Visual search and recognition for robot task execution and monitoring

Lorenzo MAURO [a], Francesco PUJA [a], Simone GRAZIOSO [a],
Valsamis NTOUSKOS [a], Marta SANZARI [a], Edoardo ALATI [a], Luigi FREDA [a],
Fiora PIRRI [a]

[a] *ALCOR Lab, DIAG, Sapienza University of Rome, Italy*
{mauro, ntouskos, sanzari, alati, pirri}@diag.uniroma1.it

Abstract. Visual search of relevant targets in the environment is a crucial robot skill. We propose a preliminary framework for the execution monitor of a robot task, taking care of the robot attitude to visually searching the environment for targets involved in the task. Visual search is also relevant to recover from a failure. The framework exploits deep reinforcement learning to acquire a *common sense* scene structure and it takes advantage of a deep convolutional network to detect objects and relevant relations holding between them. The framework builds on these methods to introduce a vision-based execution monitoring, which uses classical planning as a backbone for task execution. Experiments show that with the proposed vision-based execution monitor the robot can complete simple tasks and can recover from failures in autonomy.

Keywords. Artificial Intelligence, Computer Vision and Pattern Recognition, Robotics

1. Introduction

One of the main problems in robot task execution is the interplay between perception and execution. Indeed, it is not clear what information a robot should collect from the environment to be able to continuously interact with it.

So far, 3D reconstruction, 3D reconstructed scene segmentation and SLAM seemed the best way to provide an effective and convenient representation of the environment for task execution. Several scientific contributions have focused on solving these problems in compelling real-world scenes. Once the system is aware of both the complete environment and the robot pose, it provides to the execution monitor adequate instructions for both the planning and the required continuous updates.

A major problem in 3D reconstruction and SLAM approaches is that they do not consider the robot gaze direction. They create scene reconstructions designed for the human user and not in order to facilitate the robot reasoning about the environment. Indeed, this lack of connection between the scene and where the robot looks at creates an unsound reasoning process: the map guides the robot searching according to a global view (the human user view) and not on a saliency-based view, [1,2].

As a matter of fact, most of a 3D map reconstructions are full of unnecessary details for a robot to keep in memory, in relation to its job. It has also to be said that a represen-
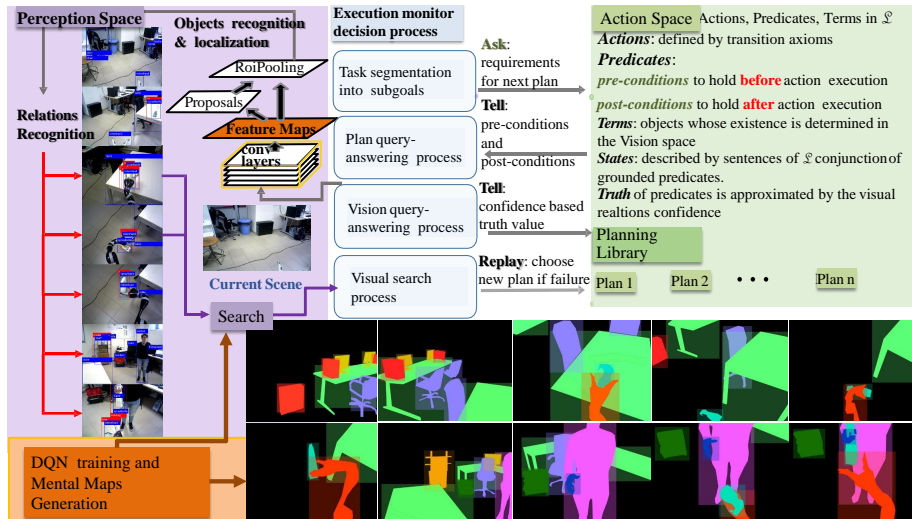
Figure 1. The execution monitor framework combining classical planning, detection and localization of objects together with visual search. Best seen on screen.

tation of the world needs to cope with frame and qualification problems [3,4] and when it comes to a full 3D map a huge amount of details can be very hard to deal with. In fact, humans learn to search and move in any complex environment essentially relying on foveated vision and on what they have learned, to limit waste of their own resources.

Basing on the human ability to hold partial representations and to fill the gap between what the robot can denote and the complexity of the world, we introduce a parsimonious representation, which we call *mental map*. Mental maps are scene snapshots keeping only objects and relations that the robot can detect and supporting the learning process for the visual search: they are the robot cartoon representation of its environment. Learning where to look at solves the problems related to both global and local maps, as it can help to assess a low cost, yet effective, representation of the environment.

