Deep Execution Monitor for Robot Assistive Tasks

Lorenzo Mauro, Edoardo Alati, Marta Sanzari, Valsamis Ntouskos, Gianluca Massimiani, and Fiora Pirri(✉)

Alcor Lab, Diag, University of Rome La Sapienza, Rome, Italy
pirri@dis.uniroma1.it

Abstract. We consider a novel approach to high-level robot task execution for a robot assistive task. In this work we explore the problem of learning to predict the next subtask by introducing a deep model for both sequencing goals and for visually evaluating the state of a task. We show that deep learning for monitoring robot tasks execution very well supports the interconnection between task-level planning and robot operations. These solutions can also cope with the natural non-determinism of the execution monitor. We show that a deep execution monitor leverages robot performance. We measure the improvement taking into account some robot helping tasks performed at a warehouse.

1 Introduction

In this paper, we present a novel approach to model high-level robot task execution. An execution monitor is a real-time decision process, which amounts to choosing at each step of the execution the next subtask and deciding whether the current task succeeded or failed [12,34]. A real-time execution monitor involves plan inference, verification of the current robot state, and choice of next goal state.

Several authors, in the planning community, have explored hierarchical task networks (HTN) (see for instance [10]) and hierarchical goal networks (HGN) (see for example [44]) to provide a way of sequencing a suitable decision process [2] at the correct level. However, both HTN and HGN require that these decisions are stacked a priory in the network, putting on the designer the burden to provide a task decomposition, for each task.

In this paper we overcome these difficulties by integrate two deep models to predict next state choice. The first model is a DCNN, identifying the objects in the scene and supporting recognition of relations holding at the current execution time. The second is a sequence to sequence model (*seq2seq*) [46] with attention [3,30,31] inferring a plausible next robot world-state given the current world-state. The interplay between the two models and classical planning grounds the specification of a world-state. The execution monitor manages the interaction amid the models at execution time. This is a very preliminary contribution, considering only the high-level robot decisions. Direct robot control is managed by state charts [48].

L. Leal-Taixé and S. Roth (Eds.): ECCV 2018 Workshops, LNCS 11134, pp. 158–175, 2019.
https://doi.org/10.1007/978-3-030-11024-6_11

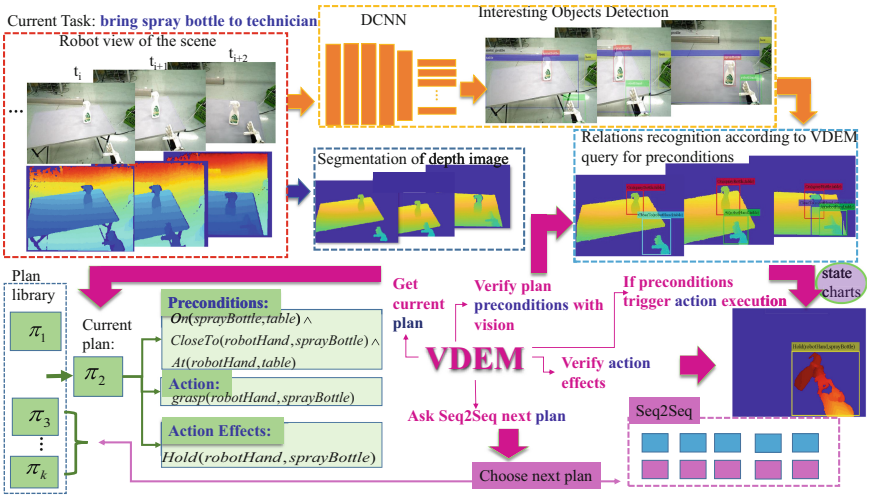

Fig. 1. The schema above presents the flow of information managed by the deep execution monitor (VDEM) for the task *bring the spray bottle to the technician*. While the robot observes the scene, the state is built by the detected relations, restricted according to what is required by the current planner. The VDEM queries the vision system to both verify the preconditions for action to be executed, and the realization of the action effects. A plan library (see e.g. [22] for a reference) provides background knowledge in a symbolic language. The seq2seq model learns to predict goal-states, according to the specific task and current state, and it is invoked by the VDEM whenever a new goal state is required.

Main Idea and Contribution. In this paper we address a *vision-based deep execution monitor* (VDEM) for robot tasks. The main idea is the introduction of a robot learning model to predict the next goal from the current one, verifying the preconditions and effects of the currently executed action. Preconditions and effects are specified in a symbolic language. Whether they hold or not at a state can be determined by the robot vision. The robot monitors the states of its execution by linking the symbolic language with the vision interpretation, such that the objects in the scene are the terms of the symbolic language, and the relations are the predicates. The next goal state is inferred by associating to each goal described by some plan in the plan library, the goal which is the most plausible successor state. Therefore, given that X is a goal descriptions, and Y is the next goal description, the seq2seq model infers $P(Y|X)$. A description is formed by the predicates and terms verified by vision, which form the current robot world-state. The seq2seq model is formed by an encoder fed by token of the symbolic language, an attention mechanism that pairs each description with the task, which is a sort of memory of the goals concerned with such a task, and a decoder, which infers the most likely successor state.

Though recent approaches [1,27,33,53,54] have considered vision based execution, our approach is novel in combining vision based execution with next

step prediction, binding the planning symbolic languages with visual instances. The binding allows the execution monitor to generate a state merging vision and planning feedbacks. Furthermore, the approach provides both depth and location for relations recognition to cope with the task dynamics.

We tested the framework at a warehouse with a humanoid robot, described in the experiment section, see Sect. 6. We provide ablation of the execution monitor functionalities to experiment the robotic performance and the advantages of the proposed vision based deep execution monitor.

