generation process. (**a**) RGB image with the bounding box, (**b**) Point cloud generated from the cropped depth image, (**c**) Point cloud after surface removal and (**d**) Per-object point cloud after clustering.

4.1.3. Pose Estimation from Per-Object Point Cloud

Once we acquire the segmented point cloud for an object, we need to estimate its position and orientation for precise pickup. We have implemented the iterative closest point (ICP) algorithm-based

pose estimator [31], which aligns known object geometries to the object point cloud to get the most probable position and orientation of the object. However, we have found that simpler method of using the 3D centroid of the point cloud works very well for most target objects HSR can handle. As this simpler method does not provide the orientation of the object, we use the principal component analysis (PCA) algorithm to get the major and minor axes of the object, as shown in Figure 5.

Figure 5. 3D pose estimation.

4.2. Previously Unknown Manipulation Objects

Our object detection pipeline described above primarily relies upon the RGB-based object detector trained with known objects. However, in some cases, the robot needs to manipulate objects that are previously unknown to robot. One way to handle this problem is using another image detector that is trained with a large number of generic images, such as [32]. If the additional image detector can detect the previously unknown object correctly, we can use the same pipeline to estimate its pose and manipulate it.

Another approach to handle a previously unknown object is using a depth image alone with contextual information. For example, we can inform the robot about the location of the children's room or dinner table, then any object detected at that area can be regarded as an unknown foreign object. The robot can then use the location context to guess what the object is and what the it should do. For example, the robot may think any unknown objects on the dinner table are dishes to clean, and move them to dishwasher. To detect unknown objects from depth images, we first build a fine-grained height map of the area of interest. Then we remove the points below the surface the object is placed upon, and apply k-means clustering and PCA algorithms to determine the best grip positions to pick up each item. Figure 6 shows the HSR picking up a number of previously unknown dishes successfully.

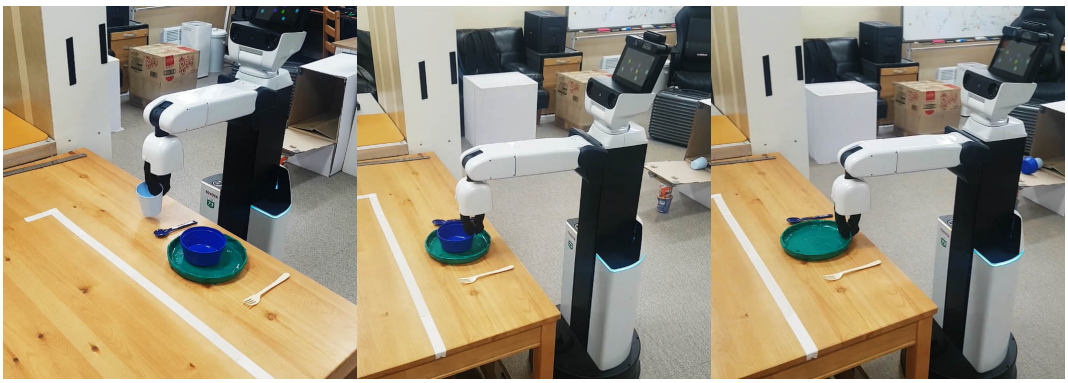


Figure 6. Detection and manipulation of previously unknown objects.

4.3. Furniture Detection

Home environments can have a number of distinctive pieces of furniture, including stationary ones like tables, bookshelves and compartment boxes, and openable ones like doors, refrigerators and drawers. To detect furniture without using any markers, we first keep track of the vertical and horizontal planar surfaces extracted from the depth image stream using RANSAC algorithm, and use those surface features to find and update furniture information.

4.3.1. Table

When a horizontal surface larger than a threshold is detected, it can be part of a table. We maintain a list of possible tables, and incrementally update their center and boundary edge information if a new surface is detected nearby. This way we can detect large tables and estimate their edge information, which is used to determine the approach angle for picking up objects on the table.

4.3.2. Bookshelf

A bookshelf is usually composed of a number of horizontal shelves and vertical side panels. Thus, we first check if a number of horizontal surfaces are detected together; then we check for vertical surfaces to find the orientation of the bookshelf. Detected bookshelf information is used to reject non-object points in the object detect pipeline.

4.3.3. Handle

Many openable pieces of furniture, such as doors, cabinets and garbage cans, have distinctive handles that need to be grabbed for operation. To detect such handles, we first detect the primary vertical surface, such as the door body, and cluster the 3D points on the surface that are farther than a threshold value from the surface. Finally, we estimate the door direction and handle orientation from the normal vector of the primary surface and the primary axis of the 3D points. Figure 7 shows the detected surfaces, primary vertical surface and all handle positions found for a drawer cabinet.

Figure 7. Handle detection example.

5. Manipulation

5.1. Human Detection

A service robot needs the ability to detect a human in his/her environment. We use the Kairos cloud-based face recognition service [33] that can detect multiple faces with a wealth of information, such as gender, age and race, from an RGB image. Once the facial features are detected from the RGB image, we use the matching depth image to get 3D positions of each face. As this cloud-based approach is only suitable for stationary people, we plan to add a local human pose estimation algorithm such as OpenPose [34] for gesture recognition and tracking a moving person.

