The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Perceiving, Learning, and Recognizing 3D Objects: An Approach to Cognitive Service Robots

S. Hamidreza Kasaei,[†] Juil Sock,[*] Luís Seabra Lopes,[†] Ana Maria Tomé,[†] Tae-Kyun Kim[*]
[†] University of Aveiro, Aveiro, Portugal
[*] Imperial College London, London, UK
{seyed.hamidreza, lsl, ana}@ua.pt {ju-il.sock08, tk.kim}@imperial.ac.uk

Abstract

There is growing need for robots that can interact with people in everyday situations. For service robots, it is not reasonable to assume that one can pre-program all object categories. Instead, apart from learning from a batch of labelled training data, robots should continuously update and learn new object categories while working in the environment. This paper proposes a cognitive architecture designed to create a concurrent 3D object category learning and recognition in an interactive and open-ended manner. In particular, this cognitive architecture provides automatic perception capabilities that will allow robots to detect objects in highly crowded scenes and learn new object categories from the set of accumulated experiences in an incremental and open-ended way. Moreover, it supports constructing the full model of an unknown object in an on-line manner and predicting next best view for improving object detection and manipulation performance. We provide extensive experimental results demonstrating system performance in terms of recognition, scalability, next-best-view prediction and real-world robotic applications.

Introduction

In recent years, the role of open-ended learning in robotics has been a topic of considerable interest. The general principle of open-ended learning is that humans learn to recognize object categories ceaselessly over time (Kim et al. 2009). This ability allows them to adapt to new environments, by enhancing their knowledge from the accumulation of experiences and the conceptualization of new object categories. Taking this as inspiration, an autonomous robot, apart from learning from a batch of labelled training data, must process visual information continuously and perform learning and recognition simultaneously. This is important since no matter how extensive the training data used for batch learning, a robot might always be confronted with an unknown object when operating in everyday environments.

This paper reports on the development of an adaptive robotic agent using a cognitive architecture that is integrated with perceptual and motor systems. The contributions presented here are the following: (*i*) a complete architecture for perceiving, learning, and recognizing 3D objects in an interactive and open-ended fashion. (*ii*) proposing a novel unsupervised Next-Best-View (NBV) prediction algorithm to predict the next best camera pose to improve object detection performance. We have tried to make the proposed architecture easy to integrate into other robotic systems. The first contribution follows our previous works on object perception and open-ended perceptual learning (Kasaei et al. 2016b; Oliveira et al. 2015; Kasaei et al. 2015). The present system aims to be more active, handle more complex scenes, and support concurrent object recognition and manipulation in highly crowded scenes. In this work, "open-ended" implies that the set of object categories to be learned is not known in advance. The training instances are extracted from on-line experiences of a robot, and thus become gradually available over time, rather than being completely available at the beginning of the learning process. The cognitive architecture provides the underlying object representations; memories to keep both temporary (working memory) and long-term (perceptual) knowledge; simultaneous access to memory; detecting, tracking, learning and recognizing objects in the environment, NBV prediction and graphical interfaces to enable human users to teach new categories and instruct the robot to perform tasks. The NBV prediction follows our work on multi-view 6D object pose estimation and camera motion planning (Sock et al. 2017).

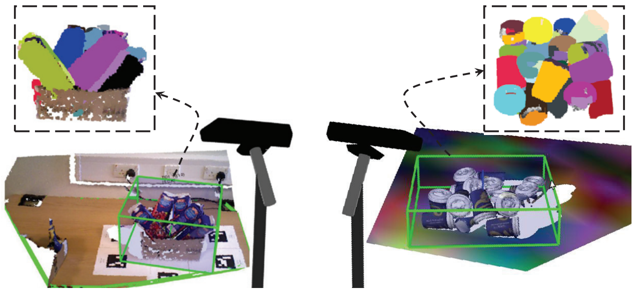
Figure 1: The proposed system tested on two bin-picking datasets that contain multiple objects in highly crowded scenes: (*left*) juice box scenario; (*right*) coffee box scenario.

Related Work

Service robots are expected to be more autonomous and work effectively in human-centric environments. This implies that robots should have special capabilities, such as learning from past experiences and real-time object category learning and recognition. Towards this end, Jain et al. (Jain and Kemp 2010) presented an assistive mobile manipulator, EL-E, that can autonomously pick objects from a flat surface and deliver them to the user. Unlike our approach, the user provides a 3D location of the target object to the robot by pointing at the object with a laser pointer. In another work, a busboy assistive robot has been developed by Srinivasa et al. (Srinivasa 2008). This work is similar to ours in that it integrates perception and motion planning for pick and place operations. However, there are some differences: their vision system is designed for detecting a single object type (mugs), while our perception system not only tracks the pose of different types of objects but also recognizes their categories. Furthermore, because there is a single object type (i. e. mug), they computed the set of grasp points off-line. In our approach, grasping must handle a variety of objects never seen before. Several state of the art cognitive architectures like DIARC (Scheutz et al. 2013) and ACT-R (Anderson, Matessa, and Lebiere 1997) use classical object category learning and recognition approaches (i.e. offline training and online testing are two separate phases), where open-ended object category learning is generally ignored (Leroux and Lebec 2013). Therefore, they work well for specific tasks, where there are limited and predictable sets of objects, and fail at any other assignment. Unlike our approach, the perceptual knowledge of these cognitive architectures are static. Therefore, these architectures are unable to adapt to dynamic environments (Laird et al. 2012).