Here we propose a novel perspective to execution monitor, which is based on the interplay between execution and perception. Our idea (see Figure 1 for an overview) combines deep learning, which is able to develop a quite mature cognitive approach to perception, with classical planning and monitoring. The proposed framework is centered on the robot execution monitor. The execution monitor is supported by the planner, see [5,6,7], by the vision system processing visual data with Faster R-CNN [8], and by the learned ability to direct the robot gaze in a plausible direction for searching the environment, via a modified version of DQN [9]. Planning is the backbone, in the sense that it provides the properties of the world that are expected to be true before and after the execution of a robot action. The planner, however, does not decide actions sequences since it cannot verify whether the prescribed requirements are true. The planner provides a sort of prior knowledge about the space of action execution, but not of their realizations. That is why the planner is enriched by a query-answering process with the execution monitor to validate whether the requirements are satisfied in the environment. The execution monitor resorts to visual search to activate the choice of next plan, which is crucial in case of failure (e.g. the loss of an object). To infer policies the execution monitor builds states representations of the scene out of the visual stream results, formed by mental
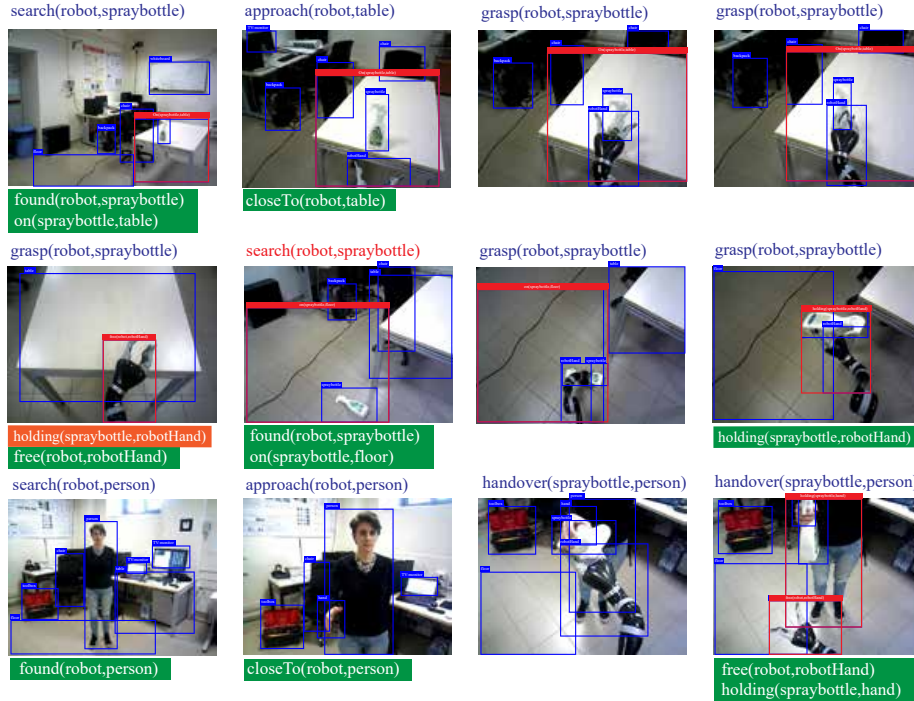
Figure 2. Execution monitoring of the task *take the spray bottle and hand it to Person*. Actions written in blue are provided by the planner and those in red by the visual search policy, once a failure is detected. The post-conditions are shown below the images, in orange if not verified. Inside the images, the bounding boxes of the detected objects are shown in blue and in red the detected relations calculated by the perception system.

maps, see Figure 1. Mental maps incorporate the essential knowledge for a robot to complete a task. To learn the search ability, we adapted the Deep Q-Network, [9], to receive RGB input and we train it on a virtual environment. Policies learned on the virtual environments, though displaying a good ability to deal with the robot representation via mental maps, can be improved by learning in real environments, and this is the objective of future work. The execution monitor is finally able to control the loop between action and perception without the need to keep the structure of the entire environment in memory, but updating its perception with what is required/asked by the planner to complete the action.

The robot operates in a laboratory environment and a schematic overview of a task execution is given in Figure 2.

2. Related Work

Visual Search Visual search is not a novel topic in robotics and a large body of work has been done so far to approach the problem, though it has been faced mainly in terms of global navigation.

Some of the approaches based on map navigation require a prior knowledge of the environment to make decisions (see for example [10,11,12,13]), all based on 3D map

reconstruction, as in [14,15]. Other methods do not actually rely on pre-saved maps but on a map reconstruction which is done either on the fly (see [16,17,18,19]) or during an initial human guided training phase as in ([20,21]). Other approaches instead face the search in terms of obstacle avoidance using input images in so not relying on any kind of map (see, [22,23,24,25]). However, all of the above-mentioned methods depend in one way or another on the navigation of a reconstructed environment. To be able to properly explore the environment in this way, the process needs to store the collected information in memory.

Differently from previous works our approach aims at updating the perception with only what is necessary at the very moment in which the information is needed to the robot to perform and complete what it is doing. This approach is more similar to a human-like behavior and it is realized with a neural network trained with reinforcement learning.

Earlier approaches to reinforcement learning (RL) (see, for example, [26,27] made the effort to extend this technique to a very large set of actions and to make it a robust method. Despite that, the breakthrough started with the work described in [9] who introduced a new amazing research area, the deep RL [28,29]. The main difficulty with this approach is the need to simulate the environment to perform the required huge collection of experiments. Hence a certain amount of research has been devoted creating new simulation environments such as [30,31]. Here we contribute with the above mentioned mental map representation of the environment to be distinguished from the V-REP [32] simulation used for training. A similar approach has been proposed by [31]. Differently from their work, which uses an auxiliary target image as input to the network, here we only provide the robot's onboard camera pictures as an input to the visual search, allowing us to search for objects regardless of their orientation.

Execution Monitoring   Managing the execution of actions involve facing several difficult challenges. A great number of researchers tackled the non-deterministic response of the environment to robot actions for high-level robot tasks [33,34]. Some introduced the role of perception in execution monitoring [35], others focused on the recovery from errors that could occur at execution time [36] and on relations recognition, which recently raised much interest in computer vision community [37,38,39] though related to 2D images. However, many problems remain to be addressed and there is still a lack of a comprehensive approach to address all the problems concerning the interleaving of perception and actions in a coherent way.

The first definitions of execution monitoring in non-deterministic environments have been presented in [40,41]. Recovery from errors that could occur at execution time was already faced by [36] and equally the role of perception in execution monitoring was already anticipated in the work of [35]. Due to the difficulties in dealing with scene recognition the efforts have been directed toward models, such as in [27,26], that manages the effects of actions allowing the execution in partially observable environments [42]. In high level monitor several authors, among which [43,44], have been recently addressing the integration of observations. Still, the most important development has been achieved with deep learning for perception and in particular with DCNN[37], deep Reinforcement Learning (RL) [9,28] and visual planning [45].

Nevertheless, despite the great number of contributions [37,38,39] for 2D images, an appropriate relations recognition method for robot execution is still not available. For this reason, we exploit the robot RGBD vision and a simple rough segmentation method

within the bounding box delivered by Faster R-CNN to obtain a more detailed object region, in so ensuring a better approximation of relations.

3. The Execution Monitor

The robot execution monitor drives the interplay between the action space, defined by actions, relations, and objects in a first order typed language and the perception space that specifies objects and relations in terms of features and measures. Figure 1 provides a schematic representation of the interplay between the two spaces.