2 Related Work

Vision Based Robot Execution. The earliest definitions of execution monitoring in nondeterministic environments were introduced by [12,34]. Since then an extraordinary amount of research has been done to address the nondeterministic response of the environment to robot actions. Several definitions of execution monitoring are reported in [38]. For high-level robot tasks, a review of these efforts is given in [24]. The role of perception in execution monitoring was already foreseen in the work of [9]. Likewise, recovery from errors that could occur at execution time was already faced by [50]. Despite this foresight, the difficulties in dealing with scene recognition have directed the effort toward models managing the effects of actions such as [4,47], allowing to execute actions in partially observable environments, similarly as in [5,13,15]. On the other hand, different approaches have studied learning policies for planning as in [28] and also for decision making, in partially observable domains [18]. Vision based planning has been studied in [54]. These approaches did not consider execution monitor and the duality between perception and learning. Likewise despite facing the integration of observations in high-level monitor [23,32] did not use perception for verifying the current state, which is crucial for both monitoring and further decision learning.

Relations Recognition in Videos. Relations in videos dynamically change, in the sense that the configuration of the involved objects is altered according to the robot vantage points. Recently a number of approaches have studied spatial relations and their grounding, such as [8,16,29,42]. Among them, only [16] faces the problem from the point of view of robot task execution. There are also recent contributions concerned with human activity recognition and human-objects interaction studying the problem regarding human dynamics such as [36,43,49,51,53], here in particular for container and containee relations. Although these latter approaches consider both videos and 3D objects they do not face general relations amid objects. The main difficulty seems that of recognizing relations in a complex scenario without overloading the perceptual scene, namely what the robot has to infer from the scene. To this end, and also to maintain real time execution, we rely on the execution monitor querying the visual interpretation at each current state about the existence of specific relations. Relations computation exploits approximate depth estimation within the object bounding box. To obtain this good performance we use DCNN trained

on different classes of models, which are retrieved by the execution monitor, and the active features of the recognized objects, involved in the relation, to estimate the object depth.

Sequence to Sequence Models and Next Step Prediction. Sequence to sequence models (seq2seq) [46] are made of two networks, one for processing the input and a second network generating the output, in an encoder-decoder configuration. They have shown an excellent performance in several sequence prediction problems especially in machine translation, image captioning and even in high-level decision processing. In planning problems, [25] have proposed recently QMDP-Net combining POMDP and LSTM to obtain a neural network architecture under partial observability. They applied their model to 2D grids to cope with 2D path planning. While we do not know of other approaches to execution monitor and high-level planning with seq2seq architecture, LSTM have been used for path planning, while [17] show that their CMP approach to navigation outperforms LSTM. The introduction of an attention mechanism [3,30,31] has improved sequence to sequence models essentially for neural translation and also for image captioning. Attention mechanisms for robot execution have been studied in [35], and here in particular we base our approach on the attention mechanism to exploit the task context.

The problem of predicting next step has not yet faced with seq2seq models. An approach to driving the focus of attention to the next useful object has been introduced by [14]. On the other hand [7] have designed a new public database including annotations also for the next action, which is relevant for execution monitor, where prediction of next state can take advantage of surrounding people actions.

3 Deep Execution Monitoring

In this section we give an overview of the execution monitor (VDEM) altogether, providing at the end of the section the main algorithms.

Preliminaries on the Environment and the Tasks. We consider robot assistive tasks related to maintenance activities at a warehouse. The robot language $\mathcal{L}$ is defined by atoms, which are formed by predicates taking terms as arguments. Terms, can be either variables or constants, and they are instantiated by the objects that the robot identify in the environment. Likewise, predicates are the relations the robot is able to identify in the environment. Predicates take also indexed terms denoting the frame as arguments. The robot language $\mathcal{L}$ is extended with meta terms denoting tasks, hence $\mathcal{L} \cup \{T\}_{i=1}^{K}$. Where T_i is a sentence specifying a task. Tasks sentences are, for example, *pass the brush and the cloth to the technician*, *help the technician to hold the guard*. Therefore a task sentence is expressed in natural language, and the execution of a task requires a number of actions to be performed, for both controlling the robot visual process and the robot motion. These actions are specified by plans collected into the plan library.

$$
\begin{aligned}
&VisionOn(robot, t_0) \wedge Free(robot\_hand, t_0),\\
&Detected(brush, t_1) \wedge Detected(ladder, t_1) \wedge On(brush, ladder, t_1),\\
&At(robot, ladder, t_2) \wedge Holding(robot\_hand, brush, t_3),\\
&Detected(technician, t_3) \wedge CloseTo(robot, technician, t_3),\\
&Detected(technician\_hand, t_4) \wedge Holding(technician\_hand, brush, t_4) \wedge\\
&Free(robot\_hand, t_4)
\end{aligned}
\tag{1}
$$

Plans and Plan Library. Let us assume that the execution of a task requires the execution of n plans, where each plan specifies a number of actions.

A plan library is a collection of plans. In a plan library, each plan defines all the actions needed to achieve a goal of a part of a task, by a suitable axiomatization. For example, to grasp an object the robot needs to be close to the object, which is a partial task.

A plan is formed by a *problem* specifying the initial state and a goal, defined in the propositional Planning Domain Definition Language (*pddl*), and by a *domain* providing an axiomatization of actions, which is first-order *pddl* with types and equality. Plans, therefore, form the background knowledge of the robot about what is needed for an action to be performed.

A state s, with respect to an action a, is formed by either the preconditions for executing a or by the effects of a execution. When s is a goal state this is the goal of the *problem*. To simplify the presentation here we assume that the preconditions and effects are conjunctions of binary or unary atoms, and a state can be reduced to $s = \bigwedge_i R_i(\nu_{i1}, \ldots, \nu_{ik}, t)$, where $(\nu_{i1}, \ldots, \nu_{ik})$, $k >= 1$, are ground terms. Plan inference amounts to deduce the goal of the problem, given the starting state. A goal of a problem is, for example, $At(robotHand, table)$, requiring to search where the table is, and reaching it.