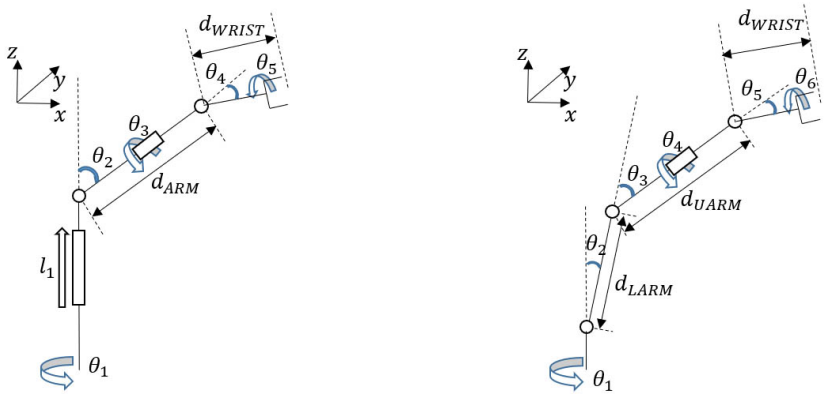
Once the robot perceives the target object, it needs to move its base and arm actuators accordingly to manipulate the object. As the total DOFs of the robot including the base are more than 6, one way to plan the motion for both the base and arm is regarding them as a single redundant DOF actuator and use a general motion planner to plan its whole body motion [35,36]. However, we have found that such a whole body motion planning is unnecessary for mobile manipulation robots with relative short reach, as the robot should always directly face the manipulation object and approach forward to minimize the collision chance with the environment. Thus we can decouple the whole body grasp motion into base and arm motions: The base motion moves the robot in front of the target object so that the target object and gripper are aligned, and the we select one of parameterized arm motions that best suits current object pose, which can be optimized in advance. Finally, we can use the analytic arm inverse kinematics solution to find the arm joint trajectories to realize the arm motion.

5.2. Arm Kinematics

To get the analytical inverse kinematic solution of HSR and MSR-1 manipulator, we include the base yaw and waist lift joints to HSR manipulator to form a 6DOF RPRRRR arm, and add a virtual base yaw joint to MSR-1 manipulator to form a 6DOF RRRRRR arm, as shown in Figure 8. If we define $T_{tr}(x, y, z)$ as a translational transform and R_x, R_y, R_z as rotational transforms on axes x, y, z respectively, the forward kinematics solution of two arm chains T_{HSR} and T_{MSR-1} can be derived as follows;

$$T_{HSR} = R_z(\theta_1)T_{tr}(0,0,l_1)R_y(\theta_2)T_{tr}(0,0,d_{ARM})R_z(\theta_3)R_y(\theta_4)R_z(\theta_5)T_{tr}(0,0,d_{WRIST}) \quad (1)$$

$$T_{MSR-1} = R_z(\theta_1)R_y(\theta_2)T_{tr}(0,0,d_{LARM})R_y(\theta_3)T_{tr}(0,0,d_{UARM})R_z(\theta_4)R_y(\theta_5)R_z(\theta_6)T_{tr}(0,0,d_{WRIST}). \quad (2)$$



(**a**) (**b**)

Figure 8. Arm configurations of HSR and MSR-1. (**a**) HSR Arm, (**b**) MSR-1 Arm.

5.2.1. Inverse Kinematics of HSR Arm

To get the inverse kinematic solution for a target transform T, we first calculate the relative wrist position (x', y', z') to get the base yaw angle θ_1;

$$\begin{bmatrix} x' & y' & z' & 1 \end{bmatrix}^T = TT_{tr}(0,0,-d_{WRIST}) \begin{bmatrix} 0 & 0 & 0 & 1 \end{bmatrix}^T \quad (3)$$

$$\theta_1 = \text{atan2}(y', x'). \quad (4)$$

Additionally, we get two possible sets of torso lift height and shoulder pitch angle (l_1, θ_2):

$$\begin{bmatrix} l_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} z' - \sqrt{d_{ARM}^2 - x'^2 - y'^2} \\ \arcsin(\sqrt{x'^2 + y'^2}/d_{ARM}) \end{bmatrix} \tag{5}$$

$$\begin{bmatrix} l_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} z' + \sqrt{d_{ARM}^2 - x'^2 - y'^2} \\ \pi - \arcsin(\sqrt{x'^2 + y'^2}/d_{ARM}) \end{bmatrix}. \tag{6}$$

To get the wrist joint angles θ_3, θ_4 and θ_5, we calculate the following matrix R with the values of l_1, θ_1 and θ_2

$$R = T_{tr}(0, 0, -d_{ARM}) R_y(-\theta_2) T_{tr}(0, 0, -l_1) R_z(-\theta_1) T T_{tr}(0, 0, -d_{WRIST}). \tag{7}$$

Then the remaining joint angles for HSR can be acquired as follows.

$$\begin{bmatrix} \theta_3 \\ \theta_4 \\ \theta_5 \end{bmatrix} = \begin{bmatrix} \text{atan2}(R_{23}, R_{13}) \\ \arccos(R_{33}) \\ \text{atan2}(R_{32}, -R_{31}) \end{bmatrix} \tag{8}$$