The action space addresses the current knowledge required by the robot in order to perform a specific action, whether this is an action not affecting the world, or an action changing the state of the world. The execution monitor, given a goal in terms of a task, has a strategy to segment it into sub-goals. This strategy is not described here. For each sub-goal, according to the terms appearing in it, the execution monitor uses naive Bayes to *ask* to the plan library the plan matching the sub-goal, according to the current state. The current state is verified by assessing what is available to the robot perception in terms of objects positions and relations holding at the specific time. Plans are defined in the classical PDDL framework [5,46], see Section 5. The inference of a sub-goal by the chosen plan results in an ordered list of actions, objects, and relations that the planner *tells* to the execution monitor. The execution monitor, in turn, asks the vision process to verify the list of actions while they are executed. If the list is verified the execution monitor triggers the next sub-goal. Otherwise, this component activates the visual search to focus on the validation of the plan requirements, and eventually repeats the current plan from the specific state. An example of a failure is given in Figure 2 where in sub figure 5 the robot hand accidentally drops the spray bottle.

4. The Perception system

The perception space is defined by features extracted from the DCNN and by the mental maps. Two processes live in this space: the object detection and localization and the visual search.

4.1. Object detection and localization

Object detection uses the well-known Faster R-CNN deep convolutional neural networks [8] and a statistic estimation of the object boundary within the bounding box detected by the trained DCNN, using the active features of the VGG-16 5.3 layer, which is a component of Faster R-CNN.

The combination of segmented depth and object labels allows inferring visual relations from the robot point of view.

Specifically, to detect both objects and relations we have trained Faster R-CNN [8] on ImageNet [47], Pascal-VOC [48] dataset, and on images taken on site. We have trained 5 models to increase the accuracy of the detection obtaining good results. The detection accuracy is based on a confidence value measured on a batch of 10 images, taken at $30 fps$, computing the most common value in the batch and returning the sampling mean accuracy for that object.

Predicate	Full	BB	GT masks	no prior	no shape	no depth feat.
CloseTo	82%	79%	89%	79%	72%	49%
Found	88%	85%	95%	85%	80%	61%
Free	91%	86%	91%	86%	83%	68%
Holding	79%	72%	88%	75%	74%	56%
Inside	69%	64%	87%	71%	65%	57%
On	85%	77%	96%	79%	78%	65%
InFront	88%	81%	95%	84%	83%	63%
Left	90%	81%	95%	85%	86%	72%
Right	83%	79%	91%	79%	80%	61%
Under	80%	76%	88%	79%	76%	59%
Behind	85%	78%	81%	76%	79%	61%
Clear	82%	75%	80%	73%	73%	60%
Empty	83%	72%	78%	79%	68%	61%
Average	**84%**	**78%**	**92%**	**80%**	**78%**	**61%**

Table 1. Accuracy and ablation study of predicate grounding. **Legend:** *BB*: bounding boxes, *GT masks*: ground truth segmentations masks, *no prior*: without use of VGG-features *no shape*: without use of statistics. *no depth feat.*: without use of depth features.

For each object ob_j detected by the DCNN we estimate a statistics of the active features with dimension $38 \times 50 \times 512$, taken before the last (5.3) convolution layer, at each pixel inside the recognized object bounding box $bb(ob_j)$. With this features set we estimate the probability that features with the highest frequency in the feature volume, fall within the object. Then reprojecting the location of maximum probability on the object cropped by the bounding boxes we obtain regions within the object. It is interesting to note, indeed, that the features (the weights learned by the convolution network) of the 5.3 convolution layer of the VGG-16, behaves like saliency in identifying the object interesting parts. Then, using nearest neighbor we find more accurate regions within the object. Since the images/videos available to robot perception are RGBD images, from this process we obtain a quite accurate segmentation of the depth map of the object. Note that this step is necessary to recognize relations between objects since we are talking about spatial relations where distance and location are crucial. Considering only the bounding boxes would hinder spatial relations: some of the objects involved in the robot tasks, such as a table, ladder, spray bottle and so on, have bounding boxes containing also other parts of the environment (see Figure 2). Without an accurate segmentation, the detection of an object returns incoherent depth and the decision for execution monitoring on controlling actions, such as *grasp*, *hand-over*, *place*, *open*, *close* and so on, cannot take place. Instead with the segmentation easily obtained via the significance of the last convolutional layer of VGG-16, we are able to define spatial relations such as *Holding, On, Below* and so on, just using relations derived by the center of mass of the objects, in close connection to the qualitative spatial relations framework [49] extended to 3D, see also [50]. A set of relations used in the framework are shown in Table 1 evaluating accuracy and ablation study. Evaluation of objects detection and localization is reported in the result Section 6.

Algorithm 1: Deep Q-Learning with Mental Maps and Experience Replay

---

    input : Faster R-CNN network

                Simulated Environment

    output: Trained DQN network

1 Load Simulated environment ;

2 Load Faster R-CNN network ;

3 Initialize replay memory D;

4 Initialize action-value function Q with random weights ω;

5 Initialize first state s_1 and Mental Map θ_1 with random values;

6 **for** $t = 1, maxSteps$ **do**

7   Calculate ε-greedy probability $\varepsilon(t) = 0.9/t_{\varepsilon_{min}} * t + 1$;

8   With probability $\varepsilon(t)$ select a random action

              a_t otherwise select $a_t = \max\limits_a Q(\theta(s_t), \omega)$;

9   Execute action a_t and observe image i_{t+1};

10   Elaborate image i_{t+1} with Faster R-CNN ;

11   Produce processed picture i^*_{t+1} and reward r_t;

12   Set state $s_{t+1} = s_t, a_t, i^*_{t+1}$;

13   Create mental map $\theta_{t+1} = \theta(s_{t+1})$;

14   Store transition $(\theta_t, a_t, r_t, \theta_{t+1})$;

15   **if** $t > startTraining$ **then**

16   Sample random minibatch of transition $(\theta_j, a_j, r_j, \theta_{j+1})$ from D;

17   Set

$$y_j = \begin{cases} r_j & \text{for terminal } \theta_{j+1} \\ r_j + \gamma \max\limits_a Q_\pi(\theta_{j+1}, \omega) & \text{for non terminal } \theta_{j+1} \end{cases}$$

              Perform gradient descent on loss $(y_i - Q(\theta_j, \omega))^2$

18   **end**

19 **end**

4.2. Visual Search

As introduced above, *visual search* is triggered both to check the current state and to recover from a failure. Visual search is modeled by deep reinforcement learning to focus the robot toward the items of interest. This step is necessary for the robot to find out whether the preconditions of the action to be executed are satisfied or to search a starting point for a recovery action.

