Tech United Eindhoven @Home 2022 Champions Paper

A. Aggarwal, M.F.B. van der Burgh, J.J.M. Lunenburg, R.P.W. Appeldoorn, L.L.A.M. van Beek, J. Geijsberts, L.G.L. Janssen, P. van Dooren, L. Messing, R.M. Núñez and M.J.G. van de Molengraft

Eindhoven University of Technology,
Den Dolech 2, P.O. Box 513, 5600 MB Eindhoven, The Netherlands
http://www.techunited.nl, techunited@tue.nl,
https://github.com/tue-robotics

Abstract. This paper provides an overview of the main developments of the Tech United Eindhoven RoboCup@Home team. Tech United uses an advanced world modeling system called the Environment Descriptor. It allows for straightforward implementation of localization, navigation, exploration, object detection & recognition, object manipulation and human-robot cooperation skills based on the most recent state of the world. Other important features include object and people detection via deep learning methods, a GUI, speech recognition, natural language interpretation and a chat interface combined with a conversation engine. Recent developments that aided with obtaining the victory during RoboCup 2022 include people and pose recognition, usage of HSR's display and a new speech recognition system.

1 Introduction

Tech United Eindhoven[1] (established 2005) is the RoboCup student team of Eindhoven University of Technology[2] (TU/e), which joined the ambitious @Home League in 2011. The RoboCup@Home competition aims to develop service robots that can perform everyday tasks in dynamic and cluttered 'home' environments. The team has been awarded multiple world vice-champion titles in the Open Platform League (OPL) of the RoboCup@Home competition during previous years, and two world champion titles in 2019 and 2022[3][4][5] in the Domestic Standard Platform League (DSPL). In the DSPL, all teams compete with the same hardware; all teams compete with a Human Support Robot (HSR), and use the same external devices. Therefore, all differences between the teams regard only the software used and implemented by the teams.

[1] http://www.techunited.nl
[2] http://www.tue.nl
[3] https://tinyurl.com/DSPLBangkok2022Stage1Score
[4] https://tinyurl.com/DSPLBangkok2022Stage2Score
[5] https://tinyurl.com/DSPLBangkok2022FinalScore

Tech United Eindhoven consists of (former) PhD and MSc. students and staff members from different departments within the TU/e. This year, these team members successfully migrated the software from our TU/e built robots, AMIGO and SERGIO, to HERO, our Toyota HSR. This software base is developed to be robot independent, which means that the years of development on AMIGO and SERGIO are currently being used by HERO. Thus, a large part of the developments discussed in this paper have been optimized for years, whilst the DSPL competition has only existed since 2017[6]. All the software discussed in this paper is available open-source at GitHub[7], as well as various tutorials to assist with implementation. The main developments that resulted in the large lead at RoboCup 2022, and eventually the championship, are our central world model, discussed in Section 2, the generalized people recognition, discussed in Section 4, the head display, discussed in Section 5.3 and the new speech recognition system in Section 5.4.

2 Environment Descriptor (ED)

The TU/e Environment Descriptor (ED) is a Robot Operating System (ROS) based 3D geometric, object-based world representation system for robots. ED is a database system that structures multi-modal sensor information and represents this such that it can be utilized for robot localization, navigation, manipulation and interaction. Figure 1 shows a schematic overview of ED.

ED has been used on our robots in the OPL since 2012 and was also used this year in the DSPL. Previous developments have focused on making ED platform independent, as a result ED has been used on the PR2, Turtlebot, Dr. Robot systems (X80), as well as on multiple other @Home robots.