To facilitate inference, the set of actions axiomatized in a plan domain are partitioned into actions that affect the state of the world (like moving objects around) and ecological actions, which affect only the state of the robot. Ecological actions are for example *search*, $verify\_vision$, *turn_head*, *look_up*, *look_down*. A plan is formed by at most a single action that can affect the world and by a number of ecological actions. This allows to partition the terms of the plan into terms denoting the world, with their types hierarchy, and terms related to the robot representation, requiring appropriate measures, for vision and motion control.

The plan library is the collection of all plans needed for the assistive tasks and it is generated together with the maintenance experts to cope with the foreseen assistive tasks, hence the hypothesis is that: *for all foreseen tasks there exists a sequence of goals factoring them.*

Task Factorization. Given a task, factorization amounts to decompose the task into plans, which are supposed to belong to the plan library. Task factorization is crucial for a number of issues. It avoids useless combinations of unrelated groups of objects, it limits the inference of a goal just to the involved objects, it ensures a high flexibility in robot execution, and allows to easily recover from failures. A top down factorization, such as HTN [10] or HGN [44], might be too costly

to be achieved in real-time, and also might not be able to take care of the state resulting after the execution of the n-th plan. An incongruence would require, in fact, to search backward for a previous reliable state.

The solution we propose here is to learn to predict the next goal, given the current goal state. In this way, given a task and its initial state goal, a successor state goal can be predicted after the success of the current goal state is confirmed.

Execution. The execution monitor loops over the following operations: (1) get the next goal; (2) identify the plan for the given goal; (3) forward the inferred actions to the *state charts* [48], as soon as the preconditions are satisfied, according to the vision process; (4) verify the effect of the inferred actions; (5) if the current plan goal is obtained ask the seq2seq model to infer next goal and go to (2) else continue with the current plan. The execution, illustrated in Fig. 1, is resumed in Algorithms 1, 2 and 3.

Algorithm 1. Vision based deep execution monitor

Input: Current task $\mathcal{T}$, plan Π, current state s, plan library Lib_{Π}
Output: end-task $\mathcal{T}$

while *not end-task* **do**
  if $\Pi \neq \emptyset$ *and* $s = \bigwedge_i^N R_i(\boldsymbol{\nu}, a)$ **then**
    $(\alpha, bounding\_box, depth) := query\_vision(s)$
    if $\alpha = True$ **then**
      if s *goal of* Π **then**
        $\Pi := query\_seq2seq(\mathcal{T}, s, Lib_{\Pi})$
      else
        Continue Π
    else
      Return end-task $\mathcal{T}$
  if $\Pi = \emptyset$ *and* $s = start(\mathcal{T}, s_0)$ **then**
    $\Pi := query\_seq2seq(\mathcal{T}, s, Lib_{\Pi})$
  if $\Pi \neq \emptyset$ *and* $s = goal(\mathcal{T})$ **then**
    Return *end-task*$\mathcal{T}$

Algorithm 2. Query seq2seq

Input: seq2seq model, plan library Lib_{Π}, current task $\mathcal{T}$, current state s
Output: subplan Π

Compute seq2seq output with input $(\mathcal{T}, s)$ and choose the goal state s_g maximizing: $p(s_g|s, \mathcal{T})$
Search in Lib_{Π} best match Π with $\boldsymbol{\nu}, \{R\}_i^M$, mentioned in s_g, goal of Π
Return Π

Algorithm 3. Query Vision

Input: video-stream at current time lapse $t_i{:}t_{i+n}$, DCNN models $\mathcal{M}_1, \ldots, \mathcal{M}_k$, current state s, thresholds μ, τ
Output: Boolean

$s = \bigwedge_i^N R_i(\boldsymbol{\nu}, a)$
Compute bounding boxes in video-stream using models $\mathcal{M}_1, \ldots, \mathcal{M}_k$
Segment objects in depth images in video-stream for each $\nu \in \boldsymbol{\nu}$ (Sect. 4)
if $confidence(\boldsymbol{\nu}) > \mu$ **then**
    Compute $R_i, i{=}1, \ldots N$
else
    while *time lapse* $T < \tau$ **do**
        Search for missed $\nu \in \boldsymbol{\nu}$
if $T \leq \tau$ **then**
    Return $True$, bounding box for $\boldsymbol{\nu}$, depth
else
    Return $(False, \{\}, -1)$

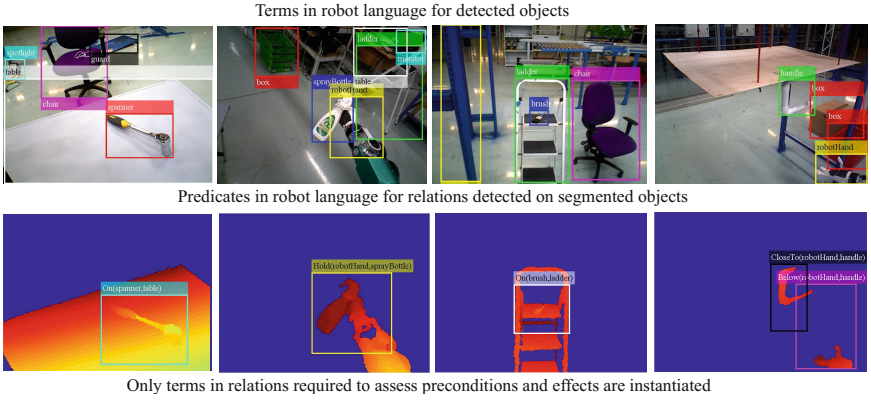

Fig. 2. Objects detected in the scene observed by the robot, while it is executing its task, are terms of the robot language. Only relations needed by the planner and queried by the VDEM to vision are considered and instantiated with detected terms.