    The reinforcement learning method allows learning by trial and error how to choose the best action to locate the target object. The visual search does not have to rely on previous knowledge of the environment but it only uses the images returned by the onboard robot camera suitably elaborated into mental maps. To achieve this result, the Deep Q-Network (DQN) proposed in [9] is used, adapted to receive as input a collection of pre-processed RGB-D camera frames called *Mental Maps*. We recall that the target of the DQN learning process is to learn an approximation of the reward function for each pos-

sible action: the Q-value. The final policy will be, therefore, the one created by selecting the action with the highest Q prediction.

Since the field of vision of the agent returns only a partial observation of the environment, the states are only partially observable. To treat this setting as a fully observable one, the current state is composed of a history of the previous frames. However, due to memory limitations, only the last 15 states are used. Together they form the *Mental Map*, which is the input of the DQN.

The raw frames that compose the *Mental Map* are preprocessed with Faster R-CNN as explained in Section 4.1. Using the output of the network we draw a colored mask for each detected object. The color is assigned in relation to the object class. A visualization of the framework can be found in Figure 3.

The creation of the mental maps is done for a variety of reasons. The strong visual clue introduced by the segmentation of the objects detach the learned behavior from the specific textures of the images. This property makes it possible to use the same DRL model in scenarios which have the same objects but different textures by just fine tuning the object detection network. In this way, it is possible to use a robot trained in a simulated environment directly in a real-world application. Strong visual clues allow also to analyze the situation regardless of the specific object orientation. This is a crucial step, in particular during the failure recovery scenario when the object orientation and position are unknown. Without this step, the DRL method would need to be trained not only on a wide variety of target object positions but also on a wide variety of target object orientations making the training process longer.

The mental maps generation process produces the reward used to calculate the training loss. This reward is calculated in relation to whether or not the robot has found the target object. In those steps in which the target is recognized a positive reward is assigned in proportion to the classification score of Faster R-CNN. Instead, the reward of any action not leading to the localization of the target is set to a fixed negative value.

At training time the policy used is ϵ-greedy. The value ϵ, which is the probability that the training process chooses a random action instead of the one calculated by the network, starts from an ϵ_{max} value and linearly descends to an ϵ_{min} value. In this way, the agent is able to explore the environment and to focus on optimizing the policy as the training goes on. An important feature of our training process is that the episode has a fixed step size. The sequence is not interrupted even when the robot sees the target object with a confidence score higher than the decision threshold. This is done in order to choose the action that maximizes the classification score and the consequent reward. In this way, if the robot sees the target object classified with a low score, it learns to investigate on the situation trying to increase the classification score by getting closer or looking at the object from a different position. Furthermore, with such investigating approach, if the classification score is the result of a false positive, the score would drop down prompting the robot to change the search behavior. Algorithm 1 illustrates the main steps of the training process.

5. Action space and the plan library

We focus here on the deterministic part of the visual execution space: the action space and the plan library. The action space is defined by a *domain* $\mathcal{D}$ formed by objects, predicates

and real numbers, which are represented in the *robot language* $\mathcal{L}$. A term of $\mathcal{L}$ can be an object ob_i of the domain $\mathcal{D}$, a variable or a function $f : \mathcal{D}^n \mapsto \mathcal{D}$, such as actions. Predicates are the counterpart of relations in the perception space and are here assumed to be only binary. Predicates are defined exclusively for the specified set of objects of the domain and are interpreted essentially as spatial relations such as *On, CloseTo, Inside, Holding*.

A *robot task* is defined by a list of plans $\langle \mathcal{P}_1, \ldots, \mathcal{P}_m \rangle$; a *plan* $\mathcal{P}_i$ is defined as usual by an initial state and a set of rules (axioms) specifying the preconditions $H(a_t)$ and postconditions $K(a_t)$ of action $a(t)$, where $t \in \mathbb{N}$ is a discrete time index. To simplify this presentation we assume that preconditions and postconditions are conjunctions of ground atoms (binary relations), hence a state $s_t = H(a_t) \cup K(a_t)$.

In our settings robot *actions* can be of two types: (1) *transforming actions*, which are actions changing the state of the world, affecting some object in the world, such as *grasp*, *hand-over*, *place*; (2) *ecological actions*, which are actions changing the state of the robot, such as *move-Forward*, *approach* ob_j along a direction, or *look-Up*, *look-Down*.

According to these definitions, a *robot plan* specifies at most one transformation-action. This action is a *durable action*, defined by an action *start* and by an action *end*. Any other action in the plan is an ecological action. In particular, given the initial state, each plan introduces at most a new primal object the robot can deal with and the relations of this object with the domain. An example is as follows: the robot is holding an object, as a result of a previous plan, and it has to put it away on a toolbox. This ensures that recovery from a failure is circumscribed for each plan to a single transformation action.

The *task goal* is unpacked into an ordered list of goals $\mathcal{G} = (G_1, \ldots, G_m)$ such that each G_i is the goal of plan $\mathcal{P}_i$ in the list of plans forming a task. Given a goal G_i, if a plan $\mathcal{P}_i$ leading from the initial state to the goal exists, then a sequence of actions leading to the goal is inferred by the search engine [46]. Here plans are defined in PDDL [51].

