Cogn Process
DOI 10.1007/s10339-017-0828-3

RESEARCH REPORT

Integrating planning perception and action for informed object search

Luis J. Manso[1] · Marco A. Gutierrez[1] · Pablo Bustos[1] · Pilar Bachiller[1]

Received: 15 November 2016 / Accepted: 5 August 2017
© Marta Olivetti Belardinelli and Springer-Verlag GmbH Germany 2017

Abstract This paper presents a method to reduce the time spent by a robot with cognitive abilities when looking for objects in unknown locations. It describes how machine learning techniques can be used to decide which places should be inspected first, based on images that the robot acquires passively. The proposal is composed of two concurrent processes. The first one uses the aforementioned images to generate a description of the types of objects found in each object container seen by the robot. This is done passively, regardless of the task being performed. The containers can be tables, boxes, shelves or any other kind of container of known shape whose contents can be seen from a distance. The second process uses the previously computed estimation of the contents of the containers to decide which is the most likely container having the object to be found. This second process is deliberative and takes place only when the robot needs to find an object, whether because it is explicitly asked to locate one or because it is needed as a step to fulfil the mission of the robot. Upon failure to guess the right container, the robot can continue making guesses until the object is found. Guesses are made based on the semantic distance between the object to find and the description of the types of the objects found in each object container. The paper provides quantitative results comparing the efficiency of the proposed method and two base approaches.

Keywords Active perception · Informed search · Perception-aware planning

Introduction

When robots have to find an object and there is previous knowledge-based information about its location, they can leverage such information to look only in those specific containers—e.g. juice bottles are found in fridges and shirts in closets. Unfortunately, in real scenarios there is often lack of such information, so robots usually inspect the locations using non-informed strategies until the objects are found (e.g. close locations first, randomly). These strategies are usually slow and might frustrate future domestic robot users. It would be desirable for domestic robots to be able to acquire and use the necessary visual information to reduce the time needed to find objects.

To guide the object search process, we propose estimating the potential objects existing in each of the containers the robots see in their environments. The images used for this purpose are acquired passively, using those obtained when performing any previous task, whether or not the robot was directly looking towards a container. The images acquired are processed using machine learning techniques, and descriptions of the potential estimated objects found on them are stored to support future guess queries. When looking for a mug, for example, it is not necessary to have a previous mug detection in any of the containers, and a glimpse of a spoon or a plate in a table can be enough for the robot to consider such table first

Handling editor: Antonio Bandera (University of Malaga); Reviewers: David Meger (McGill University), Antonio Palomino (Fundación Magtel).

This article is part of the Special Issue on 'Cognitive Robotics' guest-edited by Antonio Bandera, Jorge Dias, and Luis Manso.

✉ Luis J. Manso
lmanso@unex.es

[1] RoboLab - Robotics and Artificial Vision Laboratory, Escuela Politécnica de Cáceres, Universidad de Extremadura, Avd. de la Universidad s/n, 10071 Badajoz, Spain

Published online: 14 August 2017

when looking for the mug. In particular, we build upon the example of a domestic robot which is asked to find objects in a set of tables located in two different rooms. When the robot needs to find an object and the table where it should be found is not known in advance, it uses the information obtained while performing previous tasks (as shown in Fig. 1) to try to minimize the time spent in the search process. The container to inspect is selected based on the semantic distance between the object to find and the descriptions of the types of the objects detected in each of the different containers (there is a separate content descriptor for each container). For simplicity and without loss of generality, even though the proposed system could be applied to any other kind of container of known shape, they are from now on assumed to be tables.

The integration of the proposal in a robotic architecture enables the robot to generate plans and coordinate different software modules in such a way that object discovery can be part of more complex tasks. Although the system was integrated in a particular architecture for the experiments, it is architecture-agnostic.

The first of the two main contributions of the paper is the design of a software module that suggests the most likely location of an unknown object. The second main contribution is the development of a planning domain which allows to integrate the previously mentioned module in existing robotic architectures to efficiently search for objects.

The remainder of the paper is as follows: Sect. 2 provides a review of the current algorithms used for detecting potential objects and other related works. The active search planning domain and its integration in a robotic architecture is described in Sect. 3. Section 4 describes how labels are obtained as new images are acquired, how is the description of the contents of the tables updated and how the table to inspect is selected. The tests conducted, comparing the guess success rate and the mean time spent by the robot when approaching the correct table using different methods, are presented in Sect. 5. The conclusions drawn from the experiments are described in Sect. 6.

Fig. 1 Robot Shelly looking at one of the tables

Related work

Impressive progress has been made on bottom-up object recognition algorithms using images over the past few years (Rusu et al. 2010; Lee et al. 2015; Redmon et al. 2016), specially thanks to the advances on deep learning (He et al. 2015). Although these algorithms have a high success rate when tested against the most common datasets, their performance is not as good when working in real-world scenarios, where objects may appear at a relatively low resolution because of the distance and are often poorly illuminated.

In real scenarios, objects are found in clutter, usually next to each other. The use of attention processes such as visual attention introduces a preprocessing step that helps the robot focus its attention towards interesting parts of the image in order to improve the object learning and recognition task. Bottom-up attention mechanisms that are designed to respond to salient areas of high contrast in order to detect objects in highly cluttered scenes are often proposed (Itti et al. 1998; Walther et al. 2005). Other more recent approaches perform visual attention by training different configurations of deep neural network models (Xu et al. 2015; Canziani and Culurciello 2015; Mnih et al. 2015).

Classical visual search models can be divided into serial and parallel models (Egeth 1966). Serial search models classify objects one at a time (Sternberg et al. 1966) while parallel ones process some or even all of them at the same time. The feature integration theory (Treisman and Gelade 1980) proposes a parallel, preattentive first stage and a serial second stage controlled by visual selective attention. Also following parallel search processing, guided search-based models direct focal attention to the most promising parts of the visual field (Müller and Krummenacher 2006). Wolfe and Gray (2007) combine information from top-down and bottom-up processing of the stimulus to create a ranking of items for attentional priority.[1]

Active vision approaches arose to cope with clutter and to minimize the impact of other limitations found in real environments (Aloimonos 1993). These approaches actively direct the camera (or in general, the sensor being

[1] For more deep reviews of visual attention models from psychological and neurobiological perspectives, refer to Rothenstein and Tsotsos (2008), Carrasco (2011), Borji and Itti (2013) and Tsotsos (2017).

used) to optimize the usefulness of the sensory input. The *next best view* (NBV) family of algorithms, first appeared in 1981 (Connolly 1985), aims to estimate the best point of view that the camera of an autonomous agent should take to improve the images acquired while performing 3D modelling or object recognition. Different approaches to the NBV problem have been proposed. The effectiveness of several NBV approaches are evaluated in Bissmarck et al. (2015) in indoor and outdoor environments. Another problem associated with NBV is deciding when to stop proposing new points of view of an object of interest being classified. Methods such as the one presented in Wallenberg and Forssén (2010) deal with this problem. They generally use prior information of the objects to decide whether to continue asking for more points of view or ending with the classification of the object, giving up and accepting the object as unknown.