Note, therefore, that according to the algorithms the seq2seq model is called only if the current state is either a goal of the current plan, just concluded, or the start state of a task. Note that in case of failure a new task $\mathcal{T}'$ can be recovered from last successful state.

4 Vision Interpretation

As highlighted in the previous section, the execution monitor gets from the current plan the state to be verified in the form of a conjunction of atoms, and

query the *vision interpretation* to assess if the current state holds. An example is shown in Fig. 2.

To detect both objects and relations we have trained Faster R-CNN [40] on ImageNet [26], Pascal-VOC [11] dataset, and with images taken on site. We have trained 5 models to increase accuracy, obtaining a detection accuracy above 0.8. The good accuracy is also due to a confidence value measured on a batch of 10 images, taken at 30 fps, simply computing the most common value in the batch and returning the sampling mean accuracy for that object.

The model is called according to the state request. For example, if $On(brush, ladder)$ is requested from the plan state, the execution monitor asks the vision interpretation to call the models for *brush* and *ladder* first and for *On* relation for all the found terms, after. Though the main difficult part is searching the objects and the relations, we shall not discuss this here.

To infer spatial-relations we have introduced a look-up table for the definition of each relation of interest for the assistive task. The relations require the depth within the bounding boxes of each object denoted by the queried terms. Depth is crucial in the warehouse environment, because objects at different distances appear within the bounding box of an object, as shown in the first image of Fig. 2. There is, indeed a tradeoff between using MaskRCNN [19] and Faster. With Mask we have the depth segmentation immediately, by projecting the mask on the RGBD image, but objects of the warehouse need to be manually segmented. On the other hand Faster using Imagenet offers a huge amount of data, but depth needs to be obtained. In this version of our work we considered Faster R-CNN [40] and did a local segmentation by clustering.

We have first trained a non-parametric Bayes model to determine for each object of interest the number of feature classes. To this end, we estimate a statistics of the active features with dimension $38 \times 50 \times 512$, taken before the last pooling layer, at each pixel inside the recognized object bounding box (here we are referring to VGG, though we have considered also ZF, see [45,52]). Once the number of classes for each object is established we have trained a normal mixture model on the selected feature classes for each object, resulting in a probability map that a pixel belongs to the specific class of the object.

During execution, as the object is known, we choose the learned parameters for the model to estimate a probability that the pixel in the bounding box belongs to the object. The distribution on the bounding box is projected onto the depth map and a ball-tree is built using only the pixels with a probability greater than a threshold (we used 0.7). Using unsupervised nearest neighbor, checking the distance, a resulting segmentation is sufficiently accurate for the task at hand. Depth is considered relative to the robot-camera. See Fig. 2.

Having the depth, the relations are established, a reference are the spatial relations based on the connection calculus [6], though here distance and depth play a primary role, which are not considered in [6]. To establish the relation amid $n \leq 3$ objects we consider the distance first (within a moving visual cone with vertex the center of projection) and further the other properties consistently with the connection calculus and its 3D extensions (see [41]). See Sect. 6 for an overview of the relations and the accuracy on the recognition.

5 The seq2seq Architecture for Deep Monitoring

As gathered in previous sections the robot is given a *high level task* specified by a sentence, such as *help the technician to support the guard*. The objective, here, is to find the sequence of plans, in the plan library, ensuring the task to succeed. We have seen that relevant steps to this end are the definition of states, which are conjunctions of literals, inferred by the plans and verified by the vision interpretation to hold before or after the robot executes an action.

We have also introduced the notion of *goal state* as the state of a plan problem in which the goal holds. When a goal state is achieved, task execution requires to predict the next goal, in so ensuring to progress in the accomplishment of the assigned task.

We show that a sequence to sequence architecture is effective for mapping a current goal state, expressed as a conjunction of literals into a new goal state, where it is intended (see Sect. 3) that the predicted goal is a goal of some plan in the plan library.

A sequence to sequence system mapping a state of the robot into a new state is a network modeling the conditional probability $p(Y|X)$ of mapping a source state $x_1, \ldots, x_n$ into a target state $y_1, \ldots, y_m$. The encoder-decoder is made of two elements: an encoder which transform the source into a representation S and a decoder generating one target item at a time, so that the conditional probability is [30]:

$$\log p(y|x) = \sum_{j=1}^{m} \log p(y_j|y_1, \ldots y_{j-1}, S) \tag{2}$$

We define an input state as a set of tokens belonging to the extended robot language $\mathcal{L} \cup \{T\}_{i=1}^{K}$ with $\mathcal{L}$ the language including terms (denoting objects in the scene) and predicates, denoting relations in the scene, and with T_i a task sentence. Given an input state $\mathbf{s} = (u_1, \ldots, u_n)$, this is initially mapped into a low dimensional vector $\mathbf{x}$. With $\mathbf{x} = W\mathbf{s}$, where W is the embedding matrix, which is fine-tuned during the training of the seq2seq model.

Given the encoded sequence $\mathbf{x}$ and the true output sequence $\mathbf{y}$, encoded as well, the goal is to learn how they match in order to predict, at inference time, the correct $\mathbf{y}'$ given the input $\mathbf{s}'$.