We extend the parser introduced in [46] to infer plans so as to obtain the initial state, the sequence of actions $(a_0, \ldots, a_n)$ leading to the goal, and the set of states $S_{\mathcal{P}_i}$ for each plan $\mathcal{P}_i$. This list forms the set of deterministic states and transitions.

6. Results

6.1. Experiments setup

The framework has been tested under different conditions, in order to evaluate different aspects of the model. To begin with, all experiments have been performed with a custom-made robot. A Pioneer 3 DX differential-drive robot is used as a compact mobile base. To interact with the environment we mounted a Kinova Jaco 2 arm with 6 degrees of freedom and a reach of 900mm and finally, for visual perception, we used an Asus Xtion PRO live RGB-D camera mounted on a Direct Perception PTU-46-17.5 pan-tilt unit. We ran the experiments on two computers mounted on the robot. One was dedicated to the planning and management of the execution and another one, equipped with a Nvidia GPU, to run the perception system.

For training and testing our system, we used different objects and all the relations needed to correctly execute and monitor different tasks. A list of those elements can be seen in Table 2.

Table 2. Subset of Objects, relations, transformation actions and ecological actions of the robot language $\mathcal{L}$

Relations	Objects	Transformation actions	Ecological actions
CloseTo	Brush	Close	Look-down
Found	Chair	Grasp	Look-left
Free	Cup	Open	Look-right
Holding	Floor	Hand-over	Look-up
Inside	Hammer	Place	Move-forward
On	Person	Lift	Move-backward
InFront	Spray Bottle	Push	Turn-left
Left	Screwdriver	Spin	Turn-right
Right	Shelf	Dispose	Localize
Under	Toolbox		Rise-arm
Behind	TV-Monitor		Lower-arm
Clear	Table		Close-Hand
Empty	Door		Open-Hand

6.2. Results for Object detection, localization and relations recognition

We trained the visual system with images taken from ImageNet and Pascal VOC datasets as well as images collected with the RGB-D camera of the robot, and using predicates for the execution of the tasks. To train the DCNN models we divided the images into training and validation sets with a proportion of 80%-20%. We have chosen five different models for the objects, performing 70000 training iterations for each model on a PC equipped with 2 GPUs. The visual system has been tested under different conditions, in standalone tests and during the execution of different tasks. The accuracy has been computed considering the batch of 10 images, see in Section 2.

6.3. Results for Visual Search

Simulation in V-REP: The training phase of the *visual search* component requires a huge amount of steps and episodes in order to perform well at testing time. Training the robot in a real environment is not an efficient option. To address this problem we created a virtual environment using the V-REP simulation software [52], that allows training the robot in a setting similar to a real scenario. The simulation only generates camera images at the end of the chosen action since this is the only information available to a robot in a real scenario. In this way, the training process can be easily extended also for a real robot and the learned policy can be fine-tuned in a real scenario after a long training in a simulated environment. The robot can perform several actions, it can move backward or forward, turn right or left and it can also move the camera upward or downward and to the right or to the left. The policy chooses the action and when it is performed, the image from the real or simulated camera is given to the Mental Maps creation process as shown in Figure 3.

Training Details: To train the robot in a wide variety of situations, the target object, camera position and orientation are changed at every episode. Further, before starting an episode, a random number (between 1 and 3) of random actions are performed. In this way, we have 46720 different combinations of initial settings to prevent the overfitting problem for training.
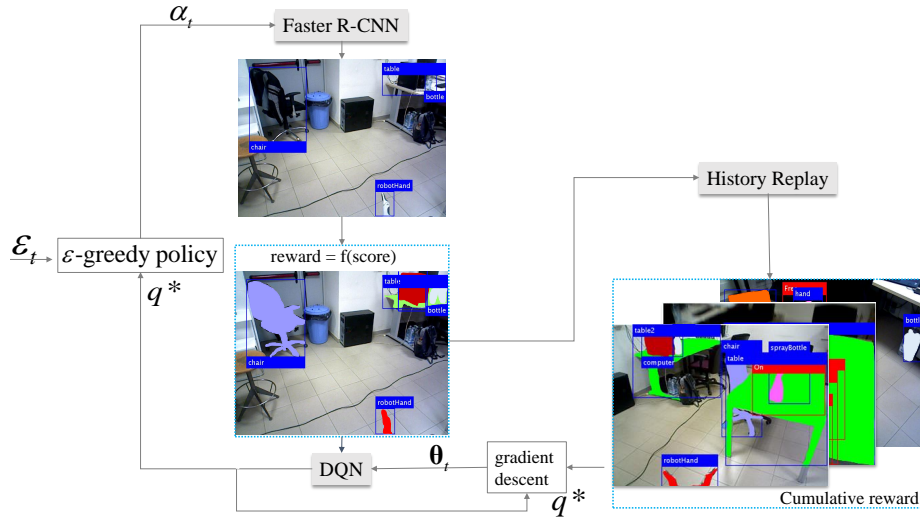
Figure 3. Visual Search Training Framework. In the figure α_t is the action chosen by the ϵ-greedy policy at time t, q^* is the q-value calculated by the DQN network and $\boldsymbol{\theta}_t$ is the vector of network weights at time t.

