Leveraging Large-Scale Semantic Networks for Adaptive Robot Task Learning and Execution

Adrian Boteanu · Aaron St. Clair · Anahita Mohseni-Kabir · Carl Saldanha · Sonia Chernova

Received: date / Accepted: date

Abstract This work seeks to leverage semantic networks containing millions of entries encoding assertions of commonsense knowledge to enable improvements in robot task execution and learning. The specific application we explore in this project is object substitution in the context of task adaptation. Humans easily adapt their plans to compensate for missing items in day-to-day tasks, substituting a wrap for bread when making a sandwich, or stirring pasta with a fork when out of spoons. Robot plan execution, however, is far less robust, with missing objects typically leading to failure if the robot is not aware of alternatives. In this paper, we contribute a context-aware algorithm that leverages the linguistic information embedded in the task description to identify candidate substitution objects without reliance on explicit object affordance information. Specifically, we show that task context provided by the task labels within the action structure of a task plan, can be leveraged to disambiguate information within a noisy large-scale semantic network containing hundreds of potential object candidates in order to identify successful object substitutions with high accuracy. We present two extensive evaluations of our work on both abstract and real-world robot tasks, showing that the substitutions made by our system are valid, accepted by users and lead to statistically significant reduction in robot learning time. Additionally, we report the outcomes of testing our approach with a large number of crowd workers interacting with a robot in real time.

A. Boteanu, A. Mohseni-Kabir
Worcester Polytechnic Institute
E-mail: aboteanu,amohsenikabir@wpi.edu

A. St. Clair, C. Saldanha, S. Chernova
Georgia Institute of Technology
E-mail: astclair,csaldanha3,chernova@gatech.edu
1 Introduction

Similar to other areas of computing, the robotics field has sought to leverage progressively larger and more complex data sources to develop more robust reasoning capabilities. In some cases, such as Levine at al.’s [24] and Oberlin et al.’s [32] work on robot grasping, large data sets have been acquired through physical robot interaction with the environment. In other works, such as Breazeal et al.’s crowdsourcing of human-robot interaction through a multiplayer game [7], the data collection has leveraged the web and online resources. Large datasets are also available for object recognition in 2D images [25] and for cloud-enabled semantic maps of object locations [34]. Unlike applications in which the robot collects data from direct observations or a limited number of sources, at large scales, information is likely obtained from multiple sources. This offers the potential of greater breadth and detail of knowledge than is feasible to collect directly, however, it also poses the significant challenge of using potentially inaccurate or conflicting information.

In this work, we present a method for leveraging large-scale semantic networks to improve robot task execution and task learning. Semantic networks, such as Cyc [23], WordNet [33] and ConceptNet [26], contain millions of hand coded and crowdsourced entries encoding assertions of commonsense knowledge encompassing the spatial, physical, social, temporal, and psychological aspects of everyday life. Our work seeks to leverage these large-scale data sources to enable robots to (1) reason more effectively about the world at a conceptual level and (2) to initiate user interaction for validating this conceptual data against the physical environment. We consider user interaction essential for applying general purpose semantic knowledge to robot execution since this information is ungrounded and cannot be used directly for execution.

The specific application we explore in this project is object substitution in the context of task adaptation during execution and learning. Humans easily adapt their plans to compensate for missing items in day-to-day tasks, substituting a wrap for bread when making a sandwich, or stirring pasta with a fork when out of spoons. Robot plan execution, however, is far less robust, with missing objects typically leading to failure. While planning-based approaches, such as dynamic planning, can enable run-time recovery, and also the environment can be observed for missing objects which the planner can account for, they require that knowledge about the substitute available to the planner. In particular, object models (such as physical models and semantic maps of probable locations) and affordances (object properties that enable them to be used in certain ways, for example, that a ball is graspable and rolls [15]) need to be known in order to substitute objects using planning methods. We argue that, through user interaction initiated by the robot, large data describing concepts and semantics can bridge this knowledge gap and bootstrap plan recovery.

