# Tech United Eindhoven @Home 2019 Team Description Paper

M.F.B. van der Burgh , J.J.M. Lunenburg, L.L.A.M. van Beek, J. Geijsberts, L.G.L. Janssen, S. Aleksandrov, K. Dang, H.W.A.M. van Rooy, A.T. Hofkamp, D. van Dinther, A. Aggarwal 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 straightforward implementation of localization, navigation, exploration, object detection & recognition, object manipulation and robot-robot cooperation skills based on the most recent state of the world. Recent developments are improved object and people detection via deep learning methods, a GUI, improved speech recognition, improved natural language interpretation, sound source localization and a chat interface combined with a conversation engine.

## 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 team has multiple World vice-champion titles to its name, two in the last three years, and is the current European vice-champion. Tech United Eindhoven consists of (former) PhD and MSc. students and staff members from different departments within the TU/e.

This year we are shifting our focus from our own robots, AMIGO and SERGIO to the Toyota HSR. This Team Description Paper is part of the qualification package for RoboCup 2019 in Sydney, Australia and describes the current status of the @Home activities of Tech United Eindhoven.

## 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

[1] http://www.techunited.nl
[2] http://www.tue.nl

a database system that structures multi-modal sensor information and represents this such that it can be utilized for robot localisation, navigation, manipulation and interaction. Figure 1 shows a schematic overview of ED.

ED has been used on our robots in the OPL for many years and now also the DSPL. Previously, developments have focused on making ED platform independent. As a result ED has been used on the PR2, Turtlebot and Dr. Robot systems (X80), as well as multiple other @Home teams. ED is a single re-usable

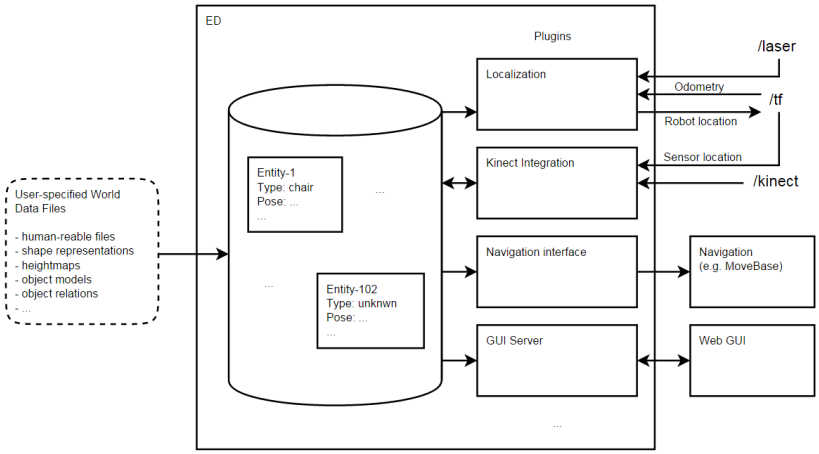


**Fig. 1.** Schematic overview of TU/e Environment Descriptor.

environment description that can be used for a multitude of desired functionalities. Improvements in ED can reflect in the performances of the separate robot capabilities. It omits 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, illustrated in Figure 8.

### 2.1 Localization, Navigation and Exploration

The *ed_localization*[3] plugin implements AMCL based on a 2D render of the central world model.
With use of the *ed_navigation* plugin[4], an occupancy grid is derived from the world model and published.
With the use of the *cb_base_navigation* package[5] the robots are able to deal with end goal constraints. The *ed_navigation* plugin allows to construct such a

[3] `https://github.com/tue-robotics/ed_localization`
[4] `https://github.com/tue-robotics/ed_navigation`
[5] `https://github.com/tue-robotics/cb_base_navigation`

