Robotics and Autonomous Systems 75 (2016) 614–626

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

3D object perception and perceptual learning in the RACE project

Miguel Oliveira[a], Luís Seabra Lopes[a,b,*], Gi Hyun Lim[a], S. Hamidreza Kasaei[a], Ana Maria Tomé[a,b], Aneesh Chauhan[c]

[a] IEETA — Instituto de Engenharia Electrónica e Telemática de Aveiro, Universidade de Aveiro, Portugal
[b] Departamento de Electrónica, Telecomunicações e Informática, Universidade de Aveiro, Portugal
[c] Center of Automation and Robotics, Universidad Politécnica de Madrid, Spain

HIGHLIGHTS

- We describe an object perception and perceptual learning system.
- The system is able to detect, track and recognize tabletop objects.
- The system learns novel object categories in an open-ended fashion.
- The Point Cloud Library is used in nearly all modules of the system.
- The system was developed and used in the European project RACE.

ARTICLE INFO

Article history:
Available online 9 October 2015

Keywords:
3D object perception
Point-Cloud Library
Dual memory systems
Open-ended learning
Interactive learning

ABSTRACT

This paper describes a 3D object perception and perceptual learning system developed for a complex artificial cognitive agent working in a restaurant scenario. This system, developed within the scope of the European project RACE, integrates detection, tracking, learning and recognition of tabletop objects. Interaction capabilities were also developed to enable a human user to take the role of instructor and teach new object categories. Thus, the system learns in an incremental and open-ended way from user-mediated experiences. Based on the analysis of memory requirements for storing both semantic and perceptual data, a dual memory approach, comprising a semantic memory and a perceptual memory, was adopted. The perceptual memory is the central data structure of the described perception and learning system. The goal of this paper is twofold: on one hand, we provide a thorough description of the developed system, starting with motivations, cognitive considerations and architecture design, then providing details on the developed modules, and finally presenting a detailed evaluation of the system; on the other hand, we emphasize the crucial importance of the *Point Cloud Library* (PCL) for developing such system.[1]

1. Introduction

One of the primary challenges of service robotics is the adaptation of robots to new tasks in changing environments, where they interact with non-expert users. The European project RACE (Robustness by Autonomous Competence Enhancement [1,2]), recently closed, assumed that versatility and competence enhancement can be obtained by learning from experiences. The project focused on acquiring and conceptualizing experiences about objects [3], scene layouts [4] and activities [5] as a means to enhance robot competence over time thus achieving robustness. Stimuli for learning can be collected, either autonomously by robots, or when they receive appropriate feedback from users.

The functional components of the RACE architecture are represented by boxes in Fig. 1. Each component may contain one or more modules, which are implemented as nodes (or nodelets) over the *Robot Operating System (ROS)* [6,7]. The *Reasoning and Interpretation* component includes a temporal reasoner, a spatial reasoner and a description logics reasoner. *Perception* contains several modules for symbolic proprioception and exteroception,

* Corresponding author at: IEETA — Instituto de Engenharia Electrónica e Telemática de Aveiro, Universidade de Aveiro, Portugal.
E-mail addresses: mriem@ua.pt (M. Oliveira), lsl@ua.pt (L. Seabra Lopes), lim@ua.pt (G.H. Lim), seyed.hamidreza@ua.pt (S.H. Kasaei), ana@ua.pt (A.M. Tomé), aneesh.chauhan@ua.pt (A. Chauhan).

[1] This paper is a revised and extended version of Oliveira et al. (2014).

http://dx.doi.org/10.1016/j.robot.2015.09.019
0921-8890/© 2015 Elsevier B.V. All rights reserved.

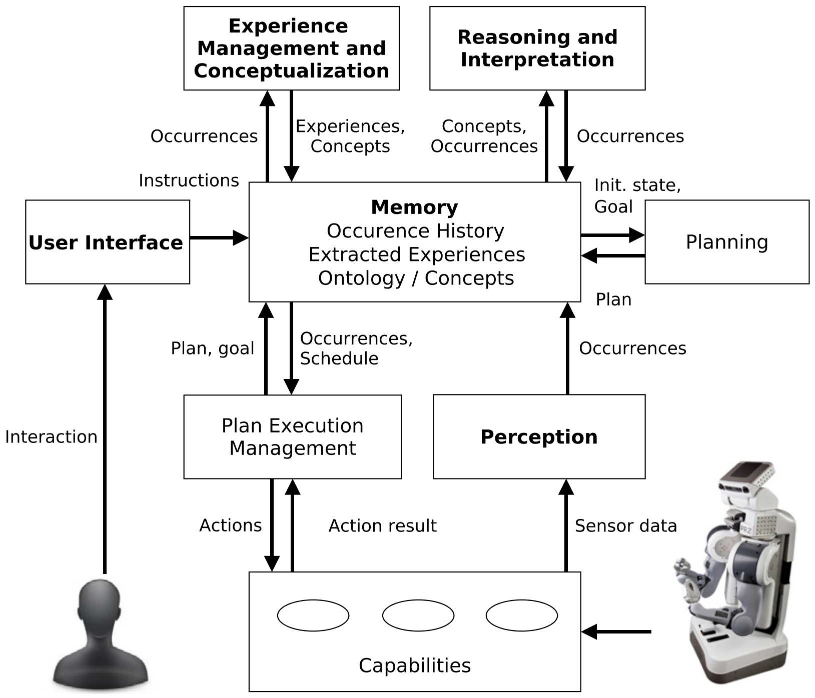

Fig. 1. A high-level overviews of the RACE architecture.

which generate occurrences. The *Experience Management and Conceptualization* component pre-processes occurrences, extracts relevant experiences, uses them to create new concepts and stores these in the *Memory* component. The *User Interface* component receives instructions from the user and relays them to the *Planning* component. Planning is carried out using *SHOP2*, a Hierarchical Task Network planner [8]. The produced plans are executed by the *Plan Execution Management* component.

One of the challenges in this type of projects is to *ground* [9] the semantic representations maintained by the robot, namely the model of the world state and the learned concepts, into the perception and action capabilities of the robot itself [10,11]. At least two types of grounding are involved here. For internal symbols that refer to real-world objects (e.g. "mug23"), the robot must maintain a perception-mediated mapping of symbols to objects. This is often called *anchoring* [12] and relies to some extent on (visual) tracking capabilities at the perception level and on semantic interpretation capabilities at the reasoning level. For category symbols (e.g. "Mug"), the robot must ground their meanings on concrete observations of instances of the categories. A related challenge is how to combine semantic (i.e. symbolic, relational, logic-based) with perceptual (numeric, pattern-based) representations, and how to store different types of representations. After analyzing the requirements of the different components of the RACE architecture, a key decision was made by the project: instead of a single memory system, two independent memory systems would be used, one for semantic information (the *Semantic Memory*) and the other for perceptual information (the *Perceptual Memory*) [2].

Through perceptual learning capabilities, the developed object perception system can be applied to *open-ended environments*. In this case, "open-ended" means that the robot does not know in advance which object categories it will have to learn, which observations will be available, and when they will be available to support this learning.

This kind of perception system must comprise a significant number of software modules, which must be closely coupled in their structure and functionality [13]. Three main design options address the key computational issues involved in processing and storing perception data. First, a lightweight, NoSQL database, is used to implement the perceptual memory. Second, a thread-based approach with zero copy transport of messages is used in implementing the modules. Finally, a multiplexing scheme, for the processing of the different objects in the scene, enables parallelization. This way, the system is capable of real time object detection, tracking and recognition. The developed perception and perceptual learning capabilities target objects in table-top scenes, e.g. in a restaurant environment. These capabilities are fully integrated in the RACE architecture and are running on the PR2 robot used by the project.

This work heavily relies on *Point Cloud Library (PCL)* functionalities, as will be detailed in Section 4. Because the developed perception system is a complex network of processing nodes, the whole paper is organized in such a way that the organization of the system and the module functionalities are well justified and presented in detail. However, since PCL is used in nearly every module of the system, the system could not have been developed easily without PCL. Therefore, this work shows the current importance of PCL in building sophisticated 3D perception systems in robotics and other domains.

The remaining part of this paper is organized as follows. In the next section, related works are discussed. Memory and cognitive architecture issues are discussed in Section 3, leading to the choice of a dual memory approach and to the development of a perceptual memory system. The RACE object perception system and the perceptual learning approach are described in Sections 4 and 5. Profiling and evaluation of the developed system is the topic of Section 6. Finally, in Section 7, the conclusion is presented and future research is discussed.

2. Related work

As robots are expected to increasingly interact and collaborate closely with humans, robotics researchers need to look at human cognition as a source of inspiration. Learning is closely related to memory in human cognition. In the cognitive science literature, the existence of multiple memory systems is widely accepted [14,15]. Biological findings about memory and learning have served as inspiration for the development of computational models and applications. Wood et al. [16] present a thorough review and discussion on memory systems in animals as well as artificial agents, having in mind further developments in artificial intelligence and cognitive science.

In [17], an open-ended object category learning system, based on one-class learning and human–robot interaction, is described. The authors also proposed a teaching protocol for performance evaluation in open-ended learning. In [18], a multi-classifier system with similar goals is described in which a meta-learning component monitors classifier performance, reconfigures classifier combinations and chooses the classifier to be used for prediction. These works are based on 2D images collected in static scenes. Since there is no continuous stream of data being stored, memory requirements are easily satisfied.

