Service Robots: A Unified Framework for Detecting, Opening and Navigating Through Doors

Service Robots: A Unified Framework for Detecting, Opening and Navigating through Doors

Tatsuya Harada[1], Antonio Tejero-de-Pablos[1],
Stefano Quer[2][0000−0001−6835−8277], and Francesco Savarese[1,2]

[1] Machine Intelligence Lab., The University of Tokyo, Tokyo, Japan
[2] Dept. of Control and Computer Engineering, Politecnico di Torino, Turin, Italy

Abstract. For an autonomous robotic system, detecting, opening, and navigating through doors remains a very challenging problem. It involves several hard-to-solve sub-tasks such as recognizing the door frame and the handle, discriminating between different type of doors and their status, and opening and moving through the doorway. Previous works often tackle single individual sub-problems, assuming that the robot is moving in a well-known static environments or it is already facing the door handle. However, ignoring navigation issues, using specialized robots, or restricting the analysis to specific types of doors or handles, reduce the applicability of the proposed approach. In this paper, we present a unified framework for the door opening problem, by taking a navigation scenario as a reference. We implement specific algorithms to solve each sub-task and we describe the hierarchical automata which integrates the control of the robot during the entire process. We build a publicly available data-set which consists in 780 images of doors and handles crawled from Google Images. Using this data-set, we train a deep learning neural network, exploiting the Single Shot MultiBox Detector, to recognize doors and handles. We implement error recovery mechanisms to add robustness and reliability to our robot, and to guarantee a high success rate in every task. We carry-out experiments on a realistic scenario, the "Help Me Carry" task of the RoboCup 2018, using a standard service robot, the Toyota Human Support Robot. Our experiments demonstrate that our framework can successfully detect, open, and navigate through doors in a reliable way, with low error rates, and without adapting the environment to the robot.

Keywords: Service Robotics · Door Opening · State Machines · Object Detection · Autonomous System.

1 Introduction

Today the greatest challenge in robotics is to create robots which are able to perform increasingly complex tasks autonomously and with little previous knowledge about the environment around them. Former approaches concentrate on static (unchanging) environments, with little or no interaction between the robot and the environment. Latter frameworks have modeled non-static environments, and at a bare minimum, they must navigate in and interact with them autonomously.

The first attempts of human-robot cooperation focused on robots capable of guiding people in human coexisting environments [4, 23, 13]. Minerva [32] was installed in the Smithsonian's National Museum of American History during two weeks in 1998. The evolutionary Mobot Museum Robot Series [31] were permanently installed robots which have operated in public spaces for many years. However, influenced by the aging population problem, service robotics has focused on the design of robots to assist elderly people, or people with mobility impairments, in their daily life at home [12]. Current approaches emphasize the ability to autonomously navigate unknown environments (such as houses or offices), to perform common tasks (such as picking up objects or delivering articles), and to interact with humans [9]. For example, the Defense Advanced Research Projects Agency (DARPA) Robotics Challenge (DRC) program recently conducted a series of prize-based competition events to develop and demonstrate technology for disaster response [11, 17, 19]. The DRC Finals required robots to perform eight tasks: Drive, Egress, Door, Valve, Wall, Surprise, Rubble, and Stairs. Door opening has drawn attention not only because it is a very common task but also because of its complexity. In the "Door" task of the DRC, the robot was supposed to open a door and to travel through a 91.4 cm (36 inches) doorway, without the human assistance. Very detailed specifications were used to simplify the task. The doorway had no physical threshold, and the door could be opened inward (away from the robot). The handle was a standard American with Disabilities Act-compliant lever, which released the latch in either the up or the down direction. The task was considered complete when all points of robot ground contact were past the door threshold.

In general, a robust unified pipeline including navigation and door opening, which does not rely on prior knowledge of the environment or on the characteristics of the door, requires the following tasks: Detection of the door, estimation of the type and status of the door, understanding of the opening direction, recognition and grasping of the handle, and navigation through the door. Many existing approaches tackle this pipeline only partially and they concentrate on independent tasks, often neglecting navigation issues. Many other techniques suppose that the robot initially faces the door. Unfortunately, the position of the robot with respect to the door can greatly influence the success of the handle detection process, meaning that the robot needs to know the position of the door within the environment to proceed correctly. Thus, these strategies are not suitable for realistic scenarios in which the robot is moving and interacting with a dynamically changing environment. Moreover, while an off-the-shelf system is desirable, most of the existing approaches use custom-made robots which imply very high costs and completely hinder reproducibility.

In this work, we present a unified framework to open doors while navigating the environment. We assume the robot navigates an unmodified house, that is, a house furnished with common furniture pieces and with non-automatic doors. We consider robot navigation in a structured environment, admitting semantic navigation. This allows studying the door opening problem from the perspective of a realistic navigation problem. We suppose no prior knowledge of the properties of the doors such that these attributes (i.e., door width, handle position, and opening direction) are estimated at run-time. We also recognize whether the door is closed or partially open, whether it has to be pushed or

pulled, and we perform appropriate actions to open it. We present a detailed hierarchical automata model of our framework. Using this model, we decompose the overall task into sub-tasks, and we perform proper error recovery during all main phases. We solve the implied sub-problems adopting a unified approach, providing detailed explanations of the resulting automata. To automatic detect doors and handles, we leverage a deep learning approach based on the Single Shot MultiBox Detector (SSD). In order to train such a detector, we build and make available the "MIL-door" data-set[3], including 780 different images of doors and handles. After the door and the handle have been detected, depth images are used to evaluate the location of the handle with a higher precision. This strategy allows our robot to recognize doors and handles even while navigating through unknown environments, that is, without previously knowing their existence.

While the majority of the proposed solutions use specific architectures, such as the Personal Robot 2 (PR2) robotic platform [21] or other custom-made robots [17], we implemented our framework into a standard general purpose robot, namely, the Toyota Human Support Robot (HSR) [33], To evaluate it, we chose a complex task among the RoboCup 2018 [26] challenges, namely the "Help Me Carry" task. In this task, the user instructs the robot to fetch an object in a specific location in a different room, and he awaits for the robot to return. We force the robot to follow different paths on the outward journey and on its way back (with different doors along the two paths) and we dynamically change the environment status during its trip. After that, we focus on the sub-task of grasping an handle, forcing our HSR platform to deal with a large variety of doors and handles. We present extensive experimentation showing low failure rate and a very efficient recovery procedure, able to rectify errors in the majority of the cases. Overall, our analysis show the high reproducibility and the broad applicability of our approach.

It has to be noticed that this work is an extended version of the conference paper [29]. While the conference paper focuses only on a few steps of the entire work-flow, the current one describes the entire process with more details, more accurate author's considerations and hints on the work done. Abstract, introduction, contributions, related works, and conclusions have been completely rewritten and are now organized in a completely different way. The core sections include new details and some extra descriptive pictures. An explicit section on future works has also been added to indicate our current effort in the area. References are now more complete and updated.

1.1 Contributions

The principle of the proposed framework is to provide a comprehensible solution to the problem of door opening in a unified fashion. That is, while most related works focus only in individual modules (e.g., door detection [7], door unlatching [25], we tackle the entire problem end to end: From the moment the robot starts navigating the environment to the moment the robot traverses the door and reaches its destination. The direct benefit of such a unified framework is its high applicability to solve a real-world door-opening problem. Moreover, to the

[3] The data set is publicly available at https://www.mi.t.u-tokyo.ac.jp/projects/mil-door.

best of our knowledge, current literature does not contain any method or evaluation for amalgamating all the required modules to solve the end-to-end door opening problem. Building such a framework is challenging, given the complexity of the system obtained when combining all modules. To successfully build the proposed framework, we identified the following design requirements:

- Comprehensibility: The end-to-end door opening problem involves multiple behaviors. A state-machine implementation should follow an understandable relationship among the modules, and consider all possible cases in the task pipeline.
- Robustness: The framework should be able to handle an error at any point of the state machine execution.
- Reproducibility: Other researchers should be able to re-implement our framework. For that, explicitly specifying the parameter values used and other implementation details is essential, but not enough. Deploying the framework on a standard platform is also preferable over a closed implementation.