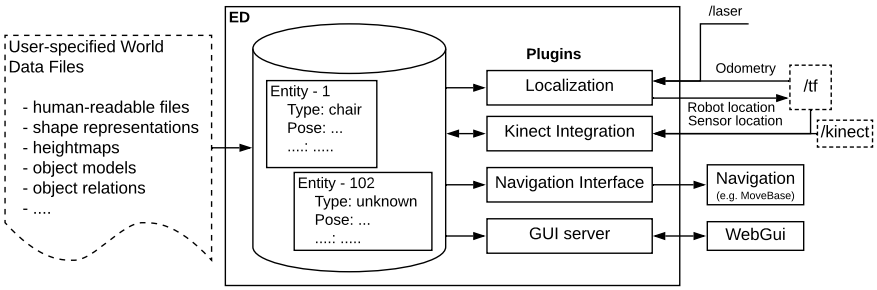

Fig. 1. Schematic overview of TU/e Environment Descriptor. Double sided arrows indicate that the information is shared both ways, one sided arrows indicate that the information is only shared in one direction.

[6] https://athome.robocup.org/robocuphome-spl
[7] https://github.com/tue-robotics

ED is a single re-usable environment description that can be used for a multitude of desired functionalities such as object detection, navigation and human machine interaction. Improvements in ED reflect in the performances of the separate robot skills, as these skills are closely integrated in ED. This single world model allows for all data to be current and accurate without requiring updating and synchronization of multiple world models. Currently, different ED plug-ins exist that enable robots to localize themselves, update positions of known objects based on recent sensor data, segment and store newly encountered objects and visualize all this in RViz and through a web-based GUI, as illustrated in Figure 9. ED allows for all the different subsystems that are required to perform challenges to work together robustly. These various subsystems are shown in Figure 2, and are individually elaborated upon in this paper.

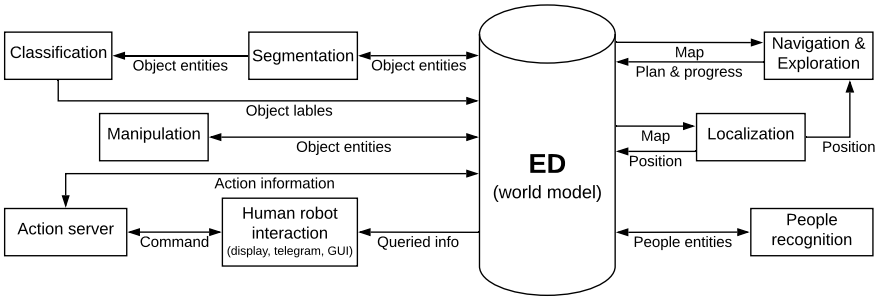

Fig. 2. A view of the data interaction with robot skills that ED is responsible for.

2.1 Localization, Navigation and Exploration

The *ed_localization*[8] plugin implements AMCL based on a 2D render of the central world model.With use of the *ed_navigation* plugin[9], an occupancy grid is derived from the world model and published. With the use of the *cb_base_navigation* package[10] the robots are able to deal with end goal constraints. The *ed_navigation* plugin allows to construct such a constraint w.r.t. a world model entity in ED. This enables the robot to navigate not only to areas or entities in the scene, but to waypoints as well. Figure 3 also shows the navigation to an area. Modified versions of the local and global ROS planners available within *move_base* are used.

[8] `https://github.com/tue-robotics/ed_localization`
[9] `https://github.com/tue-robotics/ed_navigation`
[10] `https://github.com/tue-robotics/cb_base_navigation`

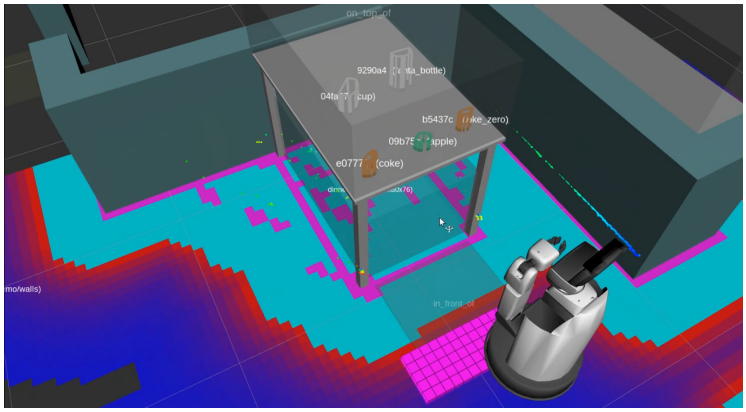

Fig. 3. A view of the world model created with ED. The figure shows the occupancy grid as well as classified objects recognized on top of the cabinet.