In this paper, we contribute a context-aware algorithm that leverages the linguistic information embedded in the task description to identify candidate substitution objects without reliance on explicit object affordance information. Specifically, we show that task context provided by the task labels within the
action structure of a task plan (in our work tasks are expressed as Hierarchical Task Networks (HTN) [39,8,9]), can be leveraged to disambiguate information within a noisy large-scale semantic network containing hundreds of potential object candidates in order to identify successful object substitutions with high accuracy. For the case in which substitute objects are not represented sufficiently in the system for the robot to use in the task, for example no grasping model is known, existing work has demonstrated effective interfaces that enable the user to teach a new grasping model that will enable the robot to resume execution [20,21,6,42]. The focus of this paper is in the method used to propose object substitutions to the user.

We present two extensive evaluations to validate our work. In the first, we evaluate our approach on nine distinct tasks, focusing on the algorithm’s ability to select valid substitutions from an unrestricted set of object candidates; two of the tasks are also performed on a physical robot. In this most difficult validation scenario, our system achieves 81% success is accurately predicting valid substitutions. In the second evaluation, we examine object substitution in the context of a user teaching through interactive demonstrations the task of packing a lunch. Our user study results show that 100% of our suggestions were accepted by participants, and that users in the suggestions condition achieved the same task quality while spending statistically significantly less time teaching the task than participants in the no-suggestions condition. Finally, we conclude the paper by considering the role that crowd workers from online micro-task markets can play in large-scale robot experimental validations. We perform a user study in which crowd workers perform experiments on our robot in real time and discuss user retention and performance quality metrics.

2 Background and Related Work

This section will focus on reviewing existing work in run-time recovery, object substitution, and related approaches (2.1). In the rest of this section we will review semantic networks (2.2) and Hierarchical Task Networks (2.3).

2.1 Task Adaptation and Object Substitution

For humans performing tasks, performing object substitutions and asking others for help have been considered natural behaviors [4]. In robotics applications, the problem of object substitution has been explored in the context of affordance reasoning as a method of recovering at run-time from exceptions created by missing objects, with conceptual substitutions being used as fallback mechanisms for cases in which affordance information is not available [4,1,3,2]. With similar high-level goals in mind, our contribution focuses on providing robot-driven interaction for performing conceptual substitution, by utilizing large general-purpose semantic networks to bridge the knowledge gap.
between user interaction and affordance representation. While conceptual semantic information is more readily available and can be collected from a broad variety of sources including unstructured data, it is not granular or specific enough to be directly used in a robot plan. Conceptual similarity has been indentified as a fall-back approach, should substitution using affordances fail [4]. In this paper, we initiate user interaction based on this conceptual object substitution, allowing for more specific models to be taught to the system. We focus on replacing individual objects in the task and not sub-tasks, thus we primarily envision as part of the recovery process teaching new grasping models for pick-and-place tasks [6,42].

General purpose semantic networks have been used in robotics applications. RoboBrain uses information from OpenCyc and WordNet together with low-level data such as object models and affordances [40]. RoboBrain has been leveraged in order to interpret the scene context, helping reduce ambiguity in perception [28]. While similar in spirit, the focus of paper is on how such conceptual information can be leveraged for adaptive task learning and execution. Thus, it is potentially compatible with emerging work in representing large-scale knowledge for robotics, such as RoboBrain.

The RoboEarth project has the goal of enabling robots to share information collected in different settings, with the goal of enabling common knowledge similar to how web users contribute to encyclopedias [44,34]. The project consists of a web service which can provide action recipes, object models and environment maps. In order to establish the semantics of the data being requested or contributed by a robot, RoboEarth uses a description language to formalize actions and their parameters, meta-information such as measuring units, requirements on the robot’s components, robot self-models and capabilities; the description language also offers methods to match requirement descriptions to the robot’s capabilities and to identify missing components. Our method differs most significantly from RoboEarth in that it does not rely on a predefined ontology of descriptive knowledge but is able to automatically leverage existing large-scale knowledge bases.

