The b-it-bots@Home 2024 Team Description Paper

Manish Saini Melvin Paul Jacob Zain Ulhaq
Shashwat Pandey Nada Aweaa Bharat Kumar
Alex Mitrevski

November 27, 2023

Abstract. This paper presents the b-it-bots@Home team and one of its mobile service robots called *Lucy* – a Human Support Robot manufactured by Toyota. We present an overview of our robot control architecture and the robot's capabilities, namely the added functionalities from various research and development projects carried out by the Autonomous Systems group at Hochschule Bonn-Rhein-Sieg.

1 Introduction

The b-it-bots@Home team[1] was established in 2007 and functions as part of the international Autonomous Systems master's program at Hochschule Bonn-Rhein-Sieg (HBRS)[2]. Our team consists of bachelor's, master's, and PhD students who are advised by three tenured professors, such that we have a long history of participation at RoboCup@Home competitions.

Our initial robot Johnny, a VolksBot platform, was used by the team from 2008 to 2010. Since 2011, the b-it-bots@Home team has been working with Jenny, a Care-O-Bot 3, and has successfully participated in multiple competitions, including RoboCup, German Open, and RoCKIn. In February 2018, the team added Lucy, a Toyota HSR, to its available platforms, and as of 2019 is participating in the RoboCup@Home Domestic Standard Platform League (DSPL).

In 2022 the team participated in the HEART-MET competition, securing an impressive array of awards: 1st place in Person Detection and Gesture Recognition, 2nd place in Object Detection, and 3rd place in Activity Recognition, ultimately claiming the Overall 1st place. The momentum continued into 2023, with our team dominating the HEART-MET Field Campaign in Florence by securing 1st place victories in all three tracks: Object Detection, Person Detection,

[1] `https://a2s-institute.de/b-it-bots/b-it-botshome/`

[2] `http://www.h-brs.de`

and Object Sorting. At the HEART-MET 2023 Physical Assistive Robot Challenge competition, our team won 2nd place in the Item Delivery episode. Lastly, in the ERL(European Robotics League) competition 2023, our team excelled in both the social acceptance and Open Door episodes, claiming 1st place in both categories.

While participation at competitions has always been an integral part of our activities, we foster a research-oriented culture above all, such that we have several ongoing PhD and master's thesis projects that are very closely related to the team. In parallel, one of our goals is to deploy service robots to real-life applications. Since our research interests are highly linked to our experiences in the field, our software both contributes to and benefits from the ongoing research projects taking place at our university.

The rest of this paper presents some of the work carried out by current and previous team members, such that we focus on some of our main research interests: execution monitoring, fault detection, and diagnosis, robust manipulation, real-time and adaptive perception, as well as knowledge-based reasoning.

2 Manipulation

2.1 Manipulation Trajectory Representation and Execution

To increase the predictability of our physical robots during manipulation, we acquire manipulation trajectories using learning by demonstration. In particular, we use dynamic motion primitives (DMPs) [1] to represent Cartesian motion trajectories, which are acquired by tracking a marker array. As the reachability of the HSR's manipulator is limited, we synchronize arm and base motions when imitating demonstrated trajectories, as described in [2]. When using demonstrated trajectories, we additionally use MoveIt![3] for simple motions between predefined positions, such as moving to a pregrasp position or retrieving the arm back after executing a demonstrated trajectory.

Since the execution of demonstrated trajectories requires base motions, a rather accurate robot odometry is needed to prevent base drift. In the case of the HSR's simulation, this has been problematic since the odometry is less accurate than on the physical robot; the inaccurate odometry leads to significant base drift, particularly for manipulation goals that require considerable base motion. For this purpose, we primarily use MoveIt! in simulation, namely trajectory planning is performed during execution; this has the disadvantage of increasing the execution time, but the execution error is generally lower than with our controller for executing DMPs.

2.2 Object Grasping

To grasp objects of varied shapes and sizes, we use an adaptive strategy that chooses either a sideways or a top-down grasp for a given object. We particularly

[3] https://moveit.ros.org

use a heuristic according to which a top-down grasp is selected if the height of an object, which is estimated based on the object's 3D bounding box, is below a predefined threshold; otherwise, a sideways grasp is selected. To align the robot's gripper for grasping, the orientation of an object is determined as the direction of the object's principal component when a top-down grasp is used, and as the object's planar orientation in the case of a sideways grasp.