Kirstein et al. [19] proposed a lifelong approach for interactive learning of multiple categories from 2D perception, in this case based on vector quantization. This involves selecting the most crucial features from a series of high dimensional feature vectors that almost exclusively belong to each specific category. However, they still follow a standard train-and-test procedure, which is not plausible in open-ended scenarios. Moreover, the authors did not provide details on their memory system or computational architecture. The work is also not integrated in a hybrid perceptual/semantic processing system.

Kruger et al. [20] use the so-called "object–action complexes" (OAC) to bind objects, actions and attributes associated with an agent in a causal way. These OACs are learnable/refinable semantic representations. To ground OACs, an agent requires an object perception and learning system, such as the one we propose below.

Heintz et al. [21] propose a hierarchical framework designed to anchor symbols to continuous streams of sensor data. The approach dynamically constructs and maintains data association hypotheses at multiple levels. A traffic monitoring application is used to illustrate the system. This work proposes a general approach to the anchoring problem, but does not address object perception and learning.

Willow Garage developed the *Object Recognition Kitchen (ORK)*,[2] a 3D object recognition system built on top of the Ecto framework.[3] Ecto organizes computation as a directed acyclic graph, which implies important limitations in the architecture of the perception system. Moreover, in ORK, training/learning and detection/recognition are two separate stages. Such approach is not suitable for developing open-ended learning agents. In contrast, our system allows for concurrent or interleaved learning and recognition, and real-time performance is achieved through nodelets and multiplexing.

Although the Point-Cloud Library (PCL) is increasingly popular, we do not know of other systems building upon PCL to integrate 3D object perception, memory, learning, recognition and interaction.

Aldoma et al. [22] reviewed several state-of-the-art 3D shape descriptors from PCL to develop 3D object recognition and pose estimation capabilities. Throughout the paper, the properties, advantages and disadvantages of different local and global shape descriptors are considered. They also proposed two pipelines for object recognition based on PCL. In the first pipeline, an object view is described by a set of local shape features, which are computed around keypoints. Afterward, each feature is compared against all the features of all models in a database using Euclidean distance. The second pipeline is based on global descriptors, i.e. high-dimensional representations usually calculated for object

2 http://wg-perception.github.io/object_recognition_core.

3 http://plasmodic.github.io/ecto/.

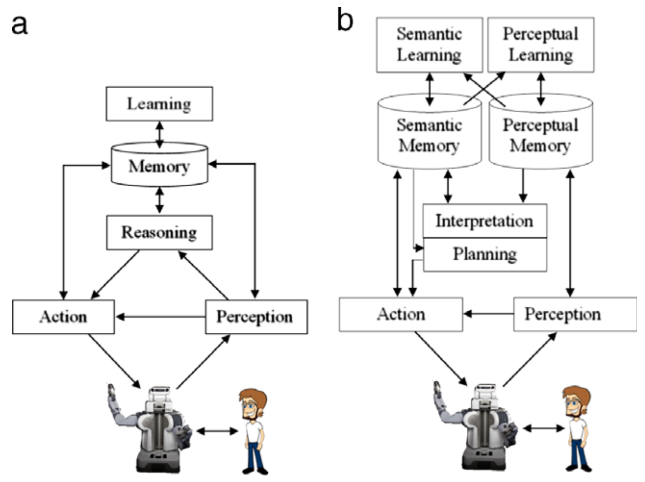

Fig. 2. Abstract cognitive architectures for hybrid reactive–deliberative robots: (a) with a single memory system; (b) with a dual memory system.

candidates (subsets of the scene's point cloud obtained through segmentation).

Clapés et al. [23] proposed an automatic surveillance system for user identification and object recognition. In this work, the position of the RGB-D camera is fixed and the authors employed a background subtraction strategy to segment users and objects in the scene. In the case of object detection, the remaining connected components (those not previously selected as being part of the user) are considered as object candidates. During the recognition stage, *Fast Point Feature Histogram* (FPFH) [24] features are computed for each detected object view and matched against the training models. There is no learning process involved.

3. A dual memory approach

Arguably, robots that interact closely with non-expert users should be [25]: *animate*, meaning that they react appropriately to different events, based on a tight coupling of perception and action; *adaptive*, to cope with changing users, tasks and environments, which requires reasoning and learning capabilities; and *accessible*, that is, they should be easy to command and instruct, and they should also be able to explain their beliefs, motivations and intentions.

In an abstract architecture for intelligent robots, as shown in Fig. 2(a), a *Perception* component processes all momentary information coming from sensors, including sensors that capture the actions and utterances of the user. A *Reasoning* component updates the world model and determines plans to achieve goals. An *Action* component reactively dispatches and monitors the execution of actions, taking into account the current plans and goals. Action processing ranges from low-level control to high-level execution management. Finally, a *Learning* component, which typically runs in the *background*, analyzes the trace of *foreground* activities recorded in a *Memory* component and extracts and conceptualizes possibly interesting *experiences*. The resulting *conceptualizations* are stored back in memory. Each component in such abstract architecture decomposes into a set of software modules, possibly distributed across multiple computers.

The reasoning component manipulates primarily semantic representations of the current world state, goals and plans, that is, representations that are symbolic and relational in nature. In RACE, where case studies were carried out in a restaurant environment, semantic representations describe tables, chairs, table-top objects, guests, the robot, etc., the categories of these objects, the relations between them, and the actions and events that change these relations. The semantic information flowing between reasoning, execution management and memory is typically of small size, and its processing tends to be slow.

One of the challenges in a project like RACE was to combine and store semantic and perceptual representations. Standard *SQL* databases do not cope well neither with semantic data nor with perception data, as both tend to be partially unstructured and/or of variable size. This suggests that modern *NoSQL* databases [26] should be used. Semantic data represents the world in terms of instances, categories and relations between them. A semantic representation of the state of the world can be simply a set of *subject–predicate–object triples*. A special kind of database, the *triplestore*, which shares some features with both *SQL* and graph databases, is especially optimized to store information in the form of a set of triples. Triplestores are clearly one of the database types to take into account when developing memory systems for robots.

An RDF triplestore was in fact the choice for the initial memory component in the RACE architecture [1]. The contents of this memory system, which is used as *blackboard* for all processes, is semantic in nature. It keeps track of the evolution of both the internal state of the robot and the events observed in the environment.

Access to the triplestore is granted via a ROS node that provides database query and write services for all other nodes (an interface node). Information exchange is performed using either publisher/subscriber or client/server mechanisms. ROS communications are a robust framework [13]. However, when the size of the messages is large (e.g., when passing 3D point clouds), the communication between processes is slow. In the case of perception related data, its large size implies large ROS messages to be passed between the database interface node and the other nodes. This is a major constraint, especially considering that, unlike semantic data, perceptual data flows continuously at the sensor output frequency. Using a database interface node creates a bottleneck for accessing the database, since it handles access requests in a first in, first out basis.

Moreover, although triplestores are well suited for storing semantic information, they can hardly be considered suited for storing perception data. In fact, the perception modules will primarily process numeric information organized in structures like vectors and matrices, possibly grouped in sets. For instance the raw perception data about an object, after detection, can be a 3D point cloud, which is a set of points described by their 3D coordinates and possibly RGB information. Based on the point cloud, shape features can be extracted, and the object can be represented by a set of local shape features, where each of them can be a 2D shape histogram. To ensure timely reaction to events in the environment, perception modules run continuously at the frame rate of the used sensors. Although raw data tends to be massive (high-dimensional), the perception modules must run fast, and whatever memory support they use, must also be lightweight.

In the context of RACE, to accommodate semantic and perceptual information in the same database, the only option would be to replace the triplestore with a more generic kind of database. However, we would loose the special features of triplestores, which are optimized for storing triples. In alternative, two different databases can be used, one for semantic information, and the other for perceptual information. The second alternative, which seems more promising, allows to use databases that are well suited for the kinds of data that each will store. In RACE, we converged to the second option.

Fig. 2(b) shows an abstract architecture diagram in which we make explicit the dual memory approach. In what concerns reasoning, we make explicit both interpretation and planning capabilities. One of the most basic interpretation capabilities is anchoring, i.e. connecting object symbols used in the semantic memory to the perception of those objects that is recorded in the perceptual memory. Interpretation also includes computing spatial relations between objects to keep an updated relational model of the scene around the robot. In turn, this scene model can be taken into account for anchoring.

The perceptual memory contains, not only object perception data, but also object category knowledge, in the form of perceptual categories that enable to recognize instances of those categories. These perceptual categories are learned in an open-ended fashion with user mediation [27]. The perceptual learning component primarily uses data from perceptual memory (e.g. shape features of objects) as well as from the semantic memory (e.g. teaching instructions from the user). In RACE, the implementation of the perceptual memory was carried out using a flexible and scalable *NoSQL* database which operates in memory (see the next section for details).

It is worth emphasizing that, although our design choices were guided primarily by engineering criteria, we converged to a solution that is biologically and cognitively plausible. In fact, as previously pointed out, human memory is not a single monolithic system, but rather a combination of several memory subsystems specialized for storing different types of information and supporting different functionalities [14,15]. In particular, our perceptual memory resembles the so-called *Perceptual Representation Memory System*, used in human cognition for enhancing the identification of objects as structured physical–perceptual entities, a process referred to as *perceptual priming* [14]. Another key distinction in cognitive science is between processes that are fast, automatic and unconscious, and processes that are slow, deliberative and conscious [28]. Our dual semantic/perceptual memory approach is also in line with these findings.

