Tech United Eindhoven @Home 2015 Team Description Paper

J.J.M. Lunenburg, S. van den Dries, L.F. Bento Ferreira 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

Abstract. This paper provides an overview of the main developments of the Tech United Eindhoven RoboCup@Home team. The main research effort of the past year has focused on developing a global, volumetric, object-oriented world model. This world model should replace all world representations that are commonly present in an architecture, e.g., for navigation and manipulation. As a result, these modules can benefit from the additional knowledge that is present in the world model. This world modeling approach is expected to i) make modules such as navigation more robust since these can benefit from the additional knowledge that is present in the world model and ii) reduce the amount of hardcoded knowledge that is commonly present.

1 Introduction

Tech United Eindhoven is the RoboCup team of the Eindhoven University of Technology, competing in the Middle Size League and the @Home League. Tech United has been competing in the @Home league since 2011, scoring second places at the 2014 RoboCup German Open and RoboCup 2014 in João Pessoa. Tech United Eindhoven consists of PhD and MSc students and staff members from different departments within the Eindhoven University of Technology.

This Team Description Paper is part of the qualification package for RoboCup 2015 in Hefei, China and describes the current status of the @Home activities of Tech United Eindhoven. First, our newly developed world model will be discussed in Section 2, followed by the use of this world model for perception (Section 3) and navigation (Section 4).

2 World Modeling

To successfully solve a RoboCup@Home challenge, a robot needs a basic understanding of the environment: it needs to know how to get from A to B without colliding, it must be able to detect humans and objects with which it has to interact and it must localize itself with respect to the objects it has to manipulate. In previous years separate world representation methods were used for each of these subtasks:

- *navigation*: down-projection of a 3D Octomap [1] representation
- *object tracking*: multiple hypothesis tracking method [2]
- *localization*: Monte Carlo method on a 2D grid representation [1]

A clear disadvantage of this approach was that all world representation had to be created and maintained separately. Furthermore, the 3D Octomap proved to not be very robust against dynamics: moving obstacles sometimes left a trail of 'clutter voxels' which resulted in a noisy 2D planning representation. Last but not least, it was clear that the separate representations could easily benefit from each other if the information they stored was combined.

To overcome these issues, a new 3D volumetric, object-oriented world model is developed that can be used for all above-mentioned subtasks. This world model, named *ED* (Environment Descriptor), represents the objects in the world as entities with a 3D shape, pose and type. If the object is known beforehand (*e.g.*, the kitchen block in a RoboCup@Home arena), its shape is represented by a detailed 3D mesh. In case the object is unknown, its shape is represented as a floor-aligned extruded polygon. An example can be seen in Fig. 2(b).

The world model is updated by comparing the point cloud obtained from a 3D sensor with the 3D shape of the world model, and applying the changes. If a set of measured points could not be associated with the existing model, a new entity is created. If an existing entity should have been measured, but could not be associated with any sensor data, the entity is removed. This process consists of the following steps (illustrated in Fig. 1):

1. Calculate the sensor pose of the 3D sensor
2. Render the world model:
   - Based on the current world model state and the sensor pose, generate a 'virtual' depth image (as if the world model was observed by the sensor)
3. Extract features:
   (a) Calculate normals of the *sensor point cloud*
   (b) Calculate normals of the *rendered (world model) point cloud*
4. Try to associate each *sensor point* to a *world model point*
   - Uses Euclidean point and normal distance
   - If a *sensor point* cannot be explained (based on a threshold), add it to a *residual point cloud*
5. Cluster the *residual point cloud* into *separate segments*
6. Try to associate the *segments* with *current entities* by calculating the overlap of their shapes
   (a) If segment *can* be associated with entity → *update* the entity shape
   (b) If segment *cannot* be associated → *add* it to the world model
7. Clear world model entities that are in view but could not be associated in step 4) or 6)

The world model obtained using the steps above contains volumetric information about known and unknown objects in the world. This allows the use of this