The Deep reinforcement learning is implemented with a modified version of the DQN implementation, see [9]. The network is adapted to take as input the last 15 RGB-D frames with a resolution of 640×480 pixels. The network has 3 convolutional layers each followed by a ReLu layer. The first one has $15 \times 4 \times 4 \times 4 \times 8$ filters and $1 \times 4 \times 2 \times 2 \times 1$ stride. The second layer has $1 \times 8 \times 8 \times 8 \times 64$ filters and $1 \times 1 \times 4 \times 4 \times 1$ stride and the third and final layer has $1 \times 4 \times 4 \times 64 \times 128$ filters and $1 \times 1 \times 2 \times 2 \times 1$ stride. The returning feature map from the convolutional layers is reshaped and passed to 2 fully connected layers. The performances reached are the results of more than 4 million iterations on a machine with 16 Gb of RAM, processor Intel Core i7-3770 and graphics card Geforce GTX 1080. The first 160 steps are used to fill enough the replay memory to allow a correct sampling of 32 steps sized mini-batches. The maximum replay memory size is set to 1000 steps and the weight update is done every 4 steps. The mental maps length is set to 15 steps and the ϵ is linearly decreased from 1 to 0.1 over the first half of the training steps. Different episode settings have been tested, changing essentially the condition to stop the episode and the reward function. The final solution is to assign a positive reward to frames in which the object is detected with both a high or low classification score, proportionally to the magnitude of the score itself. The final reward function is set to $R(score) = 50 * score^5$ for scores higher than 0.3 and -0.1 for smaller scores. The episode length is set to a fixed size of 50 steps.

Evaluating results: An accurate evaluation of the robot performances is a quite challenging task. There is no specific output value and the metrics that can be used tend to be very noisy. To better evaluate the results of the training phase, the visual search policy has been tested in different ways. 100 thousand training steps have been done, at first by following a random policy and subsequently by following the policy returned from the trained network. In this setting, if the target is found in fewer than 50 steps the episode is ended and considered successful. In the case of random action choices, the success rate was of only 2.97% with an average of 41 steps. With the trained network instead, the

	Random Policy	Trained Policy
Successful Episodes	59	3919
Unsuccessful Episodes	1950	510
Total episodes in 100k Steps	1562	4429
Success Rate	2.97%	88.47%
Average Needed Success Step	41	19

Table 3. Visual Search test results

success rate has risen up to 88.47% with an average of 19 steps as shown in Table 3. To analyze the network behavior we tested the policy in the lab environment. The pictures returned from the camera and elaborated by the object detection network were collected together, with the action done and the reward received. Figures 4 and Figure 5 show some successful episode from the trained network. The trained network has learned an important principle: to look at the height at which the object is supposed to be found and then to look around. Figure 5 also shows that sometimes the policy fails because the object detection network fails to correctly classify the target object.
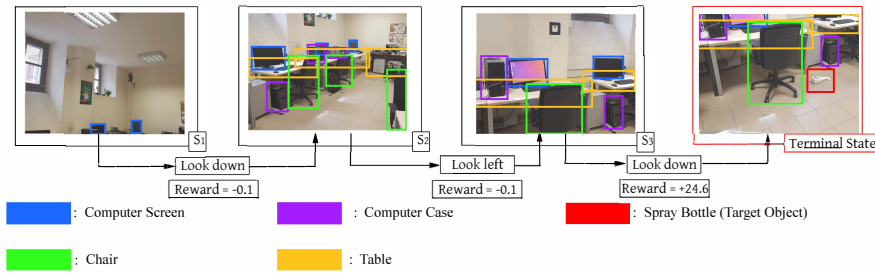

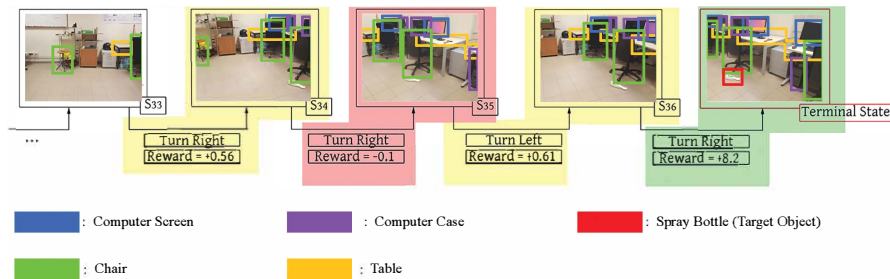**Figure 4.** Successful Episode example of the Visual Search.

Figure 5. Object Detection failure in the Visual Search.

6.4. Experiments with the whole framework

We evaluated two classes of tasks: (1) collect an object (spray bottle, screwdriver, hammer, cup) from the table or from the shelf to a subject; (2) put-away an object (screwdriver, hammer, spanner) in the toolbox. Each experiment in a class has been run 35
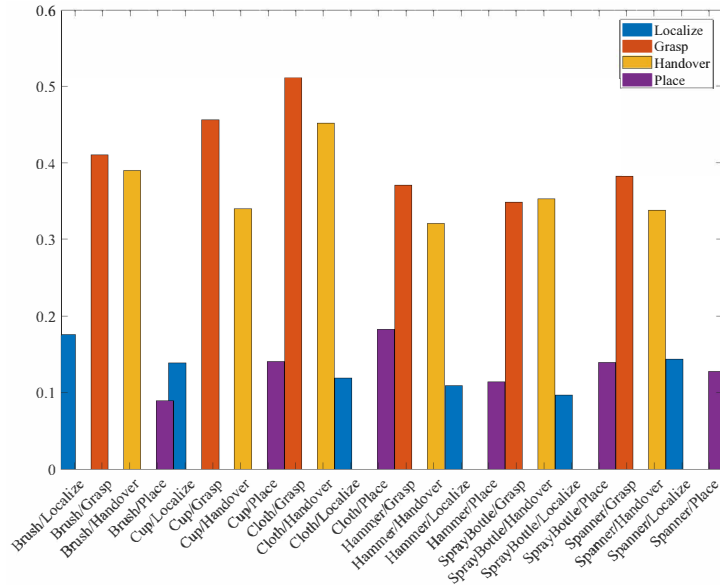
Figure 6. The histogram of action/objects failures.