Dynamic planning integrates planning and execution, with the system adapting its plan depending on execution results [14]. Two main approaches have been described as part of dynamic planning, with the goal of recovering from exceptions at runtime: replanning and plan repair [11]. The former consists of the robot discarding its original plan and generating a new plan from the current state and with the same goal state as the discarded plan. Plan repair is an established research area that explores techniques that enable an agent to locally alter an existing plan to overcome changes in the world state [13,46]. The problem of plan adaptation through object substitution, as presented in our work, can also be viewed as related to existing work on plan repair in symbolic domains. Through substitution, the planning domain can be extended before repair is attempted. Object substitution is performed prior to plan repair, with the effect that it extends the planning domain through a set of valid substitute objects before the plan is repaired in other ways. Note that the set of substitutions is generated by a model trained on similarity data.
(Section 3), and are optionally validated by the user before being used in the task. Plan repair is preferable over discarding the initial plan and re-planning, both computationally, because fewer intermediate states need to be recomputed, and from a usability standpoint, since adapting an existing plan results in a similar approach that is more predictable to a user [11, 22]. However, because of the simplifying assumptions we make on the substitution (i.e. we only consider objects and assume the substitute has the same affordances as the original), plan repair is not necessary in our implementation. We anticipate future integration to be necessary for replacing higher-level sub-tasks.

Also related is the work on plan generalization, which has been used in one-shot-learning to remove constraints falsely inferred from a small sample pool of demonstrations, such that the learned models have greater generality [47]. This method relies on semantic knowledge to propose more general scenarios derived from the demonstration, which are validated symbolically.

2.2 Semantic Networks

Since the primary purpose of the work presented in this paper is to initiate user interaction with non-expert users, our approach relies on language knowledge from general purpose semantic networks in order to generate and evaluate substitution candidates. As a data structure, a semantic network represents concepts as vertices of a graph and relations between these concepts as edges in the graph. In this work, we used WordNet [33] and ConceptNet [26]. The first is an expert-authored graph that focuses on noun taxonomy. WordNet represents words senses by associating concepts with synsets – different senses of a word belong to different synsets. It provides a concept similarity measure as the normalized path distance between senses in the hypernym/hyponym hierarchy. ConceptNet aggregates data from a variety of sources, including WordNet, DBPedia, and crowd-contributed information. ConceptNet represents edges covering a broader spectrum than WordNet, and was designed to represent “commonsense” information instead of focusing on lexical relations as is the case with WordNet. In total, edges representing 48 relation types are included in ConceptNet (e.g. part-of, is-a, used-for). Similarity between concepts in ConceptNet can be measured using the Divisi toolkit, which relies on singular value decomposition [41]. Divisi produces a normalized measure of how similar a pair of concepts is.

2.3 Hierarchical Task Networks

Hierarchical Task Networks (HTN) are non-linear plan formalizations which describe a task at different levels of abstraction: tasks are decomposed into sub-tasks recursively, down to primitive tasks which allow no further detailing and can be executed directly (example in Figure 2). In our implementation of object substitution, we replace the attribute (i.e. object) of a primitive task;
we assume that the substitute performs similarly to the original and that the same primitive task can be applied to it.

HTNs were introduced following the observation that STRIPS planners [10] impose unnecessary constraints on execution by establishing a sequence of atomic operations during initial planning [39,38]. Planning with HTNs reduces the planning space at a given sub-task level, since a sub-task can be composed only of a subset of all possible tasks. For example, in the HTN shown in Figure 2, a Store sub-task can only be composed of Get and Place primitive tasks; similarly, the top-level task FruitBasket cannot directly contain Get or Store sub-tasks. When contrasted with STRIPS-like planners, HTN planning has been considered more expressive [8,9]. However, the problem is undecidable if no restrictions are placed on non-primitive tasks or if execution order is not restricted.

HTNs have been used in human-robot interaction to enable the system to support human teachers in structuring demonstrations, which reduces the number of demonstrations necessary to teach new tasks by eliminating otherwise redundant demonstrations [29,30]. In our second evaluation, shown in Section 5, we use this task teaching framework to allow users to teach new tasks to the robot. Because in our evaluations we use HTN task descriptions either defined manually (Section 4) or taught by users (Section 5), and because the focus of this paper is to replace single objects under the assumption that they meet the same affordance requirements as the original, we do not use an HTN planner. Instead, the task teaching interface is implemented using Disco, a collaborative interaction manager which uses the ANSI/CEA-2018 standard [35,36]. In our implementation, sub-tasks are executed in pre-defined order as indicated by the temporal constraint arrows in Figures 2 and 3.