To cope with these issues, several cognitive robotics groups have started to explore how to learn incrementally from past experiences and human interactions to achieve adaptability. In (Skočaj et al. 2016), a system with similar goals is described. In (Fäulhammer et al. 2017), a perception system is presented that allows a mobile robot to autonomously detect, model, and re-recognize objects in everyday environments. They only considered isolated object scenarios, while our approach can cope with highly crowded scenarios and find a next best view to improve recognition and manipulation performance.

There are several limitations to use deep networks in open-ended domains. Deep Neural Networks (DNN) are incremental in nature but not open-ended, since the inclusion of novel categories enforces a restructuring in the topology of the network. The difference between incremental and open-ended learning is that the set of classes is predefined in incremental learning and the representation of these classes is enhanced (e.g., augmented, improved) over time, whereas in open-ended learning the set of classes is continuously growing. Moreover, DNN usually needs a lot of training data to obtain an acceptable recognition accuracy. Schwarz et.al (Schwarz, Schulz, and Behnke 2015) used DNN for 3D object category learning. They clearly showed that the performance of DNN degrades when the size of the dataset is reduced.

Overall System Architecture

A cognitive robot should process very different types of information in varying time scales. Two different modes of processing, generally labelled as System 1 and System 2, are commonly accepted theories in cognitive psychology

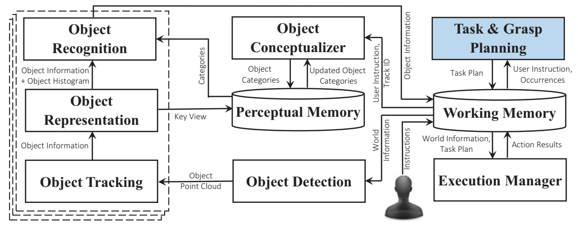

Figure 2: Architecture of the proposed cognitive system.

(Evans 2008). The operations of System 1 (i.e. perception and action) are typically fast, automatic, reactive and intuitive. The operations of System 2 (i.e. semantic) are slow, deliberative and analytic. They are governed by relatively flexible rules and are therefore explicitly represented. This paper propose a perceptual cognitive architecture, with the distinctive characteristics of System 1, to provide a proper coupling between perception and action. The overall system architecture is depicted in Fig. 2. It is a reusable framework and all modules were developed over Robot Operating System (ROS). Each box represents a module that is organized as a ROS package and typically corresponds to a node or nodelet at runtime. Information exchange is performed using standard ROS mechanisms (i.e. either publish/subscribe or server/client). The architecture consists of two memory systems namely *Working Memory* and *Perceptual Memory*. Both *Working* and *Perceptual* memory systems have been implemented as a lightweight NoSQL database namely LevelDB[1]. *Working Memory* is used for temporarily storing and manipulating information and communications of all modules. It also keeps track of the evolution of both the internal state of the robot and the events observed in the environment (i.e. word model). Description of objects and object category knowledge are stored into the *Perceptual Memory*.

For the perception of both the user and the scene, an RGB-D sensor (i.e. Kinect) is used. The starting point for the perception of the scene is *Object Detection*, which uses a hierarchical clustering procedure to isolate (partial) point clouds of the objects. *Object Detection* detects 6D poses of all objects in a scene simultaneously, assigns a *track-id* to each newly detect object, launches a dedicated object processing pipeline for every detected object and pushes the object's point cloud to the pipeline that includes *Object Tracking*, *Object Representation* and *Object Recognition* modules.

Object Tracking predicts the next probable position of an object based on a Kalman filter using geometric information as well as colour data (Oliveira et al. 2014). *Object Tracking* receives the point cloud of the detected object and computes the principal axes, an oriented bounding box for that point cloud. The pose of centre of bounding box of the object is then considered as the pose of the object. *Object Tracking* sends out the tracked pose of the object as well as its point cloud to the *Object Representation* module.

The *Object Representation* module computes and stores object representations in the *Perceptual Memory*. Object representation is one of the most challenging modules in

[1]https://code.google.com/p/leveldb/

cognitive robotics because it must provide reliable information in real-time to enable the robot to physically interact with the objects in its environment. To represent an object, GOOD descriptor is used (Kasaei et al. 2016c; 2016a). GOOD provides a suitable trade-off between descriptiveness, computation time and memory usage, allowing concurrent object recognition and pose estimation, and therefore a desirable object descriptor for 3D perception in service robots. As shown in Fig. 2, in addition to storing description of object in the perceptual memory, *Object Representation* sends the extracted representations for recognition. For recognizing an object, the perceptual categories learned so far are used to predict the category of the target object. Object information including recognition result, size of bounding box and pose of object are written to the working memory, where the *Task & Grasp Planning* module can fetch them to support object manipulation. Figure 1(left) shows a situation where several objects are segmented, tracked and recognized.