2.2 Detection & Segmentation

ED enables integrating sensors through the use of the plugins present in the *ed_sensor_integration* package. Two different plugins exist:

1. *laser_plugin*: Enables tracking of 2D laser clusters. This plugin can be used to track dynamic obstacles such as humans.
2. *kinect_plugin*: Enables world model updates with use of data from a RGBD camera. This plugin exposes several ROS services that realize different functionalities:
   (a) *Segment*: A service that segments sensor data that is not associated with other world model entities. Segmentation areas can be specified per entity in the scene. This allows to segment object 'on-top-of' or 'in' a cabinet. All points outside the segmented area are ignore for segmentation.
   (b) *FitModel*: A service that fits the specified model in the sensor data of a RGBD camera. This allows updating semi-static obstacles such as tables and chairs.

The *ed_sensor_integration* plugins enable updating and creating entities. However, new entities are classified as unknown entities. Classification is done in *ed_perception* plugin[11] package.

2.3 Object grasping, moving and placing

The system architecture developed for object manipulation is focused on grasping. In the implementation, its input is a specific target entity in ED, selected by a Python executive and the output is the grasp motion joint trajectory. Figure 4 shows the grasping pipeline.

[11] `https://github.com/tue-robotics/ed_perception`

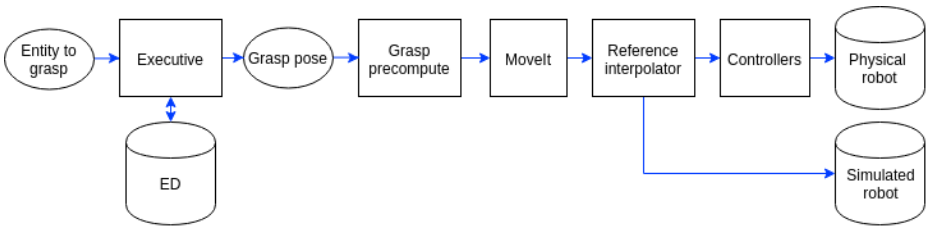

Fig. 4. Custom grasping pipeline base on ED, MoveIt and a separate grasp point determination and approach vector node.

MoveIt! is used to produce joint trajectories over time, given the current configuration, robot model, ED world model (for collision avoidance) and the final configuration.

The grasp pose determination uses the information about the position and shape of the object in ED to determine the best grasping pose. The grasping pose is a vector relative to the robot. An example of the determined grasping pose is shown in Figure 5. Placing an object is approached in a similar manner to grasping, except for that when placing an object, ED is queried to find an empty placement pose.

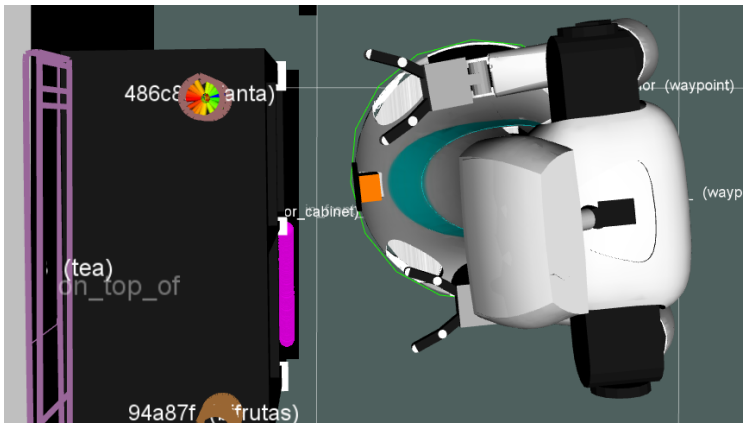

Fig. 5. Grasping pose determination result for a cylindric object with TU/e built robot AMIGO. It is unpreferred to grasp the object from behind.