5.2.2. Inverse Kinematics of MSR-1 Arm

For MSR-1 arm, we can get the relative wrist position (x', y', z') to get the base yaw angle θ_1 the same way using (3) and (4). Then the shoulder and elbow angle (θ_2, θ_3) can be found as

$$\begin{bmatrix} \theta_2 \\ \theta_3 \end{bmatrix} = \begin{bmatrix} \frac{\pi}{2} - \arccos(\frac{{d_{LARM}}^2 + (x'^2 + y'^2 + z'^2) - {d_{UARM}}^2}{2 d_{LARM} \sqrt{x'^2 + y'^2 + z'^2}}) - \text{atan2}(z', \sqrt{x'^2 + y'^2}) \\ \arccos(\frac{(x'^2 + y'^2 + z'^2) - {d_{LARM}}^2 - {d_{UARM}}^2}{2 d_{LARM} d_{UARM}}) \end{bmatrix} \tag{9}$$

To get the wrist joint angles θ_4, θ_5 and θ_6, we calculate the following matrix R with the values of θ_1, θ_2 and θ_3

$$R = T_{tr}(0, 0, -d_{UARM}) R_y(-\theta_3) T_{tr}(0, 0, -d_{LARM}) R_y(-\theta_2) R_z(-\theta_1) T T_{tr}(0, 0, -d_{WRIST}). \tag{10}$$

Then the remaining joint angles for MSR-1 can be acquired as follows:

$$\begin{bmatrix} \theta_4 \\ \theta_5 \\ \theta_6 \end{bmatrix} = \begin{bmatrix} \text{atan2}(R_{23}, R_{13}) \\ \arccos(R_{33}) \\ \text{atan2}(R_{32}, -R_{31}) \end{bmatrix}. \tag{11}$$

5.3. Parameterized Arm Motions

We have designed a number of parameterized arm motions in Cartesian space that can reach target objects with various heights and configurations, while satisfying each robot's joint limits. The designed motions are used for grasping, releasing and handing over objects. We describe each case we have considered in more detail below.

5.3.1. High Height, Horizontal Grasp

The most common case is when the target object is higher than the gripper height of the default arm configuration. As the target object is usually supported by a flat surface such as a table or a shelf, we first vertically lift the gripper to the target height and then linearly move the gripper forward to the grasp position to prevent possible collision. An example of this case is shown in Figure 9a.

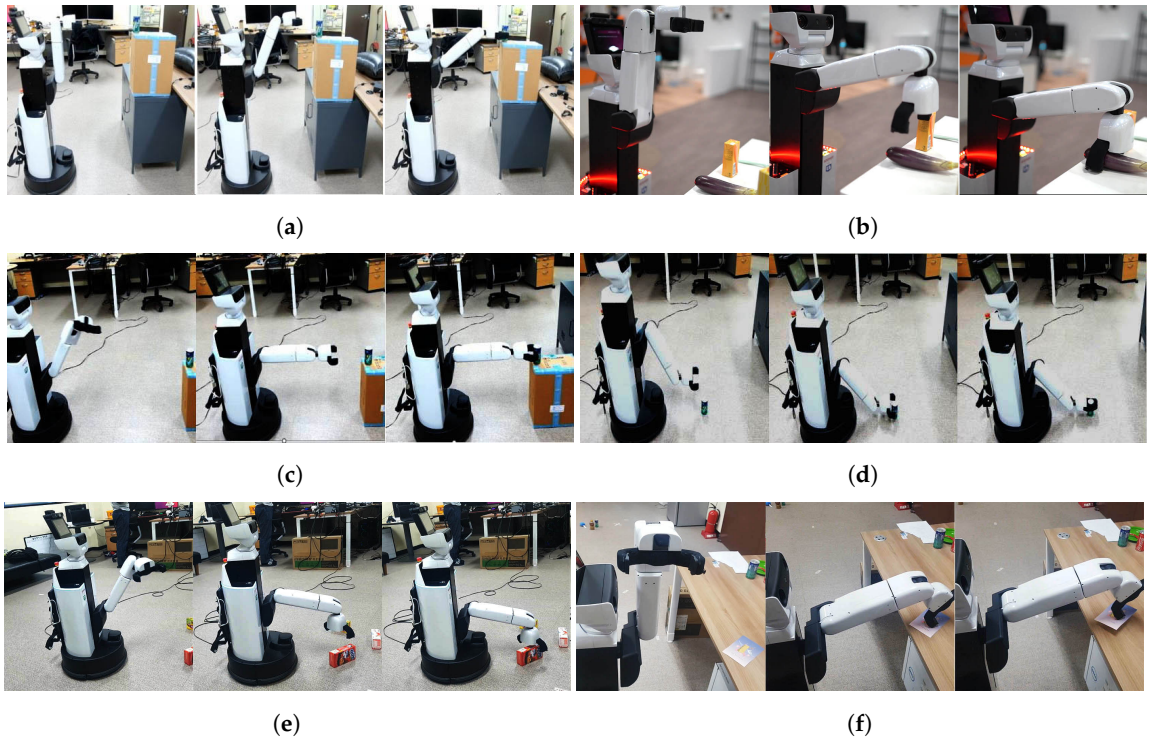


Figure 9. Various parameterized pickup motions for HSR. (**a**) High height, horizontal grasp, (**b**) High height, vertical grasp, (**c**) Medium height, straight arm grasp, (**d**) Low height, horizontal grasp, (**e**) Low height, vertical grasp and (**f**) Thin object, suction pickup.

5.3.2. High Height, Vertical Grasp

Thin and elongated objects, such as forks and spoons, are much easier to pick up with a vertical grasp. For this case we first move the gripper above the object facing downward, and then lower the gripper and pick up the object. To minimize the effect of perception error, we use the impedance control to push the gripper down on the support surface for precise positioning, and incrementally raise the gripper during the grasping to keep the fingertips closely following the support surface. An example of this case is shown in Figures 9b and 10c.

