# Tech United Eindhoven Team Description 2013

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

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 hardware and software of the AMIGO robot, which competes in the RoboCup@Home competition on behalf of Tech United Eindhoven. The main changes in hardware are new omni-wheels and EtherCAT I/O boards for the robotic arms. Among the main software improvements compared to last year are 3D navigation, reasoning and perception.

## 1 Introduction

Tech United Eindhoven is the RoboCup team of the Eindhoven University of Technology, competing in the Middle Size League, Humanoid League and @Home League. Tech United has been competing in the @Home league for two years, scoring first place at the 2012 RoboCup Dutch Open and a seventh place at RoboCup 2012. Tech United Eindhoven consists of PhD, MSc and BSc 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 2013 in Eindhoven and describes the current status of the @Home activities of Tech United Eindhoven. First, the hardware and software of the AMIGO robot platform will be introduced, followed by a description of this year's main research improvements in the fields of navigation, reasoning and world modeling and perception.

## 2 The AMIGO Robot

Our robot in the @Home League is AMIGO, which is an acronym for: Autonomous Mate for IntelliGent Operations (see Figures 1 and 2). This human-size robot has a custom made holonomic base platform, an extendable body, two 7 Degree-of-Freedom anthropomorphic arms and a 3D vision system. It is described in more detail in Section 2.1. All CAD drawings and electrical schemes can be found on Robotic Open Platform (`http://www.roboticopenplatform.org/wiki/AMIGO`), while a simulation model can be found on the ROS Wiki (`http://www.ros.org/wiki/Robots/AMIGO`). The software on AMIGO is developed in ROS and Orocos as is elaborated in Section 2.2.

### 2.1 AMIGO Hardware

The base platform of AMIGO has originally been designed for a novel Middle Size League soccer robot [1]. By upscaling this design, much knowledge and experience obtained in robot soccer could be reused in the development of a domestic service robot. The frame design of the AMIGO is based on stiffness and can be compared to the Eiffel Tower: four legs are connected to a central box. These legs and the central box are made out of aluminum and steel sheets, hence resulting in a construction that is light as well as stiff. Furthermore, using sheet metal reduces the amount of milling required for manufacturing, hence it is both easier and less expensive to manufacture. The legs are used to house the motors and the wheels: four omniwheels are used with the axes along the diagonals of the base platform. This year, the original wheels with ten rollers around the circumference have been replaced by wheels with 28 rollers. Additional bearings at the outer side of the robot keep the wheels perpendicular to the surface. As a result of these modifications,

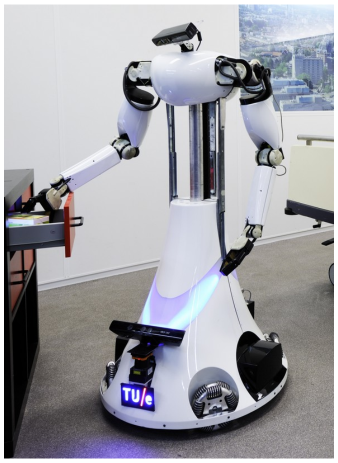
**Fig. 1.** The AMIGO robot.

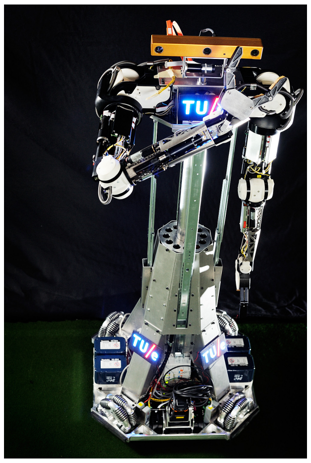
**Fig. 2.** AMIGO without covers.

AMIGO has a much smoother ride. The main advantage an omniwheel platform has is that it is holonomic, *i.e.*, it can move instantaneously in any direction, without having to turn first. To maximize stability and the available space for peripheral equipment, it is chosen to use four omniwheels.

The omniwheels are driven by 24 V Maxon motors. The required current for the motors is provided by Elmo Violin 25/60 amplifiers. The brain of AMIGO consists of four mini-PCs with Core i5 and i7 processors, which are placed on the base platform. The PCs are connected to the sensors and actuators using Beckhoff EtherCAT stacks, containing terminals for digital and analog I/O, encoder modules and RS485 modules. Finally, four 24 V, 3.3 Ah Makita power tool batteries are placed on the base platform to provide the necessary power to the robot.