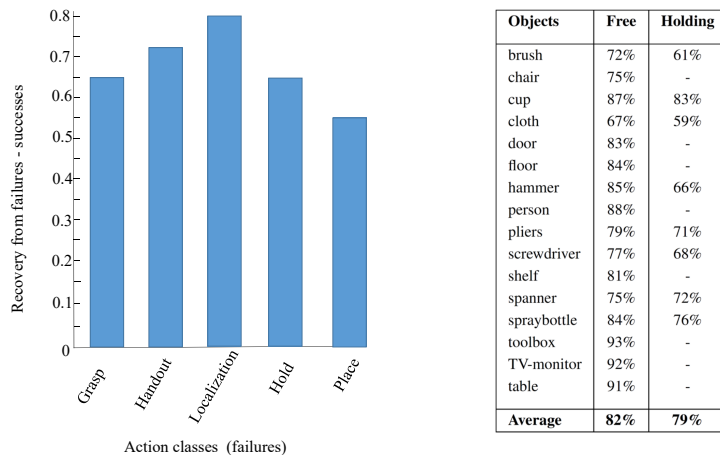| Objects	Free	Holding
brush	72%	61%
chair	75%	-
cup	87%	83%
cloth	67%	59%
door	83%	-
floor	84%	-
hammer	85%	66%
person	88%	-
pliers	79%	71%
screwdriver	77%	68%
shelf	81%	-
spanner	75%	72%
spraybottle	84%	76%
toolbox	93%	-
TV-monitor	92%	-
table	91%	-
Average	**82%**	**79%**

Figure 7. On the left, histogram of the recovery success ratios via the visual search policy. On the right, accuracy of the recognition of objects when handed by the robot or the human and when free.

times. An example is given in Figure 2, despite in a number of circumstances the grasping action has been manually helped, especially with small objects. Table 2 shows the main relations, objects, and actions considered in the tasks. The table on the right side of Figure 7 shows the object detection accuracy achieved by the DCNN models dealing with the free and held objects during the above mentioned experiments.

Failures We examined the number of failures encountered during the execution of the tasks described above. A failure is recorded as soon as the state perceived by the visual stream, via the DCNN, does not match the post conditions of the action executed. The

histogram in Figure 6 shows the probability that a failure is encountered while executing a particular action in relation to the object being involved. We noticed that, as expected, more complex actions like grasping and localizing show a higher probability of failure. Surprisingly, passing an item shows a low failure probability, this is mainly attributed to the high adaptability of the subject involved in the action.

Recovery The visual search policy is employed as soon as a failure is detected in order to find the primary objects involved in the action that failed. On the left side of Figure 7 it is shown the success rate of the recovery via the use of a visual search policy for four different types of actions. Localization and manipulation actions show a higher recovery success rate while recovering from a failed grasping seems to be the most problematic one.

7. Conclusions

In this work, we have proposed a new approach to execution monitoring by placing a strong emphasis on the role of perception to both verify the state of execution and to recover in case of failure. Robot representation relies on DCNNs to detect both objects and relations and on Deep Reinforcement Learning to ensure awareness of the environment. The perception processes are coordinated by the execution monitor, which uses them to either check the realization of the properties required by the planner or to focus the robot attention onto the elements of the scene that are needed to complete the robot task. We have shown that simple tasks can be completed in autonomy and that it is also possible to recover from failures. For future work, we are plan to explore new ways to complete the complex training phase for deep reinforcement learning, by directly learning from the environment in which the robot is operating

8. Acknowledgments

References

[1] V. Ntouskos, F. Pirri, M. Pizzoli, A. Sinha, and B. Cafaro, "Saliency prediction in the coherence theory of attention," *Biologically Inspired Cognitive Architectures*, vol. 5, no. 0, pp. 10 – 28, 2013.

[2] M. Qodseya, M. Sanzari, V. Ntouskos, and F. Pirri, "A3d: A device for studying gaze in 3d," in *The European Conference on Computer Vision (ECCV) Workshops*, 2016.

[3] J. McCarthy and P. J. Hayes, "Some philosophical problems from the standpoint of artificial intelligence," in *Readings in artificial intelligence*, 1981, pp. 431–450.

[4] F. Pirri and R. Reiter, "Some contributions to the metatheory of the situation calculus," *Journal of the ACM (JACM)*, vol. 46, no. 3, pp. 325–361, 1999.

[5] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "Pddl-the planning domain definition language," 1998.

[6] M. Helmert, "Concise finite-domain representations for pddl planning tasks," *Artif. Intel.*, vol. 173, no. 5-6, pp. 503–535, 2009.

[7] M. Cashmore, M. Fox, D. Long, D. Magazzeni, B. Ridder, A. Carrera, N. Palomeras, N. Hurtós, and M. Carreras, "Rosplan: Planning in the robot operating system." in *ICAPS*, 2015, pp. 333–341.

[8] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *NIPS*, 2015, pp. 91–99.

[9] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *Nature*, pp. 529–533, 2015.

[10] J. Borenstein and Y. Koren, "Real-time obstacle avoidance for fast mobile robots," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 19, no. 5, pp. 1179–1187, 1989.

[11] ——, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE transactions on robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[12] D. Kim and R. Nevatia, "Symbolic navigation with a generic map," *Autonomous Robots*, vol. 6, no. 1, pp. 69–88, 1999.

[13] G. Oriolo, M. Vendittelli, and G. Ulivi, "On-line map building and navigation for autonomous mobile robots," in *ICRA*, vol. 3.   IEEE, 1995, pp. 2900–2906.

[14] R. T. Whitaker, "A level-set approach to 3d reconstruction from range data," *International journal of computer vision*, vol. 29, no. 3, pp. 203–231, 1998.

[15] I. Kruijff-Korbayová, L. Freda, M. Gianni, V. Ntouskos, V. Hlaváč, V. Kubelka, E. Zimmermann, H. Surmann, K. Dulic, W. Rottner *et al.*, "Deployment of ground and aerial robots in earthquake-struck amatrice in italy (brief report)," in *Safety, Security, and Rescue Robotics (SSRR), 2016 IEEE International Symposium on*.   IEEE, 2016, pp. 278–279.

[16] R. Sim and J. J. Little, "Autonomous vision-based exploration and mapping using hybrid maps and rao-blackwellised particle filters," in *IROS*.   IEEE, 2006, pp. 2082–2089.

[17] D. Wooden, "A guide to vision-based map building," *IEEE Robotics & Automation Magazine*, vol. 13, no. 2, pp. 94–98, 2006.