3 Object Substitution within the Task Context

In this section we present our object substitution approach, which consists of three steps outlined in Figure 1. We begin with the task description encoded as a HTN, which represents plans as trees, with primitive actions on the leaf nodes and logical groupings on the higher levels [8]. Figure 2 shows an example HTN for the task of making a fruit basket. Note that our approach is not limited to HTNs, and in the results section we compare against a context generated from a flat plan representation. However, we find that HTN plans are most commonly represented in human readable form [31,37], including techniques for learning HTNs from human instructions [30].
Fig. 2 An example hierarchical task network for making a fruit basket.

Given a primitive HTN action with a missing object input, we use the term target, and variable \( t \), to denote the missing item (e.g., apple in \( \text{get(apple)} \)); thus, for the current implementation, target concepts are found only on leaf nodes in the HTN. The first step of the pipeline is to leverage ConceptNet to generate \( C \), the set of potential substitution candidates (Section 3.1). In parallel, we extract the context vocabulary set \( V \) from the HTN, which provides task context (Section 3.2).

In the third step, we score each candidate \( c \in C \) within the context of \( V \) using a supervised machine learning model. The model composes three main measures of similarity to derive features that relate \( t \) to each candidate and to \( V \) (Section 3.3). Using these similarity metrics as features, we classify candidates as either valid or not valid using a random forest classifier. In the final step, the robot suggests valid substitution candidates one at a time to the user until the user accepts one. We view interaction with the user as a key advantage of our approach because it enables users to personalize substitutions (e.g., one user may be ok with replacing turkey with ham in a sandwich, and another may not); additionally, existing work in human-robot interaction shows that providing high-quality feedback is essential in order for the dialog and teaching process to succeed [43]. Based on the user’s feedback, the final outcome of our algorithm is a subset \( \tilde{C} \subseteq C \) containing the words selected as valid object substitutions.

3.1 Candidate Generation

Given the target, we generate the candidate set \( C \) by using a semantic network to extract the names of similar objects. Specifically, we select the concepts from ConceptNet which share the same parent with the target for any of the following relations: \textit{has-property}, \textit{capable-of}, or \textit{used-for}. We selected these relations because they are indicative of object uses more than other relation types present in ConceptNet; these edges should not be considered equivalent to

\footnote{We also experimented with deriving \( C \) from sibling concepts from WordNet (i.e. concepts which share the same hypernym). However, we found this approach resulted in over constrained candidate sets with significantly fewer substitution candidates.}
affordance information, since in ConceptNet node and edge information does not include word senses or gloss. As later results show, this process does not define sufficient conditions for generating substitutions reliably. For example, two candidates we obtain for the target apple are cherry and brick, because both are also connected to red by has-property edges. Such incorrect candidates are the result of adopting a greedy and local approach in selecting them, by comparing candidate concepts only with the missing object, and highlight the importance of considering other concepts when evaluating substitutions. However, the goal of candidate generation is to include a broad variety of concepts that have semantic support for being substitutes, with later stages of the algorithm discarding unlikely candidates.

3.2 Derivation of Task Context

We model the task context, $V$, as a bag-of-words model consisting of task and input labels from the task representation itself. In this work, we utilize the HTN plan representation (Figure 3), which we selected for two reasons. First, as stated previously, techniques have been developed for formulating HTNs with human-readable labels [30]. Second, we hypothesize that the abstract, middle layers of the hierarchy (e.g. MakeBatter) provide additional context vocabulary that will improve performance of the algorithm.

To generate $V$, we extract words from labels found in nodes of the HTN, using the task label and the input and output labels of operators defined in the subtasks (for example, the target label of a perception primitive). We split these labels into words and lemmatize them using the Natural Language Toolkit (NLTK) [27]. For example, the node label GraspSpoon is converted into \{grasp, spoon\}. We do not model multi-word concepts because looking up such concepts in ConceptNet and WordNet is not sufficiently reliable.