On top of the base platform, the upper body of the AMIGO is mounted using a ball screw spindle mechanism. This way, the robot is able to adjust the height of the upper body: in its lower position, the AMIGO is able to pick up objects from the floor. In its upper position, the robot has the size of a small adult. This way, it is able to operate most features in a domestic environment, while at the same time having a friendly appearance.

For manipulation, two 7-Degree-of-Freedom (DoF) Philips robotic arms are attached to the upper body. The shoulder, elbow and wrist joint are equipped with a differential drive. Due to this construction, two motors are required to manipulate one DoF. This way, a very compact design of the arm is possible. Each arm can lift up to 1.5 kg when fully stretched. With an own (moving) mass of 3.9 kg this leads to a mass-payload ratio of $2.6 : 1$. Furthermore, force sensors are present to measure the force in the joints, so that force control can be applied. This year, the USB I/O boards have been replaced by newly developed EtherCAT based I/O boards. Each board can control three motors, with connections for encoders, force sensors, absolute position sensors, PWM output as well as spare analog and digital I/O.

For navigation and object recognition, the AMIGO relies on various sensors. On the base of the robot, a Hokuyo laser range finder (LRF) and an XBox 360 Kinect are placed. The LRF provides a 2D image of its surroundings at a 40 Hz rate, which is used for localization and obstacle avoidance. The Kinect provides 3D images to detect obstacles above or underneath the field of view of the LRF (see Section 3.2). A second Kinect camera is placed on top of the robot, mounted on a pan and tilt mechanism so that it can turn left and right as well as look up and down. This camera is used for navigation as well as people and object detection and recognition.

### 2.2 The AMIGO Software

The software on the AMIGO robot is developed within ROS [10] and Orocos [2]. ROS basically has two main features: Firstly, it provides a structured communication layer on top of the host operating system of a computer cluster. This is based on a graph architecture where processing takes place in nodes that may receive, post and multiplex messages regarding sensors, actuators, planning and control. Secondly, ROS has a large suite of user-contributed packages and stacks for various functions, *e.g.*, planning, navigation, perception and simulation. The AMIGO software is partly based on these packages, which have been extensively modified for this robot. To control the hardware, the Orocos Real-Time Toolkit (RTT) is used, which is developed to build highly configurable and interactive component-based real-time control applications.

## 3 Localization and navigation

The navigation system of AMIGO takes in data from various sensors, odometry, and a navigation goal, and outputs velocity commands to the holonomic mobile base. It is based on the ROS navigation stack. The system consists of a number of components that will be described next.

### 3.1 Localization and mapping

AMIGO's navigation can be initialized with or without an a-priori, static map. When initialized without a static map, a 2D map is acquired simultaneously with localization using openSLAM Gmapping software. Gmapping is a highly efficient RaoBlackwellized particle filter used to learn grid maps based on laser range data and odometry information [6, 7]. When using a map, AMIGO localizes itself using an Adaptive Monte Carlo Localization (AMCL) algorithm.

### 3.2 3D environment representation

AMIGO uses a 3D representation in order to navigate collision-free in a domestic environment (see Figure 3). The octree-based volumetric representation OctoMap is used to efficiently acquire a 3D voxel grid at a resolution of 5 cm. Each voxel is marked as occupied, free or unknown. The occupancy of a voxel is modeled probabilistically, which yields robustness against sensor noise and a changing environment. The 3D map is acquired by probabilistic integration of the pointcloud data from the base and top Kinect and the laser range finder. Next, each column of the 3D map is projected down into two dimensions where it is assigned a cost. Columns with an occupied cell are assigned a lethal cost, meaning that they are untraversable. The occupied cells are subsequently inflated up the radius of AMIGO's base and also marked as untraversable. Up to a user-specified inflation radius the cost decreases according to an exponential decay function. The resulting 2D map is used by the global and local planner that are described next.

### 3.3 Global planner and local planner

A minimum cost path from a start position to a goal position on the 2D map is obtained by an A* algorithm. As opposed to Dijkstra's algorithm, which is available as a standard ROS global planner, it uses an Euclidian distance to the goal position as a heuristic. As a result, the search time is significantly reduced with respect to Dijkstra's algorithm.