4. The RACE object perception system

The work presented in this paper was developed as an extension to the initial RACE architecture [1] (see also Fig. 1). In particular, the work focused on extending the perceptual capabilities of the system. The perception system developed around the perceptual memory supports the anchoring of object symbols into perceived object data as well as the grounding of category symbols into perceptual categories. Since the initial integration of the object perception system, the RACE system included basic capabilities for object symbol anchoring, allowing perceived objects to be represented not only in the perceptual memory, but also in the semantic memory. The object perception system targets tabletop scenes in a restaurant scenario. This system became a salient portion of the full RACE system [2].

4.1. Architecture

The developed perception system is composed of six functional components: *Object Detection*, *Multiplexed Object Perception*, *User Interface*, *Reasoning and Interpretation*, *Memory* and *Conceptualization*. These are represented by the dashed rectangles in Fig. 3. Functional components in Fig. 3 correspond to those highlighted in bold in the high-level RACE architecture (Fig. 1) and to the perception, interpretation, memory and perceptual learning components in Fig. 2(b). In turn, each functional component contains one or more software modules (solid line rectangles in Fig. 3). Arrows signal the exchange of information between software modules. Each software module is organized into a ROS package and will typically correspond to a node or a nodelet[4] at runtime.

[4] http://wiki.ros.org/nodelet.

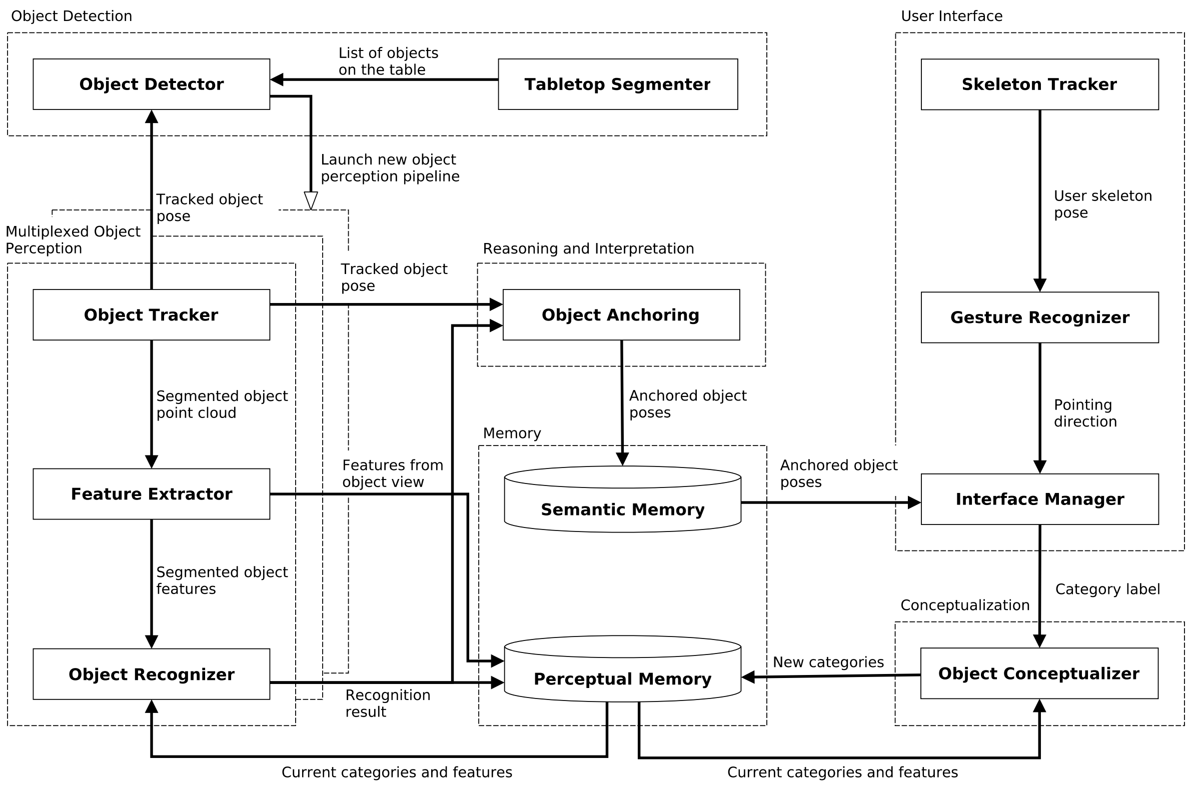

Fig. 3. Architecture of the developed object perception and perceptual learning system.

The implementation of the perceptual memory was carried out using *LevelDB*, a lightweight, flexible and scalable *NoSQL* database developed by *Google*.[5] *LevelDB* is a key–value storage database that provides an ordered mapping from string keys to string values. In addition, *LevelDB* operates in memory and is copied to the file system asynchronously. This significantly improves its access speed.

4.2. The PCL foundation of this work

This work heavily relies on PCL [29][6] functionalities. Table 1 lists the PCL classes used by the object perception and perceptual learning system along with values we typically use for the main configuration parameters. The reason why they are used as well as the modules in which they are used are also given. It can be seen from Table 1 that several PCL functionalities are used by our system. For point cloud size reduction, we use both conditional removal filter as well as voxel grid filters. Table planes are detected using RANSAC, and their boundaries extracted by estimating the 2D convex hull. Points belonging to tabletop objects are extracted using the polygonal prism extraction method. Tabletop objects are segmented using Euclidean cluster extraction. The pose of newly detected tabletop objects is estimated using PCA, which is useful to define well oriented bounding boxes. Objects are tracked over time using the particle filter tracker. Object views are represented by spin image feature descriptors, and object recognition uses an optimized view-to-view distance calculation based on K-d trees.

Since PCL is used in nearly every module of the system, the system could not have been developed easily without PCL. This work shows the current importance of PCL in building sophisticated 3D perception systems in robotics and other domains.

4.3. Object perception

An RGB-D sensor is used for the perception of both the user and the table-top scene. The starting point for the perception of the table-top scene is the *Table-Top Segmenter (TTS)* module, which uses ROS[7] and PCL functionalities to isolate (partial) point clouds of the objects placed on the table (see Table 1) [29,39]. The *Object Detector (OD)* module periodically requests the current list of objects from *TTS*. Then, *OD* will check if any of those objects is already being tracked. To do this, *OD* matches the point clouds of all objects on the table with the estimated bounding boxes of all objects currently being tracked. The percentages of points of the tabletop objects that lie inside the bounding boxes of the tracked objects are computed. A large percentage indicates that the tracked object and the segmented object are the same. Point clouds that cannot be matched with any of the tracked bounding boxes are assumed to represent new objects just added to the scene. *OD* will assign a new identifier (*track_id*) to each newly detected object. Also for each new object, *OD* will launch an object perception pipeline which contains three modules: *Object Tracking*, *Feature Extraction* and *Object Recognition*. Fig. 4 shows a situation where two objects are segmented and tracked, i.e., they have bounding boxes around them.

Object Tracking (OT) is responsible for keeping track of the target object over time while it remains visible. Tracking is an essential base for anchoring. On initialization, *OT* receives the point cloud of the detected object and computes a bounding box for that point cloud, the center of which defines the pose of the object. A particle filter tracking approach from PCL (see Table 1) is then used to predict the next probable pose of the object. In each cycle, *OT* sends out the tracked pose of the object both to *OD* (as mentioned above) and to the Interpretation component. At a lower rate, *OT*

5 https://code.google.com/p/leveldb/.
6 http://pointclouds.org/.
7 http://wiki.ros.org/tabletop_object_detector.

Table 1
PCL functionalities used in the RACE object perception and perceptual learning system. The modules listed in the third column are those represented in Fig. 3.

PCL class [refs.]/parameters	Usage in RACE	RACE modules
ConditionalRemoval	Removing points outside a 3D box[a]	Object Detector Object Tracker
VoxelGrid [30] voxel size = 0.015 m	Downsampling a point cloud[b]	Object Detector Object Tracker
ConvexHull [31,32]	Convex hulls of tables[c]	Tabletop Segmenter
RandomSampleConsensus [33] RANSAC iterations = 200	Table plane detection[d]	Tabletop Segmenter
ExtractPolygonalPrismData minimum z = 0.01 m maximum z = 0.6 m	Tabletop object detection[e]	Object Detector
EuclideanClusterExtraction [34] minimum cluster size = 10 maximum cluster size = 10,000 clustering step = 0.08 m	Object segmentation[f]	Object Detector
PCA [35]	Estimating the orientation of objects[g]	Object Tracker
ParticleFilterTracker [36] max number of particles = 200	Tracking tabletop objects using RGBD[h]	Object Tracking
SpinImageEstimation [37] support length = 0.1 m image width = 8, support angle = 90°	Features for representing object views[i]	Feature Extractor
KdTree [38] $K = 1$	Computation of distance between views[j]	Object Recognizer Object Conceptualizer

[a] http://pointclouds.org/documentation/tutorials/remove_outliers.php.
[b] http://pointclouds.org/documentation/tutorials/voxel_grid.php.
[c] http://www.pointclouds.org/documentation/tutorials/hull_2d.php.
[d] http://pointclouds.org/documentation/tutorials/planar_segmentation.php.
[e] http://docs.pointclouds.org/1.7.1/classpcl_1_1_extract_polygonal_prism_data.html.
[f] http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php.
[g] http://pointclouds.org/documentation/tutorials/normal_estimation.php.
[h] http://pointclouds.org/news/2012/01/17/tracking-3d-objects-with-point-cloud-library/.
[i] http://docs.pointclouds.org/1.7.0/classpcl_1_1_spin_image_estimation.html.
[j] http://pointclouds.org/documentation/tutorials/kdtree_search.php.

sends the point cloud of the object (i.e. containing the points inside the predicted bounding box) to Feature Extraction. As expected, the system is sensitive to the speed with which objects move. If an object moves very fast, tracking is lost, then a new object is detected, and a new object perception pipeline is initiated. Nonetheless, our experiments have shown that the system is able to cope with users picking up the objects and moving them around in natural movements with typical speeds.