5.3.3. Medium Height, Straight Arm Grasp

A third case is when the target object is between the lowest shoulder joint height and the default gripper height. As such an object usually lies inside a compartment with top and side walls, special care is needed for collision avoidance. Thus, we use the safest arm posture for this case, a fully stretched horizontal arm, and use the linear movement of the base to reach the target. To prevent catastrophic failure from collision during the base movement, we use the wrist force torque sensor to constantly adjust the base movement if necessary. An example of this case is shown in Figure 9c.

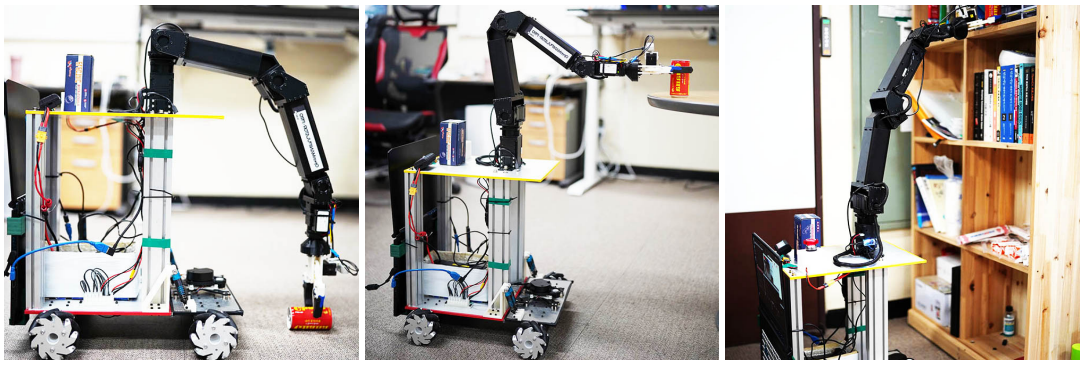
5.3.4. Low Height, Horizontal Grasp

When the target object is lower than the lowest shoulder joint height, we need to use the inverted arm configuration of Figure 8b to reach the object. As the default arm configuration is the non-inverted one, we need an additional "flip" stage before horizontally moving the gripper forward for grasping, and an "unflip" stage afterwards. An example of this case is shown in Figures 9d and 10b.

5.3.5. Low Height, Vertical Grasp

When the target object is low in height and directly reachable from above, it is faster to use the vertical grip as it does not require additional flip and unflip stages. In addition, we need to use this

motion when the horizontal access to the object is not possible, for example, when the target object is in a drawer. Figures 9e and 10a shows an example of the vertical grasping of ground objects.



(**a**) (**b**) (**c**)

Figure 10. Parameterized arm motions realized with MSR-1. (**a**) Low height, vertical grasp, (**b**) Low height, horizontal grasp and (**c**) High height, horizontal grasp.

5.3.6. Suction Pickup

Thin objects such as postcards cannot be reliably picked up using the gripper. We use the suction cup mounted on one of the gripper fingers to handle such objects. We use the same vertical grasp motion to move the gripper over the target object, and then use the impedance control to press down the suction cup over the object after turning on the suction. Figure 9f shows an example of the suction pickup.

5.4. Whole Body Motions

Due to the limited workspace of the manipulator, some manipulation tasks cannot be accomplished without moving the base and the manipulator [29,37]. We have designed three parameterized whole-body motions that can be used for various household manipulation tasks.

5.4.1. Opening an Outward-Opening Door

The door opening capability is necessary to freely navigate the indoor space without human help. We have designed a general door opening motion that can be used for outward opening doors with lever type handles. Initially we let the robot grab the door handle and rotate it, but such a strategy had issues with releasing the handle after opening in some cases. Instead, we have designed a new opening motion that moves the whole tip of the gripper in an arc around the lever pivot point, as shown in Figure 11a. Once the door opening is detected by the wrist force torque sensor, the robot pushes the door forward with body movement to fully open it.

5.4.2. Opening a Cabinet Drawer

A cabinet drawer can be opened by grabbing the handle and pulling it backward. Although it is conceptually simple, we have found it to be nontrivial, as many cabinets have fairly shallow handles which makes the grasping hard in the presence of perceptual error. We use the wrist force torque sensor to precisely position the gripper before grabbing the handle, and to detect that the drawer is fully open. Figure 11b shows the HSR opening a drawer.

5.4.3. Opening a Inward-Opening Door

To access the objects in a refrigerator, the robot needs to open the refrigerator door, which is a basically an inward opening hinged door with a magnetic closure. After grabbing the door handle, we let the robot follow a curved trajectory that makes the gripper to keep a constant distance and angle

from the door hinge, as shown in Figure 11c. Once the door is open, the robot executes additional arm sweep motions to fully open the refrigerator door.