User Interaction is essential for supervised experience gathering. For this purpose, a graphical user interface has been developed to teach the robot new object categories or to instruct the robot to perform a task such as constructing *a full model of object* or *clear_table* tasks. Whenever the instructor provides a category label for an object, the *Object Conceptualizer* retrieves the models of the current object categories as well as a description of the labelled object and improves or creates a new object category model. The goal of *Grasp Planning* is to extract a grasp pose (i.e. a gripper pose relative to the object) either from above or from the side of the object, using global characteristics of the object. The *Execution Manager* works based on a Finite-State-Machine (FSM) paradigm. It retrieves the task plan and the world model information from *Working Memory* and computes the next action (i.e. a primitive operator) based on the current context. Then, it dispatches the action to the robot as well as records success or failure in the *Working Memory*. It should be noted that *Task & Grasp Planning* is not in the scope of this paper. We previously showed how to grasp objects (Kasaei et al. 2016b; Shafii, Kasaei, and Seabra Lopes 2016) and how to conceptualize tasks using experience based robot task learning and planning (Mokhtari, Seabra Lopes, and Pinho 2017).

Perceptual Learning

This section presents in detail the object perception modules.

Object Detection

In general, object detection is a challenging task because of ill-definition of the objects (Collet et al. 2014). Since a system of boolean equations can represent any expression or any algorithm, it is particularly well suited for encoding the world and object candidates. Similar to Collet's work (Collet et al. 2014), we used boolean algebra based on the three logical operators, namely *AND* $\wedge$, *OR* $\vee$ and *NOT* $\neg$. Moreover, a set of boolean constraints, C, is defined (see Table 1). Based on these constraints, boolean expressions, ψ, are built to encode object candidates for the *Object Detection* purposes (see equation 1). Due to memory size concerns, a

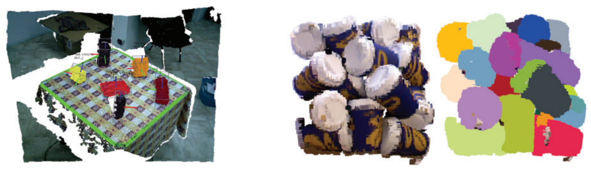

Figure 3: Examples of showing object segmentation results: (*left*) segmentation of isolated objects; (*center*) a pile of objects; (*right*) segmentation of the pile of objects.

representation of an object should only contain distinctive views. A view which is different from the current view may appear after the object is moved (i.e. the pose of the object relative to the sensor changes). An object view is selected as a key view (i.e. $C_{\text{key\_view}}$) whenever the tracking of an object is initialized (C_{track}), or when it becomes static again after being moved.Moreover, $C_{\text{instructor}}$ and C_{robot} are exploited to filter out object candidates which are part of the instructor's body or robot's body. Accordingly, the resulting object candidates are less noisy and include only data corresponding to the objects:

$$\psi = C_{\text{table}} \wedge C_{\text{track}} \wedge \neg \left(C_{\text{instructor}} \vee C_{\text{robot}} \vee C_{\text{edge}}\right). \quad (1)$$

In our current setup, we assume objects are situated on a planar surface, as this is the common pose of objects in domestic environments; but we do not consider any other assumptions about the object appearance except that transparent objects like glasses are not considered. A hierarchical clustering procedure is used to segment scenes using geometric, surface normal data and color. In this work, our segmentation pipeline is composed of two processes (see Fig. 1). The first process computes the regions of interest from the scene. It starts with extracting points which lie directly above a horizontal support plane. This is done by first finding the dominant plane in the point cloud using the RANSAC algorithm (Fischler and Bolles 1981). The scene is then segmented into individual clusters using a Euclidean Cluster Extraction[2] algorithm (see Fig. 3 *left*). In case of large object candidates (e.g. a pile of objects or a messy dinning table), as depicted in Fig. 1, Euclidean clustering algorithm is not enough to appropriately detect object candidates and further processing is required. Therefore, the second process of the segmentation pipeline is used to extract a set of object hypotheses from the results of the first process. A region of the given point cloud is considered as an object candidate whenever points inside the region are continuous in both the orientation of surface normals and the depth values. The depth continuity between every point and its neighbors is computed. If the distance between points is lower than a threshold, then the two points belong to the same region. A color-based region growing segmentation is also applied on large hypotheses. Each object hypothesis (i.e., a cluster of points) will be treated as an object candidate namely c_i, where $i \in \{1, \ldots, K\}$. It should be noted that the number of clusters is not pre-defined and it

[2]http://www.pointclouds.org/documentation/tutorials/cluster_extraction.php

Table 1: List of used constraints with a short description.

Constraints	Description
C_{table}	The interest object candidate is placed on top of a table.
C_{track}	This constraint is used to infer that the segmented object is already being tracked or not.
C_{size}	Reject large object candidate.
$C_{\text{instructor}}$	Reject candidates that belong to the user's body.
C_{robot}	Reject candidates that belong to the robot's body.
C_{edge}	Reject candidates that are near to the edge of the table.
$C_{\text{key\_view}}$	Only key-views are stored into *Perceptual Memory*.

varies for different viewpoints. Fig. 3 illustrates the results of the segmentation process in two different scenarios. A constraint on dimensions of object's bounding-box, C_{size}, has been used to define an object is manipulatable or not. A video of the robot exploring an environment[3] is available at: https://youtu.be/MwX3J6aoAX0

Object Category Learning and Recognition

Object representation is critical to any object recognition system. In the present work, for each object candidate, GOOD description (Kasaei et al. 2016c) is computed. The obtained representation is then dispatched to the *Object Recognition* module and is recorded into the *Perceptual Memory*. Whenever a new object view is added to a category (i.e. supervised learning), the object conceptualizer retrieves the current model of the category as well as representation of the new object view, and creates a new, or updates the existing category. Concerning category formation, an instance-based learning (IBL) approach is adopted, in which a category is represented by a set of views of instances of the category. An advantage of the IBL approach is to facilitate incremental learning in an open-ended fashion. This approach can incrementally update the acquired knowledge (category models) and extend the set of categories over time, which is suitable for real-world scenarios. In order to assess the dissimilarity between the target object view, **t**, and an object view, **o**, a baseline classification mechanism, in the form of a nearest neighbour classifier with a simple thresholding, is used. If, for all categories, the dissimilarity is larger than a given classification threshold, e.g. 0.75, then the object is classified as *unknown*, otherwise, it is classified as the category that has the highest similarity.