We are additionally working towards a learning-based method [3], which has been verified for a small group of objects and does not require hand-coded heuristics for choosing a grasping strategy. This method uses dedicated grasping models that have been learned for particular objects or object classes; when generalizing to new object classes, information about object similarity as encoded in an object ontology is combined with any prior generalization experiences.

3 Perception

3.1 Object Localization and Plane Detection

We detect 3D objects in the robot's view using an algorithm that processes an RGB-D point cloud. Detections from the algorithm can be used to avoid small objects lying on the floor, which are undetected by a 2D range sensor during navigation, or to obtain the poses of objects to be picked up. The algorithm expects a few parameters, such as the ground plane height, the maximum size of objects to be detected, and the minimum closeness to occupied cells in an occupancy grid map; these help in filtering out large objects, such as tables and walls, as well as objects that are very close to or touching large objects. The results from the object recognition model is used to extract object points from the original RGB-D cloud and estimate the objects' 3D positions. We use Eucledian clustering on the filtered and downsampled point cloud data to detect objects. The algorithm can also be configured to maintain a cache of previously detected objects for a certain duration of time; this particularly helps in avoiding obstacles on the ground after they have gone out of the robot's view during navigation. The algorithm is implemented using the Point Cloud Library (PCL).[4,5] For plane detection, we use the RANSAC algorithm [4] to fit a plane model to detected objects and find horizontal surfaces, such as tables or shelves, associated with the detected objects.

3.2 Object Recognition

The architecture of our object detection pipeline provides a standard and extensible way of integrating new models and approaches into our code base; new implementations only need to extend the *ImageDetector* base class for performing detection. Our current instantiation of the pipeline first extracts RGB data

[4] `http://docs.pointclouds.org/`

[5] `https://github.com/b-it-bots/mas_domestic_robotics/tree/kinetic/mdr_perception/mdr_cloud_object_detection`

from an RGB-D point cloud, an object detection model (shown in table 1), is then used to detect objects in the image. The detection result returned from the object detection model is then used in object localization, as explained in the previous section.

Model	**Classes**	**Use Case**	**Reference**
YOLOv5	Door, Handle	Door Opening/ Clossing	[5]
YOLOv5	YCB classes	Object Picking	[5]
SSD MobileNetV2	COCO classes	Person Detetction	[6]
SSD BlazeFace	Face	Face Detection	[7]

Table 1: Used Detection Models

3.3 Face Recognition

Our methodology revolves around employing a sophisticated face recognition library that operates seamlessly in real-time by harnessing facial embeddings, enhancing the identification process. We use a real-time vision stack that performs face detection, gender classification, and emotion classification simultaneously in a single step. Our pipeline initiates by employing MediaPipe's robust face detection model [6] to pinpoint human faces within the input camera stream. MediaPipe identifies facial regions swiftly and accurately, enabling subsequent processing steps.

In our perception stack we have integrated Face Recognition API [7] swiftly which performs facial identification using a singular step approach. This API employs pre-trained neural networks to generate face encodings, numerical representations capturing distinct facial features. These encodings, representing unique vectors for each face, enable seamless and rapid identification.

In addition to face recognition, our system integrates a facial expression recognition module capable of discerning emotions such as joy, surprise, sadness, neutrality, and anger. This enriches the understanding of facial dynamics

[6] `https://developers.google.com/mediapipe/solutions/vision/face_detector/python)`

[7] `https://github.com/ageitgey/face_recognition)`

beyond mere identification. This component is also being used by the SocRob team in the RoboCup@Home OPL league[8]

4 Planning, Reasoning, and Operation Monitoring

4.1 Task Planning

Robots operating in dynamic environments have to be designed to be robust and flexible; we are thus working towards a flexible plan-based architecture for high-level reasoning and recovery. In a first step, we have integrated ROSPlan[9] as a planning framework that covers the whole lifecycle of a task, starting from problem generation, task planning, plan dispatching, and plan monitoring.

In order to extend the plan generation with expert knowledge, we are also working towards integrating the hierarchical task network planner JSHOP2[10] into ROSPlan. The motivation for this comes from the work by Awaad et al. [8], as well as the task planning, execution, and monitoring system developed by Shpieva and Awaad [9], where JSHOP2 has been used successfully.