The *Feature Extraction (FE)* module computes and stores object representations in the perceptual memory. Objects are represented by sets of local shape features computed in certain keypoints. For efficiency reasons, the number of keypoints should be much smaller than the total number of points. To select keypoints, a voxelized grid approach from PCL is used (see Table 1). We select, for each voxel, the point that is closest to the voxel center [3]. Thus, there will be one keypoint per voxel. The surrounding shape in each keypoint is described by a spin-image [37]. Spin-images are pose invariant, and therefore a suitable local shape descriptor for 3D perception in service robots. They are computed by projecting the 3D surface points of the object to the keypoint's tangent plane We use an implementation of spin-image estimation available from PCL (see Table 1).

In addition to storing object representations in the perceptual memory, *FE* also sends them to *Object Recognition (OR)*. The perceptual categories learned so far and stored in the perceptual memory are used by *OR* to predict the category of the target object. *OR* is a low frequency module, which runs at 1 Hz. Accordingly, *FE* receives object point clouds from *OT* and sends the extracted representations for recognition at the same frequency. Thus, only

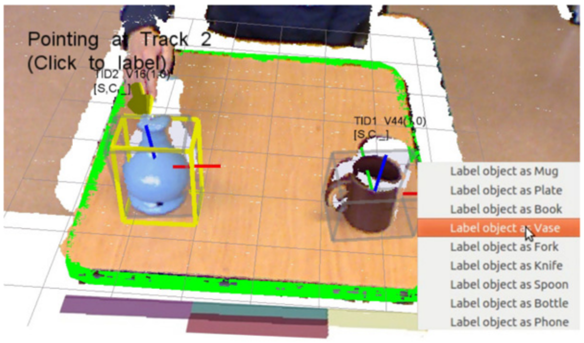

Fig. 4. Visualization of tracking, pointing and labeling.

OT itself uses object point clouds at the frame rate of the sensor (30 Hz).

For better representing an object, it is important to store different views, which is possible when the object is moved (and thus its pose relative to the sensor changes). In contrast, storing all object representations computed by *FE* while the object is static would lead to unnecessary accumulation of highly redundant data. On a different line, it is important to minimize noise effects possibly affecting object views. Thus, to optimize memory usage while keeping potentially relevant and distinctive information, a heuristic is used to select *key views*, that is, object views that should be stored. Whenever the tracking of an object is initialized, or when it becomes static again after being moved, three consecutive object views are stored, provided that the hands of the user are not detected near the object. In case the hands are detected near the object, storing key views is postponed until the hands are withdrawn. *OT* is responsible for marking object views as key views. Then, when *FE* receives a point cloud marked as key view, it will store the respective representation in the perceptual memory.

Object recognition results are also written to the perceptual memory, where the Interpretation component can fetch them to support symbol anchoring. The current implementation is capable of anchoring symbols that refer to objects only while these remain visible. Further work is ongoing to enable anchoring object symbols when the visual tracking is lost.

4.4. User perception

The perceptual memory supports, not only the anchoring of object symbols into perceived object data, but also the grounding of category symbols into perceptual categories. Perceptual categories are acquired with user mediation, that is, the user points to objects and provides their category names. Verbal input is provided through interactive markers in *RVIZ*, a 3D visualization tool for ROS. Point gesture recognition is based on tracking the skeleton of the user. The *Skeleton Tracker (ST)* module is the one available in OpenNI.[8] It tracks the user skeleton pose over time based on RGB-D data. The skeleton pose information is passed to the *Gesture Recognizer (GR)* module, which computes a pointing direction. Currently, the pointing direction is assumed to be the direction of the right forearm (see an example in Fig. 4). The pointing direction is then passed to the Interpretation component. Upon receiving verbal input, the Interpretation component checks if the received pointing direction intersects the bounding box of any of the objects currently on the table according to the world state recorded in the semantic memory. If that is the case, then a teaching instruction is recorded in the semantic memory, stating that the target object was taught to belong to the given category. Teaching instructions trigger perceptual learning to create and/or update object categories.

4.5. Addressing computational issues

In contrast with the reasoning processes supported by the semantic memory, the processes developed around the perceptual memory must run fast to cope with the continuous stream of massive sensor data. As pointed out, one of the reasons for using LevelDB to implement the perceptual memory is the fact that it operates in RAM. There is, however, the limitation that simultaneous access to LevelDB is only possible by threads within the same process. To comply with this constraint while keeping ROS as the framework for the newly developed modules, we use ROS nodelets.[9] Nodelets, which run as threads of a single process, were designed to provide a way of concurrently running different modules with zero copy transport between publisher and subscriber calls (as an example, see [40]). The motivation for ROS nodelets comes from systems with high throughput data flows as is common in perception systems. It is not surprising, therefore, that the developers of PCL and ROS nodelets are the same. In our system, in addition to handling high throughput data flows, nodelets come handy to implement modules that need to simultaneously access the perceptual memory (LevelDB).

Another way of optimizing perception is to parallelize computations. In our system, instead of tracking all objects in a single tracking module, there is a tracker for each object. Similar strategy is used for feature extraction and object recognition. In other words, object perception is designed to be multiplexed. Every time a new object is detected, a corresponding instance of the *object perception pipeline* (see Fig. 3) is launched. Thus there are as many object perception pipelines as the number of currently tracked objects, and each pipeline targets a specific object. Since the modules in an object perception pipeline run as independent nodes/nodelets, they can be distributed to different CPU cores, thus improving the overall computational efficiency of perception. Note that the three modules in the object perception pipeline are traditionally amongst the heaviest in terms of computational requirements. The parallelization is aimed at the hotspot or bottleneck of the computation flow and takes full advantage of modern multicore machines. In fact, experiments with a non-multiplexed version of this architecture show that it cannot run in real-time.

We can easily configure the perception and perceptual learning modules to be launched with different runtime configurations, that is, using ROS nodelets only, ROS nodes only, or a combination of both. By default, the object perception pipelines, the perceptual learning module and the perceptual memory run as a set of nodelets of a single process. When debugging is necessary, we use a configuration where all modules run as nodes. In this configuration, the modules access the perceptual memory (LevelDB) using ROS services provided by a database interface node.

5. Perceptual learning

Although other kinds of perceptual categories could be considered, perceptual learning currently focuses on object categories. We approach object category learning from a long-term perspective and with emphasis on open-endedness, i.e. not assuming a pre-defined set of categories [17,18]. For example, when learning how to serve a coffee [1], if the robot does not know how a mug looks like, it may ask the user to point to one. Such situation provides an opportunity to collect an experience for learning.

Concerning category formation, a purely memory-based learning approach is adopted, in which a category is represented by a set of views of instances of the category. The recording of a teaching instruction in the semantic memory triggers the perceptual learning component. If a new category was taught, the key views of the target object stored by the feature extraction module (see above Section 4.3) are used to initialize the category. If the category is previously known, the key views are added to the existing category representation only if the agent cannot correctly recognize the category of the target object. The current approach differs from our recent previous work [3,10], in the distance measures used for classification, as will be pointed out.

In order to estimate the dissimilarity between a target object view, $\mathbf{t}$, and an object view, $\mathbf{o}$, contained in a category model in the perceptual memory, the following distance function is used:

$$D(\mathbf{t}, \mathbf{o}) = \frac{\sum_{l=1}^{q} \min_k \; d(\mathbf{t}_l, \mathbf{o}_k)}{q}, \tag{1}$$

where $\mathbf{t}_l$, $l = 1, \ldots, q$, are the spin-images of the target object, $\mathbf{o}_k$ represents the spin-images of the model object, and q is the number of target object's spin-images (see Section 4.3 and [37] on spin-images). Since a linear search in high-dimensional spaces has a high computational cost, which is not suitable in the case of an autonomous service robot, a fast approximate nearest neighbor search based on k-d trees [41] from PCL (see Table 1) is used instead of a traditional nearest neighbor search. The next step is to compute the *object–category distance* between the target object,

8 http://wiki.ros.org/openni_tracker.

9 http://wiki.ros.org/nodelet.

t, and a certain category, C, as the average distance of the instances of C to **t**:

$$OCD(\mathbf{t}, C) = \frac{\sum_{\mathbf{u}\in C} D(\mathbf{t}, \mathbf{u})}{n}, \tag{2}$$

where $n = |C|$ is the total number of category instances.

In object recognition, instances are more spread in some categories than in others. Normalizing distances will help to prevent misclassification. Distance normalization is based on the following *intra-category distance*:

$$ICD(C) = \frac{\sum_{\mathbf{u}\in C}\sum_{\mathbf{v}\in C, \mathbf{v}\neq\mathbf{u}} D(\mathbf{u}, \mathbf{v})}{n\,.\,(n-1)}. \tag{3}$$

The normalized distance of target object, **t**, to the category, C, $ND(\mathbf{t}, C)$, is computed as follows:

$$ND(\mathbf{t}, C) = \frac{2 \times OCD(\mathbf{t}, C)}{ICD(C) + \overline{ICD}} \tag{4}$$

where $\overline{ICD}$ is the average of the intra-category distances of all categories, i.e. $\overline{ICD} = \sum_{i=1}^{m} ICD(C_i)/m$, and m is the number of categories.