Attention [3,39] has become, recently a hot topic for measuring similarities and dissimilarities between input and output sequences, according to the specific objective of the mapping. For example, while in neural machine translation (NMT) alignment can be quite relevant, in the case of a new state prediction alignment is not really relevant while the task at hand it is, since a new goal is looked for while a specific task is executed. In general attention computes the relevance of each token in the encoded sequence with respect to the true encoded sequence $\mathbf{y}$ via a function $\varphi(x_i, \mathbf{y})$, which returns a score whose distribution, via a softmax function, determines the relevance of each token in $\mathbf{x}$ with respect to

the encoded output $\mathbf{y}$. This can be expressed as the expectation of a token given the distribution induced by the score:

$$\sum p(z = i|\mathbf{x}, \mathbf{y})x_i \tag{3}$$

Where $p(z = i|\mathbf{x}, \mathbf{y})$ is the distribution induced by the softmax applied to the score given to each token x_i, with z the indicator of the encoded input tokens. In the literature different score function have been proposed, e.g. additive or multiplicative [3,31]:

$$\begin{array}{ll} \varphi(x_i, \mathbf{y}) = w^\top \sigma(W^{(1)}x_i + W^{(2)}\mathbf{y}) & \text{(additive)} \\ \varphi(x_i, \mathbf{y}) = \langle W^{(1)}x_i + W^{(2)}\mathbf{y})\rangle & \text{(multiplicative)} \end{array} \tag{4}$$

Where $W^{(i)}$ are learned weights. In our case we have two basic structures, the task sentence and the sequence of atoms. We have also specific separators: for the atoms $\langle eoa\rangle$, for the end of task sequence $\langle ets\rangle$ and for the end of the state description $\langle eos\rangle$. The attention mechanism required here needs to score the compatibility of each atom, namely a subsequence of the output sequence $\mathbf{y}$, with the task and with each input token. For example we expect that in the context of the task *pass the brush to technician* the output subsequence *Hold, technician, brush* has an encoding similar to *Hold, robot, brush* while this is not true in the context of the task *help the technician to hold the guard*, in which the correct subsequence would be *On, table, brush*.

To this end we formulate the input and output embedded sequences in terms of subsequences $\boldsymbol{\tau}^{\mathbf{x}} = (\tau_1^{\mathbf{x}}, \ldots \tau_K^{\mathbf{x}})$ and $\boldsymbol{\tau}^{\mathbf{y}} = (\tau_1^{\mathbf{y}}, \ldots, \tau_m^{\mathbf{y}})$, using both the $\langle eoa\rangle$ and $\langle ets\rangle$, in order to compute the weights of the attention mechanism. Weights are learned by a dense layer taking as input the concatenation of the previous predicted $\tau_{t-1}^{\mathbf{y}}$, from the decoder, the embedded task, which is always $\tau_1^{\mathbf{y}}$, and the previous hidden state of the decoder. The weights for each τ form a matrix, hence we obtain:

$$\varphi(\tau_i^{\mathbf{x}}, \boldsymbol{\tau}^{\mathbf{y}}) = W^\top \sigma(W^{(1)}\tau_i^{\mathbf{x}} + W^{(2)}\boldsymbol{\tau}^{\mathbf{y}}) \tag{5}$$

Finally, following the softmax application, we have a prediction of the importance of each token of the encoder according to the 'context' atom and according to the task. Thus we have $p(\mathbf{z} = i|\tau_i^{\mathbf{x}}, x_i, \boldsymbol{\tau}^{\mathbf{y}})$, which is a vector of the dimension of $\tau_i^{\mathbf{x}}$. This is the probability that a subsequence, namely an atom, is relevant for the current task and the predicted sequence. Then the output is obtained as the expectation over all the atoms:

$$s = \sum p(\mathbf{z} = i|\tau_i^{\mathbf{x}}, x_i, \boldsymbol{\tau}^{\mathbf{y}})\tau_i^{\mathbf{x}} \tag{6}$$

We can note that in (6) also words are made pivotal, since the probability is a vector. For example, in case the task is *bring the brush to the technician*, the *brush* is a pivotal word, and the context will most probably imply that the mapping of the predicate $Hold$ is from $Hold(robotHand, brush)$ to

$Hold(technician, brush)$ and the task sentence triggers attention to both the term *brush* and the relation *Hold*.

Data Collection for the seq2seq Model. The robot vocabulary is formed by 18 unary predicates, 13 binary predicates and 42 terms. We build the Herbrand Universe from predicates and terms, obtaining a language of more than 35k atoms. Elements of the language are illustrated in Fig. 3.

A number of the atoms does not respect the type hierarchy, which is defined in *pddl*, therefore are deleted from the language. Finally we have grown all the goal states provided in the plan library up to 20k states.

Some of the predicates from the whole set are listed in Table 1, detailing the recognition ability of the vision interpretation. We should note that a number of predicates concerns the robot inner state, such as for example *VisionOn* or the head and body positions, which are not listed in Table 1.

6 Experiments and Results

Experiments Setup. Experiments have been done at a customer fulfillment center warehouse, under different conditions in order to test different aspects of the model. To begin with, all experiments have been performed with a humanoid robot, created at the High Performance Humanoid Technologies Lab (H^2T). The robot has two 8-DoF torque-controlled arms, two 6-DoF wrists, two underactuated 5-finger hands, a holonomic mobile base and 2-DoF head with two stereo camera systems and an RGB-D sensor. The Asus Xtion PRO live RGB-D camera has been mounted on the robot head to provide the video stream to the visual system and ran the *VDEM* on two of the computers mounted on the robot. We dedicated one to the planning and management of the execution and another one, equipped with an Nvidia Titan GPU, to ran the *visual stream*. Robot control is interfaced with the VDEM via the state charts [48].

Results for the Visual Stream. We trained the visual stream system using images taken from the ImageNet dataset, Pascal VOC, as well as images collected inside the warehouse by the RGB-D camera of the robot. Most of the objects, indeed, are specific of the warehouse and cannot be found in public databases. The relations considered were essentially those relevant to the maintenance tasks (see Table 1). To train the DCNN models we split the set of images in training and validation sets with a proportion of 80%–20%. We trained a number of different models for the different types of objects, and we performed 70000 training iterations for each model on a PC equipped with 2 GPUs. The visual stream has been tested under different conditions, in a standalone tests and during the execution of different tasks. The accuracy has been computed considering the batch of 10 images, accuracy of objects recognition and relations recognition is shown in Table 1, evaluating accuracy and ablation study specifically for relations.