In this research, in order to achieve comprehensibility, our implementation of the framework follows a hierarchical structure of state machines. In order to achieve robustness, we also define an error recovery module that can deal with errors at any point of the door opening. The error recovery module behaves in a hierarchical fashion, to adapt to the state machine structure of our network. In order to achieve reproducibility, we provide the necessary details as well as sharing the datasets built for the training. In addition, we employed the Toyota HSR standard robot platform to deploy our framework and conduct our experiments.

To sum up, our contributions are the following:

- We present a unified framework to open doors while our robot navigates through an unmodified house. We suppose no prior knowledge of the property of the doors or the handles, as these characteristics are estimated at runtime. Our robot autonomously recognizes doors and handles, it performs automatic door type detection, and it executes appropriate actions to open and to traverse it.
- We describe our framework using a hierarchical automata model. The model is adopted to decompose the overall task into sub-tasks and to perform proper error recovery during all main phases. A deep learning neural network is used to detect doors and handles in the unknown environment.
- We implement our framework into a standard general purpose robot, the Toyota Human Support Robot. We analyze our robot's behavior during the "Help Me Carry" task in a realistic scenario, and we check it with several type of doors and handle. Overall, we prove the high reproducibility and the broad applicability of our approach.

1.2 Roadmap

The remainder of the paper is organized as follow. Section 2 reports details on related and recent works in the same area. Section 3 describes our hardware and software platforms, and our semantic navigation framework. Section 3.4 overviews our solution from the point of view of an automata model. It also

presents the door opening problem, and it explains our solutions to solve all sub-tasks, It finally introduces a realistic scenario, i.e., the "Help Me Carry" task, which we take as a reference. before and after making contact with the door. Section 5 describes the experiments we run to evaluate our framework. Finally, Section 6 summarizes our conclusions, and it discusses future research lines.

2 Related Works

In the field of computer vision, many research groups have proposed solutions to the problem of navigating an environment and interacting with it. Other works have focused more on recognizing door frameworks and handles, and moving through the doorways. The challenging task of door opening while navigating the environment has also received a lot of attention.

Rhee et at. [25] develop an indoor service robot equipped with a manipulator, with 6 degree of freedom and a multi-fingered hand, specifically adapted to door opening. As appropriately managing sensors and motions is essential for a service robot system, the authors propose active sensing methodologies in order to overcome uncertainty problems in real environments.

Kim et al. [6] employ cheap three-axis force sensors to successfully open a door using a home service robot called Hombot, which is equipped with an anthropomorphous manipulator arm.

Petrovskaya et al. [24] present a unified, real-time, algorithm that simultaneously models the position of the robot within the environment, as well as the objects to be manipulated. The approach is motivated by the fact that the state of an object significantly impacts the navigation task, thus the authors' goal is to simultaneously model a dynamic environment and to localize the robot within it.

Aude et al. [2] propose a new algorithm to enable a robot to autonomously find and cross doors within an unknown environment based on two main features: The identification of long straight lines and the determination of the baseboard's angle and position. They also restrict the robot's knowledge about the environment to the door's width and they detect door frames through image manipulation based on Gaussian and Sobel Filters and Hough Transforms.

Ott et al. [22] focus on the task of opening a door with no previous knowledge of the door size or on the door opening trajectory. The whole application is divided into three sub-tasks: The localization of the door handle, the turning and opening of the door handle, the movement through the door hinge until the door is sufficiently wide open. The exact localization of the door handle with respect to the mobile platform is done by using an on-board laser range scanner and a vision system.

Andreopoulos et al. [1] try to solve the door opening problem using a robotics wheelchair. They used a computer vision approach based on Viola-Jones for door and handle recognition. However, they only study handle detection and grasping, without proposing a method for door opening.

Jain et al. [10] roughly estimate the handle position using a laser scan. After that, the robot haptically searches for the door handle over the surface of the door. After the handle unlatching, the door is pushed to be opened. They do

not study the case of pulling door and they do not move the robot through the door.

Rusu et al. [28] present a laser-based approach for door and handle identification. The approach builds on a 3D perception pipeline to annotate doors and their handles solely from sensed laser data, without any a priori model learning. In particular, the authors segment the parts of interest using robust geometric estimators and statistical methods applied on geometric and intensity distribution variations in the scan.

Klingbeil et al. [15] combine a visual algorithm with laser data to locate the handle in the space. However, after handle unlatching, they do not tackle the problem of door opening.

Similar considerations can be made for Chitta et al. [5], where a planning algorithm is proposed for opening (pulling and pushing) doors, but the robot needs to know in advance if the target door is a pulling or a pushing one.

Meeussen et al. [21] propose a framework that integrates autonomous navigation and door opening. For door detection, they use a point cloud representation, while for handle recognition, they combine laser scans and a computer vision approach. Although they analyzed the entire navigation and door opening problem, their approach requires the knowledge of several details on the environment, such as the door width and the door type.

Kim et al. [14] detect doors using a context-based object recognition. The authors use the robotic context, such as the robot's viewpoint and the average height of doorknobs, to enhance the efficiency of object recognition. Robotic context is applied in the pre-processing step of object recognition to speed up the process and to reduce the false-positive rate by restricting the search space in the captured image. This approach, albeit applying for the first time both robotic context and shape-based object recognition to door detection, has a limited applicability due to the necessity to known the environment.

Gray et al. [8] present a framework that handles non-spring and spring-loaded doors, in cluttered or confined work-spaces, planning the approach to the door, pushing or pulling it open, and passing through. These task remain challenging as spring-loaded doors require making and breaking contacts with the door and preventing the door from closing while passing through. In order to plan a door-opening procedure quickly and reliably, the author start the planning using a low-dimensional, graph-based representation of the problem. However, the author do not analyze the entire problem flow. Moreover, their opening strategy requires to store additional information about the doors.

Shalaby et al. [30] build a navigation assisting tool for visually impaired people. Their based this tool on an inexpensive digital camera, such as the one used by tablets or mobile devices, able to gather information from the surrounding environment. The author also present a technique for reliable and robust door identification pairing visual information and door geometric description seen as a 4-side polygons. They implement and test the algorithm using MATLAB and its large image processing library. However, the approach requires a prior knowledge of doors details (such as the height of the handle), limiting the method applicability to only well-known scenarios.

Vertical edges have long been used by the robotics community as a first step for door detection. Fernández et al. [7] concentrate on high-level features, like

doors and corridors, which are considered as key elements in urban buildings to achieve a localization with a high semantic or symbolic processing capabilities. The authors evaluate the position of the surrounding doors by fusing the information from a monocular web-cam and a 2D laser rangefinder. By considering a real-world environment, the authors demonstrate that their technique may perform the door detection task very reliably with a computational cost that allows the procedure to be used with light on-board computers and end-user cameras.

Lee et al. [18] develop a motion planning algorithm to enable humanoids to remove an object that is blocking its path. To remove an object in its path, a humanoid must be able to reach it. Unfortunately, stretching the arms (which are shorter than the body and the legs) is not sufficient to reach an object located at some distance away or on the ground. Therefore, the authors ensured reachability by a combination of motions that include kneeling and orienting the pelvis. Indeed, they focus on the optimization of the posture of a humanoid that is reaching toward a point, which depends on the initial posture, the location of the point, and the desired manipulability of the humanoid's arms.