2.4 World model creation

A world model is described in a SDFormat file for compatibility with the Gazebo simulation engine. Currently, world models in ED are generated in a semi-automated way by manually composing and updating multiple object models

templates[12] built over the years. New ED plugins are in works that would fully automate the updation of these templates in the future.

3 Image Recognition

The *image_recognition* packages apply state of the art image classification techniques based on Convolutional Neural Networks (CNN).

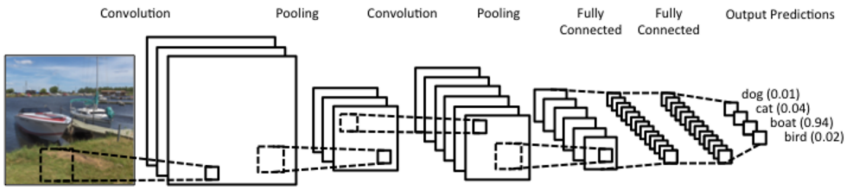

Fig. 6. Illustration of Convolutional Neural Networks (CNN) used in our object recognition nodes with use of Tensorflow.

1. **Object recognition:** Tensorflow™ with retrained top-layer of a Inception V3 neural network, as illustrated in Figure 6.
2. **Face recognition:** OpenFace[13], based on Torch.
3. **Pose detection:** OpenPose[14].

Our image recognition ROS packages are available on GitHub[15] and as Debian packages: *ros-kinetic-image-recognition*

4 People Recognition

As our robots need to operate and interact with people in a dynamic environment, our robots' people detection skills have been upgraded to a generalized system capable of recognizing people in 3D. In the people recognition stack, an RGB-D camera is used as the sensor to capture the scene information. A recognition sequence is completed in four steps. First, people are detected in the scene using OpenPose and if their faces are recognized, using OpenFace, as one of the learned faces in the robot's database, they are labeled using their known name. The detections from OpenPose are associated with the recognitions from OpenFace by maximizing the IoUs of the face ROIs. Then, for each of the recognized people, additional properties such as age, gender and the shirt color are identified. Furthermore, the pose keypoints of these recognitions are coupled

[12] `https://github.com/tue-robotics/ed_object_models`
[13] `https://cmusatyalab.github.io/openface/`
[14] `https://github.com/CMU-Perceptual-Computing-Lab/openpose`
[15] `https://github.com/tue-robotics/image_recognition`

with the depth information of the scene to re-project the recognized people to 3D as skeletons. Finally, information about the posture of each 3D skeleton is calculated using geometrical heuristics. This allows for the addition of properties such as "pointing pose" and additional flags such as 'is_waving', 'is_sitting', etc.

4.1 Pointing detection

Similar to the previous years, this year's tournament challenges too involved various non-verbal user interactions such as detecting an object the user was pointing to. In the previous section, we explained our approach to recognizing people in 3D. Once the recognition results are inserted into the world model, additional properties can be added to the people taking other entities in the world model into account, e.g. "*is_pointing_at_entity*". This information is used by the top-level state machines to implement challenges such as 'Hand Me That', the description of which can be found in the 2019 Rulebook[16]. However an additional check based on spatial queries is inserted to ensure that the correct operator is found. By using such a query it is possible to filter out people based on their location. Finally, to determine at which entity the operator is pointing to, we implemented ray-tracing, as illustrated in Figure 7.