Online Object Model Construction

In this section, we propose an approach for robots to autonomously construct models of unknown objects. This capability is necessary for cognitive robots, since it will allow robots to actively investigate their environments and learn about objects in an unsupervised and incremental way. Online construction of full surface models of objects is not only useful for improving object recognition performance by collecting several views, but also can be used for manipulations.

Our approach enables a robot to move around an object and build an increasingly complete 3D model of the object by extracting object points from different perspectives and aligning them together by considering the tracked object pose and robot pose as well as geometrical and visual information. In such a scenario, tracking object is necessary since many objects in everyday environments exhibit rotational symmetries or are lacking in distinctive geometries for matching. As stated in (Krainin et al. 2010), without having pose information, Iterative Closest Point (ICP) based approaches are not able to recover the proper transformations because of the ambiguity in surface matching. To cope with this issue, we develop a Kalman filter that uses depth and visual information for keeping track of robot motion and the target object while it remains visible over time. Afterwards, the extracted object views are unionised together using the ICP approach (Pomerleau et al. 2013) that incorporates both tracking and appearance information. It should be noted, this approach cannot provide information about object parts that are not visible based on the objects position in the environment. Since robot localization is out of the scope of this work, we use noisy ground truth information and showed that this approach can compensate for noise in robot motion and generate proper models of household objects. A video showing the robot exploring an environment for constructing a full model of an *Amita juice box* is available at: https://youtu.be/CuBS2L2q5NU

Unsupervised Next-Best-View Prediction

The ability to predict the Next-Best-View (NBV) pose is important for mobile robots performing tasks in everyday environments. In active scenarios, whenever the robot fails to detect or manipulate objects from the current view point, it is able to predict the next best view position, go there and captures a new scene to improve the knowledge of the environment. This may increase the object detection and manipulation performance (see Fig.4). Towards this end, we proposed an entropy based NBV prediction algorithm by rendering the scene using the current object hypotheses.

In the previous steps, the robot captures a point cloud of the scene and computes a list of hypothesis containing both objects' 6D pose and recognized label. The inputs to the Next-Best-View (NBV) prediction module are: the constructed 3D models of the objects; the point cloud of the scene; a set of 6D object hypotheses; $\mathbf{P} = \{\mathbf{h}_1, \ldots, \mathbf{h}_n\}$; and the possible viewing pose, $\mathbf{V}$ where $\mathbf{V} = \{\mathbf{v}_1, \ldots, \mathbf{v}_m\}$ is a finite list of possible viewing pose representing the camera rotation and translation in 3D space. The given scene is first segmented and the obtained clusters are then used to compute viewpoint

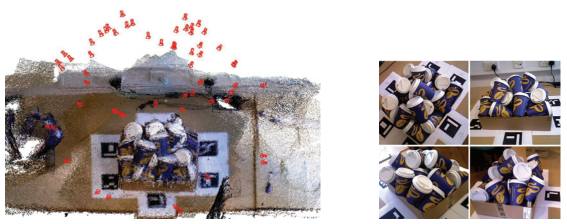

Figure 4: (*left*) Accumulated view and camera pose of different view of a pile of coffee cup scenario; (*right*) images from four different views.

[3]The ROS bag file used in this video was created by the University of Osnabrueck.

entropy for the given scene. There are various methods for computing the viewpoint entropy. In general, the number of visible voxel or points is used as an indicator of the area for entropy computation. This measure is not good enough since it only considers the coverage objective. Therefore, we propose a new formulation for viewpoint entropy calculation that takes into account both the coverage (i.e. the number of visible points) and saliency (i.e. observing a large portion of an object which can potentially reduce the pose estimation and recognition uncertainty) objectives. The viewpoint entropy of a given scene is computed as follows:

$$H = -\sum_{i=1}^{K} \frac{A_i}{S} \log \frac{A_i}{S}, \tag{2}$$

where, K is the number of clusters, A_i is the area of the i^{th} cluster and S is the total area of the given scene. Before actually moving the camera, we aim to predict the NBV from the camera pose list, $\mathbf{V}$. For this purpose, first, we have to predict what can be observed from each pose in $\mathbf{V}$ by taking a "*virtual point cloud*". Toward this goal, based on the given set of 6D objects hypothesis, the full model of objects are first added to the current scene (see Fig.5 *left*). Afterwards, for each possible camera poses, a virtual point cloud is rendered based on depth buffering and orthogonal projection methods (see Fig.5 *center* and *right*). Then, the viewpoint entropy is calculated for each rendered view as before.