Finally, the target object is classified based on the minimum normalized distance of the known categories to the object. If, for all categories, the normalized distance is larger than a given *Classification Threshold*, *CT*, then the object is predicted to belong to an *unknown* category[10]:

$$Category(\mathbf{t}) = \begin{cases} unknown, & \text{if } min_C\ ND(\mathbf{t}, C) > CT \\ argmin_C\ ND(\mathbf{t}, C), & otherwise. \end{cases} \tag{5}$$

In open-ended environments where some objects may belong to not yet known categories, recognizing that an object belongs to an unknown category is important. It may prevent the agent from making further decisions based on wrong assumptions. On the other hand, detecting that an object belongs to an unknown category may be used by the agent to trigger some exploration or interaction process leading to the acquisition of a new category.

In our recent previous work [3,10], the object–category distance was measured as the minimum distance between the target object and known instances of a given category. This measure was then normalized by an intra-category distance. If the normalized measure was larger than a given classification threshold, the target object could not belong to that category. The combination of minimum distance with normalization and threshold led to bad decisions in some limit situations. The new formulation here presented solves these problems. In addition, by taking into account the average intra-category distance, the current normalization is also more stable than the previously used normalization method.

6. Profiling and evaluation

This section presents three sets of results. First, based on a session where users manipulate objects on a table and interact with the developed perception and perceptual learning system, we carry out a profiling analysis of the main modules of the system. Second, we present an off-line evaluation of the perceptual learning approach under different configurations of the system. Finally, we report on open-ended learning experiments carried out on a public domain dataset.

Table 2
Sequence of events in the experiment (see video).

Time (s)	Event	Description
25	T1 in	A Mug (T1) is placed on the table
40	T1 is a Mug	T1 is labeled as Mug
60	T2 in	A Vase (T2) is placed on the table
75	T2 is a Vase	T2 is labeled as a Vase
90	T3 in	Another Mug (T3) is placed on the table
135	T3 out	T3 is removed from the table
140	T1 out	T1 is removed from the table
145	T2 out	T2 is removed from the table
165	T4 in	A Plate (T4) is placed on the table
170	T4 is a Plate	T4 is labeled as a Plate
175	T5 in	A Bottle (T5) is placed on the table
190	T5 is a Bottle	T5 is labeled as a Bottle
210	T6 in	A Spoon (T6) is placed on the table

6.1. Profiling

For profiling the modules, two human users interacted with the system in a short session of nearly 4 min (Table 2). All raw data from the RGB-D sensor as well as verbal input from the users during the session was recorded in a rosbag, which was then used to test different configurations of our system. Note that, when the system starts, the set of categories known to the system is empty. There is a video[11] that illustrates the behavior of the main modules of the system, from user/object tracking to learning and recognition.

As discussed in Section 4.5, nodelets can significantly improve efficiency since they support zero copy transport and they enable simultaneous access to LevelDB. Fig. 5 compares the processing time of the object perception modules. The tracker modules (Fig. 5(a) nodes and (d) nodelets) tend to display a stable processing time shortly after their initialization. This is explained by the fact that the size of the input data is more or less stable over time. In this case, nodelets are more efficient when compared to nodes: for example for pipelines 1–3 in the 100–150 time interval, nodes display an average processing time of 45 ms, compared to 25 ms in the case of nodelets. Since the trackers do not access the database, the main factor contributing to the increase in efficiency is the zero copy transport. The messages that are received (sensor point cloud) and sent (partial object point cloud) by the trackers are of large size, which explains why zero copy transport enables such a significant improvement. The feature extraction modules (Fig. 5(b) nodes and (e) nodelets) show a different behavior. These modules periodically compute the spin-image representation from the partial object point cloud. At some points, the point cloud is signaled to belong to a key view, which will trigger the writing of that representation to the perceptual memory.

The curves show these points in time with a rapid increase in processing time. Nodelets also display these peaks, but because access to the database is much faster, the peaks are smaller, as is the average processing time. The object recognition modules (Fig. 5 (c) nodes and (f) nodelets) receive a representation of the current object view from the feature extraction, and compare it against the representations of all known category views. Thus, they are continuously reading the database in the search for an update to the known categories. As a result, the larger the size of the database, the slower the reading of the complete set of categories. However, in the case of nodelets, this deterioration is minor when compared with nodes, since accessing the database is much more efficient.

Fig. 6 shows the memory usage of the system. Notice that at the end of the experiment the memory size would be above 1 MB if all object point clouds extracted by the trackers would be stored

[10] By default, CT = 2.0 is used. See Section 6.2 for an evaluation of alternatives.

[11] The video can be found at http://youtu.be/jLJqY2fKTdI.

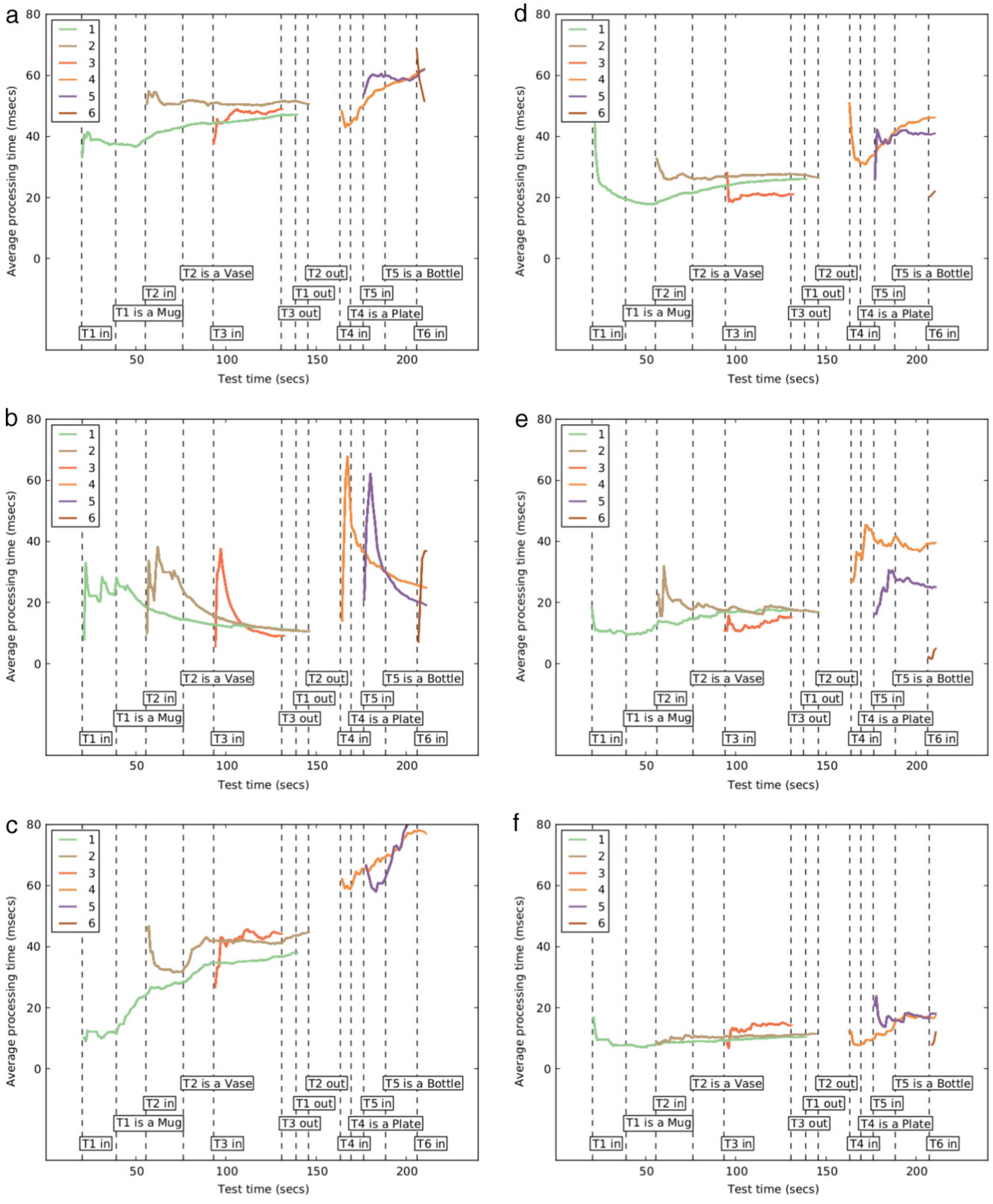

Fig. 5. Processing time in the object perception pipelines in the video sequence, comparing nodes (*left column*) with nodelets (*right column*). Six objects appear in the video, corresponding to pipelines 1 through 6. (a) tracker nodes; (b) feature extraction nodes; (c) object recognition nodes; (d) tracker nodelets; (e) feature extraction nodelets; (f) object recognition nodelets.

(roughly 5 Kb/s). In a continuously running system, this rate of data accumulation would be hard to handle, and would not bring any real benefit. The total size of the point clouds of all the selected key views is much smaller (one order of magnitude in this experiment). The data actually accumulated in memory (shape representations based on spin-images) is even smaller.