The Defense Advanced Research Projects Agency (DARPA) Robotics Challenge (DRC) [17] was motivated by the 2011 nuclear disaster at Fukushima, in Japan. This event illuminated society's vulnerability to natural and man-made disasters and the inability of existing robot technology to help avert or ameliorate the damage. Given this framework, Johnson et al. [11] discus the challenges they faced in transitioning from simulation to hardware. They also illustrate the lessons learned both during the training period and the competition, addressing the value of reliable hardware and solid software practices. Given the same framework, Jeongsoo et al. [19] run experiments on robots performing tasks in a nuclear disaster situation. The authors concentrate on a humanoid robot platform (i.e., the DRC-HUBO+) able to solve complex tasks under restricted communication conditions, as the ones in a region filled with radiation. They presented a survey of their platform including the overall hardware configuration, software architecture, various control methods for operating the robot, and the vision system. They also provide details on the task-oriented vision algorithms that were used to solve the given tasks.

Boston Dynamics [3] presented a solution based on the cooperation of two SpotMini robots. However, given the robot structure (i.e., a four-legged robot), it is hard to transfer the approach to common service robots. Moreover, their approach is not public.

3 Configuration

3.1 Hardware Platform

As our development platform, we used the Toyota *Human Support Robot* (HSR). The robot is aimed at helping elderly people and people with disabilities. Given its design, HSR is optimal for operating in home settings without any modification that facilitates its tasks (e.g., automatic doors). Toyota also provides some primitives and some basic software routine for controlling the robot.

The HSR body is cylindrical with a set of wheels that makes the robot movable in all directions. It is equipped with a folding arm capable of grabbing

objects, manipulating handles and even grasping paper sheets from the floor. Thanks to its microphone array and its speakers, HSR is able to receive voice commands and communicate with the user. Several sensors allow the robot interacting with the surrounding environment. The HSR head is equipped with a stereo video camera and a depth camera. The robot base is equipped with a collision detector. The Robot Operating System (ROS)[4] is installed on the robot, allowing communicating with the hardware layer. This way, writing low level controlling algorithms is not necessary.

3.2 Software Architecture

Fig. 1 shows our software architecture. We designed it to implement the robot's functionality, and it is the backbone of the entire system. It allows managing several basic tasks, the human-robot interaction, and easily adding new functionality on-demand (e.g., replacing voice commands with visual QR-code inputs). This improves system versatility, but it is not essential for the paper's goal.

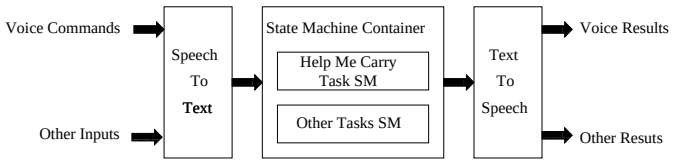

Fig. 1. Our Robot Software Architecture consists of three layers: A speech to text layer for command processing, a state machine container layer that activates state machines according to the task, and a text to speech layer for result conveying.

We defined three different layers:

- A command processing layer (speech-to-text). We use the HSR's microphone array to capture the user command, and then we internally process it.
- A container (state machine container). State machines are deployed to solve different tasks.
- A user-friendly communication layer (text-to-speech). This is used to convey the operation results to the user.

The first layer processes the user's voice command, and it forwards the result to the second layer. To interpret the voice command, and generate a command, we used the Google Cloud Speech-to-Text API[5]. This tool allows developers to convert speech into text exploiting the power of neural networks and using the Google Cloud suite. Depending on the given command, the second layer activates the proper state machine to execute the task required by the user. The third layer receives the results of the state machines, which are interpreted and communicated to the user in a user-friendly fashion. The state machine container

[4] http://www.ros.org.
[5] https://cloud.google.com/speech-to-text.

is the element that provides flexibility to the entire architecture. It is possible, in fact, to embed new state machines for executing tasks. We implement all state machines using SMACH[6].

3.3 Semantic Navigation Framework

For the path planning we rely on the ROS global and local path planners. These modules receive the desired coordinates in the space, and they convert these coordinates into commands to move the robot. Using the ROS navigation stack built-in Hector-SLAM algorithm [16] we can create a map describing the environment and the obstacles. This map allows the robot to receive coordinates and reach specific locations by automatically choosing an optimal path free of obstacles. However, semantic navigation requires a richer description of the environment to convert human understandable locations (e.g., *the kitchen table*) into suitable coordinates for the robot. As a consequence, additional information needs to be added to the map to improve the knowledge about the environment. We propose a framework for creating and managing semantic maps. This framework works as an interface layer, converting the location sent by the user to a location understandable by the motion planning module. Using RVIZ[7] we manually associate coordinates in the path planner map to human understandable locations. The association among coordinates and locations are stored as metadata into an *xml*, and a *csv* files.

We manage two different types of entities in the environment: *Rooms* and *locations*. A room is a portion of the map identified by walls or boundaries. Locations are places inside rooms. Each room can contain multiple locations. A room entity is identified by its name and it is represented by a list of corners, arranged as a polygon, plus a room center. To manage polygons and coordinates we use the python package *matplotlib.path*. A location, on the other hand, is represented by a location name, its coordinates in the map and some attributes describing the place (e.g., "isStorage" is a Boolean attribute stating if the location is a storage area). The hierarchical relationship between rooms and locations are stored in *xml* format while the room and location names with their respective coordinates are stored in *csv* format.

Fig. 2 reports an example of the files we use to store the semantic information (left and middle) and a graphical representation of a possible environment map (on the right). The hierarchical relationship between rooms and locations are stored in *xml* format while the room and location names with their respective coordinates are stored in *csv* format. In the graphical representation of a possible environment map, R1–R5 designate rooms and D1–D4 indicate doors. R3 and R5 are not separated by a wall. The position of elements in the map is retrieved with respect a fixed reference system as represented in the figure. The origin of the Cartesian system is the robot initial position, from where the entire process starts. Even though the location of the doors is indicated, the robot keeps checking for the door while approaching it, to calibrate its position and

[6] SMACH is a ROS-independent Python library for building hierarchical state machines.

[7] RVIZ is a tool for displaying sensor data using ROS.

its state (open/closed, etc.). The semantic navigation framework is also used for completing other tasks, such as localizing a person or an object.

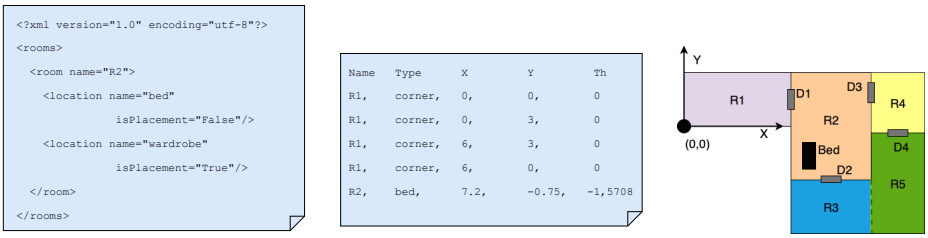

Fig. 2. The file on the left (in xml format) is an example of the rooms-to-locations relationship. The file in the middle (in csv format) is an example of the associations between rooms and locations and coordinates in the map. The map on the right, is the one for the navigation environment, with rooms (R), doors (D), and locations (Bed).

To gain planning stage flexibility, we also developed a way-points based navigation approach. In this way, to move the robot between two locations in the map, we can force it to follow intermediate points not belonging to a specific or optimal path. This is particularly useful to test motion features in specific parts of the scenario, or to reach specific places during the trajectory (e.g., to force the robot to pass through a specific door). The path between intermediate points is computed by the ROS path planner. A dictionary data structure is used to represent way-points paths: The keys are entity pairs (i.e., the source and the destination in the map), and the values are the list of places reached along the path. The way-points dictionary is stored as a *json* file. The way-points based navigation is activated if the pair source-destination is present in the dictionary. Fig. 3 is an example of dictionary to reach each room in Fig. 2, starting from room R1 and using doors as way-points.

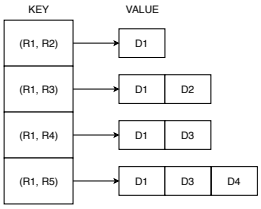

Fig. 3. Dictionary representing paths based on way-points.