In general, choosing the view with the minimum view-entropy as the next camera position has two problems. Firstly, in real system, it costs system to move the camera too far at a time. Secondly, view entropy estimation becomes less reliable if the rendering view is far from the current position, since the view entropy calculation is based on the rendered virtual point cloud. To alleviate this issue, we apply Gaussian weights to the view entropy value calculated for each view candidate:

$$H^w_{\mathbf{v}_i} = w_{\mathbf{v}_i} H_{\mathbf{v}_i} : \; w_{\mathbf{v}_i} = \frac{1}{\sigma\sqrt{2\pi}} e^{-||\mathbf{v}_i - \mathbf{v}_c||^2 / 2\sigma^2}, \tag{3}$$

where σ is a smoothness parameter which restrict the movement of the camera, $w_{\mathbf{v}i}$ is the weight applied to view entropy for $\mathbf{v}_i$, $\mathbf{v}_c$ is the current camera pose, $H_{\mathbf{v}_i}$ is the view entropy of the view $\mathbf{v}_i$ and $H^w_{\mathbf{v}_i}$ is the weighted view entropy of the view $\mathbf{v}_i$. However, minimum $H^w_{\mathbf{v}_i}$ can be use to determine the next camera pose, there is a risk of the camera moving only locally. Although a set of viewpoints which are close to each other may have good attributes, obtaining a sequence of similar viewpoints would not help to detect new objects which are visible from different viewpoints. To encourage exploratory behaviour, the following equation is introduced where viewpoints with higher entropy have a higher chance of being chosen (Sock et al. 2017):

$$p(\mathbf{v}^{\text{Next}} = \mathbf{v}_i) = H^w_{\mathbf{v}_i} \,/ \sum_{n=1}^{m} H^w_{\mathbf{v}_n}. \tag{4}$$

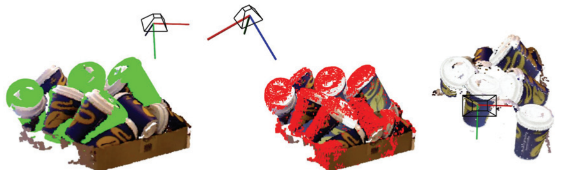
Figure 5: Rendering virtual point cloud for unsupervised NBV prediction: (*left*) the full model of detected objects are added to the scene (i.e. corresponding points are highlighted by green color); (*center*) the visible points from the virtual camera pose are highlighted by red color; (*right*) the rendered virtual point cloud. The reference frame represents the camera pose of the acquired view.

Experimental Results

Three types of experiments were performed to evaluate the proposed approach. In all results, *number of bins* parameter of GOOD descriptor has been set to 15 bins.

Open-Ended Evaluation

We used a teaching protocol designed for experimental evaluation in open-ended learning (Chauhan and Seabra Lopes 2011). The idea is to emulate the interactions of a recognition system with the surrounding environment over long periods of time. The teacher interacts with the learning agent using three basic actions: ***teach***, used for introducing a new object category; ***ask***, used to ask the system what is the category of a given object view; and ***correct***, used for providing corrective feedback in case of misclassification. The idea is that, for each newly taught category, the simulated teacher repeatedly picks unseen object views of the currently known categories from a dataset and presents them to the system. It progressively estimates the recognition performance of the system and, in case this performance exceeds a given threshold, introduces an additional object category. This way, the system is trained, at the same time the recognition accuracy of the system is continuously estimated. The simulated teacher must be connected to an object dataset. In this work, the simulated teacher was connected to the Washington RGB-D Object Dataset (Lai et al. 2011). The performance of an open-ended learning system is not limited to the object recognition accuracy. Therefore, when an experiment is carried out, learning performance is evaluated using three distinct measures, including: (*i*) the number of learned categories at the end of an experiment (TLC), an indicator of ***How much does it learn?***;

Table 2: Summary of experiments.

EXP#	#QCI	#TLC	#AIC	GCA (%)	APA (%)
1	1669	37	18.35	0.65	0.74
2	2014	40	20.50	0.65	0.71
3	1759	43	16.91	0.66	0.72
4	1952	43	18.67	0.66	0.73
5	1592	37	17.35	0.67	0.72
6	1366	37	15.38	0.66	0.72
7	1547	35	17.97	0.66	0.72
8	2087	41	20.39	0.66	0.72
9	1066	34	13.94	0.65	0.74
10	1540	45	13.36	0.70	0.77
Mean±STD	**1659±311**	**39.20±3.73**	**17.28±2.44**	**0.66±0.01**	**0.73±0.02**

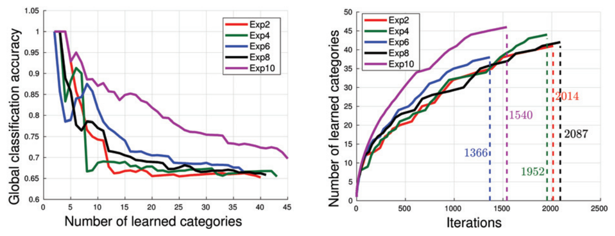

Figure 6: System performance during open-ended evaluations: (*left*) global classification accuracy versus number of learned categories;(*right*) number of learned categories versus number of question/correction iterations.