Fig. 7 shows the evolution of object recognition performance throughout the experiment. When the first Mug (T1) is placed on the table the system recognizes it as Unknown. After some time, the instructor labels T1 as a Mug and the system starts displaying a precision of 1.0. However, the recall score is under 0.2, because the system classified T1 as Unknown several times before

Table 3
Average object recognition performance (F1 measure) for different parameters: voxel size (VS), image with (IW), support length (SL) and classification threshold (CT).

Parameters	VS		IW		SL		CT		
Values	0.03	0.06	4	8	0.075	0.100	1.00	1.50	2.00
Average F1	0.92	0.89	0.89	0.91	0.90	0.91	0.89	0.91	0.91

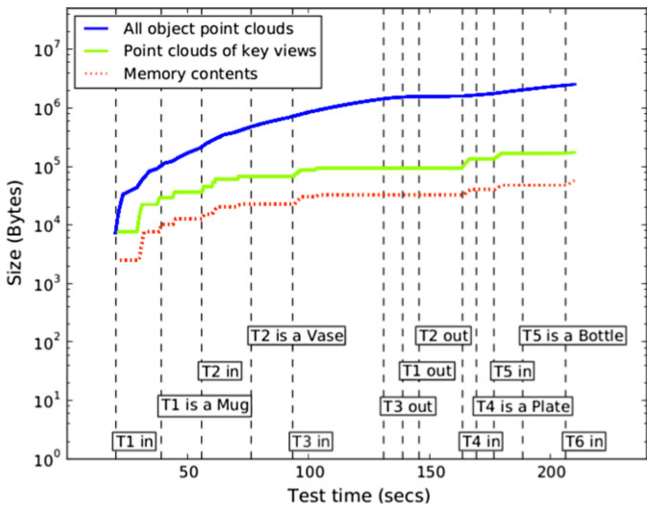

Fig. 6. Perceptual memory usage during the experiment, in logarithmic scale. The blue (upper) curve represents the total size of all point clouds of object views extracted by the trackers. The green (middle) curve represents the total accumulated size of all point clouds of key views. The red (bottom) curve represents the actual perceptual memory content (shape-based representations of key views).

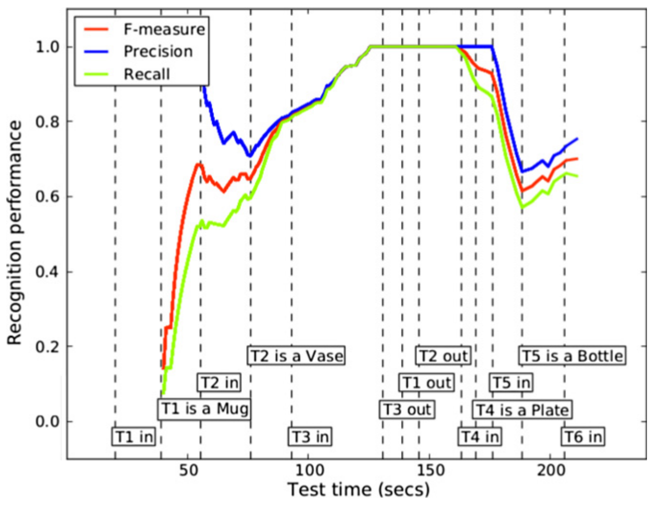

Fig. 7. Object recognition performance (precision, recall and F-measure) during the experiment. Each point in these curves is computed based on the object recognition results in the previous 20 s.

the user labeled the object. After the labeling, the recall starts improving continuously. The instructor then places a Vase (T2) on the table. Because the category Vase has not been taught yet, the performance goes down. After labeling T2 as Vase, performance starts going up again. When a second Mug (T3) enters the scene, the system can correctly recognize it and the scores continue to increase. Then, a Plate (T4) enters the scene, causing recall to drop. Successively, the Plate is taught, a Bottle is placed on the table and then taught, and eventually performance starts going up again. This illustrates the process of acquiring categories in an open-ended fashion with user mediation.

6.2. Off-line evaluation of the perceptual learning approach

More systematic experiments have been performed to evaluate the object category learning and recognition approach. An object dataset has been acquired, which contains 339 views of 10 categories of objects: *bottle*, *bowl*, *flask*, *fork*, *knife*, *mug*, *plate*, *spoon*, *teapot* and *vase*. In addition, there are 31 views of unknown and false objects. The point clouds of the objects were segmented using an offline version of the *Object Detection (OD)* module. Detected objects were manually labeled. The performance of the system was measured using a leave-one-out cross validation scheme.

A total of 24 experiments were performed for different values of four parameters of the system, namely the voxel size (VS) which is related to number of key points extracted from each object view, the image width (IW) and support length (SL) of spin images, and the classification threshold (CT). Results are presented in Table 3. The parameters that obtained the best average F1 score were selected as the default system parameters. They are the following: VS = 0.03, IW = 8, SL = 0.1 and CT = 2. The F-measure of the proposed system with the default parameters was 0.94. Results show that the overall performance of the recognition system is promising. Spin images are capable of collecting distinctive traits of the local surface patches of each object.

6.3. Open-ended learning evaluation

Although there are well established methodologies to evaluate learning systems, e.g., k-fold cross validation, leave-one-out, etc., these approaches follow the classical train-and-test procedure, i.e., two separate stages, training followed by testing. Training is accomplished offline, and once it is complete the testing is performed. These methodologies are not well suited to evaluate open-ended learning systems, because they do not abide to the simultaneous nature of learning and recognition and because the number of categories must be predefined.

A *teaching protocol* for evaluating open-ended learning systems was proposed in [17,42]. The protocol relies on three basic actions to interact with an object recognition system: *teach*, used for teaching a new object category; *ask*, used to ask the system what is the category of an object view; and *correct*, used for providing the system with an additional (labeled) view of an existing category. The idea is to continuously ask the system to recognize previously unseen views of known categories and provide corrections when needed. This way, the system is trained, and at the same time the recognition performance of the system is continuously estimated. A *simulated teacher* was developed to automate experiments following the teaching protocol [3]. To operate, the simulated teacher must be connected to a data-set of object views. In this work, we use the *Washington RGB-D Object Dataset* [43]. This dataset contains images of 300 common household objects from 51 categories. In the experiments presented, the system begins without category knowledge, i.e., it knows zero categories at start, and the training instances are gradually given to the system. Thus, object category models are incrementally built. Although the evaluation protocol is designed to test the system only using views of known categories, the system is designed to identify

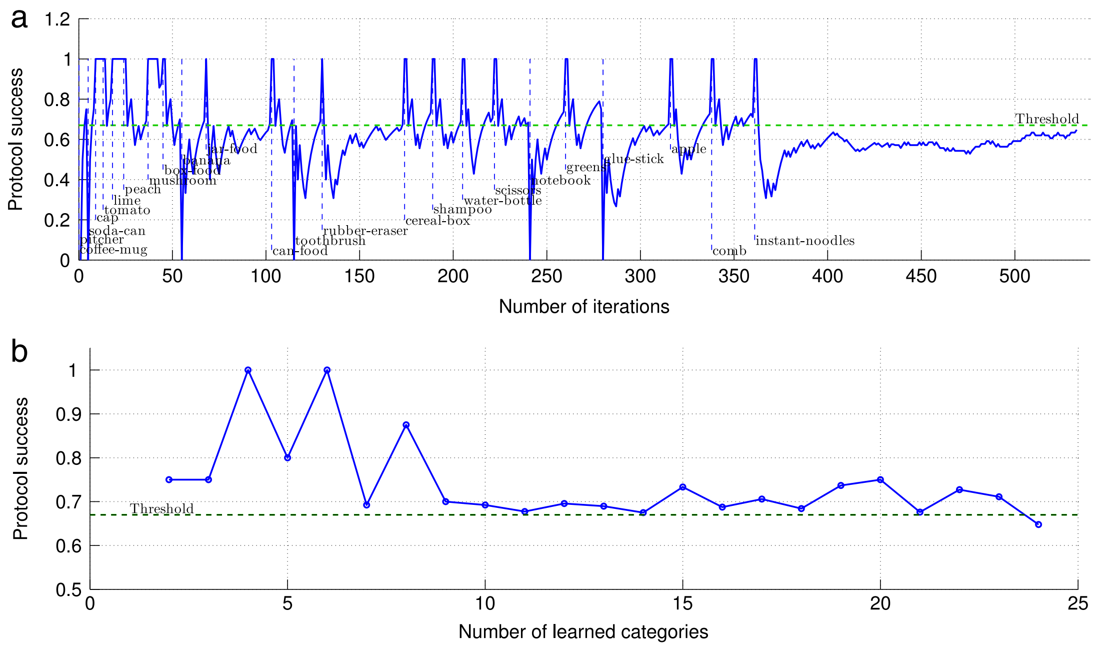

Fig. 8. (a) Simulated teacher experiment no. 3; (b) protocol success versus the number of learned categories, for the same experiment.

Table 4
Summary of simulated teacher experiments[a].