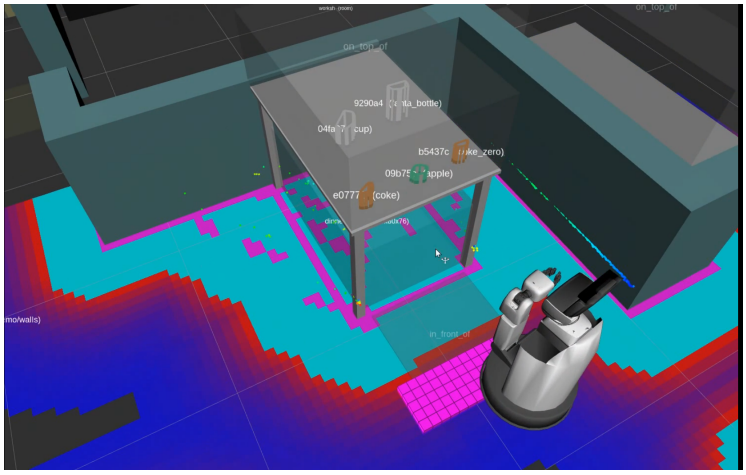


**Fig. 2.** A view of the world model created with ED. The figure shows the occupation grid as well as classified objects recognized on top of the cabinet.

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 2 also shows the navigation to an area. Modified versions of the local and global ROS planners available within *move_base* are used.

### 2.2 Object detection

**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[6] package.

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

### 2.3 Object grasping, moving and placing

As for manipulating objects, the architecture is only focused on grasping. The input is a specific target entity in ED, selected by a Python executive. The output is the grasp motion joint trajectory. Figure 3 shows the grasping pipeline. MoveIt! is used to produce joint trajectories over time, given the current con-

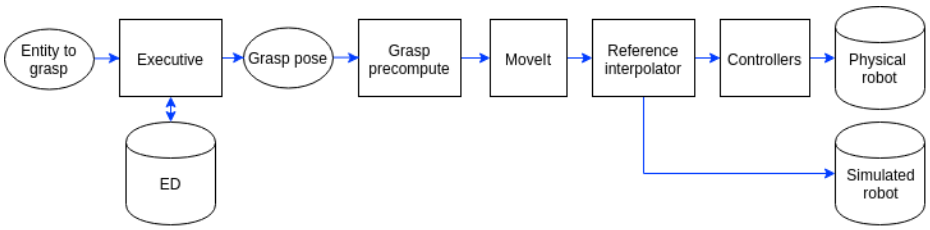


**Fig. 3.** Custom grasping pipeline base on ED, MoveIt and a separate grasp point determination and approach vector node.

figuration, robot model, ED world model (for collision avoidance) and the final configuration.
Before placing, ED is queried to find an empty placement pose. 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 4.

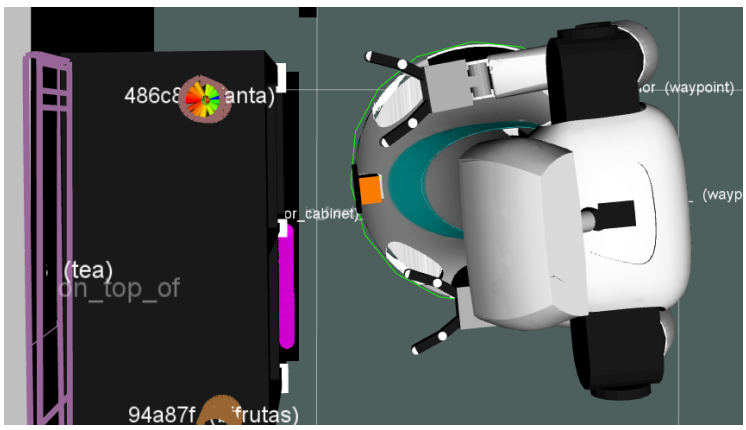


**Fig. 4.** Grasping pose determination result for a cylindric object. It is unpreferred to grasp the object from behind.

## 3 Image Recognition

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

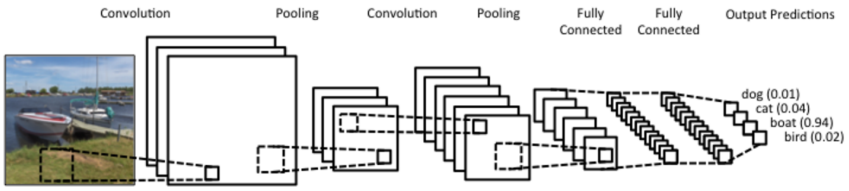


**Fig. 5.** Illustration of Convolution Neural Networks (CNN) used in our object recognition nodes with use of Tensorflow.