Other research efforts such as our proposal have been directed towards a step of the perception process previous to actually perceiving the object: the estimation of the location of an object sought. Solutions to this problem, known as *informed visual search*, use prior information collected by the robot to improve later search. The work presented in Forssén et al. (2008) is the most similar work to our proposal. To obtain the best viewpoint of the object in search, they use a combination of an exploration behaviour that moves the robot towards unexplored areas, a coverage behaviour that explores the environment with a peripheral camera and a viewpoint selection behaviour, in charge of selecting the proper viewpoint that contains the searched object. Although it is very interesting, it has certain limitations. Firstly, it only works with known objects that have already been seen. Secondly, it was not designed to be integrated in a cognitive architecture, and the robot only knows the geometric location of the object (e.g. the container in which the object might be located is not known).

Other approaches estimate the approximate location of objects by using RFID sensors and generic semantic relationships (Martinez Mozos et al. 2012). A work towards extracting these semantic relationships is presented in Gutierrez et al. (2015), where different images grouped with room names are given to a machine learning algorithm that learns in which kind of room to find each object. The main limitation for applying this last technique in an autonomous robot is that it would not differentiate between two different rooms of the same type (e.g. the robot would not have any preference regarding in which bedroom should it look for a shoe). Even though semantic information obtained using generic training is not useful to differentiate between rooms (or any kind of object container in general), it can still be useful when there are no additional or recent data available to the robot.

Integration of an object oracle in a robotic architecture

To perform household chores and communicate with humans, domestic robots need to model the environment, acquiring knowledge about each object such as its name, use or appearance. This still represents an extremely challenging goal that can only be addressed by experienced research groups reusing already existing software.

Among other numerous pieces of software, to study how robots can use passively acquired images to perform informed visual search, we built on top of the *active grammar-based modelling* (AGM) architecture (Manso et al. 2015, 2016). The AGM architecture was chosen because it was the one in use in the robot used for the experiments and it provides all the required features. The module in charge of estimating the best container to inspect, *oracle* from now on, can be integrated in any other architecture as long as it has planning capabilities and allows using hybrid models combining symbolic and geometric information.

Although designed with perceptual issues in mind, AGM is a general-purpose robotic architecture that can use RoboComp (Manso et al. 2010) and ROS (Quigley et al. 2009) as the middleware connecting the modules that the software of the robot is composed of. The AGM architecture proposes communicating higher-level modules—*agents* from now on—through a shared model with a graph-like structure. This model combines symbolic and metric information as necessary Manso et al. (2016) (see Fig. 2), and is managed and monitored by an *executive* module in charge of providing agents with a sequence of actions that, if executed correctly, would take the robot to a state where its goal has been achieved. Through coordinated cooperation, the agents execute the steps of the plan and make the corresponding changes in the world model so that the executive can update the plan as the mission advances. To fulfil its purpose, in addition to the model, the *executive* module holds the *mission* of the robot and a formal description of the physical and perceptual actions that the robot can execute to modify its own world model—*world model grammar* from now on. The actions of the plan are executed by the agents and the outcome of their execution is acknowledged by modifying the model. Although agents can be composed of lower-level modules, these modules cannot modify the AGM model. An overview of the AGM architecture is shown in Fig. 3.

Domain: node types and actions

The plan that a robot uses to fulfil its mission depends on its domain knowledge (i.e. the actions that can be performed by the robot) but also on the internal world

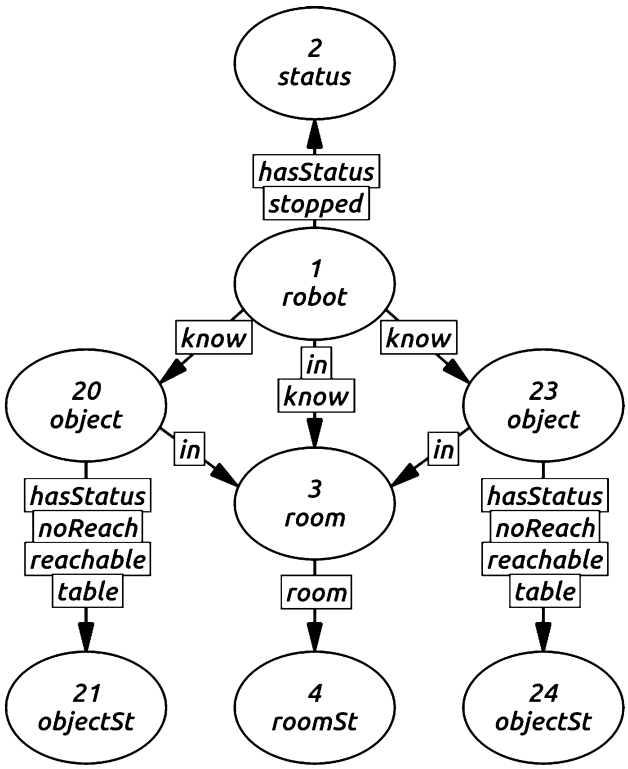

Fig. 2 AGM models represent the world as understood by the robots. The nodes of these graphs represent symbols. Each of them contains two strings: the upper one is a unique identifier and the one in the bottom represents the type of the node. Edges, which can be interpreted as binary predicates, can represent any kind of symbolic property. In this particular case, it is shown the model of a robot which is stopped and—to the knowledge of the robot—is located in a room with two more tables. Nodes and edges can hold string-to-string mappings used to store geometric information

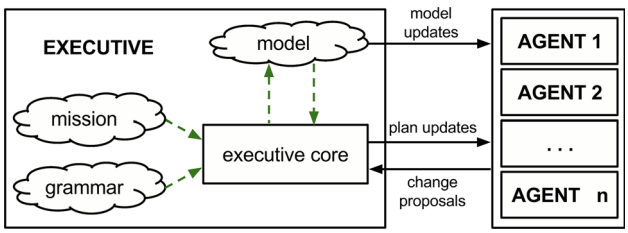

Fig. 3 The *executive* holds the *mission* of the robot, the *grammar* and the current *model*. With each world model modification proposed by any of the agents, the executive updates the plan correspondingly

model of the robot. Therefore, understanding the world model of a robot is essential to understand how it works, and AGM-based architectures are no exception. AGM world models are hypergraphs (i.e. graphs where any pair of nodes can be linked multiple times). AGM models have the particularity that both nodes and edges can have any number of additional attributes. These attributes are used to encode metric information which is not taken into account by the planner. Each node has a unique identifier and a type which determines its purpose.