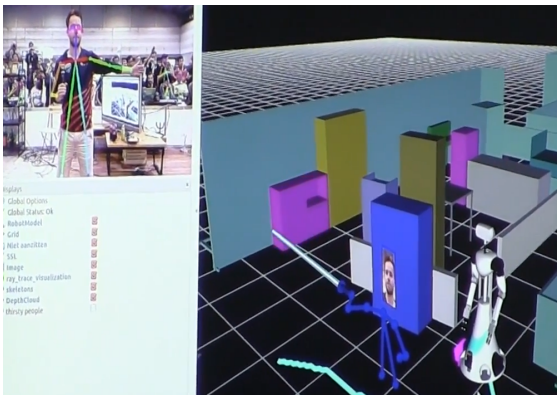

Fig. 7. Ray-tracing based on pose detection with AMIGO.

5 Human-Robot Interface

We provide multiple ways of interacting with the robot in an intuitive manner: WebGUI, subsection 5.1, and Telegram™ interface, subsection 5.2, which uses our *conversation_engine*, subsection 5.2.

[16] `http://www.robocupathome.org/rules`

5.1 Web GUI

In order to interact with the robot, apart from speech, we have designed a web-based Graphical User Interface (GUI). This interface uses HTML5[17] with the Robot API written in Javascript and we host it on the robot itself.

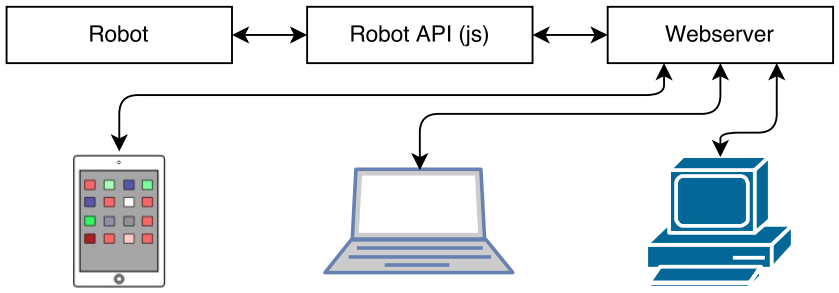

Fig. 8. Overview of the WebGUI architecture. A webserver that is hosting the GUI connects this Robot API to a graphical interface that is offered to multiple clients on different platforms.

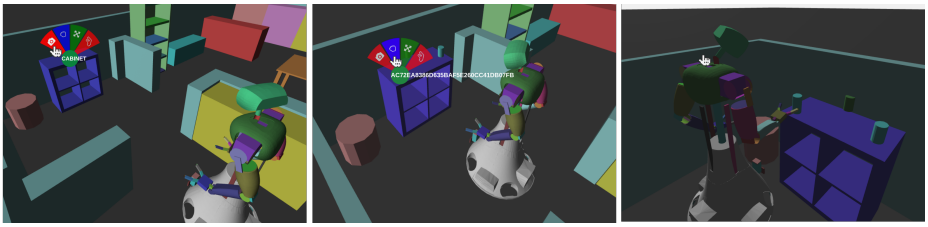

Fig. 9. Illustration of the 3D scene of the WebGUI with AMIGO. User can long-press objects to open a menu from which actions on the object can be triggered

Figure 8 gives an overview of the connections between these components and Figure 9 represents an instance of the various interactions that are possible with the Robot API.

5.2 Telegram™

The Telegram interface[18] to our robots is a ROS wrapper around the *python-telegram-bot* library. The software exposes four topics, for images and text resp. from and to the robot. The interface allows only one master of the robot at a time. The interface itself does not contain any reasoning. This is all done by the *conversation_engine*, which is described in the following subsection.

[17] `https://github.com/tue-robotics/tue_mobile_ui`
[18] `https://github.com/tue-robotics/telegram_ros`

Conversation Engine The *conversation_engine*[19] bridges the gap between text input and an action planner (called action_server). Text can be received from either Speech-to-Text or from a chat interface, like Telegram™. It is then parsed according to a (Feature) Context Free Grammar, resulting in an action description in the form of a nested mapping, along with (sub)actions and their parameters are filled in. This mapping may include references such as "it".