The local planner is seeded with the plan produced by the global planner, and attempts to follow it as closely as possible while taking into account the kinematics and dynamics of the robot as well as the obstacle information stored in the 2D map. It smooths the global path and generates velocity commands for the mobile base that will safely move AMIGO towards a goal.

### 3.4 Planning Interface

AMIGO uses an interface to deliberate a navigation goal received from a high-level executive to the global and local planner. This interface monitors the progress of execution and has a replanning strategy to avoid detected obstacles.

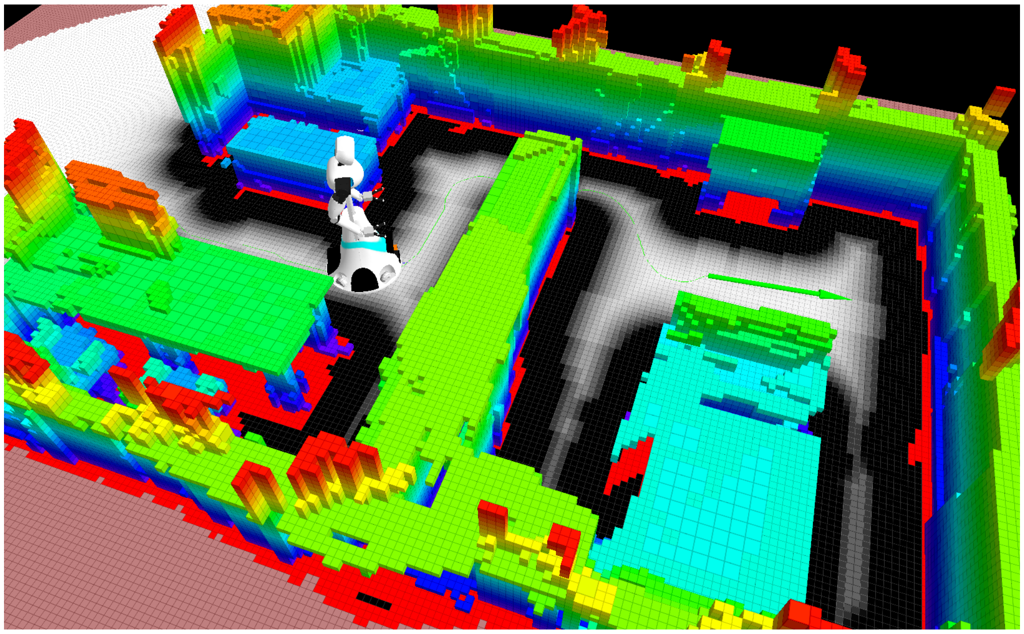

**Fig. 3.** 3D environment representation using an octomap. Each column of the 3D map is projected down to form a 2D costmap for navigation. Light red cells indicate completely unknown space. Bright red cells represent an occupied column in the 3D map and are assigned a lethal cost. Black cells represent inflated obstacle cells also having a lethal cost. From black to white cells (free space) the cost decays towards zero.

When an obstacle is encountered that blocks the global path, the interface allows AMIGO to keep moving up to a predefined distance to the obstacle. Next, AMIGO has a prespecified amount of time to find a new plan. If this time is exceeded the interface will report failure to the high-level executive. If a new plan is found it will depend on the length of this new plan whether it is executed immediately or not: the larger the difference in distance between the original plan and the re-plan, the longer AMIGO will wait before executing it. This ensures that if a path is blocked for a short time (*e.g.*, by a moving obstacle), a re-plan leading to an unnecessarily long plan is not executed immediately. If the new plan is of approximately equal length to the original plan it will execute it immediately to avoid unnecessarily long waiting. Furthermore, the interface automatically updates the goal position within a predefined goal area size. This ensures that if the goal pose is in collision with an obstacle it will be updated to closest by feasible pose. This is especially desirable if an initially free pose close by obstacles becomes infeasible during navigation due to, *e.g.*, sensor noise.

## 4 Reasoning about the World

In the RoboCup-setting, AMIGO operates in a complex and dynamically challenging environment: it is not known beforehand which objects are present, where objects are located, and objects and persons may move over time. Furthermore, different types of objects may need to be treated differently. To deal with such an environment, AMIGO maintains a model of the world in which it stores attribute information, such as position, velocity, color and name of the objects and persons it encounters, and with which it keeps track of those attributes over time. Furthermore, a reasoning component links the stored and tracked objects and properties to the concepts which the executive uses to make decisions with. This enables the executive to query the world model in a semantic way, *e.g.*, by asking: *“Are there any objects on the table which belong in the kitchen?”*, or *“what is the class of the nearest object?”*.