In addition to other node types that are not relevant for this paper, the models of the robot presented use the following node types: *room*, *robot*, *object*, *imgObject* and additional node types to store node-specific properties, *robotSt* and *objectSt*. The edges between the nodes represent symbolic or relationships. As an example, Fig. 2 shows part of one of these models. Edges labelled as "*in*" represent that one object is inside or over another object. Edges labelled as "*know*" represent objects which are known by the robot. Edges linking an object and their *status symbol* represent dynamic properties of the objects (e.g. whether or not they are reachable or being seen at the moment).

The domain of the robot (i.e. the set of actions that can be executed by the robot, along their preconditions and consequences) is expressed using a *grammar* similar to those used to define formal languages. These grammars are sets of grammar rules that are used by the executive to compute the plans to achieve the missions of the robot. The rules used to proactively find *coffee mugs* will be described in the following paragraphs. There are additional rules designed to include mugs when the robot finds them by chance, rules for perceiving other objects and rules for other different activities such as human–robot interaction or manipulation. In particular, five rules have special interest:

- **setObjectReach** is used for getting close to objects such as tables or mugs. See Fig. 4.
- **changeRoom** is used to make the robot change from a room to an adjacent one. See Fig. 5.
- **imagineMugInPosition** is used to imagine a mug in the most likely container so it can later be inspected and confirmed or discarded. As shown in Fig. 6, it generates an *imgObject* node associated with an already existing container.
- **verifyImaginaryMug** is used to confirm that a previously imagined mug has been successfully modelled as an actual mug. It basically changes the type of the node associated with the imaginary mug from *imgObject* to *object*. See Fig. 7.

Agents

As aforementioned, the actions and their corresponding modifications in the model are carried out by a set of software modules named *agents*. Depending on the current plan of the robot each agent can be in charge of performing an action. Generally the action to execute will depend on the first action of the plan, but it can also depend on any or

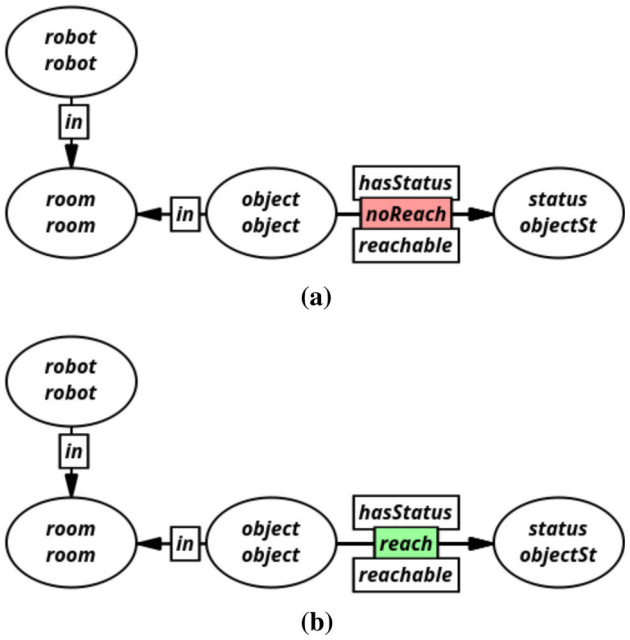

Fig. 4 The action "*setObjectReach*" describes the preconditions to make an object reachable (**a**) and how performing such action would affect the internal world model (**b**). **a** Action "*setObjectReach*": LHS. **b** Action "*setObjectReach*": RHS

all the actions.[2] Similarly, each agent can perform modifications in the model, whether these are the result of the action of the robot or an exogenous event. Exogenous events are those which are not the result of an action of the robot. For example, a "low battery" event would be exogenous: robots cannot generally reason about how to get to such state, and they can only detect it. Therefore, depending on the task a robot has to deal with, different agents will be active.

The following agents are used in the specific implementation of the architecture used for the experiments: a *navigation* agent in charge of moving the robot; a *localization* agent in charge of indicating which is the current room where the robot is located and its specific coordinates; a *proprioception* agent which updates the angles of the joints of the robot in the model; an *object* agent in charge of detecting and updating the position of the objects seen; a *human* agent which includes the persons nearby in the world model; a *dialog* agent which updates the mission of the robot upon user request; and an *oracle agent* which is the main contribution of the present paper. Figure 8 shows the architecture with the agents that were developed for the robot.

[2] Let us assume that a robot located in a room r_1 is supposed to approach a table t_1, located in room r_2 to fetch a bottle of water for a user. A possible plan could comprise, moving to room r_2, then approaching table t_1 and finally detecting a bottle of water on it. Let us also assume that another bottle of water gets into the field of view of the robot as it moves towards room r_2. If and only if the *bottle of water detector* is activated before approaching table t_1, it could be detected and the plan could be optimized using such bottle instead.

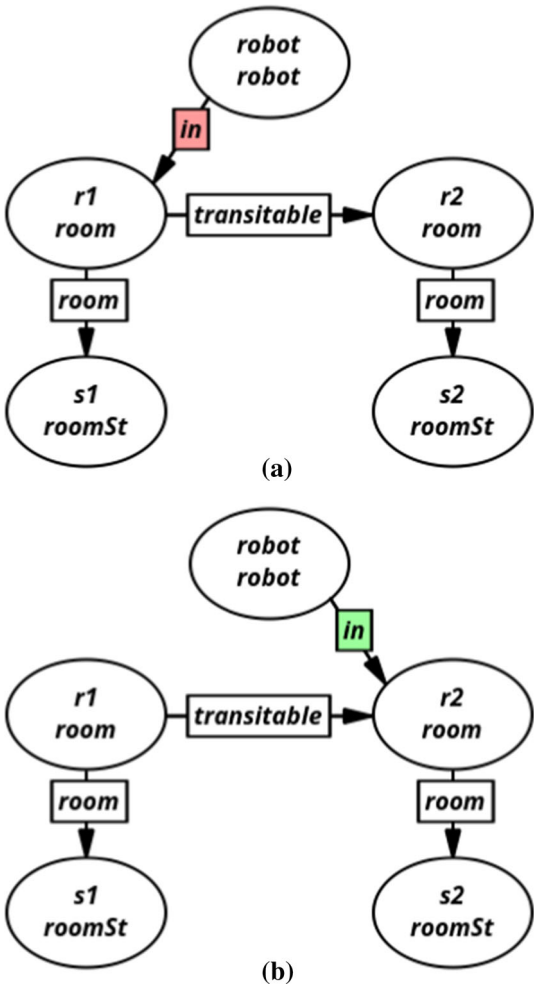

Fig. 5 The action "*changeRoom*" describes the preconditions to make the robot move from a room to another adjacent room (**a**) and how performing such action affects the internal world model (**b**). **a** Action "*changeRoom*": LHS. **b** Action "*changeRoom*": RHS

When the robot needs to find an object the executive provides the agents with the step sequence required to fulfil the mission, which generally entails:

- imagining the object in its most likely position: actions *imagine* $< OBJ > InPosition$, where $< OBJ >$ depends on the type of object to detect.
- reach the container where an object is expected to be found: actions *reachObject* and, in case the container is located in other room, *changeRoom*.
- inspect the container: actions *verifyImaginary* $< OBJ >$.