In this paper, we compare five strategies for generating $V$ in order to evaluate the impact that context scope, in terms of both depth and breadth, has on the quality of produced substitutions:
Context derivation strategies.

- Node: words only from the task node of the HTN that contains the substitution target (e.g., pick up);
- Task Name: words derived only from the task name at the root of the HTN (e.g., make pancake);
- Root Path: the union of words derived from the current sub-task and all parent sub-tasks on the path between the target and the root of the HTN;
- Leaves: the union of words derived from lead task nodes; note that this is identical to a context generated from a flat plan representation;
- Full: the union of words from all task notes in the HTN.

Figure 4 presents a visualization of the context strategies.

3.3 Candidate Similarity

Having obtained substitution candidate set $C$, context vocabulary set $V$, and target word $t$, we now evaluate each candidate, $c_i \in C$, with respect to its similarity to $t$ within the context $V$. We utilize three similarity metrics: WordNet path similarity ($WPS$) [33], Divisi pairwise similarity ($CND$), and Semantic Similarity Engine’s analogical similarity ($SSE$) [5]. $WPS$ and $CND$ are standard metrics introduced in Section 2. $SSE$ is a similarity metric recently introduced in [5], originally designed to perform analogical reasoning by searching for relational parallelism between two pairs of words using a semantic network. The analogies SSE targets have the form “$A$ is to $B$ as $C$ is to $D$” or $A:B::C:D$. Within the context of object substitution, we use SSE to evaluate the $t : v_j :: c_i : v_j$ for all $c_i \in C$ and $v_j \in V$. In other words, we measure the similarity in the target-to-context and candidate-to-context relationships to evaluate how suitable the candidate is for replacing the target.

Each of the above three measures outputs a score measuring the similarity of $t$ and $c_i \in C$ within $V$. Based on these numerical values, we are interested in predicting the suitability of substituting $c_i$ for $t$. To make this prediction we train a classifier (details in Section 4.1) based on the following features (described using the notation defined in the above paragraph):

- Similarity between target and context:
WPS(t, V) = \sum_{v_j}^{WPS(t, v_j)} and CND(t, V) = \sum_{v_j}^{CND(t, v_j)};  

- Similarity between candidate and context:
  WPS(c_i, V) = \sum_{v_j}^{WPS(c_i, v_j)} and CND(c_i, V) = \sum_{v_j}^{CND(c_i, v_j)}

- Target and context similarity difference w.r.t. context:
  \Delta WPS(t, c_i, V) = WPS(t, V) - WPS(c_i, V) and \Delta CND(t, c_i, V) = CND(t, V) - CND(c_i, V)

- Analogical similarity:
  SSE(t : V :: c_i : V) = \sum_{v_j}^{SSE(t, v_j :: c_i, v_j)}
  and the proportion of above analogies for which SSE finds no similarity, denoted as no_analogy. The latter relies on a feature of SSE which identifies analogies which are not represented at all in the semantic network, indicating either that there is no analogical similarity between the two pairs, or that the semantic network does not contain the necessary information.

Additionally, we compare our approach against a baseline similarity. To investigate the assumption that task context is important for evaluating substitution candidates, we use a baseline which compares the original and substitute objects directly, ignoring V, and evaluates similarity using only WPS(t, c_i) and CND(t, c_i).

4 Evaluation 1: Object Substitution in Task Execution

In this section we evaluate object substitution in the case of it being used while the robot is executing a task. The target scenario, also shown in the video\(^2\), is for the robot to attempt to perform a task, initiate runtime substitution in the case of a missing object, optionally obtain user validation of the suitability of the planned substitution, and finally complete the task using the substituted item. Note that in this scenario we consider the most difficult substitution scenario, in which the set of substitution candidates is not restricted to a predefined set of object candidates; the entire set of objects within ConceptNet is available for substitution selection. The substitution space can be reduced by limiting the available substitutions to objects the robot already knows it has access to in its environment, which we do in our second evaluation.