Based on the action description, the *action_server* tries to devise a sequence of actions and parameterize those with concrete object IDs. To fill in missing information, the *conversation_engine* engages with the user and parses any additional inputs in context to the missing info. Lastly, it keeps the user "informed" whilst actions are being performed by reporting on the current sub-task.

Custom Keyboard, Telegram HMI The user interface modality as explained above has been extended to reduce the room for operator error by only presenting the user with a limited number of buttons in the Telegram app. This has been realized through Telegram's *custom_keyboards*[20] feature. This feature is especially useful when there are only a few options, like a predetermined selection of drinks, as shown in our RoboCup@Home 2019 Finals.

We have employed this custom keyboard to compose commands word-for-word (*hmi_telegram*[21]). After a user input has been received, either via text or previous buttons, for example "Bring me the ...", the user is presented with only those words as options that might follow the input according to the grammar, eg. "apple", "orange" etc. This process iterates until a full command has been composed.

5.3 Head Display

Most people find interacting with robots a very challenging task, especially when they do not deal with them on a regular basis. It is often difficult for people to hear what the robot is saying and not always intuitive to know when to respond to it. As a solution, we use the integrated screen on the Toyota HSRs' 'head' to display useful information. Through the *hero_ display*[22] we have integrated a few different functionalities. As a default, our Tech United @Home logo with a dynamic background is shown on the screen, as depicted in Figure 10. When the robot is speaking, the spoken text is displayed, and when it is listening, a spinner along with an image of a microphone is shown. It is also possible to display custom images on this screen.

[19] `https://github.com/tue-robotics/conversation_engine`
[20] `https://github.com/tue-robotics/telegram_ros`
[21] `https://github.com/tue-robotics/hmi_telegram`
[22] `https://github.com/tue-robotics/hero-display`

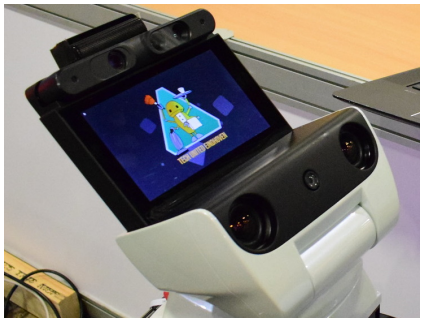

Fig. 10. The default status of HERO's head display.

5.4 Speech Recognition

Over the years, with the change of base hardware and operating systems, we implemented and experimented with multiple speech recognition systems to allow our robots to hear and understand the operator in noisy environments. We started with the Dragonfly speech recognition framework [23] using Windows Speech Recognition engine as the backend on a Windows 10 virtual machine. This system proved to not be robust against noisy environments primarily due to the default microphone of HERO. As an alternative, we investigated Kaldi-ASR [4], but finally settled with Picovoice[24], alongside the existing Dragonfly system, as it provided seamless support for our custom context-free grammars.

6 Task Execution

In the previous sections, we have described the various modules that contribute towards the functioning of our robots. However, for a challenge to be successfully performed by HERO, these modules need to be strategically integrated together in context to the said challenge. We do this by creating hierarchical state machines using SMACH[25]. Over the years, we have extracted all the commonly used integrations into the modules "*robot_smach_states*" and "*robot_skills*", and started retaining only challenge specific behaviors within the challenges.

7 Re-usability of the system for other research groups

Tech United takes great pride in creating and maintaining open-source software and hardware to accelerate innovation. Tech United initiated the Robotic Open Platform website[26], to share hardware designs. All our software is available

[23] `https://dragonfly2.readthedocs.io/en/latest/`
[24] `https://picovoice.ai/`
[25] `https://github.com/tue-robotics/tue_robocup`
[26] `http://www.roboticopenplatform.org`

on GitHub[27]. All packages include documentation and tutorials. The finals of Robocup@Home 2022[28] demonstrates all the capabilities of HERO, as described in the previous sections. Tech United and its scientific staff have the capacity to co-develop (15+ people), maintain and assist in resolving questions.