Actions related to navigation (*reachObject* and *changeRoom*) are implemented in the navigation agent, and those related to object modelling (*verifyImaginary* $< OBJ >$) are implemented in the object agent. Actions *imagine*

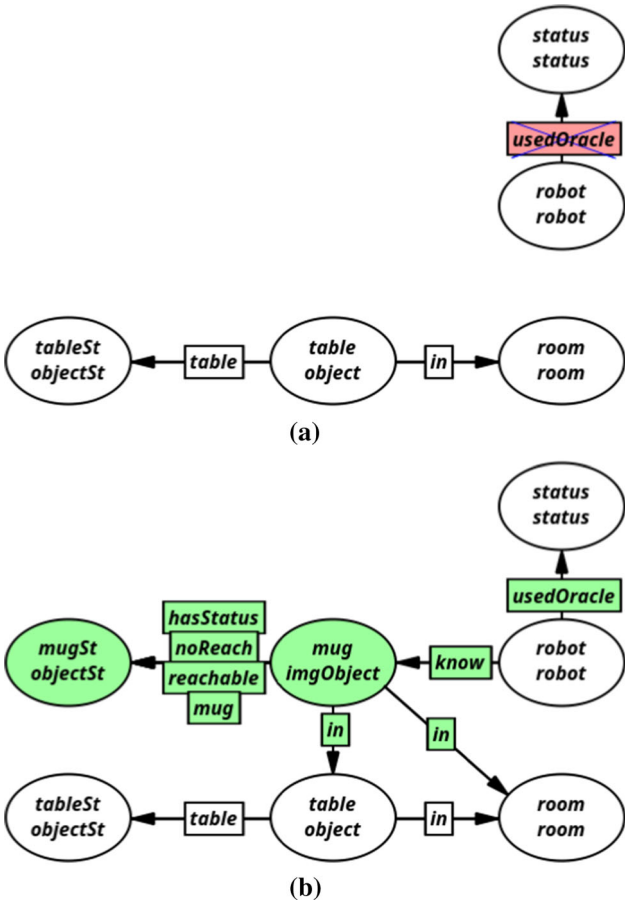

Fig. 6 The action "*imagineMugInPosition*" describes the preconditions to imagine a mug (**a**) and how performing such action would affect the internal world model (**b**). **a** Action "*imagineMugInPosition*": LHS. **b** Action "*imagineMugInPosition*": RHS

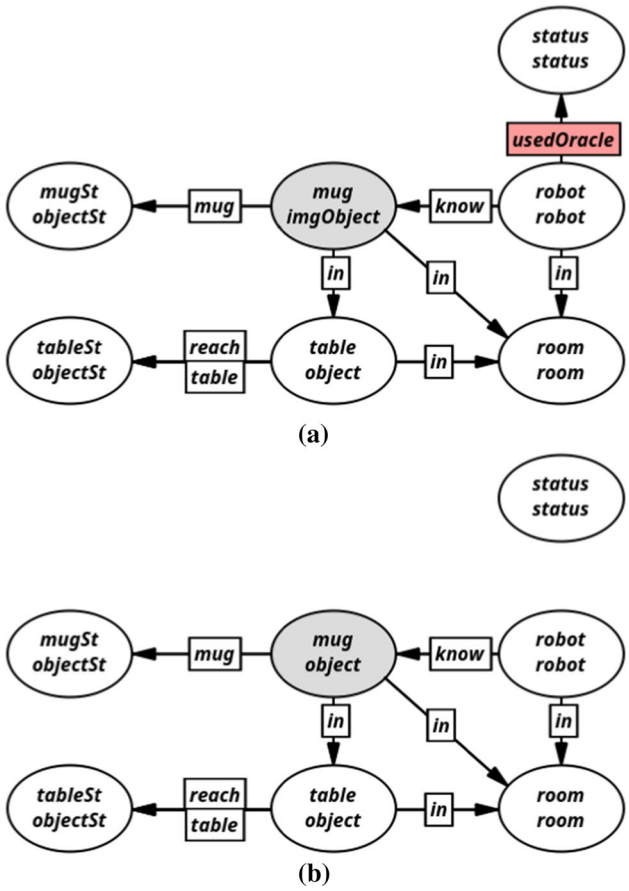

Fig. 7 The action "*verifyImaginaryMug*" describes the preconditions to verify the existence of an imaginary mug (**a**) and how performing such action affects the internal world model (**b**). **a** Action "*verifyImaginaryMug*": LHS. **b** Action "*verifyImaginaryMug*": RHS

$< OBJ > InPosition$ are implemented in the oracle agent. The next section is devoted to explaining how the oracle agent works.

Design of the oracle agent

As aforementioned, two concurrent processes take place in the oracle agent: one that passively generates descriptions of the types of the objects detected in the containers and another one that uses such information to select and indicate in the model which container most likely contains the object sought when the current plan involves imagining an object. The descriptions of the contents of the containers generated by the oracle are average semantic vectors which take as input the list of objects detected by a deep neural network (DNN). The remainder of this section describes how contents are detected, how semantic vectors are generated and updated, and how the oracle performs the query when it is asked to execute an *imagine* $< OBJ > InPosition$ action.