In our analysis, we present results on different strategies for constructing context from the task vocabulary, showing that including all words from the task outperforms the baseline and other approaches. We then investigate the performance of our method relative to each semantic resource and similarity metric, and to omissions in the task vocabulary. Such omissions may arise from tasks that are not fully annotated with human-readable labels, or if some unusual words are not present in the semantic network. We present two experiments on the transfer-learning potential and generality of the models produced by our method. Finally, we give details on the setup of the tasks we

\(^2\) https://youtu.be/4JMWsSgHF00
Table 1 Vocabulary sizes for each task, the valid candidate counts are according to the expert annotation.

<table>
<thead>
<tr>
<th>Task</th>
<th>Total words</th>
<th>Targets</th>
<th>Candidates</th>
<th>Valid cand.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Eat Soup</td>
<td>8</td>
<td>4</td>
<td>156</td>
<td>6</td>
</tr>
<tr>
<td>Make Pancakes</td>
<td>6</td>
<td>7</td>
<td>94</td>
<td>8</td>
</tr>
<tr>
<td>Set the Table</td>
<td>14</td>
<td>6</td>
<td>302</td>
<td>10</td>
</tr>
<tr>
<td>Cook Soup</td>
<td>17</td>
<td>6</td>
<td>307</td>
<td>30</td>
</tr>
<tr>
<td>Make Hot Drink</td>
<td>26</td>
<td>9</td>
<td>173</td>
<td>8</td>
</tr>
<tr>
<td>Rotate Car Tires</td>
<td>15</td>
<td>6</td>
<td>44</td>
<td>0</td>
</tr>
<tr>
<td>Make Fruit Basket</td>
<td>8</td>
<td>5</td>
<td>305</td>
<td>31</td>
</tr>
<tr>
<td>Wipe Table</td>
<td>7</td>
<td>4</td>
<td>84</td>
<td>17</td>
</tr>
<tr>
<td>Pack School Bag</td>
<td>7</td>
<td>5</td>
<td>308</td>
<td>23</td>
</tr>
</tbody>
</table>

implemented on the physical robot in order to demonstrate a user interaction scenario.

4.1 Experimental Setup

Our evaluation focuses on two aspects: (1) investigating the benefit of using context information in addition to the baseline similarity metrics, and (2) analyzing the resilience our method has to variations in the context constituency. We apply our evaluation to HTNs derived for nine different tasks. To allow for a greater variety of concepts than what is possible to accomplish with our physical robot, we manually encoded the nine HTNs used in this section, whereas in Section 5 we use HTNs taught by the user. The datasets we used for training classifiers contain an instance for each (task, target, candidate) tuple, for which we computed the context-average similarity scores according to each similarity metric. The classifier was trained only on the numerical similarity metric values and did not take the task or target words into account.

We encoded our HTNs following established notations taken from prior applications of HTNs for robotic tasks [37,30]. Table 1 summarizes word information for each task: the number of targets is the number of input words for primitive tasks, the number of candidates is the set of substitution candidates derived using the method described in Section 3.1, and valid candidates are those annotated as such for supervised training. The average set overlap between the vocabulary of each task (Jaccard index) is 8.9%. The average overlap of the sets of targets between tasks is 3.2%. Thus, the tasks are mostly orthogonal in the vocabularies they use, with the majority of the overlap being in the names of the primitive actions, all related to manipulation. Note that the Rotate Car Tires task, replicated from [30], has no allowable substitutions in our system, serving as an example of a task in which only very specific tools/items can be used.

4.1.1 Labeling Valid Substitutions

In selecting substitution targets, we assume that only objects directly manipulated can be substituted, which in our representation reside in the leaf nodes,
which contain a primitive action (e.g. get) and an input (e.g. cup). Our total set of tasks contained 37 substitution targets ($M = 5.9; SD = 1.5$). Starting from these targets, we generated a total of 1832 substitution candidates ($M = 203.5; SD = 111.8$), which greatly exceeds the number of targets, but as we will see, only a small percentage of these represent valid substitutions.

To establish the ground truth for classification, we had all candidates labeled as suitable or not by two experts familiar with the task definition and the scope of this work. For 31 candidates out of 1832 in total, the experts disagreed, in which case the final label was decided after discussion. In total, 6.16% (113) of the total number of substitution candidates were annotated as valid ($M = 14.7; SD = 10.4$).