Mean average precision mAP for object detection is 0.87 and localization in depth is 0.98 accurate up to 3 m.

Table 1. Accuracy and ablation study of predicate grounding. **Legend:** *BB*: bounding boxes only, *masks*: segmentations masks only, *no prior*: without use of distance *no shape*: without use of shape properties *no depth*: without use of depth.

Predicate	Full	BB	Masks	No prior	No shape	No depth
CloseTo	89%	79%	82%	79%	72%	49%
Found	95%	85%	81%	85%	80%	61%
Free	91%	86%	91%	86%	83%	68%
Hold	88%	72%	82%	75%	74%	56%
Inside	87%	64%	78%	71%	65%	57%
On	96%	77%	85%	79%	78%	65%
InFront	95%	81%	85%	84%	83%	63%
Left	95%	81%	88%	85%	86%	72%
Right	91%	79%	88%	79%	80%	61%
Under	88%	76%	69%	79%	76%	59%
Behind	81%	78%	78%	76%	79%	61%
Clear	82%	75%	80%	73%	73%	60%
Empty	83%	72%	78%	79%	68%	61%
Average	**89%**	**77%**	**83%**	**79%**	**77%**	**62%**

Results of the seq2seq. We used for the seq2seq network the encoder decoder structure with LSTM [21], in particular a multilayer bidirectional LSTM for the encoder. The maximum input sentence length is set to 17 predicates and a task, which is equivalent to 72 words among relations and terms. The embedding layer transforms the index encoding of every word in the input into a vector of size 20, the encoder then uses a bidirectional LSTM and an LSTM to transform the input question in a vector of size 10. This vector is repeated 3 times, as the length of output sentence and then it is fed to the decoder network. A fully connected layer is then applied to every time sequence returned and then it is passed to a softmax activation layer. The attention function is modeled by a fully connected two layers network.

The seq2seq training uses the Categorical Cross Entropy loss and Adam as an optimizer using batches of 5 sequences for a total of 100 epochs. The total size of the dataset is of 20 thousand sequence pairs.

The accuracy, calculated as the percentage of correct prediction made on a test set extracted from the dataset is used to evaluate the training results. The measurement is done under three different hypotheses. First we considered only the best combination, then we considered the first three combinations, randomly changing the length of the input sequences, finally we considered the accuracy under the local attention model. As shown in Fig. 3 we vary the number of predicates from one to nineteen, which is equivalent to a sequence length varying from 4 to 72 considering both relations and terms. It is possible to see

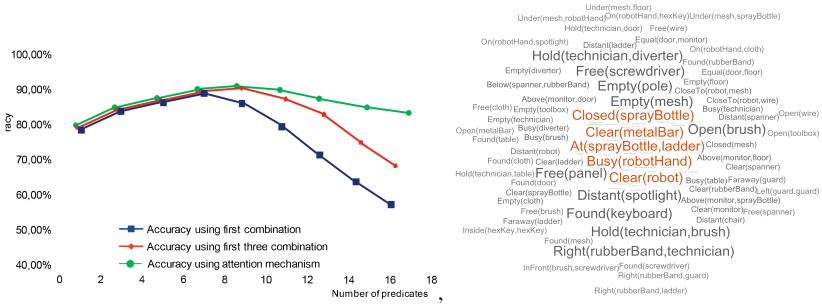

Fig. 3. Accuracy plot at variable number of predicates in the input sequence, considering the first combination, the first three and, finally with attention. On the right a cloud representation of the robot language expressed in the form of an Herbrand Universe, namely, all predicates are instantiated with all terms.

that initially the accuracy increases as the amount of atoms increases, this is caused by the fact that with more than one atom the sequence is more specific and characteristic. The maximum accuracy is reached at seven atoms with 94,2% of accuracy for the first combination and 97,9% using the first three. After this point the accuracy starts to decrease with the increase of the atoms in the input sequence. On the other hand we can note that by adding the attention mechanism the accuracy keeps high also with a large number of atoms.

Experiments of the VDEM Framework at Warehouse. In this section, we report the results of the experiments carried out with the VDEM deployed on the humanoid robot inside the warehouse. In the absence of other frameworks to make a comparison with, we perform a comprehensive ablation study. Table 2 shows the results. We identify the components of our framework with: PL = Planning, Ex = Execution, M = Monitoring (Visual Stream), GPr = Goals Prediction (LSTM). Furthermore, we indicate with Kn the complete knowledge of the world.

The experiments were performed on 5 tasks: *remove panel, support panel, clean diverter, bring object, find object*. Snapshots taken from two of these tasks are shown in Fig. 4. Each task was executed 50 times for assessing the accuracy, excluding failures caused by the robot controllistic part (grasping failure, platform movement error, etc.). The tasks have been tested for each framework configuration, making 750 total experiments. Note that for Task 5, there are no values related to the first configuration. This is because this task intrinsically requires perceptive and search skills, which can not be tested in the first configuration.

Starting from the PL + Ex + Kn case, the framework is tested with the FastDownward (FD) [20] based planning system and the execution component. FD was adopted as it proved to be the fastest among the other planners that were considered, i.e. POMDP and PKS [37]. In this configuration a complete knowledge of the world was provided. We note that the system in this case

Fig. 4. Recognition during tasks execution. The sequence shows the detection of *guard* (panel), *handle* and its manipulation to lower it helping the technician to hold the guard for inspecting the rollers. The involved relations are *At*, *Hold*, *InFront*, *On*, and *CloseTo*.

Table 2. Accuracy and average execution time according to task and configuration.