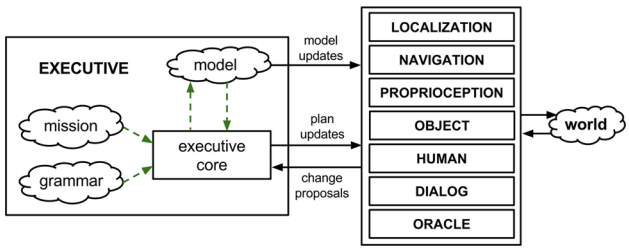

Fig. 8 The figure shows the specific architecture implemented, the executive along with the agents that were developed for the robot: localization, navigation, proprioception, object, human, dialog and oracle

Cue acquisition

Estimating the potential objects in a container and their corresponding probability is proposed as a passive process. The information is obtained providing segmented images to a deep learning model every time the robot has a good point of view of a container which has not been seen recently from a similar perspective. As illustrated in Fig. 9,

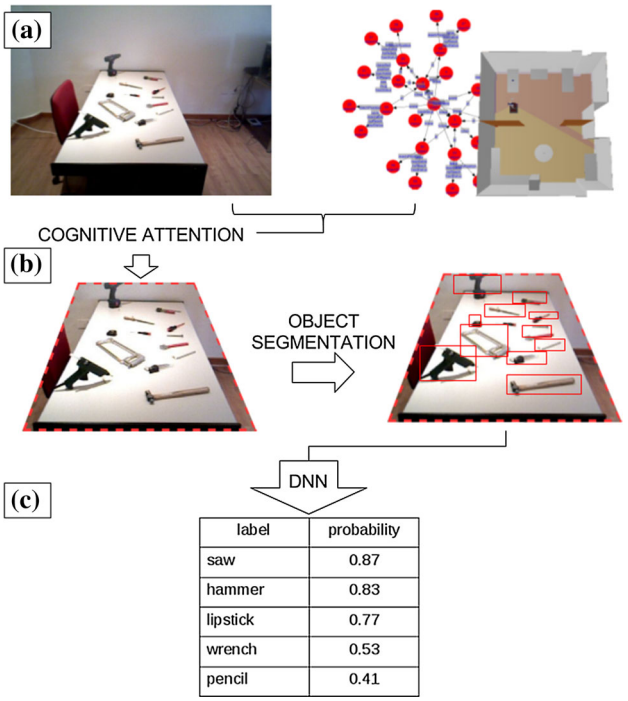

Fig. 9 Pipeline used to extract labels from the camera of the robot: **a** the geometric model extracted from the semantic network and the robot image is used to extract the region of interest; **b** then, the object segmentation is fed to the deep neural network; **c** the result of this process is a list of the possible objects detected in the figure and their associated probability

a first step called cognitive attention (CA) selects the regions of interest (ROIs) in the image corresponding to the containers by projecting the contours of the tables in the model into the image. For each container seen, the ROI defined by its contour is extracted to a separate image. After selecting the container section of the image a cognitive subtraction (CS) step is applied in order to segment the set of regions potentially containing objects into separate images. Finally, these object proposal images are fed to a convolutional network for labelling purposes.

For the cognitive attention step to extract the ROI of the container, the oracle needs to compute where the contour of each container would be projected in the camera of the robot. To this end, the oracle needs an estimation of the size and shape of the container (the table in this case), its pose from the frame of reference of the camera and the focal length. Thanks to the use of AGM models (Manso et al. 2016), all this information can be easily extracted (i.e. assuming the robot has modelled its environment previously). Other valid alternatives to AGM can also be used, as long as they allow storing the necessary information and support computing the coordinates of the contour of the container in the images of the camera of the robot (Milliez et al. 2014; Foote 2013). The environment is assumed to be modelled and updated by another module out of the scope of the paper, since it is a basic requirement of a household robot.

Even though some deep neural networks are able to segment scenes, to detect objects the oracle uses its own 3D pipeline which does not require a DNN with segmentation capabilities. Although this step increases the computational resources required, thanks to the a priori information available, the proposed combination of segmentation and classification yields better results than a CNN-based detector for small objects (Gutiérrez et al. 2017). This issue is well known, and it is explicitly mentioned by the authors of YOLO (Redmon et al. 2016). Once the region of interest for the container is computed and extracted, the objects within it are segmented as follows:

1. A method based on random sample consensus is used to estimate the plane of the table using the point cloud of the scene acquired with the RGBD camera of the robot.
2. A convex hull is performed and all the points lying in a threshold-based bounding box over it are considered objects lying on the table (in the experiments presented in the paper, those between 0.03 and 0.5 m).
3. Different point clouds corresponding to separate objects are segmented using euclidean distance clustering. For the experiments conducted, a threshold of 0.01 m was used.
4. Candidate object point clouds are transformed to image coordinates and the image ROI corresponding to the object candidate is segmented.

Once the segmentation of the objects is obtained, the corresponding parts of the image with potential objects are provided to the neural network for labelling purposes. This provides the system with a probability-ordered list of potential labels from the segmented object image. To obtain these labels, the robot uses deep residual networks by He et al. (2015), trained on the generic ImageNet dataset. It is a well-known network which scored first in the ImageNet classification, ImageNet detection, ImageNet localization, COCO detection and COCO segmentation challenges. It contains up to 152 layers; however, its authors claim it has a lower complexity than others. This is mostly due to the fact that in a traditional approach layers have to generate a whole desired output while layers on this residual networks are only responsible for fine-tuning the output from a previous layer by just adding a learned residual to the input. Therefore, as shown in Fig. 10, the $F(x) + x$ layer is adding in, element-wise, the input x to the $F(x)$ layer (being $F(x)$ the residual). This makes the DNN easier to optimize and more accurate as shown by their results.

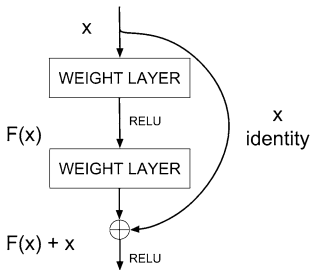

Fig. 10 A building block of the residual learning process in the DNN used in the experiments, proposed in He et al. (2015)

Semantic container vector representation

The deep neural network (DNN) provides the oracle agent with a list of potential objects in the ROI of the object segmented and their estimated existence probability. Since the training is generic and false positives might appear, a careful combination of these labels must be performed, so only the label with the highest degree of confidence is used. Even if the robot might not be able to work with all of the objects embedded in the DNN model (e.g. it is not prepared to manipulate heavy or tiny objects), it is still interesting to be able to detect their existence. Since humans tend to place related things in similar places, specially when working in household environments, we take advantage of likelihood of the coexistence of similar or related objects in nearby locations.

In order to exploit this, the oracle manages the label information through word vector representations, in particular the word2vec model by Mikolov and Dean (2013). This model is generally used for learning vector representations of words, called *word embeddings*. It uses a corpus as an input and produces vectors of 300 dimensions for each word as a result. In a first step, it uses the corpus to create a vocabulary, and then, it learns the vector representation of each word. The resulting word vector can be used for different research purposes. One of the most common ones is the use of the distance between word vector representations as a measure of word semantic similarity, because in the corpus used, similar words appear close to each other. The model used was trained over a Google News corpus with 3 billion running words and it contains the vector representation of these words.