1. **Object recognition:** Tensorflow™ with retrained top-layer of a Inception V3 neural network, as illustrated in Figure 5.
2. **Face recognition:** OpenFace[7], based on Torch.
3. **Pose detection:** OpenPose[8]. We use ray-tracing to find the object an operator in pointing to. Figure 6 shows an example of the ray-tracing.

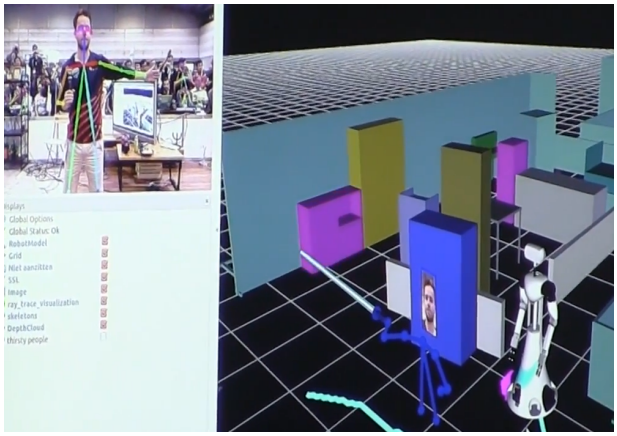

**Fig. 6.** Ray-tracing based on pose detection

Our image recognition ROS packages are available on GitHub[9] and as Debian packages: *ros-kinetic-image-recognition*

[7] `https://cmusatyalab.github.io/openface/`
[8] `https://github.com/CMU-Perceptual-Computing-Lab/openpose`
[9] `https://github.com/tue-robotics/image_recognition`

## 4 Sound source localization

We localize the sound source by determining the Direction of Arrival (DOA) using a Matrix Creator[10] board. The detection is done by cross-correlating between pairs of opposing microphone, combined with finding the two microphones with the lowest mutual phase shift and using the energy level of the microphones. This improvement is contributed back into the upstream repository. The DOA is published as a ROS Pose. Because the DSPL doesn't allow hardware changes to the robot, this software needs to be re-implemented for the Toyota HSR.

## 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 5.3.

### 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[11] with the Robot API written in Javascript and we host it on the robot itself.

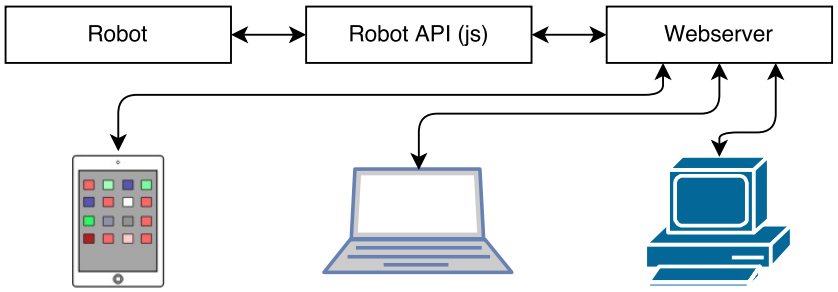


**Fig. 7.** Overview of the WebGUI architecture. A webserver that is hosting the GUI connects this Robot API to a graphical interface that is offered to multiple clients on different platforms.

[10] `https://creator.matrix.one`
[11] `https://github.com/tue-robotics/tue_mobile_ui`

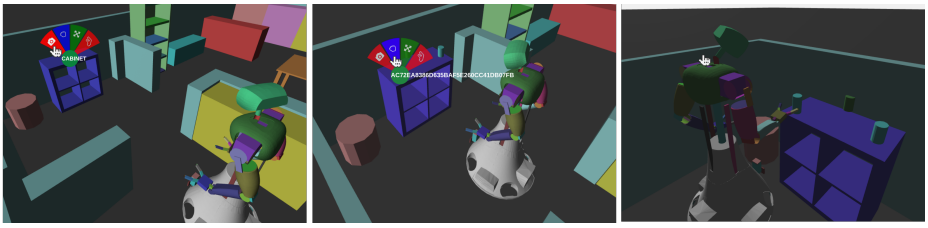

**Fig. 8.** Illustration of the 3D scene of the WebGUI. User can long-press objects to open a menu from which actions on the object can be triggered

Figure 7 gives an overview of the connections between these components and Figure 8 represents an instance of the various interactions that are possible with the Robot API.