### 4.1 World Modeling

The world model is constructed based on observations received from sensors by fusing the received information into one global, consistent belief state. Due to measurement noise, partial observability and incompleteness and fallibility of perception routines (*e.g.*, sometimes only basic features such as color information and position can be retrieved from a sensor reading), the world model has to be able to cope with uncertainty. Therefore, a probabilistic, multi-hypothesis approach is adopted, based on multi-target tracking literature [4, 11].

Objects are represented in the world model by sets of attributes and object IDs. Each object attribute (typically `position`, `color` and `class`) is represented by a probability density function which explicitly describes the uncertainty of the given object property. For example, the position of a certain `object-3` can be represented by a Gaussian or Mixture of Gaussians, while its class is described by a discrete distribution over all possible classes. If a new observation is associated with a given object, the properties of the object are updated using Bayesian update rules. For example, in the case of a single Gaussian representation, a Kalman update step is performed.

Before object attributes are updated, a data association method based on multi-hypothesis tracking [11] is used to determine which observations originate from which objects. Maintaining multiple plausible measurement/object associations prevents the world model from making incorrect association decisions: instead of picking one possibility, the algorithms postpones its final decision until enough evidence is collected to make an appropriate choice. This approach is particularly useful in situations with dynamic environments, occlusions or partial visibility of a scene. An anchoring [3] based approach is combined with the multiple hypothesis based data association. This enables associating measurements with semantically annotated objects, *e.g.*, measurement $z$ originates from object `object-2` which is associated with `red` and `coke-can`, and it facilitates symbol grounding, *e.g.*, the mapping of the color *green* to a region in Hue Saturation Value (HSV) color space.

The world model is described in more detail in [5]. Furthermore, the world model was recently released under the name *WIRE* (World Informer for Robot Environments) within the ROS framework. The source code and extensive documentation and tutorials can be found at `http://www.ros.org/wiki/wire`.

### 4.2 Reasoning

To provide an expressive, semantic interface to the world model, a reasoning component was developed and implemented in SWI-Prolog[1] which acts as 'glue' between the WIRE world model and executive. While the world model represents and maintains the properties of objects in the world, the reasoning component contains knowledge to interpret the property values and link them to a set of predicates and facts. For example, the reasoner can be used to express that a plate belongs to the kitchen, that X is on top of Y if X's position is above the area that Y occupies, etc. The reasoner advertises a ROS service that can be used to query the world model in such a way, and provides interface implementations for both C++ and Python. This allows all C++ and Python modules to execute queries such as:

```
?- property(X, position, P), type(X, bottle)
```

which binds `X` to the ID of an object which is of type `bottle` (if it exists) and binds `P` to the corresponding location. Similarly:

```
?- property(X, near, amigo), belongs_to(X, kitchen)
```

queries the world model for objects near AMIGO that belong to the kitchen.

The possibility of combining a probabilistic world model with a logical representation language provides AMIGO with an expressive and powerful tool to reason about the world.

[1] http://www.swi-prolog.org/

### 4.3 Executive Integration

AMIGO's executive component, *i.e.*, the module that decides which actions need to be taken to fulfill given task, is implemented as a SMACH state machine. Since the reasoning component described above provides a semantic interface to the world model, the state machine itself does not need to maintain information in a dedicated memory. Instead, the reasoner is queried whenever information about the world is needed. For example, the explore state can ask the reasoner for nearby interesting locations. Which location is interesting depends on the challenge, the rules for which can be coded in a SWI-Prolog knowledge file. The combination of a world model, reasoner and state machine executive realizes a powerful decision-making module: the rules and dynamics of the challenge can be coded in a transparent manner within a state machine, while the world information and world dynamics are taken care of by a probabilistic framework.

### 4.4 Human-Robot Interaction

For many challenges human-robot interaction is a key prerequisite. We have developed a basic speech interpreter that is used during all challenges. As an input it gets a parameterized request from the executive, *e.g.*, give me a name within 10 s. The module asks the necessary questions and asks for validation. For robustness reasons, the speech interpreter loads multiple dictionaries in parallel and only activates the ones associated with the request.