The first time the oracle inspects a container, whether or not this inspection is intentional, it associates the list of objects and their existence probabilities with that container. All the labels are transformed into its vector representation V_i and an average vector C_t is computed and associated with the container as in Eq. 1, being n the number of labels obtained for a container C in a point in time t. Note that only one average vector is obtained per container. This vector is the result of averaging the semantic representations of all the labels assigned to the objects found on that specific container.

$$C_t = \frac{1}{n}\sum_{i}^{n} V_i \tag{1}$$

As the robot keeps moving, it will eventually see the same container again, at point in time $t+1$, probably with another point of view. In this case, a new average vector is obtained using the new label representations V along with the previous vector assigned to the container. The new resulting vector C_t is then assigned to the container in substitution of the old one (see Eq. 2).

$$C_{t+1} = \left(C_t + \frac{1}{n}\sum_{i}^{n} V_i\right)/2 \tag{2}$$

Having an average vector representation of the labels seen helps reinforce the labels that have been detected from different views of a certain container. Since correctly applied labels are seen repeatedly, they will have a higher impact on the average vector value assigned to the container. On the other hand, false positives are expected to be isolated and not be repeated much on different views; therefore, their impact on the average vector of the container will be negligible. Only the label with the highest certainty value obtained from the classifier is used to compute the average vectors, so the distance from average vector to the word2vec representation of labels is used as a measure of likelihood of finding the object in that container.

For a better visualization of the vector representations of objects, t-SNE (Maaten and Hinton 2012) was used. It is a dimensionality reduction technique well suited for the visualization of high-dimensional datasets. Using this technique, Fig. 11 shows a plot of the vector representations of labels obtained for the table containing office objects. The figure shows that most of them are located next to each other in the semantic search space.

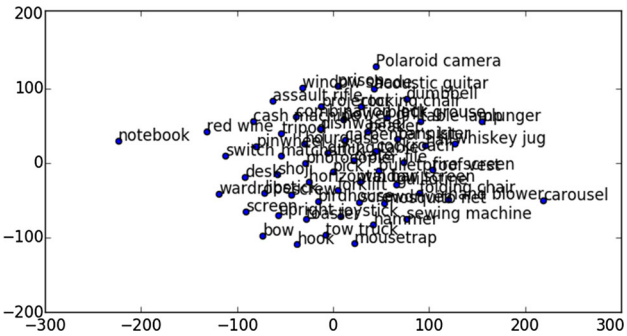

Fig. 11 A 2D plot of the vector representations of labels obtained for a table containing office objects

Querying oracle for object locations

The oracle agent provides the ordered list of most likely containers for a certain type of object. When queried, the first step is to find the vector representation of the object label using the previously mentioned word2vec model trained with a Google News corpus. After that, the oracle computes the euclidean distance between this representation and the vector representation associated with each of the containers, which mathematically corresponds to the dot product of the vectors. A shorter distance means a higher semantic similarity between the object sought and the container. Therefore, the container with the shorter distance is considered as the one having a higher probability of containing the object. The oracle returns the list of containers ordered by this distance value.

Experimental results

A series of experiments were performed to test the possible benefits of using the oracle when looking for objects. As aforementioned in Sect. 2, to the knowledge of the authors the informed visual search method proposed in Forssén et al. (2008) is the closest alternative found in the literature. Although it is a interesting work, its aim differs from the one of the oracles presented in this paper and has several limitations that make impossible a fair comparison. Because of this, the experiments compared the performance of the robot when using the proposed system with the performance without it. In particular, it was compared with a non-informative criterion and another one minimizing the time spent in the worst-case scenario. The non-informative criterion is implemented by inspecting tables in a random order. The third selection criterion uses the position of the robot and the tables known by the robot to estimate the order that minimizes the time spent by the robot if required to inspect all the tables. It is implemented as a travelling salesman problem where all nodes (tables) are connected. For each of the three methods tested, the experiments measured the success rate for the different attempts and the time spent by the robot when looking for objects. It is worth noting that the robot always succeeds in finding the objects because there are a finite number of tables in the environment (5). The goal is to succeed with as few container inspections as possible. The success rate is provided for the maximum of the five possible attempts.

The environment in which the experiments were run is a two-room apartment with a total of five tables: three tables in a room and the rest on the other. There were 65 objects distributed among the different tables during the execution of the experiments. Although the proposed system can be used with any kind of object container, in these experiments we focused on the case of looking for objects located in tables. The tables were filled with objects of five different categories: a) hardware tools, b) desktop with a computer, c) office objects, d) kitchen gear and e) toys. See Fig. 12 for an overview of the experimental setting.

Although the object classifier is trained with generic data and might not contain a specific label, the selection of the container to inspect is carried on using the cosine similarity of the corresponding word2vec vectors. Since there are available trainings of word2vec which take into account 3 billion words, it is highly unlikely that the word used to name an object is not covered by word2vec. For example, even if a user asks the robot to find a *biscuit* and the classifier was trained using the term *cookie*, the system will work properly because the semantic distance between both terms is relatively small.

The robot used for the experiments is driven by an omnidirectional platform and has an RGBD camera mounted in a motorized head which is used for monitoring the contents of the tables. Before running the experiments, the robot wandered over the apartment for approximately 150 s to obtain initial labels corresponding to the objects on each table. Since the robot continuously learns as it moves and analyses the contents of the tables, the effectiveness of the proposed oracle could improve while performing the experiments. Therefore, to prevent the label detection process from learning while doing the experiments (having only *learned* the contents of the tables in the initial wandering stage), the learning capability of the oracle was disengaged before the experiments started. It is disengaged

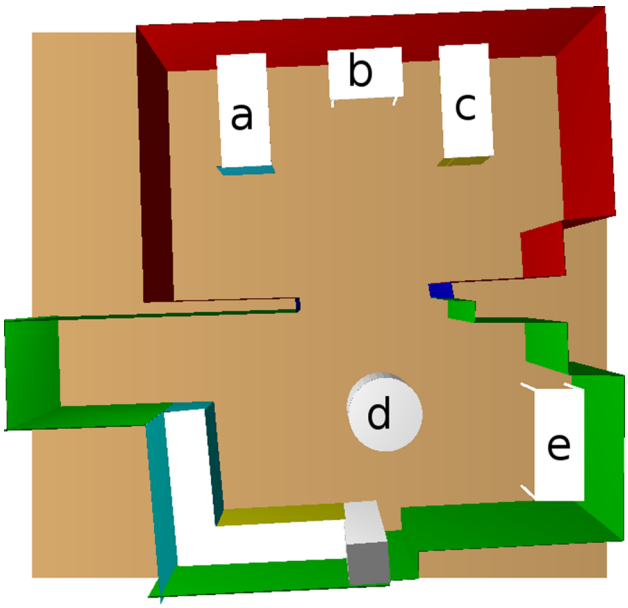

Fig. 12 View of the geometric model of the experimental setting. The highlighted tables contained: **a** hardware tools, **b** desktop with a computer, **c** office objects, **d** kitchen gear and **e** toys