Exp#	#Iterations	#Categories	#Instances	GS	APS
1	706	22	14.55	0.65	0.76
2	217	13	8.69	0.68	0.86
3	533	24	10.46	0.68	0.74
4	699	21	15.57	0.63	0.74
5	747	29	11.31	0.69	0.79
6	711	31	9.83	0.72	0.75
7	1041	36	12.06	0.70	0.77
8	252	13	10.08	0.64	0.87
9	412	19	10.37	0.68	0.81
10	393	20	8.9	0.70	0.84

[a] Exp#, experiment number; #Categories, number of categories learned; #Iterations, number of iterations in the experiment; #Instances, average number of instances per category at the end of the experiment; GS, global success; APS, average protocol success.

unknown categories when the views to be recognized are too far from all models in memory. Therefore, we use F-measure, which combines precision and recall, as the indicator of recognition performance. Protocol success is a local F-measure, computed in a sliding window of size $3n$ (as defined in [42]), where n is the number of categories that have already been introduced. According to the teaching protocol, the system is ready to learn a new object category when the protocol success is above a threshold (0.67 in all experiments presented), and at least one instance of every known category has been tested. Simulated teacher experiments can be used to evaluate the performance of open-ended learning systems using several measures, namely:

- The number of learned categories at the end of an experiment, an indicator of how much the system is capable of learning;
- The number of question/correction iterations required to learn those categories and the average number of stored instances per category, indicators of time and memory resources required for learning;
- Global success, an F-measure computed using all predictions in a complete experiment, and the (local) protocol success defined above, indicators of how well the system learns.

Since the order of introducing new categories may affect the performance of the system, ten experiments were performed using random sequences of introduction of categories. Table 4 summarizes the 10 experiments. Fig. 8(a) shows the evolution of the teaching protocol success in experiment 3. The introduced categories are signaled in the plot. Fig. 8(b) shows the protocol success as a function of the number of learned categories.

Fig. 9(a) shows the global success (i.e. since the beginning of the experiment) as a function of the number of learned categories. In this figure we can see that the global success decreases as more categories are learned. This is expected since the number of categories known by the system makes the classification task more difficult. Finally, Fig. 9(b) shows the number of learned categories as a function of the number of protocol iterations. This gives a measure of how fast the learning occurred in each of the experiments.

7. Conclusions

This paper describes the 3D object perception and learning system developed for the RACE project. The system is designed to detect and track tabletop objects. It is also capable of classifying the tracked objects according to the object categories that are known by the system. Furthermore, this object category knowledge may be enhanced in real time through an RVIZ interface that enables humans to teach additional object categories. The paper describes a dual memory approach in which two databases with different characteristics are used to store semantic and perceptual data. The proposed nodelet software architecture is designed for efficiency purposes, since it enables actual simultaneous access to the perceptual database. Results show that the nodelet based approach is significantly faster when compared to the standard node approach. In addition, we also propose a multiplexed object perception pipeline, in which a set of nodes is launched during execution to handle each newly detected object. This mechanism allocates separate computational resources for each tracked object, which creates a dynamic computation graph in run time, and facilitates the parallelization of processing. As a consequence, the system is capable of simultaneously tracking several objects moving in the scene. The RACE object perception and learning system contains a large number of nodes that process 3D data for different purposes, e.g., segmentation, feature extraction, tracking, etc. Thus, it also

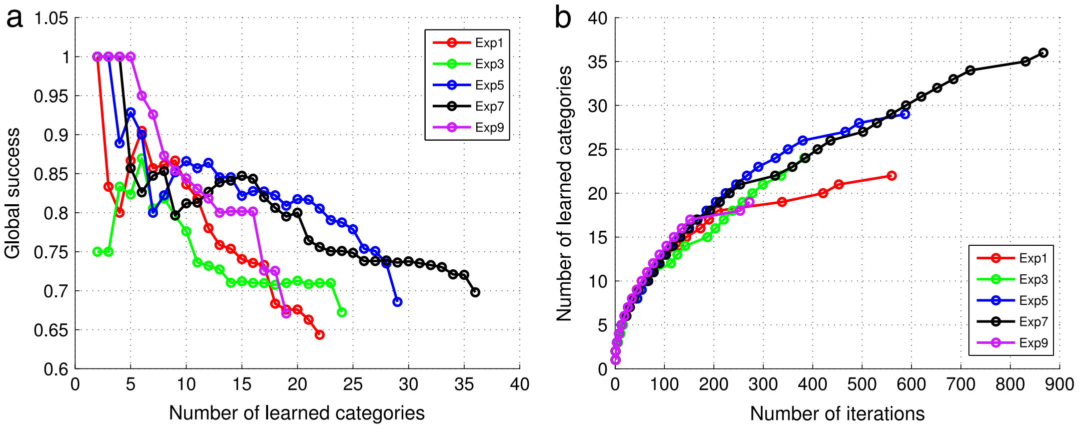

Fig. 9. System performance during simulated user experiments: (a) global success vs. number of learned categories, a measure of how well the system learns; (b) number of learned categories vs. number of question/correction iterations, represents *how fast* the system learned object categories.

serves as a good example of how PCL functionalities can be used to create an efficient and complex 3D perception system.

This paper presented a perceptual memory system designed to enable grounding of object symbols and object category symbols in an open ended fashion. This system is integrated in a dual memory architecture which also includes a semantic memory component. The dual memory approach contributes to the optimization of both the semantic and perceptual components, and is in line with findings in cognitive science regarding the human memory system. The perceptual memory implementation was carried out using a lightweight, NoSQL database which, when combined with a nodelet based infrastructure, allows the simultaneous access of several modules to the storage. The system also supports runtime multiplexing of the object perception pipelines, which leads to the parallelization of the bottlenecks in the data processing. Results show that the perceptual memory combined with a nodelet infrastructure significantly outperforms a node based approach. The presented architecture can seamlessly integrate user-mediated experience acquisition, conceptualization and recognition. The system is open-ended since it can continuously acquire new object categories. Because the system is open-ended and receives a continuous stream of data, ongoing work is using data stream clustering methods to update a dictionary of local features [44].

Acknowledgments

This work was funded by the EC 7th FP theme FP7-ICT-2011-7, Grant agreement No. 287752 (project RACE — Robustness by Autonomous Competence Enhancement). We would like to thank the other RACE project partners for their efforts in the integration and the demonstrations, and especially to the Technical Aspects of Multimodal Systems (TAMS) group, University of Hamburg, for making the PR2 robot available to the project.

References

[1] S. Rockel, B. Neumann, J. Zhang, K.S.R. Dubba, A.G. Cohn, Š. Konečný, M. Mansouri, F. Pecora, A. Saffiotti, M. Günther, S. Stock, J. Hertzberg, A.M. Tomé, A.J. Pinho, L. Seabra Lopes, S. von Riegen, L. Hotz, An ontology-based multi-level robot architecture for learning from experiences, in: Designing Intelligent Robots: Reintegrating AI II, AAAI Spring Symposium on, Stanford, USA, 2013.
[2] J. Hertzberg, J. Zhang, L. Zhang, S. Rockel, B. Neumann, J. Lehmann, K.S.R. Dubba, A.G. Cohn, A. Saffiotti, F. Pecora, M. Mansouri, Š Konečný, M. Günther, S. Stock, L. Seabra Lopes, M. Oliveira, G.H. Lim, H. Kasaei, V. Mokhtari, L. Hotz, W. Bohlken, The RACE project, KI—Künstliche Intelligenz 28 (4) (2014) 297–304.
[3] H. Kasaei, M. Oliveira, G.H. Lim, L. Seabra Lopes, A.M. Tomé, Interactive open-ended learning for 3D object recognition: An approach and experiments, J. Intell. Robot. Syst. (2015) 1–17.
[4] K. Dubba, M. de Oliveira, G. Lim, H. Kasaei, L. Seabra Lopes, A. Tomé, A. Cohn, Grounding language in perception for scene conceptualization in autonomous robots, in: AAAI 2014 Spring Symposium on Qualitative Representations for Robots, 2014.
[5] V. Mokhtari Hassanabad, G.H. Lim, L. Seabra Lopes, A.J. Pinho, Gathering and conceptualizing plan-based robot activity experiences, in: E. Menegatti, N. Michael, K. Berns, H. Yamaguchi (Eds.), Intelligent Autonomous Systems 13: Proceedings of the 13th International Conference IAS-13, in: Advances in Intelligent Systems and Computing, vol. 302, Springer, 2015.
[6] S. Cousins, B. Gerkey, K. Conley, W. Garage, Welcome to ROS topics, IEEE Robot. Autom. Mag. 17 (1) (2010) 12–14.
[7] S. Cousins, B. Gerkey, K. Conley, W. Garage, Sharing software with ROS, IEEE Robot. Autom. Mag. 17 (2) (2010) 12–14.
[8] D.S. Nau, T.-C. Au, O. Ilghami, U. Kuter, J.W. Murdock, D. Wu, F. Yaman, Shop2: An htn planning system, J. Artificial Intelligence Res. 20 (2003) 379–404.
[9] L. Barsalou, Perceptual symbol systems, Behav. Brain Sci. 22 (4) (1999) 577–609.
[10] M. Oliveira, G.H. Lim, L. Seabra Lopes, H. Kasaei, A. Tome, A. Chauhan, A perceptual memory system for grounding semantic representations in intelligent service robots, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE, Chicago, Illinois, 2014.
[11] G.H. Lim, I.H. Suh, H. Suh, Ontology-based unified robot knowledge for service robots in indoor environments, Systems, IEEE Trans. Syst. Man Cybern. 41 (3) (2011) 492–509.
[12] S. Coradeschi, A. Saffiotti, An introduction to the anchoring problem, Robot. Auton. Syst. 43 (2–3) (2003) 85–96. special issue on perceptual anchoring.
[13] S. Zaman, G. Steinbauer, J. Maurer, P. Lepej, S. Uran, An integrated model-based diagnosis and repair architecture for ROS-based robot systems, in: IEEE International Conference on Robotics and Automation, Karlsruhe, Germany, 2013.
[14] E. Tulving, Concepts of human memory, in: L. Squire, G. Lynch, N. Weinberger, J. McGaugh (Eds.), Memory: Organization and Locus of Change, Oxford Univ. Press, 1991, pp. 3–32.
[15] E. Tulving, Episodic memory and autonoesis: Uniquely human? in: J.M.H.S. Terrace (Ed.), The Missing Link in Cognition, Oxford Univ. Press, NewYork, NY, 2005, pp. 4–56.
[16] R. Wood, P. Baxter, T. Belpaeme, A review of long-term memory in natural and synthetic systems, Adapt. Behav. 20 (2011) 81–103.
[17] L. Seabra Lopes, A. Chauhan, How many words can my robot learn?: An approach and experiments with one-class learning, Interact. Stud. 8 (1) (2007) 53–81.
[18] L. Seabra Lopes, A. Chauhan, Open-ended category learning for language acquisition, Connect. Sci. 20 (4) (2008) 277–297.
[19] S. Kirstein, H. Wersing, H.-M. Gross, E. Körner, A life-long learning vector quantization approach for interactive learning of multiple categories, Neural Netw. 28 (2012) 90–105.
[20] N. Krüger, C. Geib, J. Piater, R. Petrick, M. Steedman, F. Wörgötter, A. Ude, T. Asfour, D. Kraft, D. Omrčen, A. Agostini, R. Dillmann, Object-action complexes: Grounded abstractions of sensory-motor processes, Robot. Auton. Syst. 59 (10) (2011) 740–757.
[21] F. Heintz, J. Kvarnstrom, P. Doherty, A stream-based hierarchical anchoring framework, in: Intelligent Robots and Systems, 2009, IROS 2009, IEEE/RSJ International Conference on, 2009, pp. 5254–5260.
[22] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R.B. Rusu, S. Gedikli, M. Vincze, Point cloud library, IEEE Robot. Autom. Mag. 1070 (9932) (2012).
[23] A. Clapés, M. Reyes, S. Escalera, Multi-modal user identification and object recognition surveillance system, Pattern Recognit. Lett. 34 (7) (2013) 799–808.
[24] R.B. Rusu, N. Blodow, M. Beetz, Fast point feature histograms (FPFH) for 3D registration, in: Robotics and Automation, 2009, ICRA'09, IEEE International Conference on, IEEE, 2009, pp. 3212–3217.