### 5.2 Telegram™

The Telegram interface[12] 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 doesn't contain any reasoning. This is all done by the *conversation_engine*, which is described in the following subsection.

### 5.3 Conversation Engine

The *conversation_ engine*[13] 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™. The text is parsed according to a (Feature) Context Free Grammar, resulting in an action description in the form of a nested mapping. In the action description, (sub)actions and their parameters are filled in. This 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.

When the user supplies more information, the additional input is parsed in the context of what info is missing.

Lastly, it keeps the user 'informed' while actions are being performed by reporting on the current subtask.

## 6 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

[12] `https://github.com/tue-robotics/telegram_ros`
[13] `https://github.com/tue-robotics/conversation_engine`

Platform website[14], to share hardware designs. All our software is available on GitHub[15]. All packages include documentation and tutorials. Tech United and its scientific staff have the capacity to co-develop (15+ people), maintain and assist in resolving questions.

## 7 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 has been a member of the Technical Committee during the period: 2014-2017. 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[16], and interacts on many social media like: Facebook[17], Instagram[18], YouTube[19], Twitter[20] and Flickr[21]. Our robotics videos are often shared on the IEEE video Friday website.

## References

1. Morgan Quigley, Ken Conley, Brian P. Gerkey, Josh Faust, Tully Foote, Jeremy Leibs, Rob Wheeler, and Andrew Y. Ng. ROS: an open-source robot operating system. In *ICRA Workshop on Open Source Software*, 2009.
2. D. Fox. Adapting the sample size in particle filters through kld-sampling. *The International Journal of Robotics Research*, 22(12):985–1003, 2003.
3. D. Fox, W. Burgard, and S. Thrun. The dynamic window approach to collision avoidance. *IEEE Magazine on Robotics & Automation*, 4(1):23–33, 1997.

[14] `http://www.roboticopenplatform.org`
[15] `https://github.com/tue-robotics`
[16] `http://www.techunited.nl`
[17] `https://www.facebook.com/techunited`
[18] `https://www.instagram.com/techunitedeindhoven`
[19] `https://www.youtube.com/user/TechUnited`
[20] `https://www.twitter.com/TechUnited`
[21] `https://www.flickr.com/photos/techunited`

## 8 HSR's Software and External Devices

We use a standard Toyota™ HSR robot. To differentiate our unit, we named it HERO. We wanted to link it's name to our AMIGO and SERGIO domestic service robots.

**Fig. 9.** The Toyota™ HSR Robot, HERO

**HERO's Software Description** An overview of the software used by the Tech United Eindhoven @Home robots can be found in Table 1. All our software is developed open-source at GitHub[22].

**Table 1.** Software overview

| | |
|---|---|
| Operating system | Ubuntu 16.04 LTS Server |
| Middleware | ROS Kinetic [1] |
| Simulation | Gazebo |
| World model | Environment Descriptor (ED), custom<br>`https://github.com/tue-robotics/ed` |
| Localization | Monte Carlo [2] using Environment Descriptor (ED), custom<br>`https://github.com/tue-robotics/ed_localization` |
| SLAM | Gmapping |
| Navigation | CB Base navigation<br>`https://github.com/tue-robotics/cb_base_navigation`<br>Global: custom A* planner<br>Local: modified ROS DWA [3] |
| Arm navigation | MoveIt! |
| Object recognition | image_recognition_tensorflow<br>`https://github.com/tue-robotics/image_recognition/tree/master/image_recognition_openface` |
| People detection | Custom implementation using contour matching<br>`https://github.com/tue-robotics/ed_perception` |
| Face detection & recognition | image_recognition_openface<br>`https://github.com/tue-robotics/image_recognition/tree/master/image_recognition_openface` |
| Speech recognition | Julius Speech Recognition |
| Speech synthesis | Toyota™ Text-to-Speech |
| Task executors | SMACH<br>`https://github.com/tue-robotics/tue_robocup` |

This is our current software implementation, which matches our own robots, AMIGO and SERGIO, which have participated with in the open platform league.

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

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

– Official Standard Laptop

**Cloud Services** *HERO connects the following cloud services:* None