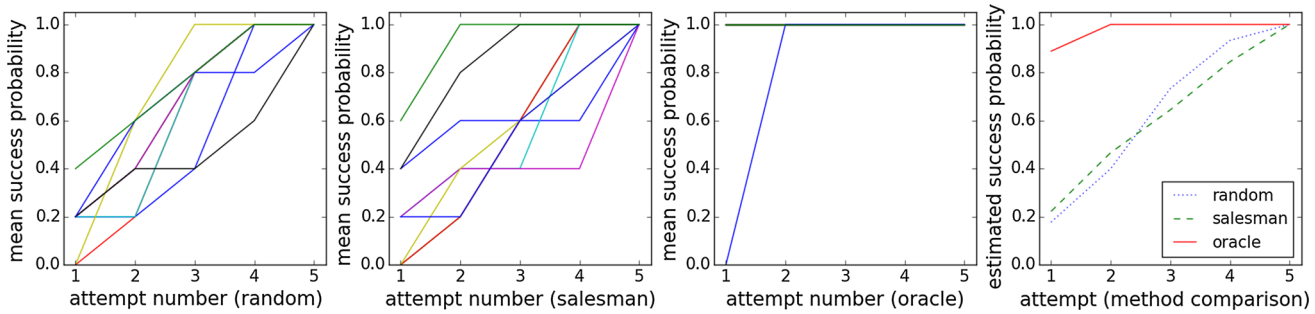

Fig. 13 Mean accumulated success for different objects, from *left* to *right* of the random, travelling salesman and oracle policies. The graphic on the rightmost position compares the general mean accumulated success rates of the three different policies. The success rate is provided for the five possible attempts needed

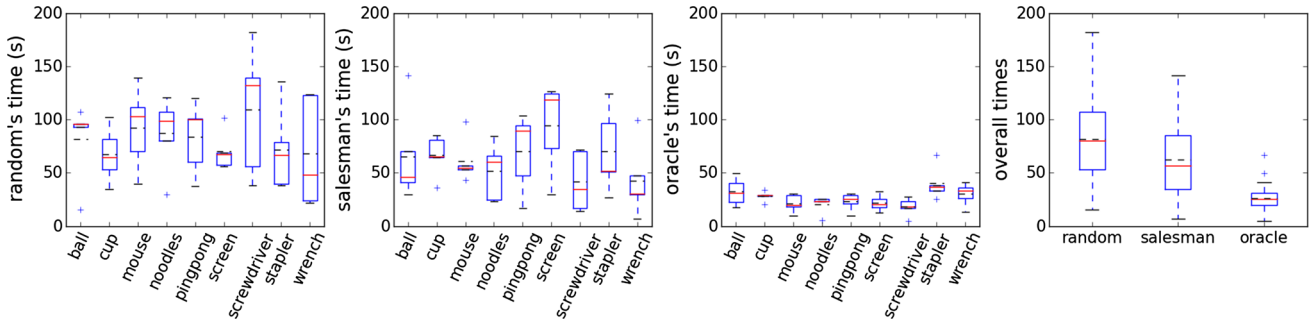

Fig. 14 Boxplots showing the time spent by the robot when approaching the tables containing the different objects, from *left* to *right* of the random, travelling salesman and oracle policies. The boxplot on the rightmost position compares the times of the three methods regardless of the objects

only for testing purposes; under normal operation, this capability would be always enabled.

To perform all the experiments under the same conditions, they were not moved at all. Only the initial position of the robot was changed after performing each search with the three different methods. In the experiments the robot was asked to find five objects from five different locations, erasing the object from the memory of the robot after each execution (otherwise the robot would not need to find the object because it would be already known). Since this paper only proposes a method to approach to the correct table, taking into account the table inspection time would be undesirable. To avoid this problem, once the robot approaches the table, the oracle is automatically informed about whether it made the right guess or it had to approach a different table. The initial success rates measured after performing the experiments (i.e. the probabilities of choosing the right container in the first attempt) were of 0.36, 0.35 and 0.9 for the random, travelling salesman and oracle-based solutions, respectively. Similarly, the average time that the robot used to approach the correct table was of 82.68, 61.0 and 26.13 s for the random, travelling salesman and oracle-based solutions. It is worth noting that, even though the random and travelling salesman policies have similar same success rates, the latter is able to approach faster the right table because it optimizes the order in which tables are inspected.

Figure 13 contains four graphics showing how the success rate evolved with the different methods as attempts were made for all the objects considered. It can be appreciated in the figure that the success rate evolves similarly for the random and travelling salesman policies. On the other hand, the success rate of the oracle was almost perfect, the only case in which the oracle failed was with the stapler. Instead of looking for the stapler in the table corresponding to the office objects, the robot first looked for it in the table corresponding to the hardware tools. This is because the stapler was not detected as such while the robot wandered around the apartment and, using *word2vec*, the semantic distance from stapler to the objects found in the table corresponding to the hardware tools is shorter than to the one of the office objects. The table corresponding to the office objects was the second choice of the robot. Unlike the other approaches, which for some of the objects needed 5 attempts, 2 was the maximum number of attempts needed by the robot to approach the right table.

Regarding hardware requirements, the oracle runs in an Intel NUC with an i7-5557U processor, at a peak frequency

of 94.11 Hz per request. However, since the robot moves at a maximum speed of 0.3 m/s, when working in real conditions its main loop is configured to work at 1 Hz. Running at a higher frequency does not provide much more information and this allows saving computational resources for other modules of the robot. Additionally, it must be taken into account that the computational resources used by the object detection process can be used for other purposes, not only the oracle (e.g. simultaneous mapping and localization Pillai and Leonard 2015).

To demonstrate the general utility of each of the methods, Fig. 14 provides boxplots for the time spent by each of the methods for each of the objects. Although the random and travelling salesman policies have similar success rates (they do not even use the images acquired), the time needed by the travelling salesman is significantly shorter than using the random policy. The proposed oracle system clearly outperforms the other policies regarding the time needed to approach the correct table.