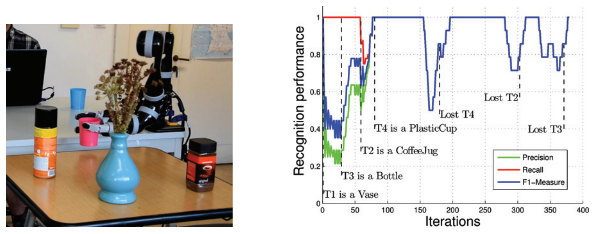

Figure 7: System performance during the first *clear_table* demonstration; (*left*) JACO arm manipulates a *plasticCup*; (*right*) Object recognition performance. Each point in these curves is computed based on the object recognition results in the previous 15 iterations.

(*ii*) The number of question / correction iterations (QCI) required to learn those categories and the average number of stored instances per category (AIC), indicators of ***How fast does it learn?*** (Fig.6 (*right*)); (*iii*) Global classification accuracy (GCA), an accuracy computed using all predictions in a complete experiment, and the average protocol accuracy (APA), indicators of ***How well does it learn?*** (Fig.6 (*left*)).

Since the order of the categories introduced may have an affect on the performance of the system, ten experiments were carried out in which categories were introduced in random sequences. Results are reported in Table 2. By comparing all experiments, it is visible that in the tenth experiment, the system learned more categories than other experiments. Results showed that both evaluation measures (GCA and APA) for this experiment are higher than in all other experiments. In the case of experiment 8, the number of iterations required to learn 41 object categories was greater than other experiments.

The left column of the Fig.6 shows the global classification accuracy obtained by the proposed approach as a function of the number of learned categories. One important observation is that accuracy decreases in all experiments as more categories are introduced. This is expected since a higher number of categories known by the system tends to make the classification task more difficult. The right column of the Fig.6 illustrates how fast the learning occurred in each of the experiments. It shows the number of question / correction iterations required to learn a certain number of categories.

Our approach learned faster than Schwarz et. al (Schwarz, Schulz, and Behnke 2015) approach; i.e. our approach requires much fewer examples than Schwarzs work. Furthermore, we achieved accuracy around 70% by storing less than 20 instances per categories (see Table 2), while Schwarz et.al used more than 1000 training instances per category (see Fig.8 in (Schwarz, Schulz, and Behnke 2015)). In addition, they clearly showed the performance of DNN degrades when the size of dataset is reduced.

System Demonstration

To show the functionalities of the system, three real demonstrations were performed. In the first two demonstrations, a JACO robotic arm manufactured by KINOVA has been used (Fig.7). During the session, a user presents objects to the system and provides the respective category labels. The user then instructs the robot to perform a *clear_table(.)* task. (i.e. puts the table back into a clear state). To achieve this task, the robot must be able to detect, learn and recognize different objects and transport all objects except standard table items (e.g. *Vase*, etc.) to the predefined areas. While there are active objects on the table, the robot retrieves the world model information from the *Working Memory* including label and position of all active objects. The robot then selects the nearest object to the arms base and clear it from the table. Figure 7 shows the evolution of object recognition performance throughout the first *clear_table* demonstration. First, the system recognizes all table-top objects as *Unknown*. After some time, the instructor labels T1 as a *Vase* and the system starts displaying a recall of 1.0. However, the precision starts to decrease, because the category *Bottle*, *CoffeeJug* and *PlasticCup* have not been taught yet, the performance goes down. After the labelling objects, the precision starts improving continuously. As it is shown in the Fig. 7 (*right*), whenever the robot grasps an object (i.e. iterations 155, 280, 332), the shape of the object is partially changed, misclassification is happened and the performance goes down. The grasped object is then transported to the placing area and the tracking of the object is lost (i.e. iterations 181, 302, 375). Afterwards, the performance starts going up again. A video of the second *clear_table* demonstration is available at: https://youtu.be/LZtI-s95uTk

In the third demonstration, two human users interact with the system. Initially, the system only had prior knowledge about the *Mug* and *Dish* categories, learned from batch data (i.e. set of observations with ground truth labels), and there is no information about other categories (i.e. *Vase*, *Bottle*, *Spoon*). Throughout this session, the system must be able to recognize instances of learned categories and incrementally learn new object categories. A video of this demonstration is available at: https://youtu.be/eP0lwqW55Iw

These demonstrations show that the developed system is capable of detecting new objects, tracking and recognizing as well as manipulating objects in various positions.

Evaluation using scene datasets

Object Recognition Performance: Imperial College Dataset (Doumanoglou et al. 2016) is related to domestic environments, where everyday objects are placed on a kitchen table. It consists of variety of different scenes with a set of

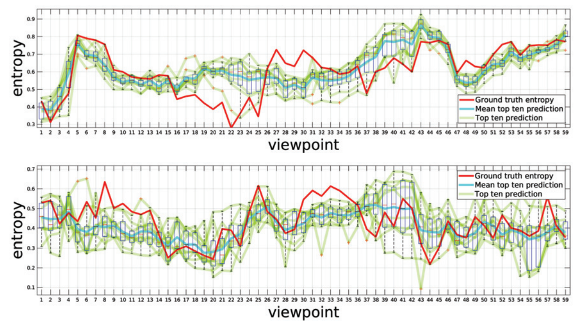

Figure 8: Viewpoint entropy for the bin-picking dataset: (*top*) coffee-cup and (*bottom*) juice-box scenarios.

table top objects. This dataset contains 6 different sets of table-top scenes from two different heights which has 353 scenes in total (see Fig.9 *left*). This is a especially suitable dataset to evaluate the system since the object dataset was collected under various clutter conditions and distances. The objects were extracted from the scenes by running the proposed object perception (see Fig.3 *left*). All detected objects were manually labelled by the authors. To examine the performance of the proposed approach, a 10-fold cross-validation has been used. The F1-score of the object recognition system was 0.92 for the extracted objects.