4.2 High-Level Knowledge Representation and Reasoning

In order to perform purposeful tasks, domestic robots need to be able to understand their environment and reason about it. While several aspects about the world, such as the locations of objects or people, have to be estimated and updated dynamically as a robot is operating, it would be suboptimal to let a robot learn everything about the environment from scratch; instead, it is more pragmatic to guide the robot's reasoning process by using an ontology that represents encyclopedic knowledge about the world (namely known facts about objects, their properties, and relations between each other). In this respect, we are working towards an ontology for domestic environments that is motivated by the KnowRob ontology [10], but represents a stripped-down version of it. Just as KnowRob, our ontology is written in the OWL Web Ontology Language[11], such that we are working on an RDFLib-based[12] interface for interacting with the encoded knowledge.

4.3 Execution and Component Monitoring

Due to the complexity of domestic environments, such as the high variability of objects and the presence of other agents, domestic robots are quite failure-prone. This raises the need for both appropriate recovery strategies during execution as well as learning mechanisms for improving the execution process, but also means

[8] https://github.com/socrob/face_classification

[9] https://github.com/KCL-Planning/ROSPlan

[10] http://www.cs.umd.edu/projects/shop/description.html

[11] www.w3.org/TR/owl-ref/

[12] https://github.com/RDFLib/rdflib

that robots need to be transparent about their actions so that their policies can be understood more easily. In order to model and express action execution knowledge, we are using an action execution library[13] for (i) representing knowledge about actions, namely their inputs, outputs, and known failure cases and (ii) logging execution-relevant data. Motivated by the work in [11] and [12], we are currently using this library for executing placing actions. In addition to that, we are in the process of adapting the component monitoring framework described in [13] so that component failures can be detected early enough, which should allow us to prevent undesired events (e.g. the robot colliding with a table due to a malfunctioning arm joint) and, if automatic recovery is not possible, communicate such failures to a human operator.

5 Human Robot Interaction

For robots in domestic environments, interaction with humans, particularly in a verbal manner, is an indispensable component. This section provides an overview of how we are addressing the challenge of understanding humans and responding to them.

5.1 Speech Recognition

For detecting speech from an audio snippet and transforming it into a machine-readable format, we differentiate between online and offline methods. Online speech recognition is conducted using Google's speech recognition API [14], thereby leveraging the model's high recognition rates and robustness; however, since we cannot assume that our robot will always have a stable internet connection, we have also integrated PocketSphinx[15], a speech recognition library developed by Carnegie Mellon University, for understanding commands even when our robot is offline. The results of a project that has compared open source speech recognition toolkits for domestic environments[16] point out that the speech recognition toolkit Kaldi [14] is more suitable for everyday household tasks; we are thus in the process of switching from PocketSphinx to Kaldi as our primary offline speech recognition tool.

5.2 Speech Matching

Even with a highly accurate speech recognition model, recognized speech might be faulty or incomplete, which generally means that it needs to be further processed by comparing what was recognized with what a robot already knows

[13] `https://github.com/b-it-bots/action-execution`

[14] `https://cloud.google.com/speech-to-text/`

[15] `https://github.com/cmusphinx/pocketsphinx`

[16] Conducted as part of the RoboLand project: https://www.h-brs.de/de/roboland-telepraesenz-roboter-im-haeuslichen-lebens-und-pflegearrangement-von-personen-mit-demenz-im

(such as a database of known questions and commands) and reacting to that accordingly. For comparing recognized and known speech, we use the *Levenshtein distance*, which is a string comparison metric that measures the difference between two sentences; if the similarity between the recognized speech and a sentence in a previously created database is above a certain threshold, the robot would reply in case of a question or execute a corresponding action in case of a command.

Although the Levenshtein distance provides acceptable practical performance, we have additionally implemented *Double Metaphone*, which is an approach for indexing words by sound, as an additional speech verification method. This allows us to refer to the results of two different methods, thereby increasing the robustness of the system even if the speech recognition was only partially correct.

5.3 Natural Language Understanding

As part of an ongoing project, we are also investigating natural language processing toolkits for use in our robots. The project is still ongoing, but some preliminary results suggest that SocRob's NLU ROS package[17] is a promising candidate, so we have started integrating it for further testing.