[18] A. J. Davison, "Real-time simultaneous localisation and mapping with a single camera," in *null*.   IEEE, 2003, p. 1403.

[19] M. Tomono, "3d object map building using dense object models with sift-based recognition features," in *IROS*, 2006, pp. 1885–1890.

[20] K. Kidono, J. Miura, and Y. Shirai, "Autonomous visual navigation of a mobile robot using a human-guided experience," *Robotics and Autonomous Systems*, vol. 40, no. 2, pp. 121–130, 2002.

[21] E. Royer, J. Bom, M. Dhome, B. Thuilot, M. Lhuillier, and F. Marmoiton, "Outdoor autonomous navigation using monocular vision," in *IROS*, 2005, pp. 1253–1258.

[22] H. Haddad, M. Khatib, S. Lacroix, and R. Chatila, "Reactive navigation in outdoor environments using potential fields," in *ICRA*, vol. 2, 1998, pp. 1232–1237.

[23] S. Lenser and M. Veloso, "Visual sonar: Fast obstacle avoidance using monocular vision," in *IROS*, vol. 1.   IEEE, 2003, pp. 886–891.

[24] A. Remazeilles, F. Chaumette, and P. Gros, "Robot motion control from a visual memory," in *ICRA*, vol. 5.   IEEE, 2004, pp. 4695–4700.

[25] P. Saeedi, P. D. Lawrence, and D. G. Lowe, "Vision-based 3d trajectory tracking for unknown environments," *IEEE transactions on robotics*, vol. 22, no. 1, pp. 119–136, 2006.

[26] D. P. Bertsekas and J. N. Tsitsiklis, "Neuro-dynamic programming: an overview," in *Decision and Control*, vol. 1, 1995, pp. 560–564.

[27] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction (Second Edition)*, 1998, 2017, vol. 1, no. 1.

[28] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *ICML*, 2016, pp. 1928–1937.

[29] S. Levine, C. Finn, T. Darrell, and P. Abbeel, "End-to-end training of deep visuomotor policies," *Journal of Machine Learning Research*, vol. 17, no. 39, pp. 1–40, 2016.

[30] R. Mottaghi, H. Bagherinezhad, M. Rastegari, and A. Farhadi, "Newtonian scene understanding: Unfolding the dynamics of objects in static images," in *CVPR*, 2016, pp. 3521–3529.

[31] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, L. Fei-Fei, and A. Farhadi, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*.   IEEE, 2017, pp. 3357–3364.

[32] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IROS*, 2013, pp. 1321–1326.

[33] O. Pettersson, "Execution monitoring in robotics: A survey," *Robotics and Autonomous Systems*, vol. 53, no. 2, pp. 73–88, 2005.

[34] F. Ingrand and M. Ghallab, "Deliberation for autonomous robots: A survey," *Artificial Intelligence*, vol.

247, pp. 10–44, 2017.

[35] R. J. Doyle, D. J. Atkinson, and R. S. Doshi, "Generating perception requests and expectations to verify the execution of plans," in *AAAI*, 1986, pp. 81–88.

[36] D. E. Wilkins, "Recovering from execution errors in sipe," *Computational. Intelligence*, vol. 1, no. 1, pp. 33–45, 1985.

[37] S. Guadarrama, L. Riano, D. Golland, D. Go, Y. Jia, D. Klein, P. Abbeel, T. Darrell *et al.*, "Grounding spatial relations for human-robot interaction," in *IROS*, 2013, pp. 1640–1647.

[38] C. Lu, R. Krishna, M. Bernstein, and L. Fei-Fei, "Visual relationship detection with language priors," in *ECCV*, 2016, pp. 852–869.

[39] A. Das, H. Agrawal, C. L. Zitnick, D. Parikh, and D. Batra, "Human attention in visual question answering: Do humans and deep networks look at the same regions?" *arXiv:1606.03556, preprint*, 2016.

[40] R. E. Fikes, "Monitored execution of robot plans produced by strips," SRI, Tech. Rep., 1971.

[41] N. J. Nilsson, *A hierarchical robot planning and execution system*. SRI, 1973.

[42] C. Boutilier, R. Reiter, M. Soutchanski, S. Thrun *et al.*, "Decision-theoretic, high-level agent programming in the situation calculus," in *AAAI/IAAI*, 2000, pp. 355–362.

[43] A. Hornung, S. Böttcher, J. Schlagenhauf, C. Dornhege, A. Hertle, and M. Bennewitz, "Mobile manipulation in cluttered environments with humanoids: Integrated perception, task planning, and action execution," in *Humanoids*, 2014, pp. 773–778.

[44] J. P. Mendoza, M. Veloso, and R. Simmons, "Plan execution monitoring through detection of unmet expectations about action outcomes," in *ICRA*, 2015, pp. 3247–3252.

[45] Y. Zhu, D. Gordon, E. Kolve, D. Fox, L. Fei-Fei, A. Gupta, R. Mottaghi, and A. Farhadi, "Visual semantic planning using deep successor representations," *CoRR*, 2017.

[46] M. Helmert, "The fast downward planning system." *JAIR*, vol. 26, pp. 191–246, 2006.

[47] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *NIPS*, 2012, pp. 1097–1105.

[48] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[49] A. G. Cohn, B. Bennett, J. Gooday, and N. M. Gotts, "Representing and reasoning with qualitative spatial relations about regions," in *Spatial and temporal reasoning*. Springer, 1997, pp. 97–134.

[50] L. Kunze, C. Burbridge, and N. Hawes, "Bootstrapping probabilistic models of qualitative spatial relations for active visual object search," in *AAAI Spring Symposium*, 2014, pp. 24–26.

[51] S. Edelkamp and J. Hoffmann, "Pddl2. 2: The language for the classical part of the 4th international planning competition," *ICAPS*, 2004.

[52] E. Rohmer, S. P. Singh, and M. Freese, "V-rep: A versatile and scalable robot simulation framework," in *IROS*. IEEE, 2013, pp. 1321–1326.