Next-Best-View Prediction: We test the proposed NBV prediction on the bin-picking dataset (Doumanoglou et al. 2016), which is one of few datasets that contains multiple objects in a highly crowded scene (Fig. 4). The coffee-cup scenario contains 59 different views of the scene with 15 cups in a pile. The juice box scenario contains 5 juice boxes and also has 59 views. Ground truth view point entropy (i.e. highlighted by the red lines in Fig.8) is calculated based on the proposed viewpoint entropy and ground truth objects positions provided by the dataset. In this evaluation, an object pose estimator based on sparse auto-encoder (Doumanoglou et al. 2016) is first used to generate multiple object hypotheses. The proposed method for rendering *virtual point cloud* is then used and an entropy for the rendered view is calculated.

Top-ten views, in terms of predicting view point entropy, are selected based on the square error to the ground truth. Boxplot for the selected views in both scenarios are depicted in Fig.8, which displays the range of variation. The NBV algorithm works well in both scenarios since the standard deviation (SD) of the view entropy is small (i.e. SD for the coffee-cup was 0.105 and for the juice-box was 0.067) and the mean is near to the ground truth in both scenarios. The mean squared error (MSE) was 5.22 and 3.95 for the coffee-cup and the juice-box scenarios respectively. Note, coffee cup scenario is much more complex as it has more objects and many of them are occluded in different views. In contrast, objects in juice box scenario are visible in most of the views.

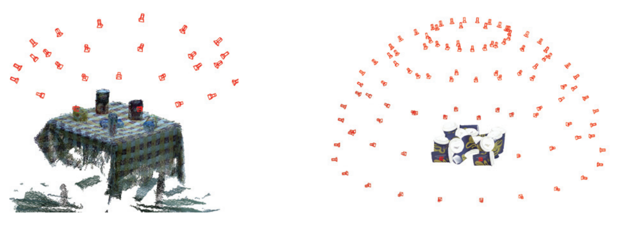

Figure 9: Accumulated view and camera pose: (*left*) Imperial College dataset; (*right*) generated synthetic dataset.

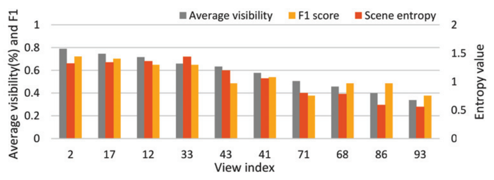

Figure 10: Graph showing the correlation between visibility, viewpoint entropy and detection performance.

Correlation between visibility and viewpoint entropy: To verify the correlation between the object detection performance and the proposed viewpoint entropy, a synthetic dataset is built for the following reasons: (*i*) more dense and even sampling of camera viewpoint can be obtained; (*ii*) perfect knowledge on object ground truth, camera pose and calibration parameters are known. Towards this end, 20 object models are randomly thrown into a virtual box using MuJoCo physics engine(Todorov, Erez, and Tassa 2012). As depicted in Fig.9 (*right*), RGBD scenes are rendered at 100 evenly sampled viewpoints around upper hemisphere. For more details on dataset see (Sock et al. 2017). The dataset is publicly available at: https://goo.gl/BSr2mU

For each object, ratio of visible pixel to the total number of pixel if the object were not occluded is calculated and these values of every objects in a scene are averaged to quantify the average visibility score for each viewpoint. Detector (Doumanoglou et al. 2016) is used to obtain the F1-score for each viewpoint and the proposed view entropy is also used to calculate viewpoint entropy. Results are plotted in Fig. 10. The view indices are ordered in descending average visibility score and the graph shows the F1 score and viewpoint entropy decreases along with the visibility of the viewpoint. The viewpoint entropy and F1 score are positively correlated with the correlation coefficient of 0.6644 for the dataset.

Conclusion

In this paper, we presented a cognitive architecture designed to enhance a proper perception for service robots. In particular, an interactive open-ended learning approach for perceiving and and recognizing 3D object categories has been presented, which enables robots to adapt to different environments. We also introduced view entropy, which can be used to predict the NBV in an environment where robot movement is costly and the scene is complex. Results showed that the proposed system supports classical learning from a batch of train labelled data and open-ended learning from on-line experiences of robots. Moreover, we have tried to make the proposed architecture easy to integrate on the other robotic systems. In the continuation of this work, we would like to investigate the possibility of overcoming the mentioned limitations to use DNN in open-ended domains.

Acknowledgments

This work was funded by National Funds through FCT project PEst-OE/EEI/UI0127/2017 and FCT scholarship SFRH/BD/94183/2013.

References

Anderson, J. R.; Matessa, M.; and Lebiere, C. 1997. Act-r: A theory of higher level cognition and its relation to visual attention. *Human-Computer Interaction* 12(4):439–462.

Chauhan, A., and Seabra Lopes, L. 2011. Using spoken words to guide open-ended category formation. *Cognitive processing* 12(4):341–354.