[1] Used implementation: `http://wiki.ros.org/amcl`

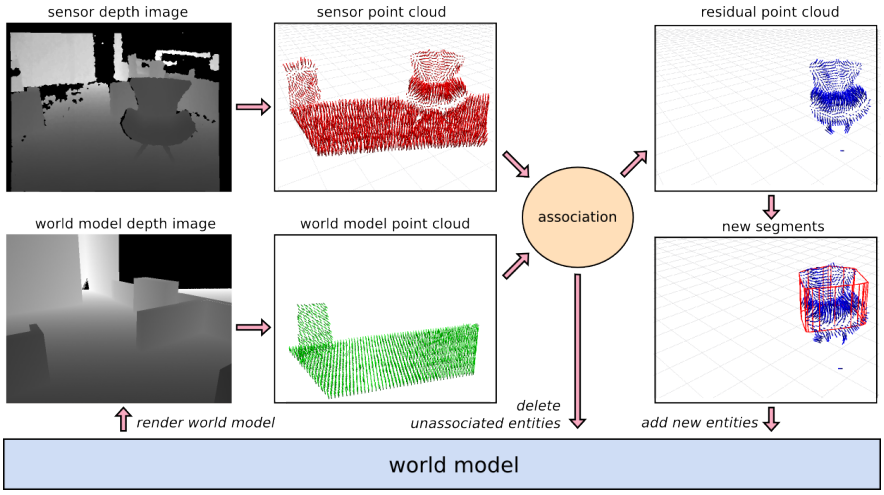


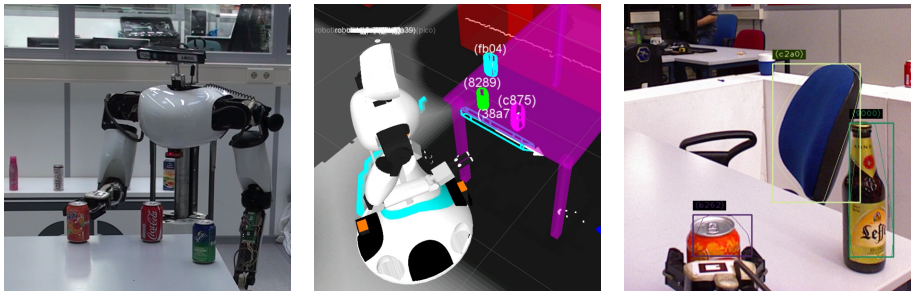
Fig. 1. Overview of the world model update mechanism. From left to right: world model rendering, normal estimation, association and clustering

world model for safe 3D navigation, as is explained in Section 4. Furthermore, due to the association in step (4) and clustering in step (5) unknown objects are automatically segmented. For example, an object situated on top of table will automatically be segmented and labeled as new if the table is correctly modeled. This allows perception routines to focus on relative small, segmented parts of the image. The volumetric information of the objects also enables manipulation tasks. An example of a manipulation task using the world model can be seen in Fig. 2.

In its current status, the world model does not yet contain the features of previous semantic world models as described in [2, 3]. A future research direction would be to include motion models to allow for object tracking as well as integrating an algorithm to do reassociation. The former can be used by navigation to decide, *e.g.*, on which side to pass a moving obstacle while the latter will improve clearing. This demonstrates one of the advantages of a single world model that is used for multiple tasks: the navigation module will directly benefit from improvements in the world model.

3 Perception

As stated before, the world model segments the sensor data into separate entities. This means that each entity is related to a certain part of both the color image and the depth image. These two sources can provide information essential for the classification. The classification process is initiated by assigning a *'perception worker'* to each entity. These workers are independent from each other and run in multiple processing threads, which allows for recognition and labeling of several entities simultaneously in real-time.



(a) Pick up task (b) World model (c) Sensor view

Fig. 2. The robot picks up the object with use of the information available in the newly designed world model

Each perception worker is responsible for calling one or several classification algorithms on the segmented color and depth image belonging to corresponding entity. At the time of writing the following perception algorithms are used:

- *Size Matcher*: matches the dimensions, width and height, of the current entity with the one from previously learned models
- *Objects of Daily Use*: provides a matching between the current entity and previously learned models based on SIFT features [4]
- *Color Matcher*: performs color histogram comparison between the current entity and previously learned models
- *Human Contour Matcher*: applies a Template Matching technique to determine if the entity's shape matches that of a human head and shoulders
- *Face Detector*: searches for faces in the segmented area of a given entity using OpenCV face detection[2]
- *Face Recognition*: if a face is found in an entity, then an attempt to find a corresponding match is performed among previously learned faces based on OpenCV face recognition[3]

Each of these perception algorithms tries to determine the type of the entity and generates hypotheses and according scores. Then, the perception worker collects all hypothesis and fuses them into one final hypothesis. This allows for a robust classification of household objects and people recognition. More perception algorithms can be added to this pipeline as required. The ones presented above allow for an effective recognition and labeling of entities present in most RoboCup challenges.

[2] `http://docs.opencv.org/trunk/doc/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html`

[3] `http://docs.opencv.org/trunk/modules/contrib/doc/facerec/facerec_tutorial.html`

4 Navigation

In our new navigation system, the world model introduced in Section 2 has replace the time-dependent Octomap representation in [5, 8]. Experiments have shown that this representation is indeed more robust against clutter. Furthermore, the local planner that has been used in previous years has been replaced by a modified implementation of the ROS DWA planner. The details of this approach will be published at a later time.