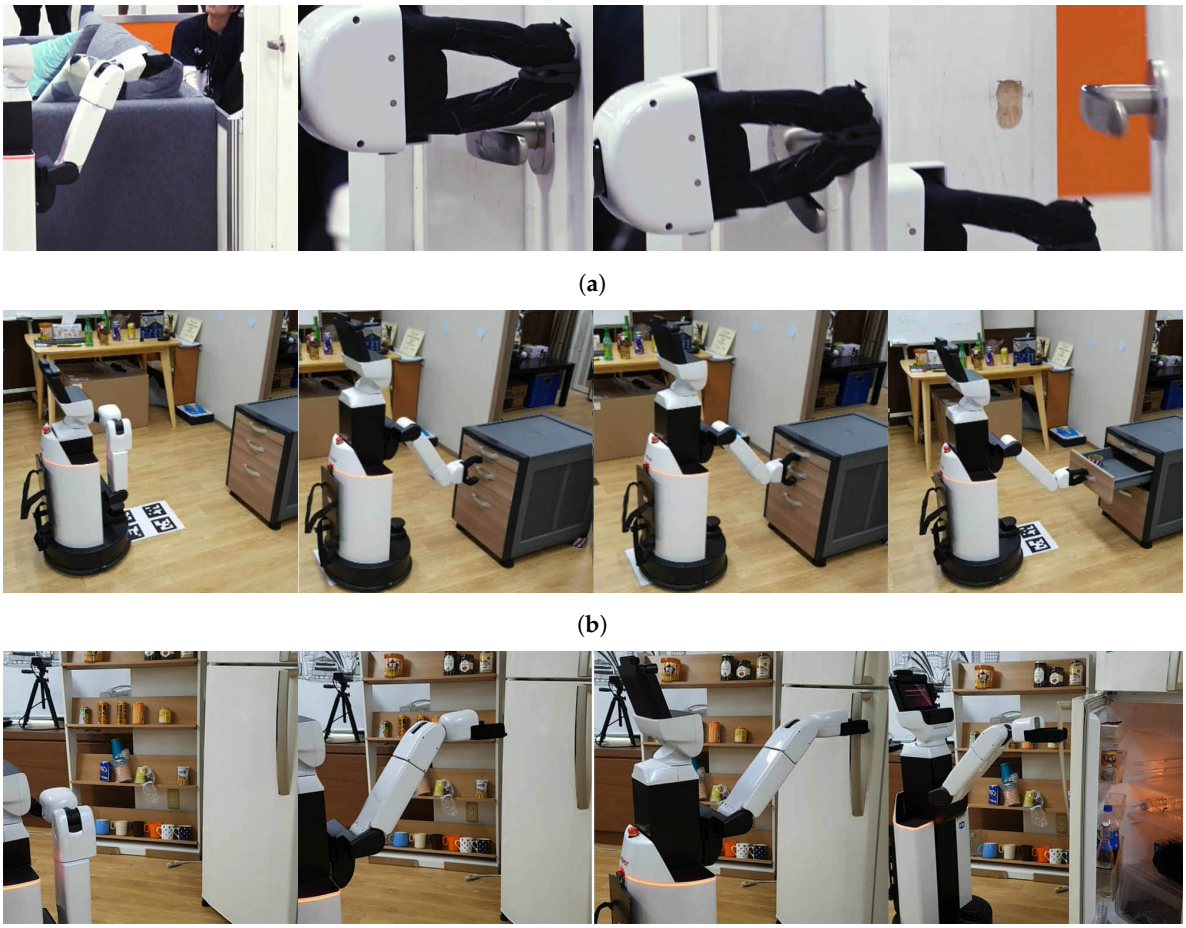


Figure 11. Whole body manipulation examples. (**a**) Opening a Door, (**b**) Opening a Drawer and (**c**) Opening a Refrigerator.

6. Autonomy and the Human–Robot Interaction

6.1. SLAM

Currently we use the ROS amcl package which relies upon a pre-built 2d map of the environment. Although the default package works fairly well in most situations, its reliance upon a static pre-built map is a big limitation. Currently we are working on an iterative closest point (ICP)-based slam algorithm that uses an incrementally updated dynamic traversal map, which we have successfully used in various crowded indoor environments. In addition, we also plan to let the robot learn a semantic map of the environment from detected objects, which will ultimately let the robot operate in a previously unknown environment without any prior knowledge.

6.2. Navigation

Many service tasks require the robot to navigate to known locations while avoiding obstacles. We have found the default navigation module of HSR has many issues for indoor navigation—it does not fully utilize the maximum speed of the robot, is sensitive to dynamic obstacles which can completely stop the robot unnecessarily and tends to detect the robot's own arm as an obstacle. Thus,

we have built a new hierarchical navigation module, which uses the A* algorithm and visibility augmentation. The new algorithm favors open spaces where the robot is allowed to accelerate to the maximum velocity, and dynamic obstacles are continuously avoided without stopping using the potential field algorithm. The new navigation module allows our robots to move significantly faster than other teams' robots, which was crucial for some tasks with tight time limits. Figure 12 shows the HSR navigating through tight indoor environment during the RoboCup@Home competition.

Figure 12. The HSR navigating in a tight indoor environment.

6.3. Behavioral Autonomy

The behavior of the robot is governed by a number of hierarchical finite state machines (FSMs) [38], which manage behaviors of each robot part such as the head, arm and base movements. We have found that this FSM-based autonomy ensures a reliable and predictive behavior of the robot, and is highly customizable for various tasks. In addition to the behavior FSMs, we use tasks which are the highest level actions for the robot such as navigating to a place, searching for container handle and picking up a object, which are stored in a task queue and executed sequentially.

6.4. Human–Robot Interaction

We use the verbal communication as the primary method of human–robot interaction. For voice synthesis, we use the provided HSR text-to-speech (TTS) module, which supports both the English and Japanese languages and runs offline. For speech recognition, we use the Google Cloud Speech API that provides a excellent sentence-level voice recognition in real time from a streaming audio input. The recognized sentences are fed to task specific parser, which translates the given voice command into a series of action tasks and enqueue them to the task queue.