5.4 Visual Interaction

We use the robot's integrated display to show relevant images during its interactions with humans. The robot's display becomes a canvas for visual communication, showcasing images that are directly pertinent to the conversation or task at hand. This feature allows for a more dynamic and intuitive exchange between humans and robots. Whether it's displaying step-by-step instructions, providing visual aids for better comprehension, or presenting relevant data in a comprehensible manner, this feature significantly elevates the user experience.

5.5 Gesture Recognition

To achieve gesture recognition we use MediaPipe holistic landmarks detection [18] to obtain landmarks from the face and hands which are then passed on to a Convolutional Neural Network (CNN) architecture for gesture recognition, with accuracies of 90% on a custom gesture dataset. Currently, our models are able to identify two sets of gestures including numbers from 1 to 5, as shown in figure 1 and gestures like nodding, stop signs, thumbs down, waving, pointing, calling someone, thumbs up, waving someone away, and shaking head. These allow users to interact with the robot non-verbally. For example, the user can select items from a menu by indicating the number corresponding to the item, or can even confirm his/her order using a thumbs-up sign.

[17] `https://github.com/socrob/mbot_natural_language_processing`

[18] `https://developers.google.com/mediapipe/solutions/vision/holistic_landmarker`

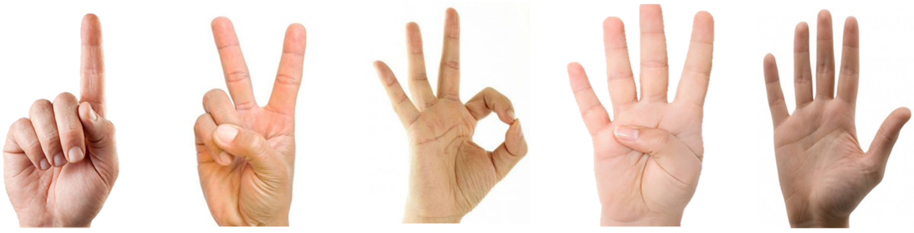

Fig. 1: Gestures of numbers 1 to 5

6 Interacting with doors

In the latest series of advancements, our team has focused on developing and refining the capability of our humanoid robot, Lucy, to open and close doors autonomously. This functionality is crucial for enhancing the robot's ability to navigate and operate in human environments, especially in domestic settings.

6.1 Sensor Integration and Door Detection

Initially, we integrated advanced sensory capabilities to enable Lucy to detect and localize doors and their levers within its environment. Using a combination of RGB-D sensors and advanced computer vision algorithms, Lucy can now accurately identify the door and its lever's position and orientation.

6.2 Kinematic Planning for Door Manipulation

The next challenge was to enable Lucy to interact with doors physically. This involved developing a kinematic model that allows for precise manipulation of door handles. By integrating this model with our existing control architecture, Lucy can move its manipulator arm toward door levers, adjust its grip, and apply the necessary forces to turn door handles and push or pull doors.

6.3 Adaptive Control Strategies

We implemented adaptive control strategies to ensure successful door manipulation in various scenarios. These strategies enable Lucy to adjust its approach based on the type of door handle and the force required to open or close the door.

Our successful implementation of the door opening and closing task marks a significant milestone in our project, paving the way for more complex and practical applications in domestic environments.

7 Open-Source Contributions

Our team is committed to making our work accessible to the robotics community; most of our code is thus available through our official GitHub organization.[19] Our HSR-specific code is not publicly available to the nature of our contract with Toyota, but all of our core robot-independent software can be found there, including the following major components:

- **mas_domestic_robotics**: Core robot-independent components for domestic applications (`https://github.com/b-it-bots/mas_domestic_robotics`)
- **mas_execution_manager**: A library for creating state machines and managing their execution [15] (`https://github.com/b-it-bots/mas_execution_manager`)
- **mas_knowledge_base**: A library exposing interfaces for interacting with an OWL ontology, the ROSPlan knowledge base, and mongodb_store (`https://github.com/b-it-bots/mas_knowledge_base`)
- **mas_perception_libs**: Robot-independent perception components, including a Boost Python wrapper to allow executing point cloud processing functionalities from Python, since the original implementation uses the C++-based PCL (`https://github.com/b-it-bots/mas_perception_libs`)
- **ros_dmp**: A library for learning and executing dynamic motion primitives (`https://github.com/b-it-bots/ros_dmp`)
- **dataset_interface**: A library with various utilites for data augmention, object detection, and face recognition (`https://github.com/b-it-bots/dataset_interface`)
- **explainable-robot-execution-models**: An implementation of the learning-based object grasping algorithm described above (`https://github.com/alex-mitrevski/explainable-robot-execution-models`)
- **ftsm**: A library for generically implementing components with a so-called fault tolerant state machine (`https://github.com/ropod-project/ftsm`)