Another benefit of using a volumetric world model is that instead of defining hardcoded waypoints throughout the RoboCup@Home scenario, navigation goals can be derived from the world model. Although the concept of a *goal region* is common in motion planning, see, *e.g.*, [7], navigation goals are still typically defined as waypoints, possibly with a 'goal area radius' and an independent orientation constraint, *i.e.*, the desired orientation with respect to the world frame is independent of the position within the goal region. When taking the task context into account, the shape and size of the goal region $\mathcal{G}$ as well as the desired orientation within that region depend on the task at hand. To put this into practice, the navigation goals are not simply defined as waypoints but as constraints. These can be defined in any frame of reference, *e.g.*, the combination of $x_{\text{obj}}^2 + y_{\text{obj}}^2 > 0.45^2$ and $x_{\text{obj}}^2 + y_{\text{obj}}^2 < 0.60^2$ directs the robot to a radius $0.45 < r < 0.60$ as can be seen in Fig 3. This is a convenient distance from an object to grasp it. The orientation constraint can be defined similarly, which can be used to make the robot face a person or when looking for an object on a table (see Fig. 4).

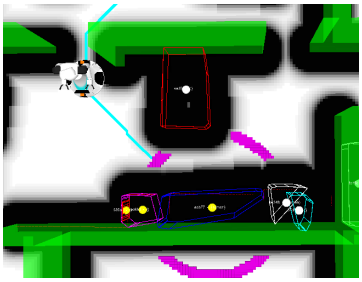


Fig. 3. Visualization of the goal constraint $\mathcal{G}$ to grasp the object.

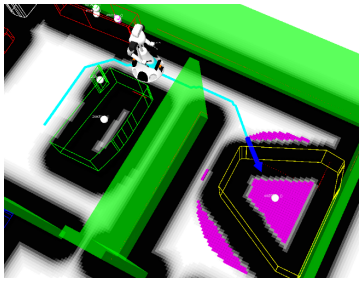


Fig. 4. Visualization of the goal constraint $\mathcal{G}$ to search the table for objects.

The corresponding position and orientation constraints $\mathcal{G}$ are subsequently computed based on the purpose of navigation and are sent to the global planner. The A* planner searches a path to $\mathcal{G}_{\text{lc}} = \{q \in \mathcal{G} | c(q) < c_{\text{min}}\}$, *i.e.*, the subset of the goal region of which the costs $c(q)$ are below threshold c_{min}. If no path can found, the search is repeated towards $\mathcal{G}_{\text{hc}} = \mathcal{G} \setminus \mathcal{G}_{\text{lc}}$. If a path is returned, it will be send to the local planner that will start moving the robot along this path.

5 Conclusions

In this paper, this year's main developments of Tech United Eindhoven have been discussed:

- A new world model has been developed: Environment Descriptor. By combining semantic and volumetric information about the environment, this world model can be used not only for task planning and execution but also for motion planning.
- Perception algorithms are implemented as 'workers' on world model entities which allows for real-time classification of objects.
- One of the benefits of using a volumetric world model is that predefined waypoints have become obsolete. Instead, a goal area can be defined depending on the task and the object at hand, which greatly robustifies navigation.

With these improvements, we hope to improve on last year's performance. We are looking forward to RoboCup 2015 in Huefei!

Robot Hardware Descriptions

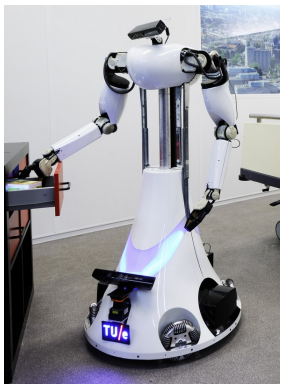


Fig. 5. The AMIGO robot.

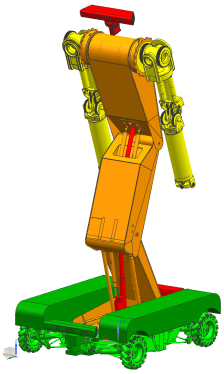


Fig. 6. CAD drawing of SERGIO.

AMIGO (Autonomous Mate for Intelligent Operations, see Fig. 5) has competed in RoboCup@Home since 2011. Its design is based on a Middle Size League soccer robot, equipped with two Philips Experimental Robotic Arms mounted on an extensible upper body. Based on our experiences with AMIGO, SERGIO (Second Edition Robot for Generic Indoor Operations, see Fig. 6) has been developed. The main differences with AMIGO are the use of Mecanum wheels which are compliantly suspended, the torso with two degrees of freedom and the modular setup. The core specifications of AMIGO and SERGIO can be found in

Table 1. More details about the robots can be found on the Robotic Open Platform[4], where all CAD drawings, electrical schemes and CAD files are published.