		$PL + Ex + Kn$	$PL + Ex + M$	$PL + Ex + M + GPr$
a. ex. time	Task 1	540 s	135 s	135 s
	Task 2	260 s	70 s	70 s
	Task 3	596 s	147 s	147 s
	Task 4	477 s	121 s	121 s
	Task 5	x	52 s	52 s
accuracy (%)	Task 1	23	72	81
	Task 2	52	78	80
	Task 3	24	68	79
	Task 4	26	75	86
	Task 5	x	85	93

suffers from long planning times caused by considering knowledge of the entire scene. Furthermore, this setting excludes dynamic and non-deterministic tasks.

Considering the *PL + Ex + M* setting, the robot is able to complete all the tasks correctly, as it is possible to manage the non-deterministic nature of the tasks in this case. An example of the detection and monitoring capacity is shown in the first row of Fig. 4.

A limitation of this setting concerns the management of failures due to the inability to predict the correct sequence of the goals.

Finally, the complete configuration of the framework is taken into consideration, *PL + Ex + M + GPr*. In this setting tasks are decomposed and executed dynamically, identifying in real time different ways to complete a task. A direct consequence of this greater flexibility, as can be seen in Table 2, is the improvement of the accuracy on the successful execution of the tasks.

An example is shown at the bottom row of Fig. 4. In this case the task is to find, grab and bring the brush to the technician. Based on experience, the seq2seq system first suggests *on(brush, table)*.

The goal fails, as another object is found (*on(spraybottle, table)* detected). At this point the possibility of recovery using seq2seq comes into play. The execution monitor takes the second proposal (regarding the first goal to be achieved) made by the seq2seq-based proposal system, namely *on(brush, ladder)*.

7 Conclusions

We have presented an approach to vision based deep execution monitor for a robot assistive task. Both the idea and the realization are novel and promising. The experiment with the humanoid robot created at the High Performance Humanoid Technologies Lab (H2T) have proved that the framework proposed works as far as the specific tasks are considered and as far as the high level actions are taken into account. Weak elements of the approach are the ability of the robot to search the environment, which should cope with the limitation of vision at distances greater than 2.5 m. We are currently facing this problem by modeling search with deep reinforcement learning, so that the robot can optimize its search of objects and relations.

Acknowledgments. The research has been granted by the H2020 Project Second Hands under grant agreement No. 643950. We thanks in particular our partners: the team at Ocado, Graham Deacon, Duncan Russel, Giuseppe Cotugno and Dario Turchi, the team of KIT led by Tamim Asfour, the team at UCL with Lourdes Agapito, Martin Runz and Denis Tome, and the group at EPFL led by Aude Billiard.

References

1. Al-Omari, M., Chinellato, E., Gatsoulis, Y., Hogg, D.C., Cohn, A.G.: Unsupervised grounding of textual descriptions of object features and actions in video. In: KR 2016, pp. 505–508 (2016)
2. Alford, R., Shivashankar, V., Roberts, M., Frank, J., Aha, D.W.: Hierarchical planning: relating task and goal decomposition with task sharing. In: IJCAI 2016, pp. 3022–3029 (2016)

3. Bahdanau, D., Cho, K., Bengio, Y.: Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 (2014)
4. Bertsekas, D.P., Tsitsiklis, J.N.: Neuro-dynamic programming: an overview. Decis. Control **1**, 560–564 (1995)
5. Boutilier, C., Reiter, R., Soutchanski, M., Thrun, S., et al.: Decision-theoretic, high-level agent programming in the situation calculus. In: AAAI/IAAI 2000, pp. 355–362 (2000)
6. Cohn, A.G., Hazarika, S.M.: Qualitative spatial representation and reasoning: an overview. Fun. Inf. **46**(1–2), 1–29 (2001)
7. Damen, D., et al.: Scaling egocentric vision: the epic-kitchens dataset. In: ECCV 2018 (2018)
8. Das, A., Agrawal, H., Zitnick, C.L., Parikh, D., Batra, D.: Human attention in visual question answering: do humans and deep networks look at the same regions? arXiv preprint arXiv:1606.03556 (2016)
9. Doyle, R.J., Atkinson, D.J., Doshi, R.S.: Generating perception requests and expectations to verify the execution of plans. In: AAAI 1986, pp. 81–88 (1986)
10. Erol, K., Hendler, J.A., Nau, D.S.: UMCP: a sound and complete procedure for hierarchical task-network planning. In: AIPS, vol. 94, pp. 249–254 (1994)
11. Everingham, M., Eslami, S.M.A., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The Pascal visual object classes challenge: a retrospective. IJCV **111**(1), 98–136 (2015)
12. Fikes, R.E.: Monitored execution of robot plans produced by strips, SRI, Technical report (1971)
13. Finzi, A., Pirri, F.: Combining probabilities, failures and safety in robot control. In: International Joint Conference on Artificial Intelligence, vol. 17, no. 1. Lawrence Erlbaum Associates Ltd., pp. 1331–1336 (2001)
14. Furnari, A., Battiato, S., Grauman, K., Farinella, G.M.: Next-active-object prediction from egocentric videos. J. Vis. Commun. Image Represent. **49**, 401–411 (2017)
15. Gianni, M., Kruijff, G.-J.M., Pirri, F.: A stimulus-response framework for robot control. ACM Trans. Interact. Intell. Syst. (TIIS) **4**(4), 21 (2015)
16. Guadarrama, S., et al.: Grounding spatial relations for human-robot interaction. In: IROS 2013, pp. 1640–1647 (2013)
17. Gupta, S., Davidson, J., Levine, S., Sukthankar, R., Malik, J.: Cognitive mapping and planning for visual navigation. arXiv preprint arXiv:1702.03920, vol. 3 (2017)
18. Haarnoja, T., Ajay, A., Levine, S., Abbeel, P.: Backprop KF: learning discriminative deterministic state estimators. In: Advances in Neural Information Processing Systems, pp. 4376–4384 (2016)
19. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask R-CNN. In: IEEE International Conference on Computer Vision (ICCV), pp. 2980–2988. IEEE (2017)
20. Helmert, M.: The fast downward planning system. JAIR **26**, 191–246 (2006)
21. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Comput. **9**(8), 1735–1780 (1997)
22. Hofmann, T., Niemueller, T., Lakemeyer, G.: Initial results on generating macro actions from a plan database for planning on autonomous mobile robots (2017)
23. Hornung, A., Böttcher, S., Schlagenhauf, J., Dornhege, C., Hertle, A., Bennewitz, M.: Mobile manipulation in cluttered environments with humanoids: integrated perception, task planning, and action execution. In: Humanoids 2014, pp. 773–778 (2014)
24. Ingrand, F., Ghallab, M.: Deliberation for autonomous robots: a survey. Artif. Intell. **247**, 10–44 (2017)