Collet, A.; Xiong, B.; Gurau, C.; Hebert, M.; and Srinivasa, S. S. 2014. Herbdisc: Towards lifelong robotic object discovery. *The International Journal of Robotics Research*.

Doumanoglou, A.; Kouskouridas, R.; Malassiotis, S.; and Kim, T.-K. 2016. Recovering 6d object pose and predicting next-best-view in the crowd. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3583–3592.

Evans, J. S. B. 2008. Dual-processing accounts of reasoning, judgment, and social cognition. *Annu. Rev. Psychol.* 59:255–278.

Fäulhammer, T.; Ambruş, R.; Burbridge, C.; Zillich, M.; and Folkesson, J. 2017. Autonomous learning of object models on a mobile robot. *IEEE Robotics and Automation Letters* 2(1):26–33.

Fischler, M. A., and Bolles, R. C. 1981. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24(6).

Jain, A., and Kemp, C. C. 2010. El-e: an assistive mobile manipulator that autonomously fetches objects from flat surfaces. *Autonomous Robots* 28(1):45–64.

Kasaei, S. H.; Oliveira, M.; Lim, G. H.; Seabra Lopes, L.; and Tomé, A. M. 2015. Interactive open-ended learning for 3D object recognition: An approach and experiments. *Journal of Intelligent & Robotic Systems* 80(3):537–553.

Kasaei, S. H.; Lopes, L. S.; Tomé, A. M.; and Oliveira, M. 2016a. An orthographic descriptor for 3d object learning and recognition. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 4158–4163. IEEE.

Kasaei, S. H.; Shafii, N.; Lopes, L. S.; and Tomé, A. M. 2016b. Object learning and grasping capabilities for robotic home assistants. In *Robot World Cup*, 279–293. Springer.

Kasaei, S. H.; Tomé, A. M.; Lopes, L. S.; and Oliveira, M. 2016c. Good: A global orthographic object descriptor for 3d object recognition and manipulation. *Pattern Recognition Letters* 83:312–320.

Kim, J. G.; Biederman, I.; Lescroart, M. D.; and Hayworth, K. J. 2009. Adaptation to objects in the lateral occipital complex (loc): shape or semantics? *Vision research* 49(18):2297–2305.

Krainin, M.; Henry, P.; Ren, X.; and Fox, D. 2010. Manipulator and object tracking for in hand model acquisition. In *Proceedings, IEEE International Conference on Robots and Automation*.

Lai, K.; Bo, L.; Ren, X.; and Fox, D. 2011. A large-scale hierarchical multi-view rgb-d object dataset. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, 1817–1824.

Laird, J. E.; Kinkade, K. R.; Mohan, S.; and Xu, J. Z. 2012. Cognitive robotics using the soar cognitive architecture. *Cognitive Robotics AAAI Technical Report WS-12-06. Accessed* 46–54.

Leroux, C., and Lebec. 2013. Armen: Assistive robotics to maintain elderly people in natural environment. *IRBM* 34(2):101–107.

Mokhtari, V.; Seabra Lopes, L.; and Pinho, A. J. 2017. Learning robot tasks with loops from experiences to enhance robot adaptability. *Pattern Recognition Letters* 99(Supplement C):57 66. User Profiling and Behavior Adaptation for Human-Robot Interaction.

Oliveira, M.; Lim, G. H.; Lopes, L. S.; Kasaei, S. H.; Tomé, A. M.; and Chauhan, A. 2014. A perceptual memory system for grounding semantic representations in intelligent service robots. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 2216–2223. IEEE.

Oliveira, M.; Seabra Lopes, L.; Lim, G. H.; Kasaei, S. H.; Tomé, A. M.; and Chauhan, A. 2015. 3D object perception and perceptual learning in the RACE project. *Robotics and Autonomous Systems*.

Pomerleau, F.; Colas, F.; Siegwart, R.; and Magnenat, S. 2013. Comparing icp variants on real-world data sets. *Autonomous Robots* 34(3):133–148.

Scheutz, M.; Briggs, G.; Cantrell, R.; Krause, E.; Williams, T.; and Veale, R. 2013. Novel mechanisms for natural human-robot interactions in the diarc architecture. In *AAAI Workshop on Intelligent Robotic Systems*.

Schwarz, M.; Schulz, H.; and Behnke, S. 2015. Rgb-d object recognition and pose estimation based on pre-trained convolutional neural network features. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, 1329–1335. IEEE.

Shafii, N.; Kasaei, S. H.; and Seabra Lopes, L. 2016. Learning to grasp familiar objects using object view recognition and template matching. In *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, 2895–2900. IEEE.

Skočaj, D.; Vrečko, A.; Mahnič, M.; et al. 2016. An integrated system for interactive continuous learning of categorical knowledge. *Journal of Experimental & Theoretical Artificial Intelligence* 28(5):823–848.

Sock, J.; Kasaei, S.; Seabra Lopes, L.; and Kim, T.-K. 2017. Multi-view 6D object pose estimation and camera motion planning using rgbd images. In *International Conference on Computer Vision (ICCV), 3rd International Workshop on Recovering 6D Object Pose*. IEEE.

Srinivasa, S. 2008. The robotic busboy: Steps towards developing a mobile robotic home assistant. In *International Conference on Intelligent Autonomous Systems*, 2155–2162.

Todorov, E.; Erez, T.; and Tassa, Y. 2012. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 5026–5033. IEEE.