Table 1. Core specifications of AMIGO and SERGIO

	AMIGO	SERGIO
Name	Autonomous Mate for IntelliGent Operations	Second Edition Robot for Generic Indoor Operations
Base	Fully holonomic omni-wheel platform based on a soccer robot	Fully holonomic Mecanum wheel platform with independent wheel suspension system
Torso	1 vertical DoF using a ball screw	1 nearly vertical DoF using a coupled ankle and knee joint, 1 rotational hip joint
Manipulators	2 7-DoF Philips Experimental Robotic Arms	2 7-DoF custom arms
Neck	Pan-tilt unit using two Dynamixel RX-28 servo actuators	Pan-tilt unit using two Dynamixel RX-64 servo actuators
Head	Kinect for XBox 360	Kinect for XBox 360
External devices	Wireless emergency button	Wireless emergency button
Dimensions	Diameter: 0.75 m, height: ±1.5 m	Base: 0.7 m × 0.6 m, height: ±1.65 m
Weight	±70 kg	±70 kg
Additional sensors	Hokuyo UTM-30LX laser range finder on base and torso	Hokuyo UTM-30LX laser range finder on base and torso (tilting)
Microphone	RØDE Videomic	RØDE Videomic Pro
Batteries	4× Makita 24 V, 3.3 Ah	4× Makita 24 V, 3.3 Ah
Computers	4× AOpen Mini PC with Core-i7 processor and 8 Gb RAM	3× Gigabyte mini ITX board with Core-i7 processor and 16 Gb RAM

Robot Software Description

An overview of the software used by the Tech United Eindhoven @Home robots can be found in Table 2. Our recent software developments can be found on GitHub[5].

[4] `http://www.roboticopenplatform.org/`
[5] `https://github.com/tue-robotics`

Table 2. Software overview of the robots.

Operating system	Ubuntu 12.04 LTS Server
Middleware	ROS Hydro
Low-level software	Orocos Real-Time Toolkit
World model	ED (Environment Descriptor), custom `https://github.com/tue-robotics/ed`
Localization	Monte Carlo using ED, custom `https://github.com/tue-robotics/ed_localization`
SLAM	Gmapping: `http://wiki.ros.org/gmapping`
Navigation	Global: custom A* planner Local: modified ROS DWA
Arm navigation	Custom implementation using Orocos KDL
Object recognition	Combination of size matching (custom), color matching (custom) and Objects-of-Daily-Use Finder `http://wiki.ros.org/objects_of_daily_use_finder`
People detection	Custom implementation using contour matching
Face detection	See Section 3
Face recognition	See Section 3
Speech recognition	Dragonfly + Windows Speech Recognition `http://code.google.com/p/dragonfly/`
Speech synthesis	Philips Text-to-Speech
Task executors	SMACH `http://wiki.ros.org/smach`

References

1. A. Hornung, K.M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: an efficient probabilistic 3d mapping framework based on octrees. *Autonomous Robots*, pages 1–18, 2013.
2. J. Elfring, S. van den Dries, M.J.G. van de Molengraft, and M. Steinbuch. Semantic world modeling using probabilistic multiple hypothesis anchoring. *Robotics and Autonomous Systems*, 61(2):95 – 105, 2013.
3. J. Elfring, R. Van de Molengraft, and M. Steinbuch. Semi-task-dependent and uncertainty-driven world model maintenance. *Autonomous Robots*, 38(1):1–15, 2014.
4. Dejan Pangercic, Vladimir Haltakov, and Michael Beetz. Fast and robust object detection in household environments using vocabulary trees with sift descriptors. In *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, San Francisco, CA, USA, September, 25–30 2011.
5. J.J.M. Lunenburg, S.A.M. Coenen, S. van den Dries, J. Elfring, R.J.M. Janssen, J.H. Sandee, and M.J.G. van de Molengraft. Tech United Eindhoven team description. *RoboCup 2013*, 2013.
6. R. Janssen, E. van Meijl, D. Di Marco, R. Van De Molengraft, and M. Steinbuch. Integrating planning and execution for ros enabled service robots using hierarchical action representations. In *16th Int. Conf. on Advanced Robotics*, pages 1–7, 2013.
7. Jean-Claude Latombe. *Robot motion planning*. Springer, December 1990.
8. S.A.M. Coenen, J.J.M. Lunenburg, M.J.G. van de Molengraft, and M. Steinbuch. A representation method based on the probability of collision for safe robot navigation in domestic environments. In *Proc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2014.