3.4 The "Help Me Carry" context

As a realistic scenario for door opening, we based our study on the "Help Me Carry" task included in *Robocup 2018*. To complete it, the robot has to memorize

locations, move following user commands, avoid obstacles, and open doors. The task description is as follows. The user went shopping, and needs the robot's help for bringing inside all the bags. To complete the task the robot will:

1. Follow the owner to the bags.
2. Memorize the bags location.
3. Understand the owner's command to bring the bags to a specific different location.
4. Bring all bags to that desired specified location.

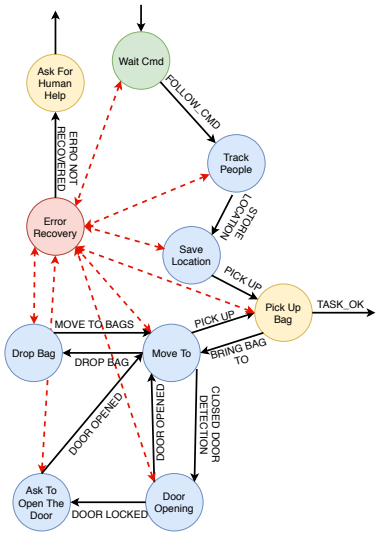

Fig. 4. Automaton representing the "Help Me Carry" task. It shows the problem of door opening in the context of a more complex task, which involves human interaction and navigation.

The automaton designed to perform the task is shown in Fig. 4. Blue circles indicate operational states, green ones are initial states, and yellow ones represent ending states. The red color represents error recovery states. Black and red dashed arrows indicate transitions between states and transitions between a state and the error recovery state, respectively. The red lines are bi-directional because after the error handling the control may be given back to the calling state. The text on the arrows represent the event causing the transition. Each state is implemented as an automaton, hence the overall architecture is a hierarchical state machine. For the sake of readability, we did not used the often used "double border" notation to identify nested state machines. This structure is quite flexible and it is easy to maintain.

As an example of behavior, the robot is activated in the state named "Wait Cmd" (wait for command). In this state the robot simply waits for commands coming from the user. If the command for following the user is received, the state

machine transit to the "Track People" state. Otherwise, if the command cannot be correctly interpreted, the state machine transits to the "Error Recovery" state. The general policy of the "Error Recovery" state is that, if the error is rectified, the control is given back to the incoming state. If the error cannot be rectified, the state returns the control to a higher level state machine or directly interacts with the user asking for help.

4 Nesting Automata

Detecting, opening and navigating through doors is a complex problem that involves many algorithms. In our approach, we decomposed the problem into different stages. The flowchart in Fig. 5 (left-hand side) describes the algorithmic approach we followed.

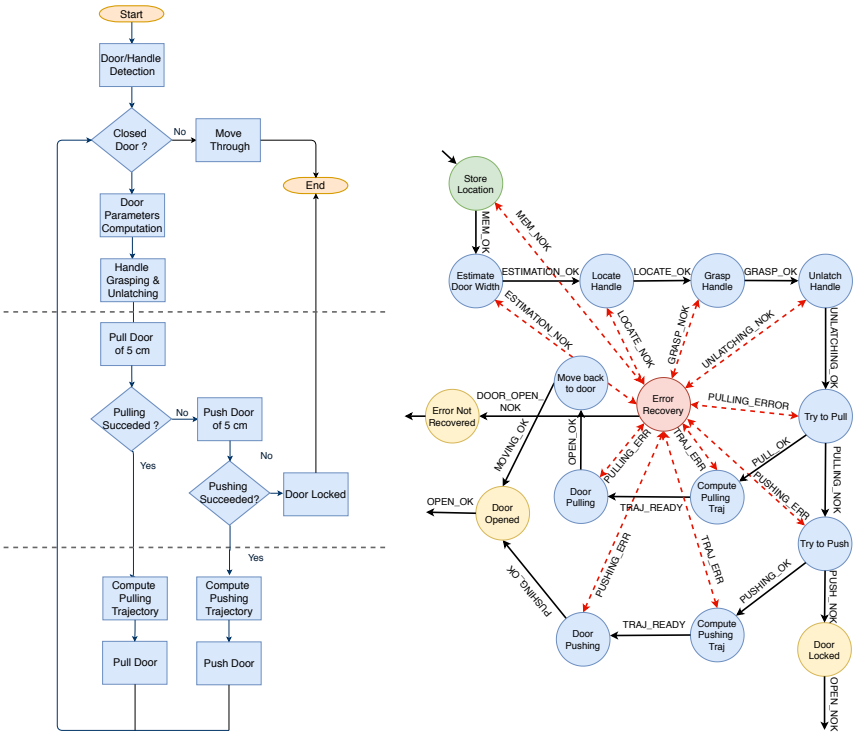

Fig. 5. On the left, we report the operational flowchart for door opening. It includes the entire flow from the moment if which the robot detects a door to the one in which it crosses the door or it understands that the door is locked. On the right, we illustrate our automaton for door opening. The names over the red dashed lines indicate the type of transition between a state and the "Error Recovery" state.

Each block involves different technologies and techniques. The top part represents the overall door/handle detection, and the door parameters estimation. The door type (pulling or pushing) is checked in the central part, whereas the opening phase is executed at the bottom part. In summary, the robot autonomously recognizes the door, it localizes the handle for grasping, and it decides the opening action (i.e., pulling or pushing). To open the door, the robot needs to know two parameters, i.e., the opening direction (pushing or pulling), and the door width. Following many other approaches, these characteristics could be annotated in advance in the environment description. However, we want to achieve a flexible and completely autonomous interaction with the door. Therefore, our robot computes the door width and the opening direction at run-time. The automaton implementing our door opening approach is shown on the right-hand side of Fig. 5. Notice that this state machine is nested in the automaton designed for the overall "Help Me Carry" task and previously described in Fig. 4. The door opening state machine is launched when the robot detects a closed door. In the first state the current location is memorized. The following states complete the entire process described in the flowchart. The automaton has 3 ending states:

- "Door Opened": Reached when the door is open.
- "Door Locked": Reached if the door is locked.
- "Error Not Recovered": Reached if an error that prevents door opening occurs.

If the "Door Locked" or the "Error Not Recovered" states are reached, the door can not be opened. This situation is managed by the state machine working at a higher hierarchical level (i.e., the one in Fig. 4). Our error recovery approach plays an essential role to reach robustness and flexibility against unexpected situations. First of all, the error is handled locally within the state in which occurs. For the sake of usability, the robot should not rely on human help for solving minor issues. Thus, in our framework, each state stores enough knowledge of the situation to handle minor problems. Examples of minor errors are: A wrong handle recognition in the 3D space, a grasping failure, a wrong location spelling from the user, etc. If local error correction is not possible, the control flow jumps to the previous (higher) hierarchical level, in which the error recovery state tries more drastic error rectification procedures. Only after the system has attempted all error recovery procedures, the robot will ask for help from the human operator.

4.1 Door and Handle Detection

For the door and handle detection we use a deep learning approach. Several deep neural networks have been proposed for object detection, and more specifically for door and handle recognition. Among state-of-the-art networks, we decided to exploit the Single Shot MultiBox Detector (SSD) neural network [20]. Authors proved that this network outperforms other well know networks, like Yolo and Faster R-CNN in terms of speed and accuracy. Moreover, since SSD performs better on embedded systems, the network can work correctly at run-time, and it guarantees a fast interaction with the environment. Compared to other single shot methods, SSD provides a much better accuracy, even with a smaller input

image size. The input to SSD is a monocular color image, and the output is a list of bounding boxes containing the detected objects in the image, namely, the top left angle of each detected object plus its height and width (*object detection* part). Each detected object has an associated label indicating which class the object belongs to (*object recognition* part).