For more advanced human robot communication as needed during, *e.g.*, the General Purpose Service Robot challenge, the module has a notion of the knowledge needed for interpreting more advanced commands:

- object/location categories and their members
- actions and their synonyms, *e.g.*, both the *get* and *pick* action require to *transport* an object from location A to B

The module keeps asking questions until all the available knowledge is received and confirmed and then sends the results to the executive that uses the knowledge for generating a plan.

### 4.5 Plan Generation

For certain challenges, *e.g.*, the General Purpose Service Robot, the order in which actions have to be performed can not be pre-programmed in a static state-machine. In these challenges, the desired action sequence depends on a combination of the given instruction, the state of the robot and the state of the environment. To compose a viable action sequence, therefore a search method is applied to a symbolic representation of the robots capabilities. To parametrise the search, the initial state of the world is derived through the reasoner (Section 4.2). The desired goal state is derived through a refinement dialog controlled by the Human-Robot Interface (Section 4.4). The used search method is based on a simple Means-End Analysis, and encoded as unification rules in SWI-Prolog.

## 5 Perception

Any autonomous robot performing tasks needs some collection of perceptual routines for recognizing and detecting objects and persons. This section explains the Perception Infrastructure (PeIn) which accommodates all perceptual routines used for this purpose.

### 5.1 Perception Infrastructure

The main observation that led to the development of PeIn was that many subparts of perceptual system contained similar or identical pieces of code, *e.g.*, receiving (synchronized) sensor data, mapping images to depth data, communication with the world model. It was decided to develop an infrastructure that was able to perform the general parts and implement modules for the actual detection or recognition of objects and persons. More specifically, we have implemented a base class implementing the basic functionality mentioned above. The base class uses ROS nodelets to avoid needles data copying. We have defined different derived classes for segmentation, learning and recognition of objects or persons. All modules can be switched on or off individually by sending requests to the PeIn supervisor. The supervisor therefore acts as an interface to the executive. All requests are validated and constraints are checked, *e.g.*, are the required models available or are the required parameters provided.

### 5.2 PeIn modules

Various algorithms are implemented in PeIn. For people detection/recognition, existing methods are wrapped in PeIn wrappes:

- Leg detection using a 2D laser [9]
- Face detection
- Eigenfaces based face recognition

In addition, we have implemented OpenCV and PCL based modules for

- Colored blob detection
- Tabletop segmentation
- Viewpoint Feature Histogram matching [12]

Finally, we have implemented an adapted version of Linemod [8], a template based object recognition algorithm that allows for combining multiple modalities. A screenshot can be seen in Figure 4.

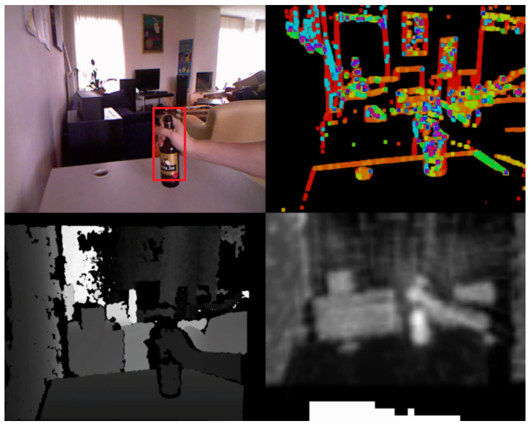

**Fig. 4.** Example of the Linemod object recognition algorithm. The light area on the bottom right figure indicates the beer bottle.