A diagram showing the interaction between our main open-source components is shown in Fig. 2. We additionally have a list of freely available tutorials for using some of our components.[20]

8 Conclusions and future work

This paper presents the b-it-bots@Home team and various functionalities that are in active use and development by the team. While the described functionalities have already been integrated into the HSR and in its simulation, the integration and development of new functionalities is a continuous process driven by our research goals, which are reflected through several ongoing PhD projects, master's theses, and funded projects. Our ongoing work is aligned with the aspects

[19] `https://github.com/b-it-bots`

[20] `https://github.com/b-it-bots/mas_tutorials`

[21] Diagram taken from `https://github.com/b-it-bots/mas_domestic_robotics`

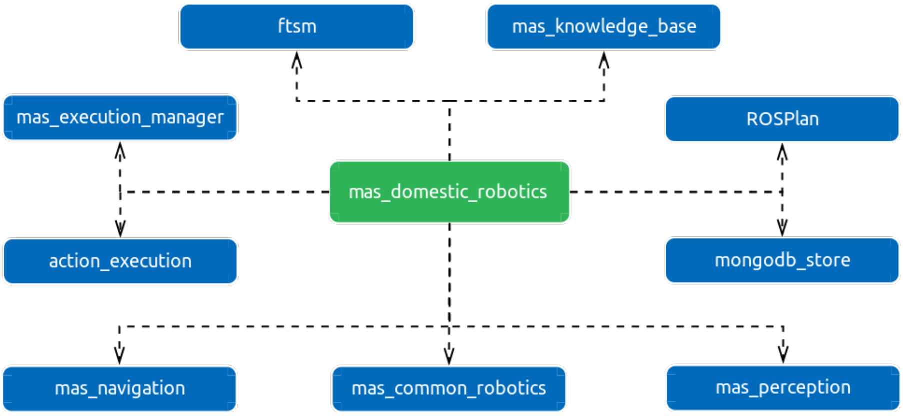

Fig. 2: Various components developed and/or used by our team[21]

described in this paper and includes long-term experience acquisition, execution monitoring, and communicating robot intentions for transparent execution, aspects that are particularly important for increasing the practical acceptance of domestic robotics.

Acknowledgement

We gratefully acknowledge the continued support of the team by the b-it Bonn-Aachen International Center for Information Technology and Hochschule Bonn-Rhein-Sieg.

References

1. A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal. Dynamical Movement Primitives: Learning Attractor Models for Motor Behaviors. *Neural Computation*, 25(2):328–373, 2013.
2. A. Mitrevski, A. Padalkar, M. Nguyen, and P. G. Plöger. "Lucy, Take the Noodle Box!": Domestic Object Manipulation Using Movement Primitives and Whole Body Motion. In *Proceedings of the 23rd RoboCup International Symposium*, 2019.
3. A. Mitrevski, P. G. Plöger, and G. Lakemeyer. Ontology-Assisted Generalisation of Robot Action Execution Knowledge. In *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems (IROS)*, pages 6763–6770, 2021.
4. M. A. Fischler and R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. *Communications of the ACM*, 24(6):381–395, June 1981.
5. Glenn Jocher, Alex Stoken, Jirka Borovec, NanoCode012, ChristopherSTAN, Liu Changyu, Laughing, tkianai, Adam Hogan, lorenzomammana, yxNONG, AlexWang1900, Laurentiu Diaconu, Marc, wanghaoyang0106, ml5ah, Doug, Francisco Ingham, Frederik, Guilhen, Hatovix, Jake Poznanski, Jiacong Fang, Lijun

Yu, changyu98, Mingyu Wang, Naman Gupta, Osama Akhtar, PetrDvoracek, and Prashant Rai. ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements, October 2020.

6. Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.
7. Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019.
8. I. Awaad, G. K. Kraetzschmar, and J. Hertzberg. The role of functional affordances in socializing robots. *International Journal of Social Robotics*, 7(4):421–438, March 2015.
9. E. Shpieva and I. Awaad. Integrating Task Planning, Execution and Monitoring for a Domestic Service Robot. *Information Technology*, 57(2):112–121, March 2015.
10. M. Tenorth and M. Beetz. KnowRob – A Knowledge Processing Infrastructure for Cognition-enabled Robots. *Int. Journal of Robotics Research (IJRR)*, 32(5):566–590, Apr. 2013.
11. A. Kuestenmacher, N. Akhtar, P. G. Plöger, and G. Lakemeyer. Towards Robust Task Execution for Domestic Service Robots. In *Journal of Intelligent & Robotic Systems*, volume 76, pages 5–33, September 2014.
12. A. Mitrevski, A. Kuestenmacher, S. Thoduka, and P. G. Plöger. Improving the Reliability of Service Robots in the Presence of External Faults by Learning Action Execution Models. In *Proceedings of the 2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4256–4263, 2017.
13. A. Mitrevski, S. Thoduka, A. Ortega Sáinz, M. Schöbel, P. Nagel, P. G. Plöger, and E. Prassler. Deploying robots in everyday environments: Towards dependable and practical robotic systems. In *29th Int. Workshop Principles of Diagnosis DX'18*, 2018.
14. D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz, J. Silovsky, G. Stemmer, and K. Vesely. The kaldi speech recognition toolkit. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 1–4. IEEE Signal Processing Society, 2011.
15. A. Mitrevski and P. G. Plöger. Reusable Specification of State Machines for Rapid Robot Functionality Prototyping. In *Proceedings of the 23rd RoboCup International Symposium*, 2019.
16. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. B. Foote, J. Leibs, R. Wheeler, and A. Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
17. W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C. Fu, and A. C. Berg. SSD: Single Shot MultiBox Detector. In *Computer Vision – ECCV 2016*, pages 21–37, 2016.

Lucy (Toyota HSR) Software and External Devices

We use a standard Human Support Robot (HSR) from *Toyota*. No modifications have been applied.

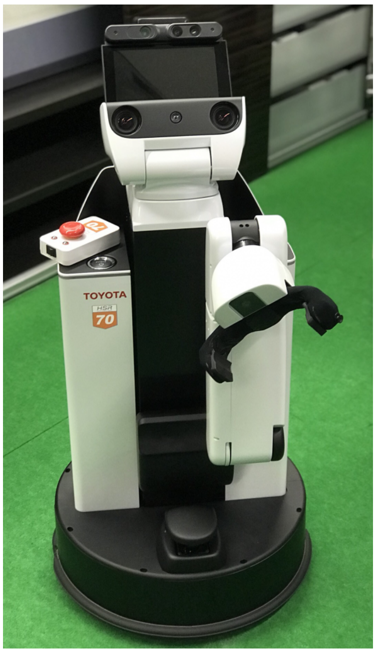

Fig. 3: Lucy: Toyota HSR

Robot's Software Description

For our robot, we are using the following software:

- *Platform*: Robot Operating Systems (ROS) [16]
- *Navigation*: Built-in ROS-based functionalities provided by Toyota
- *Arm control*: In-house imitation learning framework and MoveIt![22]
- *Task planning*: ROSPlan using the LAMA planner
- *Object recognition*: Single-shot Multi-box Detector (SSD) [17]
- *Speech recognition*: Google Speech (online), PocketSphinx and Kaldi (offline)
- *Natural language processing*: SocRob's NLU[23]
- *Gender recognition*: In-house CNN model[24]

Most of our software is publicly available at `https://github.com/b-it-bots`.

External Devices

Lucy relies on the following external hardware:

- Alienware 15", Intel Core i9 processor and GTX 1080

Cloud Services

Lucy connects to the following cloud services:

- Speech recognition: Google Speech[25]

[22] `moveit.ros.org`

[23] `https://github.com/socrob/mbot_natural_language_processing`

[24] `https://github.com/oarriaga/face_classification`

[25] `https://cloud.google.com/speech-to-text/`