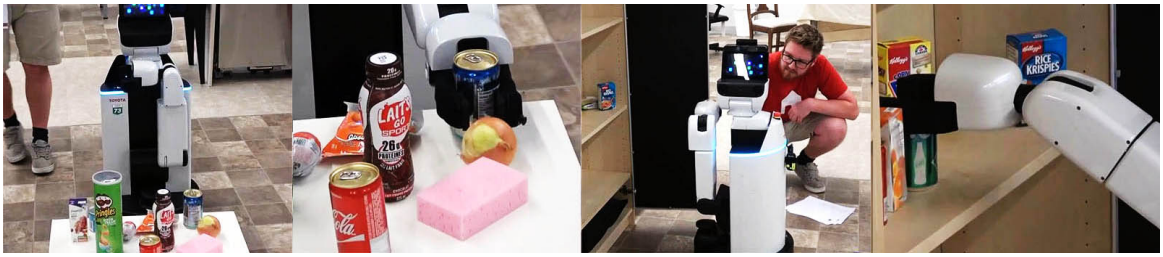
7. Experimental Results

The suggested software framework was implemented on the HSR platform and used for two major international service robot competitions with promising results.

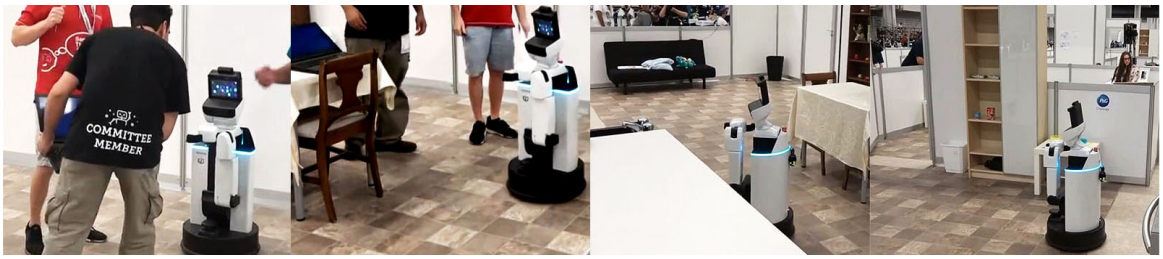
7.1. RoboCup@Home Domestic Standard Platform League

The RoboCup@Home league, the largest international annual competition for autonomous service robots, was established in 2006 with the goal of providing a framework for testing and comparing solutions for the development of service robots aiming personal and domestic applications. We have participated the RoboCup@Home DSPL 2018 held in Montreal, Canada and successfully executed a number of tasks. Figure 13 shows three of the tasks our team has scored at; the Speech and Person Recognition (SPR) task where the robot visually analyzes a group of people and answers questions from them; the Grocery task wherein the robot arranges a number of groceries on a table into the shelf; and finally, the General Purpose Service Robot (GPSR) task wherein the robot performs a series of tasks according to a long and complex command sentence from human operators.

(**a**)



(**b**)



(**c**)

Figure 13. The HSR doing various tasks at the RoboCup@Home 2018 Competition. (**a**) Speech and Person Recognition Task, (**b**) Grocery Task and (**c**) General Purpose Service Robot Task.

7.2. World Robot Summit Partner Robot Challenge

The World Robot Summit is another big international robot competition hosted by the Japanese government. The WRS 2018 was held in Tokyo, Japan in October 2018, and the WRS 2020 is scheduled to be held along with the Tokyo Olympics. The service robot league, named the Partner Robot Challenge, uses the HSR as a standard platform and is heavily focused on autonomy and mobile manipulation. Figure 14 shows three tasks used for the WRS 2018, which include the Tidy Up Here task where the robot navigates to commanded area and tidies up the area by placing various objects into appropriate storage places; the Bring Me task where the robot fetches the asked object stored in one of the containers and brings it back to the operator; and finally the Show Me the Future task where each team demonstrates what a future home with service robots will be like.

7.3. Discussion

Experimental results from two robotic competitions show that the suggested software framework can handle a wide range of service tasks with excellent results. The system lets the robot navigate through tight indoor spaces with dynamic obstacles, use whole body motion to operate furniture, reliably detect and pickup various objects that have not been known before the competition and communicate with the human operator using natural language. The flexibility of the framework has allowed a quick adoption to multiple hardware platforms, where only the low-level libraries and motion parameters need to be modified.

(**a**)

(**b**)

(**c**)

Figure 14. The HSR doing various tasks at the WRS 2018 Competition. (**a**) Tidy Up Here Task, (**b**) Bring Me Task and (**c**) Show Me the Future Task.

As we have presented in the Section 2, the distinctive contributions of the software framework we suggest in this work are as follows: (a) Full integration of various software modules required for accomplishing highly complex home service tasks with full autonomy, which includes deep learning-based 3D perception, a natural language-based human–robot interface and manipulation of various objects and furniture. (b) Out-of-the-box support of Toyota HSR, the widely adopted standard hardware platform for two international robotic competitions; (c) quick adoption to any mobile manipulation robot due to the modular nature and (d) was rigorously tested and proven to work reliably in non-controlled real environments of robotic competitions.