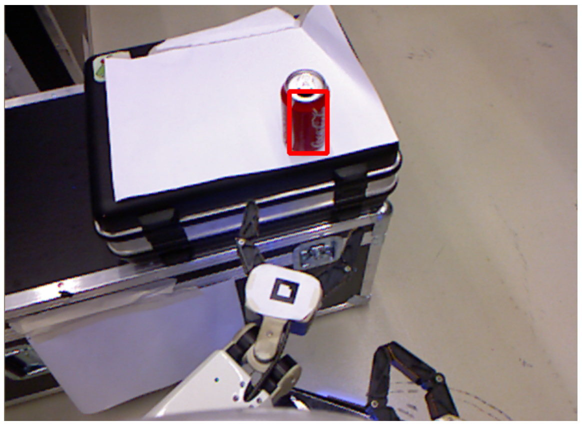

**Fig. 5.** Camera view with the gripper with ar-marker as well as object to grasp in sight.

### 5.3 One sample visual feedback in manipulation

In a normal grasping procedure, the robot first perceives the object, then repositions itself such that it can easily grasp the object and finally performs the actual manipulation. However, last year's perception framework did not work robustly when standing too close to a table. As a result, the robot could not perceive the object after repositioning, introducing localization errors in the manipulation accuracy. Together with the significant backlash in the robotic arms and small inaccuracies in the kinematic model, this often resulted in the robot grasping next to the object it was supposed to grasp.

To improve this, two improvements have been implemented:

- With the new perception algorithms, the object can now also be seen after the robot has repositioned itself, preventing localization inaccuracies from affecting manipulation accuracy;
- After moving the end-effector close to the object, an ar-marker on the gripper is used to determine the position of the object relative to the end-effector, which can be seen as one sample visual feedback. Assuming that the pose-error of the end-effector does not change during the final grasping movement, this is sufficient to reliably grasp objects.

See Figure 5 for a screenshot of the gripper with the ar-marker with the object.

## 6 Conclusions

Compared to the previous year, there are improvements on various topics. The main hardware modifications are omni-wheels with more rollers on their circumference, resulting in a smoother ride, and EtherCAT I/O boards to replace USB interfaces. More improvements can be found in software:

- Navigation of the base platform has been completely re-implemented with a new planning interface, 3D environment representation and both global as well as local planner.
- A reasoning component has been developed that forms an intelligent interface between the world model, containing a global, consistent belief state of the environment, and the task executive which can query the reasoner in a semantic way.
- A new modular perception infrastructure has been developed which contains a collection of perception algorithms for detecting, segmenting and recognizing people and objects. Furthermore, an adapted version of Linemod is used for object recognition.

With these improvements, we hope to improve on last year's performance and are looking forward to RoboCup 2013 in Eindhoven!

## References

1. R. Alaerds, Mechanical design of the next generation Tech United TURTLE, Master's thesis, Eindhoven University of Technology (2010).
2. H. Bruyninckx, Open robot control software: the orocos project, in: Proceedings of the 2001 IEEE International Conference on Robotics & Automation, 2001.
3. S. Coradeschi, A. Saffiotti, An introduction to the anchoring problem, Robotics and Autonomous Systems 43 (2–3) (2003) 85–96.
4. I. J. Cox, S. L. Hingorani, An efficient implementation and evaluation of reid's multiple hypothesis tracking algorithm for visual tracking, IEEE Transaction on Pattern Analysis and Machine Intelligence 18 (2) (1996) 138–150.
5. J. Elfring, S. van den Dries, M. van de Molengraft, M. Steinbuch, Semantic world modeling using probabilistic multiple hypothesis anchoring, Robotics and Autonomous Systems 61 (2) (2013) 95 – 105.
6. G. Grisetti, C. Stachniss, W. Burgard, Improving grid-based slam with rao-blackwellized particle filters by adaptive proposals and selective resampling, in: Proceedings of the 2005 IEEE International Conference on Robotics and Automation (ICRA), 2005.
7. G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with Rao-Blackwellized particle filters, IEEE Transactions on Robotics 23 (1) (2007) 34–46.
8. S. Hinterstoisser, S. Holzer, C. Cagniart, S. Ilic, K. Konolige, N. Navab, V. Lepetit, Multimodal templates for real-time detection of texture-less objects in heavily cluttered scenes, in: Computer Vision (ICCV), 2011 IEEE International Conference on, 2011.
9. O. M. Mozos, R. Kurazume, T. Hasegawa, Multi-layer people detection using 2D range data, in: Proceedings of the ICRA 2009 Workshop: People Detection and Tracking, Kobe, Japan, 2009.
10. M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng, ROS: an open-source robot operating system, in: ICRA Workshop on Open Source Software, 2009.
11. D. B. Reid, An algorithm for tracking multiple targets, IEEE Transactions on Automatic Control AC-24 (6) (1979) 843–854.
12. R. B. Rusu, G. Bradski, R. Thibaux, J. Hsu, Fast 3d recognition and pose using the viewpoint feature histogram, in: Proceedings of the 23rd IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2010.