In our version of SSD, the object recognition part is based on the VGG16 model pre-trained on the ILSVRC CLS-LOC data-set [27]. Then, we trained the object detection part, and fine-tuned the object recognition part, by constructing our own data-set, the "MIL-door" data-set. The "MIL-door" data-set consists of images of "doors" and "handles" crawled from Google Images. After filtering the erroneous results, MIL-door contains 462 images of doors and 318 images of handles, for a total of 780 images. The height and width of the images range from 400 to 1200 pixels. For each image, we manually annotated bounding boxes delimiting the area corresponding to doors and handles. Annotations are not inserted on top of the images, but stored in a separate text file. Fig. 6 shows three example images extracted from our annotated data-set.

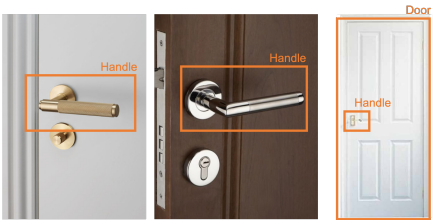

Fig. 6. Sample images from the "MIL-door" data-set.

When training our SSD network with the MIL-door data-set, we performed data augmentation on the training data, namely, 90 degrees rotations and horizontal flips. This increases the size of our data-set eight times, for a total of 6240 images. Considering that the object detection part of the original SSD was trained with 9963 images for 20 object classes, we believe our data size is reasonable for our 2 object class detection problem.

As training parameters, we used the following configuration (please refer to [20] for more details on the meaning of these parameters): Batch size 32, maximum iterations $120,000$, learning rate 0.001 (the original learning rate is decayed by 10 at iterations $80,000$, $100,000$ and $120,000$), weight decay 0.0005, γ 0.1, momentum 0.9.

We used a low learning rate to assure convergence during training and we selected it empirically. We evaluated our door and handle detection with our MIL-door data-set using a 10-fold cross-validation setting. We consider that the door (or handle) has been correctly detected if the intersection over union (IoU) between the estimated bounding box and the annotation is greater than 85%. The detection accuracy in this controlled setting is of 94.7% for doors, and 86.3% for handles. However, during the evaluation in a real setting, the IoU recognition accuracy was slightly lower than using the data-set images. This was mainly due

to three factors: The large diversity of doors that exist in the real world, the small size of some handles, and sporadic image quality loss due to poor lighting conditions.

Since there are cases in which the door is detected but the handle is not, we designed an error recovery algorithm to add robustness. When a door is detected but the handle is not, the robot moves slightly forward, backwards, and laterally to change the perspective until the recognition succeeds. If the handle is not detected after a certain number of trials (5 in our case), the error is passed to the above error recovery state in the state machine hierarchy.

4.2 Door Width Computation

The door width is an important parameter to correctly estimate the robot's trajectory. To compute it, we combine the door size in the image, taken from the robot camera, and the door to robot distance computed using the depth camera. Assuming that the object width on the image is $width_{image}$, and the detected distance is d, we can obtain the relative size in the real world using the following formula:

$$width_{real\ [pixel]} = width_{image} \cdot d. \tag{1}$$

However, Equation 1 measures the door size using the pixels as measurement unit. To transform the computed value from pixel into centimeters, we empirically calibrated our camera and we computed a conversion factor $conversion_{coeff}$. The door width, expressed in length units (centimeters), is thus given by:

$$width_{real\ [cm]} = d \cdot conversion_{coeff} \cdot width_{image}. \tag{2}$$

We measured the quality of our method by comparing our estimated widths against ground truth values, on four different types of doors. These doors differ in terms of color, surface material, and shape. We also varied the distance of the robot from the door from 1 m to 3 m, measures that are somehow reasonable in a home environment. We used the root mean square error to evaluate the error. Our results show that we reached an average error of $\pm$ 6 cm. As observed in our experiments, this value does not affect the door opening noticeably.

4.3 Opening Direction

To open the door, the robot should move backward from left to right if the hinges are on the right, and vice-versa. Anyway recognizing the hinge position is not robust enough, since hinges are often undefined or barely visible. However, our handle and door detector provides the handle location with respect to the door, and thus, inferring the opening direction is straightforward. The opening direction is used to compute the opening trajectory for both pulling and pushing doors (see Sections 4.7 and 4.8, respectively).

4.4 Closed Door

The door detected in the door recognition phase may be already open. To check this, we use the HSR's RGB-D sensor, the Xtion PRO LIVE. First, we obtain

the depth image corresponding to the frame where the door has been located. Then, we take two horizontal rows (e.g., one in the lower half and one in the upper). Finally, we compute the Sobel derivative along the horizontal direction of these lines, and we check if it contains values above a certain threshold t. This allows our method to detect if there are edges where the depth suddenly increases, which translates into the door being open.

We experimentally established that the door can be considered open if the log_{10} of the derivatives exceed a threshold $t = 3.5$.

4.5 Handle Grasping and Unlatching

Once the door opening direction has been established, and the distance from the door d evaluated, the robot can approach the handle enough to get a more precise measure of its location with the depth sensor. If some error occurs while evaluating the handle position, we retrieve a new depth measurement from the sensor to get the right location. The robot, with its grip open, gets in front of the door, and when it reaches the handle location, the grip closes and the robot grasps the handle. To unlatch the handle, we combine the robot hand rotation with a downward movement. We rotate the hand 20 degrees, and we move it downwards 10 *cm*. We empirically found that HSR does not have a strong grip and a rotation plus a downward movement can improve the pressure that the hand can apply to the handle. This allows a robust unlatching even if the handle is not grasped perfectly at its end, or the surface of the handle is slippery (e.g., metallic).

4.6 Door Type Checking

Before computing the opening trajectory the robot has to understand the door type, i.e., whether the door is a pulling or a pushing door. To discriminate between the two categories, after the grasping and the unlatching, the robot tries to move backwards and forward to test the opening type. First, it attempts to pull the door back 5 *cm* while monitoring the force acting on the wrist torque sensor. The measure of 5 *cm* has been heuristically selected as a good compromise among several requirements. If during this movement, the torque on the wrist sensor grows continuously, the door cannot be pulled. In this case, the HSR attempts to push the door by moving forward and it checks the force acting on the wrist sensor as before. In case the torque force does not increase in one of these two attempts, the robots start the opening phase (see Sections 4.7 and 4.8). On the other hand, if the door cannot be pulled or pushed, the robot assumes that the door is locked. The "Error Recovery" state handles this case by calling for human help.

We also considered other approaches for testing the door type. One of those involves monitoring movement of the robot's base while performing the test. This approach did not succeed mainly because, to measure a significant movement of the base, we have to move the robot more than 5 *cm*. This in turn can damage both the robot and the door (e.g., by pulling a pushing door too hard). Another approach implies the classification of the door type using a computer vision approach. However, this solution depends largely on the size of the training dataset, which should contain a wide variety of doors and annotations indicating their

type. Unfortunately, many available images are not annotated, and manually create a large data-set is very time consuming.

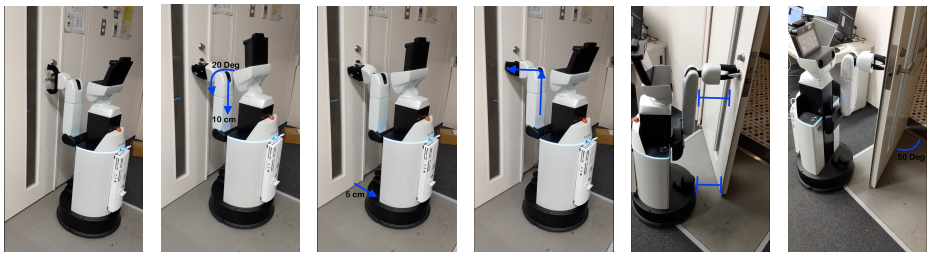

Fig. 7. A visual example of our door opening approach. The HSR first grasps the handle and then it unlatches it. After that, HSR tries to move back for 5 *cm* to pull the door. If the door cannot be pulled, the robot moves the handle back to its neutral position, and the door is opened by moving backwards and drawing an angle with respect to the door closing position. During the entire process the door-to-robot distance is maintained constant.