Conclusions

The paper described the two processes that allow the robot to reduce the time spent when looking for objects and how these processes are integrated in a robotic architecture. The first one is a passive process that generates and maintains a feature vector describing the contents detected for each object container. The second one, which is only triggered when required by the plan of the robot, uses the information generated by the first process to select which container should be inspected. The results obtained from the experiments show that the proposed oracle system has the highest success rate (0.9) and requires the shortest time for the robot to approach the correct table (26.13 s). The problem addressed in this paper is relatively unexplored, so two basic container selection criteria were used for comparison. Even though the travelling salesman approach does not have a higher success rate than the random policy, it allows performing the task in shorter times. This suggests that it is worth conducting future research where the oracle takes into account the cost of inspection of the containers.

Generic semantic information obtained using generic training data or readily available semantic networks as suggested by Martinez Mozos et al. (2012) or Gutierrez et al. (2015) can be useful when there are no additional or recent data available to the robot. Another reasonable improvement would be to include attention mechanisms enabling the robot to actively look the containers nearby when the robot is moving and there is no other activity of higher priority requiring the attention of the robot. Future versions of the oracle system are expected to incorporate these features.

Acknowledgements This work has been partially supported by the MICINN Project TIN2015-65686-C5-5-R, by the Extremaduran Government Project GR15120, by the Red de Excelencia "Red de Agentes Físicos" TIN2015-71693-REDT and by MEC project PHBP14/00083. Funding was provided by Junta de Extremadura (Ayudas Consolidación Grupos Investigación Catalogados).

References

Aloimonos Y (1993) Active perception. Lawrence Erlbaum, Hillsdale

Bissmarck F, Svensson M, Tolt G (2015) Efficient algorithms for next best view evaluation. In: IEEE/RSJ international conference on intelligent robots and systems

Borji A, Itti L (2013) State-of-the-art in visual attention modeling. IEEE Trans Pattern Anal Mach Intell 35(1):185–207

Canziani A, Culurciello E (2015) Visual attention with deep neural networks. In: Information sciences and systems (CISS), 2015 49th annual conference on, pp. 1–3, March 2015

Carrasco M (2011) Visual attention: the past 25 years. Vis Res 51(13):1484–1525

Connolly C (1985) The determination of next best views. In: Robotics and automation. Proceedings. 1985 IEEE international conference on, vol 2, pp 432–435. IEEE

Egeth HE (1966) Parallel versus serial processes in multidimensional stimulus discrimination. Atten Percept Psychophys 1(4):245–252

Foote T (2013) TF: the transform library. In: Technologies for practical robot applications (TePRA), 2013 IEEE international conference on, open-source software workshop, pp 1–6, April 2013

Forssén P-E, Meger D, Lai K, Helmer S, Little JJ, Lowe DG (2008) Informed visual search: combining attention and object recognition. In: Robotics and automation, 2008. icra 2008. IEEE international conference on, pp 935–942. IEEE

Gutierrez MA, Banchs RE, D'Haro LF (2015) Perceptive parallel processes coordinating geometry and texture. In: Proceedings of Workshop on Multimodal Semantics for Robotic Systems 2015, Hamburg, pp 30–35

Gutiérrez MA, Manso LJ, Pandya H, Núñez P (2017) A passive learning sensor architecture for multimodal image labeling: an application for social robots. Sensors 17(2):353

He K, Zhang X, Ren S, Sun J (2015) Deep residual learning for image recognition. In arXiv:1512.03385

Itti L, Koch C, Niebur E (1998) A model of saliency-based visual attention for rapid scene analysis. IEEE Trans Pattern Anal Mach Intell 20(11):1254–1259

Lee S, Lim J, Suh IH (2015) Incremental learning from a single seed image for object detection. In: Intelligent robots and systems (IROS), 2015 IEEE/RSJ international conference on, pp 1905–1912. IEEE

Manso LJ et al (2010) RoboComp: a tool-based robotics framework. In: Simulation, modeling and programming for autonomous robots, pp 251–262. Springer

Manso LJ, Bustos P, Bachiller P, Núñez P (2015) A perception-aware architecture for autonomous robots. Int J Adv Robot Syst 12(174):13

Manso LJ, Calderita LV, Bustos P, Bandera A (2016) Use and advances in the active grammar-based modeling architecture. In: Proceedings of the workshop of physical agents, pp 1–25

Martinez Mozos O, Chollet F, Murakami K, Morooka K, Tsuji T, Kurazume R, Hasegawa T (2012) Tracing commodities in indoor environments for service robotics. In: IFAC Proceedings Volumes, vol 45, Elsevier, pp 71–76

Mikolov T, Dean J (2013) Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems

Milliez G, Warnier M, Clodic A, Alami R (2014) A framework for endowing an interactive robot with reasoning capabilities about perspective-taking and belief management. In: The 23rd IEEE international symposium on robot and human interactive communication, pp 1103–1109. IEEE

Mnih V, Heess N, Graves A, Kavukcuoglu K (2015) Recurrent models of visual attention. In: Advances in neural information processing systems, vol 27

Müller HJ, Krummenacher J (2006) Visual search and selective attention. Vis Cognit 14(4–8):389–410

Pillai S, Leonard J (2015) Monocular slam supported object recognition. arXiv preprint arXiv:1506.01732

Quigley M et al (2009) ROS: an open-source robot operating system. In: Proc. of ICRA workshop on open source software

Redmon J, Divvala S, Girshick R, Farhadi A (2016) You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788

Rothenstein AL, Tsotsos JK (2008) Attention links sensing to recognition. Image Vis Comput 26(1):114–126

Rusu RB, Bradski G, Thibaux R, Hsu J (2010) Fast 3d recognition and pose using the viewpoint feature histogram. In: Intelligent robots and systems (IROS), 2010 IEEE/RSJ international conference on, pp 2155–2162. IEEE

Sternberg S et al (1966) High-speed scanning in human memory. Science 153(3736):652–654

Treisman AM, Gelade G (1980) A feature-integration theory of attention. Cognit Psychol 12(1):97–136

Tsotsos JK (2017) Attention and cognition: principles to guide modeling. In: Computational and cognitive neuroscience of vision, pp 277–295. Springer

Van der Maaten L, Hinton G (2012) Visualizing non-metric similarities in multiple maps. Mach Learn 87(1):33–55

Wallenberg M, Forssén P-E (2010) Embodied object recognition using adaptive target observations. Cognit Comput 2(4):316–325

Walther D, Rutishauser U, Koch C, Perona P (2005) Selective visual attention enables learning and recognition of multiple objects in cluttered scenes. Comput Vis Image Underst 100(1):41–63

Wolfe JM, Gray W (2007) Guided search 4.0. Integrated models of cognitive systems, pp 99–119

Xu K, Ba J, Kiros R, Courville A, Salakhutdinov R, Zemel R, Bengio Y (2015) Show, attend and tell: neural image caption generation with visual attention. arXiv preprint arXiv:1502.03044,