8 Community Outreach and Media

Tech United has organised 3 tournaments: Dutch Open 2012, RoboCup 2013 and the European Open 2016. Our team member Loy van Beek was a member of the Technical Committee between 2014-2017 and Peter van Dooren has been since 2022. We also carry out many promotional activities for children to promote technology and innovation. Tech United often visits primary and secondary schools, public events, trade fairs and has regular TV appearances. Each year, around 50 demos are given and 25k people are reached through live interaction. Tech United also has a very active website[29], and interacts on many social media like: Facebook[30], Instagram[31], YouTube[32], Twitter[33] and Flickr[34]. Our robotics videos are often shared on the IEEE video Friday website.

References

1. Amos, B., Ludwiczuk, B., Satyanarayanan, M.: Openface: A general-purpose face recognition library with mobile applications. Tech. rep., CMU-CS-16-118, CMU School of Computer Science (2016)
2. Fox, D.: Adapting the sample size in particle filters through kld-sampling. The International Journal of Robotics Research **22**(12), 985–1003 (2003)
3. Fox, D., Burgard, W., Thrun, S.: The dynamic window approach to collision avoidance. IEEE Magazine on Robotics & Automation **4**(1), 23–33 (1997)
4. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K.: The kaldi speech recognition toolkit. IEEE Signal Processing Society (2011), `http://infoscience.epfl.ch/record/192584`, iEEE Catalog No.: CFP11SRW-USB
5. Quigley, M., Conley, K., Gerkey, B.P., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: ROS: an open-source robot operating system. In: ICRA Workshop on Open Source Software (2009)
6. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A.: Going deeper with convolutions. In: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1–9 (2015). https://doi.org/10.1109/CVPR.2015.7298594

[27] `https://github.com/tue-robotics`
[28] `https://tinyurl.com/TechUnited2022AtHomeFinals`
[29] `http://www.techunited.nl`
[30] `https://www.facebook.com/techunited`
[31] `https://www.instagram.com/techunitedeindhoven`
[32] `https://www.youtube.com/user/TechUnited`
[33] `https://www.twitter.com/TechUnited`
[34] `https://www.flickr.com/photos/techunited`

A HSR's Software and External Devices

A standard Toyota™ HSR robot is used. To differentiate our unit, it has been named HERO. This name also links it to our AMIGO and SERGIO domestic service robots.

HERO's Software Description An overview of the software used by the Tech United Eindhoven @Home robots can be found in Table 1.

Table 1. Software overview

Operating system	Ubuntu 20.04 LTS Server
Middleware	ROS Noetic [5]
Simulation	Gazebo
World model	Environment Descriptor (ED), custom `https://github.com/tue-robotics/ed`
Localization	Monte Carlo [2] using Environment Descriptor (ED), custom `https://github.com/tue-robotics/ed_localization`
SLAM	Gmapping
Navigation	CB Base navigation `https://github.com/tue-robotics/cb_base_navigation` Global: custom A* planner Local: modified ROS DWA [3]
Arm navigation	MoveIt!
Object recognition	Inception based custom DNN [6] `https://github.com/tue-robotics/image_recognition/image_recognition_tensorflow`
People detection	Custom implementation using contour matching `https://github.com/tue-robotics/people_recognition`
Face detection & recognition	OpenFace [1] `https://github.com/tue-robotics/image_recognition/image_recognition_openface`
Speech recognition	Windows Speech Recognition, Picovoice `https://github.com/reinzor/picovoice_ros.git`
Speech synthesis	Toyota™ Text-to-Speech
Task executors	SMACH `https://github.com/tue-robotics/tue_robocup`

External Devices *HERO relies on the following external hardware:*

- Official Standard Laptop
- Gigabit Ethernet Switch
- Wi-Fi adapter