Notice that all checks performed by our approach are done to assure robustness and to minimize the number of errors. We emphasize the importance of robustness in such a complicated scenario, since an error in door type recognition could lead to hard-to-manage situations or risks for the robot or the handle and the door integrity.

4.7 Door Pulling

Fig. 7 shows the entire flow for opening a pulling door, from the moment the robot must grasp the handle to the one in which the door is open. Fig. 8 shows the corresponding code flow.

When the robot stands in front of the door, and before starting the door pulling phase, the application stores the current robot position. These coordinates will be used when the door is open, as the robot will move back to the stored position to pass through the door. The first three images, from left to right, are part of the door type understanding process described in Section 4.6. In the latter phase, the robots moves backward 5 *cm* to check whether the door is a pulling one. In the affirmative case, the robot moves the handle back to its neutral position. A visual representation is given in the forth picture. This action emulates typical human behavior, and it effectively reduces the load on the robot wrist that does not need to hold the handle down. At this point, the robot computes the pulling trajectory as shown in the fifth image. The final trajectory is an arc-shaped sequence of map coordinates that form an angle of 80 degrees with respect to the door hinges. In this way, the door is opened wide enough for the robot to pass through it.

Because the HSR's arm has less than six degrees-of-freedom (DoF), we have to move the base and the arm together, keeping the robot hand in a fixed position. As a consequence, the door-to-robot distance remains constant. In this

Fig. 8. Schematic code flow for opening a pulling door (left), and opening a pushing door (right). The code flows are encoded as SMACH state machines, and they are fully integrated in our software framework.

way, we do not need to continuously check for collision between the robot and the door. This situation is shown in Fig. 7(e). Once the robot completes the trajectory, it releases the handle, and it moves back in front of the door to continue the navigation toward the final goal. The robot position saved in the first state is used as a target position to cross the door.

4.8 Door Pushing

Following the flowchart of Fig. 5, if the robot detects that the door cannot be pulled, it checks whether it can be pushed, and, in this latter case, the pushing process starts. The pushing door action flow is detailed on the left-hand side image of Fig. 8. As in the pulling door case, our robot attempts to push the door to check the opening type. After the handle releasing phase, the robot moves in front of the door at a fixed distance of 50 *cm*. Once this position is reached, the robot first extends its arm to reach the door, which is already open a few centimeters after pushing it to check its type. As the robot is going to move forward, reaching the door is not strictly necessary. At the same time, we also monitor the wrist sensor to assure that no unexpected collision occurs. During the pushing phase, the HSR moves forward, and when the phase finishes, the robot is on the other side of the door. The last action executed by the robot before restarting the normal navigation, is to retract its arm into its original and safer position.

To succeed in the pushing action, the handle position is an important parameter. When unlatching the handle, the robot faces it, but during the pushing action, some collisions may occur. Since HSR is a left-handed robot, the most unfavorable scenario is when the handle is on the right side. A schematic top-view of this situation is given in Fig. 9. Since HSR is a left-handed robot, the most unfavorable scenario is when the handle is on the right side of the door.

While pushing the door, a collision check is performed in the robot base to prevent HSR from hitting the door frame. If a potential collision situation is detected, the robot is moved slightly to the left with respect to the handle. If a collision is detected, the “Error Recovery” state stops the robot and moves

it back to the beginning of the pushing stage. These strategies were validated empirically, and allowed for a safe and robust navigation through doors, as described in the next section.

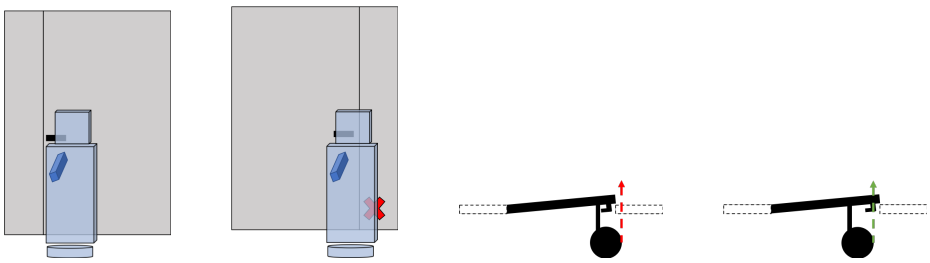

Fig. 9. The figure shows two ways of pushing a door and passing through it, depending on the handle position (left or right-hand side). Since HSR is a left-handed robot, the most unfavorable scenario is when the handle is on the right side of the door. In this case the HSR may suffer a collision. To avoid hitting the door frame, the sensor on the robot base is activated. If HSR detects a possible collision, its position is slightly shifted to the left.

5 Experimental analysis

We evaluated our unified framework by means of two set of experiments. These experiments were designed to verify two main aspects: 1) Our framework's robustness in a real navigation scenario, and 2) The quality of the entire door opening process with different doors, handles, materials, etc.

First, we evaluated the door opening process in a realistic navigation scenario by using a simplified version of the "Help Me Carry" task previously described. In this task, the user instructs the robot to fetch an object in a specific location in a different room, and he awaits for the robot to return. We also imposed waypoints during navigation, i.e., we force the robot to follow a different path on the way back. To run this scenario, we arranged a house environment similar to the one in Fig. 2. Initially, the HSR robot is in a location within room R1. The robot is supposed to reach room R4 by passing through doors D1, D2, and D4. Then, it should go back to the initial position by passing through doors D3 and D1. The doors in this task have different characteristics. When moving from R1 to R2, door D1 is a pushing door with the handle on the left. Door D2 is open. when moving from R5 to R4, and door D4 is a pulling door with its handle on the left. On the way back, when moving from R4 to R2, door D3 is a pulling door with its handle on the left. Finally, when the robot moves back from R2 to R1, D1 is still open. The robot detected the doors during navigation, following a route determined by the ROS path planner. Since the experiment does not involve any obstacles, we did not employed the way-points navigation approach. Notice that the door type and handle position affects the door opening process

in terms of the selected trajectories and the final success rate. In order to show the robustness of our framework, the door and handle attributes are unknown by the robot.

We commanded the robot to execute the task 50 times. In all cases, the robot reached R4 without navigation errors, and it successfully detected and discriminated between closed and opened doors. The accuracy of the door and the handle detection in the real scenario does not vary significantly with respect to the detection accuracy reported for our MIL-door data-set. Whenever a handle was not initially recognized, the error recovery procedure forced the robot to move slightly forward, backwards, and laterally to change the perspective until the recognition was successful. This procedure provided a recognition success rate up to 95%. In the remaining 5%, the error persisted so the higher hierarchical automata level dealt with it. Moreover, even if initially the location of the detected handle was not aligned perfectly, the location was refined when approaching the handle and using depth images. Regarding the handle grasping, every time the HSR could not hold the grip on a handle, the error recovery procedure reactivated the detection phase and the "door opening" phase restarted from the beginning.

In light of these results, we designed a second experiment with an emphasis on the handle grasping sub-task. In this experiment, the HSR had to deal with a variety of doors and handles, which differ in terms of door type (pushing or pulling), handle position (left or right), and material (slippery or non-slippery). We commanded the robot to move from room R1 to room R2 while modifying the configuration of D1. The robot starts in front of the door ready to grasp the handle, and it stops after the door is open (passing through is not required). As above, the robot does not know the door and handle attributes. We conducted 20 runs for each door and handle configuration. Notice that the door type influences the robot trajectory, whereas the handle material influences the quality of the handle grasp and its holding process. Moreover, some metallic handles may cause noise in the depth image due to reflections. We separate the door opening results for slippery handles (metallic), and non-slippery handles (wood or plastic-like material), and their location with respect to the door (i.e., left or right). Similarly, we also consider spring loaded doors, that is, doors that close by themselves after they are open. We do not evaluate opening pushing spring loaded doors since, once the robot arm releases the handle after the unlatching, the door closes again before the HSR has the chance to push it.