[25] L. Seabra Lopes, J. Connell, Semisentient robots: routes to integrated intelligence, IEEE Intell. syst. 16 (5) (2001) 10–14.
[26] S. Sahib, A review of non relational databases, their types, advantages and disadvantages, Int. J. Eng. Technol. 2 (2) (2013).
[27] G.H. Lim, M. Oliveira, V. Mokhtari, S. Hamidreza Kasaei, A. Chauhan, L. Seabra Lopes, A.M. Tomé, Interactive teaching and experience extraction for learning about objects and robot activities, in: Robot and Human Interactive Communication, 2014 RO-MAN: The 23rd IEEE International Symposium on, IEEE, 2014, pp. 153–160.
[28] J.S. Evans, Dual-processing accounts of reasoning, judgment, and social cognition, Ann. Rev. Psychol. 59 (1) (2008) 255–278.
[29] R. Rusu, S. Cousins, 3D is here: Point cloud library (PCL), in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, 2011, pp. 1–4.
[30] D. Cohen-Or, A. Kaufman, Fundamentals of surface voxelization, Graph. Models Image Process. 57 (6) (1995) 453–461.
[31] C.B. Barber, D.P. Dobkin, H. Huhdanpaa, The Quickhull algorithm for convex hulls, ACM Trans. Math. Softw. 22 (4) (1996) 469–483.
[32] N.M. Amato, F.P. Preparata, An NC parallel 3D convex hull algorithm, in: Proceedings of the ninth annual symposium on Computational geometry, SCG'93, 1993, pp. 289–297.
[33] M.A. Fischler, R.C. Bolles, Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography, Commun. ACM 24 (6) (1981) 381–395.
[34] R.B. Rusu, Semantic 3D object maps for everyday manipulation in human living environments (Ph.D. thesis), Computer Science department, Technische Universitaet Muenchen, Germany, 2009, October.
[35] W. Yi, S. Marshall, Principal component analysis in application to object orientation, Geo-spatial Inf. Sci. 3 (3) (2000) 76–78.
[36] Y. Salih, A. Malik, 3D tracking using particle filters, in: Instrumentation and Measurement Technology Conference, I2MTC, 2011 IEEE, 2011, pp. 1–4.
[37] A. Johnson, M. Hebert, Using spin images for efficient object recognition in cluttered 3D scenes, IEEE Trans. Pattern Anal. Mach. Intell. 21 (5) (1999) 433–449.
[38] M. Connor, P. Kumar, Fast construction of k-nearest neighbor graphs for point clouds, IEEE Trans. Vis. Comput. Graphics 16 (4) (2010) 599–608.
[39] R. Rusu, N. Blodow, Z. Marton, M. Beetz, Close-range scene segmentation and reconstruction of 3D point cloud maps for mobile manipulation in domestic environments, in: Intelligent Robots and Systems, 2009, IROS 2009, IEEE/RSJ International Conference on, 2009, pp. 1–6.
[40] M. Munaro, F. Basso, S. Michieletto, E. Pagello, E. Menegatti, A software architecture for RGB-D people tracking based on ROS framework for a mobile robot, in: Frontiers of Intelligent Autonomous Systems, in: Studies in Computational Intelligence, vol. 466, 2013, pp. 53–68.
[41] J.L. Bentley, Multidimensional binary search trees used for associative searching, Commun. ACM 18 (9) (1975) 509–517.
[42] A. Chauhan, L. Seabra Lopes, Using spoken words to guide open-ended category formation, Cogn. Process. 12 (2011) 341–354.
[43] K. Lai, L. Bo, X. Ren, D. Fox, A large-scale hierarchical multi-view RGB-D object dataset, in: Robotics and Automation, ICRA, 2011 IEEE International Conference on, 2011, pp. 1817–1824.
[44] M. Oliveira, L. Seabra Lopes, G.H. Lim, H. Kasaei, A.D. Sappa, A. Tome, A. Chauhan, Concurrent learning of visual codebooks and object categories in open-ended domains, in: Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE (in press).

Miguel Oliveira received the B.Sc., and M.Sc., degrees in Mechanical Engineering from the University of Aveiro, Portugal, in 2004 and 2007, where later in 2013 he obtained the Ph.D. in Mechanical Engineering with specialization in Robotics, on the topic of autonomous driving systems. Currently he is a researcher at the Institute of Electronics and Telematics Engineering of Aveiro, Portugal, where he works on visual object recognition in open-ended domains. His research interests include multimodal sensor fusion, computer vision and robotics.

Luís Seabra Lopes is an Associate Professor of Informatics in the Department of Electronics, Telecommunications and Informatics of the University of Aveiro, Portugal. He received a Ph.D. in Robotics and Integrated Manufacturing from the New University of Lisbon, Portugal, in 1998. He has longstanding interests in robot learning, cognitive robotic architectures, and human–robot interaction.

Gi Hyun Lim received the B.S. degree in Metallurgical Engineering and the M.S. and Ph.D degrees in Electronics and Computer Engineering from Hanyang University, Seoul, Korea, in 1997, 2007 and 2010, respectively. He is currently a Post-doctoral Researcher at the Institute of Electronics and Telematics Engineering of Aveiro, Portugal. His research interests lie in the area of intelligence and learning for robots, including perception and semantics. From 1997 to 1998, he was an Engineer with the Dongbu Electronics, Ltd., Chungcheongbuk-do, Korea, where he was involved in research on semi-conductor factory automation. From 1999 to 2005, he was a Senior Research Engineer at a venture business, where he was involved in research on real-time operating systems and embedded systems.

S. Hamidreza Kasaei obtained B.Sc. (2010) and M.Sc. (2012) in Computer Engineering field of Artificial Intelligence from the University of Isfahan (Iran). Currently he is a Ph.D. student at the University of Minho, Aveiro and Porto (Portugal), where he works on 3D object category learning and recognition in open-ended domains as a research student at the Institute of Electronics and Telematics Engineering of Aveiro. He worked on Middle size soccer robot and Humanoid robot and obtained different ranks in Robocup competition. His main research interest are in computer vision, robotics and multi agent systems.

Ana Maria Tomé is an Associate Professor of Electrical Engineering with the DETI/IEETA of the University of Aveiro. Her research interests include digital and statistical signal processing, independent component analysis, and blind source separation, as well as machine learning applications.

Aneesh Chauhan is a Post-doctoral Researcher in the Computer Vision Group at the Centre of Automatics and Robotics at Universidad Politecnica de Madrid. He holds a Bachelor of Engineering degree in Computer Science and Engineering from Baba Ambedkar Marathwada University, Maharashtra, India, a Master of Science degree in Autonomous Systems from the University of Exeter, UK and Ph.D. in Informatics Engineering from Universidade de Aveiro, Aveiro, Portugal. His research interests include intelligent robotics, language grounding, human–robot interaction as well as the application of computer vision and machine learning approaches for autonomous perception tasks.