At the time of writing, we are working on the public release of our software framework, and iterating on other hardware modules for our modular MSR-1 hardware platform. We hope the open source release of the software framework will help other researchers to quickly set up and deploy their mobile manipulation robot and start researching on various subtopics. Additionally, we also present valuable lessons we have learned from two robotic competitions we have participated in with the suggested software framework.

8. Conclusions

In this work, we described a unified software framework for intelligent service robots that can communicate with humans using voice commands; navigate in tight indoor spaces; perceive the

environment, household objects, furniture and people; and finally, manipulate objects. The suggested software has been implemented on physical HSR and MSR-1 platforms and thoroughly evaluated in two international robot competitions. Future work will include applying a machine learning approach with both simulated and real environments to optimize the behavior of the robot, and improving the semantic learning capability so that the robot can operate in totally unknown environments.

Author Contributions: Methodology, S.-J.Y.; software, J.-B.Y.; hardware, T.K. and D.S. All authors have read and agreed to the published version of the manuscript.

Funding: This work was partly supported by the Institute for Information and Communications Technology Promotion (2018-0-00622-RMI), the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIP) (number 2018R1C1B5045389) and Pusan National University Research Grant, 2017.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

HSR	Human Support Robot
MSR-1	Modular Service Robot-1
DOF	Degree of Freedom
IMU	Inertial Measurement Unit
ROS	Robot Operating System
DSPL	Domestic Standard Platform League
WRS	World Robot Summit
ICP	Iterative Closest Point
PCA	Principal Component Analysis
RANSAC	Random Sample Consensus
TTS	Text to Speech
SPR	Speech and Person Recognition
GPSR	General Purpose Service Robot

References

1. Savioke Relay Delivery Robot. Available online: http://www.knitec.com/savioke-relay-delivery-robot/ (accessed on 29 July 2020).
2. Starship Robot. Available online: https://www.starship.xyz/ (accessed on 29 July 2020).
3. Pandey, A.K.; Gelin, R. A Mass-Produced Sociable Humanoid Robot: Pepper: The First Machine of Its Kind. *IEEE Robot. Autom. Mag.* **2018**, *25*, 40–48. [CrossRef]
4. Cusomized, Cloud-Based, Intelligent Humanoid Service Robot. Available online: https://ubtrobot.com/products/cruzr?ls=en (accessed on 29 July 2020).
5. Yamaguchi, U.; Saito, F.; Ikeda, K.; Yamamoto, T. HSR, Human Support Robot as Research and Development Platform. *Abstr. Int. Conf. Adv. Mechatron.: Towar. Evol. Fusion IT Mechatron. (ICAM)* **2015**, *6*, 39–40. [CrossRef]
6. Wise, M.; Ferguson, M.; King, D.; Diehr, E.; Dymesich, D. Fetch and freight: Standard platforms for service robot applications. In Proceedings of the IJCAI-2016 Workshop on Autonomous Mobile Service Robots, New York City, NY, USA, 11 July 2016.
7. Pages, J.; Marchionni, L.; Ferro, F. TIAGo: The modular robot that adapts to different research needs. In Proceedings of the International Workshop on Robot Modularity, Daejeon, Korea, 10 October 2016.
8. Yamamoto, T.; Terada, K.; Ochiai, A.; Saito, F.; Asahara, Y.; Murase, K. Development of the Research Platform of a Domestic Mobile Manipulator Utilized for International Competition and Field Test. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 7675–7682.
9. Stretcht RE1. Available online: https://hello-robot.com/product (accessed on 29 July 2020).
10. Srinivasa, S.; Ferguson, D.; Helfrich, C.; Berenson, D.; Collet, A.; Diankov, R.; Gallagher, G.; Hollinger, G.; Kuffner, J.; Weghe, M. HERB: A home exploring robotic butler. *Auton. Robot.* **2010**, *28*, 5–20. [CrossRef]