Table 1 summarizes the results for this second experiment. The handle localization using depth images proved to be robust with different handle shapes and materials. After the handle grasping, our approach recognized in 100% of the cases the door type, i.e., whether the HSR had to pull or push the door. As the HSR grip did not have enough strength to hold slippery handles (in particular, those in spring loaded doors) the door opening did not always succeed. However, when an error arose, the robot was able to retry the task by itself by following the error recovery procedure previously described. The robot asked for human help only in a total of 3 occasions. This results are very promising for a practical application, as the recovery procedure is able to rectify errors in most cases. However, for the sake of fairness, Table 1 considers runs as failed whenever an error arose, even if the robot recovered from the error autonomously. Overall,

we reached a 98% of success rate for non-slippery handles, and 94% for slippery metal-like handles. Notice that these results are influenced not only by the robot's grasping ability, but also by the handle detection under different types of light reflection on the handle surface. Regarding pulling spring loaded doors, holding the handle when opening was quite challenging for the robot, specially in the case of slippery handles. This is due to the limited strength of the HSR's grip. Moreover, handles on the right side of pushing doors are more challenging due to the reasons explained in Section 4.8.

Table 1. Results of our door opening approach. The table presents the number of successes out of 20 opening attempts, with 4 different handle types. T_1: Slippery handle on the door left side. T_2: Slippery handle on the door right side. T_3: Non-slippery handle on the door left side. T_4: Non-slippery handle on the door right side.

Action Type	Handle Type			
	T_1	T_2	T_3	T_4
Pulling non-spring loaded door	16	18	18	19
Pulling spring loaded door	16	18	19	19
Pushing non-spring loaded door	20	18	20	17

5.1 Final Considerations

As seen in the evaluation, the door opening task takes advantage of the proposed framework in multiple ways:

- The benefit of our unified structure: The proposed framework encompasses the entire task of opening a door, from the start of the navigation to the goal after the door is traversed. It provides a comprehensive view of the task and the connection between subtasks, and the modules that implements them.
- The benefit of our door detection module: Our deep learning-based door detection module trained with our MIL door dataset allows recognizing a more variety of doors than rule-based methods. Moreover, it is possible to fine-tune it for adaptation to other environments if necessary.
- The benefit of the error recovery module: Our error recovery procedure is adapted to the layered structure of our state machine implementation. This allows recovering from multiple errors in different parts of the framework, making it possible to return to an upper layer if there is a problem in the current subtask that cannot be overcome.
- The benefit of using a standard platform: To solve the door opening scenario, we leverage the functionalities (e.g., sensors, navigation, etc.) of Toyotas HSR, so our proposed solution can be easily reproduced for researchers using the same or a similar platform. There is also a community for HSR[8] that supports developers and provides useful software.

[8] https://newsroom.toyota.co.jp/jp/detail/8709536

Most importantly, by evaluating the framework in an end-to-end manner, we came across several errors and situations that cannot be observed when evaluating individual parts of the problem. For example, readjusting the door position while approaching the door, collisions during door traversing, etc. Therefore, this work is a valuable contribution to the community of software developers for robots, in particular, those participating in robot competitions (e.g., Robocup), who value practical application over theoretical discussion.

6 Conclusions

In this paper we present a unified robotic framework for approaching, opening, and navigating through doors. The paper covers the analysis, design, and synthesis of such a system and our experiments on a real scenario. To the best of our knowledge, this is one of the first attempts to solve the door opening problem in a navigation scenario.

Our unified framework integrates an automata model and its state machine hierarchy. The state machine includes techniques for error recovery, enabling a robust door opening framework. We propose a deep learning-based method for door and handle detection. To appropriately train our neural network, we create, and we made publicly available, a large door and handle image dataset. To facilitate the reproducibility of our work, we implement our framework on a standard platform, i.e., the Toyota Human Support Robot (HSR). Handle grasping, door type checking, door unlatching and opening have been performed with techniques optimized for our HSR framework, but they are extrapolable to similar off-the-shelf platforms with moderate effort.

We evaluate our application in a challenging realistic scenario, named the "Help Me Carry" task within the RoboCup 2018 challenge. To complete its task, the robot was required to memorize locations, move around in an unknown environment, follow user commands, avoid obstacles, and open doors. We tested our platform against different types of doors, different types of handles, and both door opening directions (inward and outward). The robot successfully identifies the door state, distinguishing between totally open, widely open, slightly open and closed doors. The robot is also able to judge if the doorway is suitable for crossing and it is capable to drive itself across the door. Our results show the robustness and flexibility of our approach and its high reproducibility on standard service robotic platforms.

7 Future Works

Among the possible extensions of this work, we report the following.

Our current framework relies on all HSR features, such as the depth camera, the base sensor, and the wrist torque sensor. Currently, a robot missing any of these devices may not be able to perform its duty. We are working on some specific steps of the overall framework to make the application even more flexible in terms of hardware requirements.

As approaches that adapt well to changing environments are increasingly important, we plan to improve the robustness and the flexibility of our application against greater environment modifications, such as recognizing and opening

a wider variety of doors and handles. Within this framework, we also have to improve our robot's ability to recognize and adapt its behavior to moving obstacles. A recognition algorithm with the ability to identify removable obstacles and determine the positions of grasping points is required to develop a fully autonomous system. Crowded environments are also a potential target, as occasional passersby cause small unmodeled effects which become more frequent in highly crowded or cluttered environments.

ACKNOWLEDGMENTS

We would like to thank Yusuke Kurose, Yujin Tang, Jen-Yen Chang, James Borg, Takayoshi Takayanagi, Yingy Wen and Reza Motallebi for their help implementing this research. This work was partially supported by JST CREST Grant Number JPMJCR1403, Japan. The authors have been part of the HSR developer community[9], and they made use of HSR hardware and software platforms.

References

1. Andreopoulos, J.A., Tsotsos, J.K.: A framework for door localization and door opening using a robotic wheelchair for people living with mobility impairments. In: Robotics: Science and systems, Workshop: Robot manipulation: Sensing and adapting to the real world (2007)
2. Aude, E.P.L., Lopes, E.P., Aguiar, C.S., Martins, M.F.: Door Crossing and State Identification Using Robotic Vision. IFAC Proceedings Volumes **39**(15), 659–664 (2006). https://doi.org/https://doi.org/10.3182/20060906-3-IT-2910.00110, http://www.sciencedirect.com/science/article/pii/S1474667016385895, 8th IFAC Symposium on Robot Control
3. Boston Dynamics: Robots: SPOT. https://www.bostondynamics.com/spot-mini (2019), accessed: 2019-11-10
4. Burgard, W., Cremers, A.B., Fox, D., Hähnel, D., Lakemeyer, G., Schulz, D., Steiner, W., Thrun, S.: The Interactive Museum Tour-guide Robot. In: Conference on Artificial Intelligence/Innovative Applications of Artificial Intelligence. pp. 11–18. AAAI '98/IAAI '98, American Association for Artificial Intelligence, Menlo Park, CA, USA (1998), http://dl.acm.org/citation.cfm?id=295240.295249
5. Chitta, S., Cohen, B., Likhachev, M.: Planning for Autonomous Door Opening with a Mobile Manipulator. In: IEEE International Conference on Robotics and Automation. pp. 1799–1806 (may 2010). https://doi.org/10.1109/ROBOT.2010.5509475
6. Dongwon, K., Ju-Hyun, K., Chang-Soon, H., Gwi-Tae, P.: Mobile Robot for Door Opening in a House. In: Knowledge-Based Intelligent Information and Engineering Systems. pp. 596–602. Springer Berlin Heidelberg, Berlin, Heidelberg (2004)
7. Fernández-Caramés, C., Moreno, V., Curto, B., Rodríguez-Aragón, J.F., Serrano, F.: A real-time door detection system for domestic robotic navigation. Journal of Intelligent & Robotic Systems **76**(1), 119–136 (2014). https://doi.org/10.1007/s10846-013-9984-6, https://doi.org/10.1007/s10846-013-9984-6