25. Karkus, P., Hsu, D., Lee, W.S.: QMDP-Net: deep learning for planning under partial observability. In: Advances in Neural Information Processing Systems, pp. 4697–4707 (2017)
26. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: NIPS 2012, pp. 1097–1105 (2012)
27. Lenz, I., Lee, H., Saxena, A.: Deep learning for detecting robotic grasps. Int. J. Robot. Res. **34**(4–5), 705–724 (2015)
28. Littman, M.L., Sutton, R.S.: Predictive representations of state. In: Advances in Neural Information Processing Systems, pp. 1555–1561 (2002)
29. Lu, C., Krishna, R., Bernstein, M., Fei-Fei, L.: Visual relationship detection with language priors. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9905, pp. 852–869. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46448-0_51
30. Luong, M.-T., Pham, H., Manning, C.D.: Effective approaches to attention-based neural machine translation. arXiv preprint arXiv:1508.04025 (2015)
31. Luong, M.-T., Sutskever, I., Le, Q.V., Vinyals, O., Zaremba, W.: Addressing the rare word problem in neural machine translation. arXiv preprint arXiv:1410.8206 (2014)
32. Mendoza, J.P., Veloso, M., Simmons, R.: Plan execution monitoring through detection of unmet expectations about action outcomes. In: ICRA 2015, pp. 3247–3252 (2015)
33. Mirowski, P., et al.: Learning to navigate in complex environments. arXiv:1611.03673 (2016)
34. Nilsson, N.J.: A hierarchical robot planning and execution system. SRI (1973)
35. Ntouskos, V., Pirri, F., Pizzoli, M., Sinha, A., Cafaro, B.: Saliency prediction in the coherence theory of attention. In: Biologically Inspired Cognitive Architectures, vol. 5, pp. 10–28 (2013)
36. Ntouskos, V., et al.: Component-wise modeling of articulated objects. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2327–2335 (2015)
37. Petrick, R.P., Bacchus, F.: PKS: knowledge-based planning with incomplete information and sensing. In: Proceedings of the System Demonstration session at ICAPS (2004)
38. Pettersson, O.: Execution monitoring in robotics: a survey. Robot. Auton. Syst. **53**(2), 73–88 (2005)
39. Raffel, C., Luong, M.-T., Liu, P.J., Weiss, R.J., Eck, D.: Online and linear-time attention by enforcing monotonic alignments. arXiv preprint arXiv:1704.00784 (2017)
40. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: towards real-time object detection with region proposal networks. In: NIPS 2015, pp. 91–99 (2015)
41. Sabharwal, C.L., Leopold, J.L., Eloe, N.: A more expressive 3D region connection calculus. In: DMS, pp. 307–311. Citeseer (2011)
42. Santoro, A., et al.: A simple neural network module for relational reasoning. In: Advances in Neural Information Processing Systems, pp. 4974–4983 (2017)
43. Sanzari, M., Ntouskos, V., Pirri, F.: Bayesian image based 3D pose estimation. In: Leibe, B., Matas, J., Sebe, N., Welling, M. (eds.) ECCV 2016. LNCS, vol. 9912, pp. 566–582. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46484-8_34
44. Shivashankar, V.: Hierarchical goal networks: formalisms and algorithms for planning and acting, Ph.D. dissertation, University of Maryland, College Park (2015)
45. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv:1409.1556 (2014)

46. Sutskever, I., Vinyals, O., Le, Q.V.: Sequence to sequence learning with neural networks. In: Advances in Neural Information Processing Systems, pp. 3104–3112 (2014)
47. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction, 2nd edn. MIT Press, Cambridge (2017)
48. Wächter, M., Ottenhaus, S., Kröhnert, M., Vahrenkamp, N., Asfour, T.: The ArmarX statechart concept: graphical programing of robot behavior. Front. Robot. AI **3**, 33 (2016)
49. Wang, H., Liang, W., Yu, L.-F.: Transferring objects: joint inference of container and human pose. In: CVPR 2017, pp. 2933–2941 (2017)
50. Wilkins, D.E.: Recovering from execution errors in SIPE. Comput. Intell. **1**(1), 33–45 (1985)
51. Wu, C., Zhang, J., Sener, O., Selman, B., Savarese, S., Saxena, A.: Watch-n-patch: unsupervised learning of actions and relations. In: TPAMI 2017 (2017)
52. Zeiler, M.D., Fergus, R.: Visualizing and understanding convolutional networks. In: Fleet, D., Pajdla, T., Schiele, B., Tuytelaars, T. (eds.) ECCV 2014. LNCS, vol. 8689, pp. 818–833. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10590-1_53
53. Zhu, L., Xu, Z., Yang, Y., Hauptmann, A.G.: Uncovering the temporal context for video question answering. IJCV **124**(3), 409–421 (2017)
54. Zhu, Y., et al.: Visual semantic planning using deep successor representations. CoRR abs/1712.05474 (2017)