11. Savage, J.; LLarena, A.; Carrera, G.; Cuellar, S.; Esparza, D.; Minami, Y.; Peñuelas, U. ViRbot: A System for the Operation of Mobile Robots. In Proceedings of the RoboCup 2007: Robot Soccer World Cup XI, Atlanta, GA, USA, 9–10 July 2007; pp. 512–519.
12. Graf, B.; Parlitz, C.; Hägele, M. Robotic Home Assistant Care-O-bot® 3 Product Vision and Innovation Platform. In Proceedings of the Human-Computer Interaction. Novel Interaction Methods and Techniques, San Diego, USA, 19–24 July 2009; pp. 312–320.
13. King, C.H.; Chen, T.L.; Fan, Z.; Glass, J.D.; Kemp, C.C. Dusty: An assistive mobile manipulator that retrieves dropped objects for people with motor impairments. *Disabil. Rehabil. Assist. Technol.* **2012**, *7*, 168–179. [CrossRef] [PubMed]
14. Hendrich, N.; Bistry, H.; Zhang, J. Architecture and Software Design for a Service Robot in an Elderly-Care Scenario. *Engineering* **2015**, *1*, 27–35. [CrossRef]
15. Chen, A.; Yang, B.; Cui, Y.; Chen, Y.; Zhang, S.; Zhao, X. Designing a Supermarket Service Robot Based on Deep Convolutional Neural Networks. *Symmetry* **2020**, *12*, 360. [CrossRef]
16. Nam, C.; Lee, S.; Lee, J.; Cheong, S.H.; Kim, D.H.; Kim, C.; Kim, I.; Park, S. A Software Architecture for Service Robots Manipulating Objects in Human Environments. *IEEE Access* **2020**, *8*, 117900–117920. [CrossRef]
17. Yi, S.J.; McGill, S.G.; Vadakedathu, L.; He, Q.; Ha, I.; Han, J.; Song, H.; Rouleau, M.; Zhang, B.T.; Hong, D.W.; et al. Team thor's entry in the DARPA robotics challenge trials 2013. *J. Field Robot.* **2015**, *32*, 315–335. [CrossRef]
18. McGill, S.G.; Yi, S.J.; Yi, H.; Ahn, M.S.; Cho, S.; Liu, K.; Sun, D.; Lee, B.; Jeong, H.; Huh, J.; et al. Team THOR's Entry in the DARPA Robotics Challenge Finals 2015. *J. Field Robot.* **2017**, *34*, 775–801. [CrossRef]
19. Kitano, H.; Asada, M.; Kuniyoshi, Y.; Noda, I.; Osawa, E.; Matsubara, H. RoboCup: A Challenge Problem for AI. *AI Mag.* **1997**, *18*, 73.
20. Gouaillier, D.; Hugel, V.; Blazevic, P.; Kilner, C.; Monceaux, J.; Lafourcade, P.; Marnier, B.; Serre, J.; Maisonnier, B. Mechatronic design of NAO humanoid. In Proceedings of the 2009 IEEE International Conference on Robotics and Automation, Kobe, Japan, 12–17 May 2009; pp. 769–774.
21. McGill, S.G.; Brindza, J.; Yi, S.J.; Lee, D.D. Unified Humanoid Robotics Software Platform. In Proceedings of the 5th Workshop on Humanoid Soccer Robots, Nashville, TN, USA, 7 December 2010.
22. Röfer, T.; Laue, T.; Baude, A.; Blumenkamp, J.; Felsch, G.; Fiedler, J.; Hasselbring, A.; Haß, T.; Oppermann, J.; Reichenberg, P.; et al. B-Human Team Report and Code Release 2019. 2019. Available online: http://www.b-human.de/downloads/publications/2019/CodeRelease2019.pdf (accessed on 29 July 2020).
23. Keleştemur, T.; Yokoyama, N.; Truong, J.; Allaban, A.A.; Padir, T. System Architecture for Autonomous Mobile Manipulation of Everyday Objects in Domestic Environments. In Proceedings of the PETRA '19 12th ACM International Conference on PErvasive Technologies Related to Assistive Environments, Rhodes, Greece, 5–7 June 2019; pp. 264–269.
24. Jiang, Y.; Walker, N.; Kim, M.; Brissonneau, N.; Brown, D.S.; Hart, J.W.; Niekum, S.; Sentis, L.; Stone, P. LAAIR: A Layered Architecture for Autonomous Interactive Robots. *arXiv* **2018**, arXiv:1811.03563.
25. Shah, R.; Jiang, Y.; Karnan, H.; Briscoe-Martinez, G.; Mulder, D.; Gupta, R.; Schlossman, R.; Murphy, M.; Hart, J.W.; Sentis, L.; et al. Solving Service Robot Tasks: UT Austin Villa@Home 2019 Team Report. *arXiv* **2019**, arXiv:1909.06529.
26. Chen, H.; Li, Y.; Su, D. Attention-Aware Cross-Modal Cross-Level Fusion Network for RGB-D Salient Object Detection. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 6821–6826.
27. Denninger, M.; Triebel, R. Persistent Anytime Learning of Objects from Unseen Classes. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 4075–4082.
28. Zakharov, S.; Shugurov, I.S.; Ilic, S. DPOD: 6D Pose Object Detector and Refiner. In Proceedings of the 2019 IEEE/CVF International Conference on Computer Vision (ICCV), Seoul, Korea, 27 October–2 November 2019; pp. 1941–1950.
29. Adjigble, M.; Marturi, N.; Ortenzi, V.; Rajasekaran, V.; Corke, P.; Stolkin, R. Model-free and learning-free grasping by Local Contact Moment matching. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 2933–2940.
30. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767.

31. Sun, H.; Meng, Z.; Du, X.; Ang, M.H. A 3D Convolutional Neural Network Towards Real-Time Amodal 3D Object Detection. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 8331–8338.
32. Lin, T.Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; Zitnick, C.L. Microsoft COCO: Common Objects in Context. In Proceedings of the European Conference on Computer Vision (ECCV), Zurich, Switzerland, 6–12 September 2014.
33. Kairos Face Recognition Service. Available online: https://www.kairos.com/docs/ (accessed on 29 July 2020).
34. Cao, Z.; Simon, T.; Wei, S.E.; Sheikh, Y. Realtime multi-person 2D pose estimation using part affinity fields. In Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, 21–26 July 2017; Volume 2017, pp. 1302–1310.
35. Cruciani, S.; Smith, C. Integrating Path Planning and Pivoting. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 6601–6608.
36. Dai, S.; Orton, M.; Schaffert, S.; Hofmann, A.; Williams, B. Improving Trajectory Optimization Using a Roadmap Framework. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 8674–8681.
37. Safeea, M.; Bearee, R.; Neto, P. Reducing the Computational Complexity of Mass-Matrix Calculation for High DOF Robots. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 5614–5619.
38. Rovida, F.; Wuthier, D.; Grossmann, B.; Fumagalli, M.; Krüger, V. Motion Generators Combined with Behavior Trees: A Novel Approach to Skill Modelling. In Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Madrid, Spain, 1–5 October 2018; pp. 5964–5971.