[9] https://newsroom.toyota.co.jp/jp/detail/8709536

8. Gray, S., Chitta, S., Kumar, V., Likhachev, M.: A single planner for a composite task of approaching, opening and navigating through non-spring and spring-loaded doors. In: IEEE International Conference on Robotics and Automation. pp. 3839–3846 (may 2013)
9. Hernandez, K., Bacca, B., Posso, B.: Multi-goal Path Planning Autonomous System for Picking up and Delivery Tasks in Mobile Robotics. IEEE Latin America Transactions **15**(2), 232–238 (2017)
10. Jain, A., Kemp, C.C.: Behaviors for robust door opening and doorway traversal with a force-sensing mobile manipulator. In: RSS Workshop on Robot Manipulation: Intelligence in Human Environments (2008)
11. Johnson, M., Shrewsbury, B., Bertrand, S., Wu, T., Duran, D., Floyd, M., Abeles, P., Stephen, D., Mertins, N., Lesman, A., Carff, J., Rifenburgh, W., Kaveti, P., Straatman, W., Smith, J., Griffioen, M., Layton, B., de Boer, T., Koolen, T., Neuhaus, P., Pratt, J.: Team IHMC's Lessons Learned from the DARPA Robotics Challenge Trials. Journal of Field Robotics **32**(2), 192–208 (mar 2015). https://doi.org/10.1002/rob.21571, https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21571
12. Khatib, O.: Mobile manipulation: The robotic assistant. Robotics and Autonomous Systems **26**(2), 175–183 (1999). https://doi.org/https://doi.org/10.1016/S0921-8890(98)00067-0, http://www.sciencedirect.com/science/article/pii/S0921889098000670, field and Service Robotics
13. Kim, G., Chung, W., Kim, K.R., Kim, M., Han, S., Shinn, R.: The autonomous tour-guide robot Jinny. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). vol. 4, pp. 3450–3455 (01 2004). https://doi.org/10.1109/IROS.2004.1389950
14. Kim, S., Cheong, H., Kim, D.H., Park, S.: Context-based object recognition for door detection. In: 15th International Conference on Advanced Robotics (ICAR). pp. 155–160 (jun 2011)
15. Klingbeil, E., Saxena, A., Ng, A.Y.: Learning to open new doors. In: IEEE/RSJ International Conference on Intelligent Robots and Systems. pp. 2751–2757 (oct 2010). https://doi.org/10.1109/IROS.2010.5649847
16. Kohlbrecher, S., von Stryk, O., Meyer, J., Klingauf, U.: A flexible and scalable SLAM system with full 3D motion estimation. In: IEEE International Symposium on Safety, Security, and Rescue Robotics. pp. 155–160 (Nov 2011). https://doi.org/10.1109/SSRR.2011.6106777
17. Krotkov, E., Hackett, D., Jackel, L., Perschbacher, M., Pippine, J., Strauss, J., Pratt, G., Orlowski, C.: The DARPA Robotics Challenge Finals: Results and Perspectives. Journal of Field Robotics **34**(2), 229–240 (mar 2017). https://doi.org/10.1002/rob.21683, https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21683
18. Lee, I., Oh, J.H.: Humanoid Posture Selection for Reaching Motion and a Cooperative Balancing Controller. Journal of Intelligent & Robotic Systems **81**(3), 301–316 (mar 2016). https://doi.org/10.1007/s10846-015-0225-z, https://doi.org/10.1007/s10846-015-0225-z
19. Lim, J., Lee, I., Shim, I., Jung, H., Joe, H.M., Bae, H., Sim, O., Oh, J., Jung, T., Shin, S., Joo, K., Kim, M., Lee, K., Bok, Y., Choi, D.G., Cho, B., Kim, S., Heo, J., Kim, I., Lee, J., Kwon, I.S., Oh, J.H.: Robot System of DRC-HUBO+ and Control Strategy of Team KAIST in DARPA Robotics Challenge Finals. Journal of Field Robotics **34**(4), 802–829 (2017). https://doi.org/10.1002/rob.21673, https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21673
20. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: SSD: Single Shot MultiBox Detector. In: European conference on computer vision. pp. 21–37 (2016), http://arxiv.org/abs/1512.02325

21. Meeussen, W., Wise, M., Glaser, S., Chitta, S., McGann, C., Mihelich, P., Marder-Eppstein, E., Muja, M., Eruhimov, V., Foote, T., Hsu, J.M., Rusu, R.B., Marthi, B., Bradski, G.R., Konolige, K., Gerkey, B.P., Berger, E.: Autonomous door opening and plugging in with a personal robot. In: IEEE International Conference on Robotics and Automation. pp. 729–736. Anchorage, Alaska, USA (may 2010)
22. Ott, C., Bäuml, B., Borst, C., Hirzinger, G.: Autonomous opening of a door with a mobile manipulator: A case study. 6th IFAC PSymposium on Intelligent Autonomous Vehicles **40**(15), 349–354 (2007). https://doi.org/https://doi.org/10.3182/20070903-3-FR-2921.00060, http://www.sciencedirect.com/science/article/pii/S1474667016346857
23. Peterson, L., Austin, D., Kragic, D.: High-level control of a mobile manipulator for door opening. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). vol. 3, pp. 2333–2338 (oct 2000). https://doi.org/10.1109/IROS.2000.895316
24. Petrovskaya, A., Ng, A.Y.: Probabilistic Mobile Manipulation in Dynamic Environments, with Application to Opening Doors. In: Proceedings of the 20th International Joint Conference on Artifical Intelligence. pp. 2178–2184. IJCAI'07, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (2007), http://dl.acm.org/citation.cfm?id=1625275.1625627
25. Rhee, C., Chung, W., Kim, M., Shim, Y., Lee, H.: Door opening control using the multi-fingered robotic hand for the indoor service robot. In: Proc. IEEE International Conference on Robotics and Automation. vol. 4, pp. 4011–4016. IEEE (01 2004). https://doi.org/10.1109/ROBOT.2004.1308898
26. RoboCup 2018: http://www.robocup2018.com (2018), accessed: 2019-11-10
27. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. Int. J. Comput. Vision **115**(3), 211–252 (Dec 2015). https://doi.org/10.1007/s11263-015-0816-y, http://dx.doi.org/10.1007/s11263-015-0816-y
28. Rusu, R.B., Meeussen, W., Chitta, S., Beetz, M.: Laser-based perception for door and handle identification. In: International Conference on Advanced Robotics. pp. 1–8 (jun 2009)
29. Savarese, Francesco and Tejero-de-Pablos, Antonio and Quer, Stefano and Harada, Tatsuya: Detecting, Opening and Navigating through Doors: A Unified Framework for Human Service Robots. In: 14th International Conference on Software Technologies (ICSOFT 2019). pp. 416–427 (01 2019). https://doi.org/10.5220/0007947604160427
30. Shalaby, M.M., Salem, M.A., Khamis, A., Melgani, F.: Geometric model for vision-based door detection. In: 9th International Conference on Computer Engineering Systems. pp. 41–46 (dec 2014). https://doi.org/10.1109/ICCES.2014.7030925
31. Sunspiral, V., Kunz, C., Nourbakhsh, I.: The History of the Mobot Museum Robot Series: An Evolutionary Study. pp. 514–518 (01 2001)
32. Thrun, S., Bennewitz, M., Burgard, W., Cremers, A.B., Dellaert, F., Fox, D., Hahnel, D., Rosenberg, C., Roy, N., Schulte, J., Schulz, D.: MINERVA: a second-generation museum tour-guide robot. In: IEEE International Conference on Robotics and Automation. vol. 3, pp. 1999–2005 (1999)
33. Toyota: Partner Robot. https://www.toyota-global.com/innovation/partner_robot/robot/#link02 (2019), accessed: 2019-11-10