Validation of expert labels: As an additional step to verify the expert labels, we randomly selected 228 of the candidates to be annotated on the Crowdflower\textsuperscript{3} platform. The survey included a brief background description, the name of the task, and the target and candidate objects. We noted the final annotation result as the majority vote from at least 5 participants (16.1 on average). We selected workers only from English-speaking countries (111 workers in total). The expert and crowd annotations match for 80.7% of the substitution candidates. The experts rejected 35 candidates that the crowd accepted: e.g. replacing a knife with a spoon in the cook soup task, or glass with chalk for the set the table task. On the other hand, there were 9 instances in which the crowd rejected a substitution that the experts accepted, for example that a knife can be replaced with a cleaver for the same task. We attribute these distinctions to personal preference, and also to crowdsourcing noise \cite{19}. Overall, following this survey we concluded that the expert annotation is representative of what a common person might accept in terms of substitution.

4.1.2 Performance Metrics and Classification Approach

We tracked two metrics for evaluating classification accuracy. First, the global accuracy of the classifier, i.e. on both valid and invalid substitutions. Second, we report the rate of success on valid substitutions, since this value most accurately represents the user-observable performance of our method:

$$\frac{\text{valid substitutions}}{\text{valid substitutions} + \text{false positives} + \text{false negatives}}$$ \hspace{1cm} (1)

Having established our metrics, the first step was to choose a classification model. We compared the performance of our method for full context metrics with the baseline over six common classification methods\textsuperscript{4}. With 95% confidence according to a paired t-test, the Random Forest Classifier outperforms the other methods for both the baseline and for our method (the second best is $J_4.8$ Decision Tree). Therefore we use Random Forest for the rest of our experiments, applying the same classifier in all our experiment. Thus, the baseline performance is obtained using a classifier trained only on the baseline similarity metrics (WordNet path similarity and Divisi similarity between the

\textsuperscript{3}http://www.crowdflower.com
\textsuperscript{4}Weka 3.7.12 [17]
Fig. 5 Relative classification performance for different methods of generating context vocabularies from the HTN and the task-agnostic baseline.

4.2 Evaluation of Context-creation Strategies

A central hypothesis of our work is that taking into account the broader context of the other objects and actions in the task is important for evaluating whether a substitution candidate is viable or not. To evaluate this hypothesis, we compared the five context generation methods introduced in Section 3.2, as well as the context-agnostic baseline. Figure 5 reports the substitution prediction accuracy of all six methods, calculated using 10-fold cross-validation averaged over all nine tasks (tested on 10% randomly withheld instances from the candidate set). As can be seen, due to the skewed distribution of negative to positive samples in our dataset, the global classification accuracy (Total) is less informative than the valid substitutions metric.

The results show that the full context condition outperforms all other methods (81% valid-only accuracy, compared to 66% for the baseline). The results support our hypothesis that context is beneficial, but only when the context contains higher-level concepts and a larger number of words. The \textit{Node} and \textit{RootPath} contexts do not perform as well, possibly because of the similarity noise introduced by having a small number of words in the context, out of which most are generic manipulation actions. However, even in this case we can observe that adding higher-level concepts benefits the classification accuracy, since \textit{RootPath} outperforms \textit{Node}.

Another example of the power of using the full context is the rotate tires task, which has no valid substitution candidates. For this task, the full-context classifier outputs no false positives and clearly outperforms the baseline, which attempts to replace the metal nuts that affix the wheels with various types of
Table 2 Classification performance for using similarity metrics from only WordNet, only ConceptNet, or both, for the full context classifier. “Global Accuracy” represents the total classification accuracy, on both valid and invalid substitution candidates, while “Valid Substitution Accuracy” is computed using Formula (1). Values reported for a random forest classifier over 10-fold cross-validation.

<table>
<thead>
<tr>
<th></th>
<th>ConceptNet</th>
<th>Global Accuracy</th>
<th>Valid Substitution Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>•</td>
<td>92%</td>
<td>59%</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>95%</td>
<td>72%</td>
</tr>
<tr>
<td>•</td>
<td>•</td>
<td>95.7%</td>
<td>76%</td>
</tr>
</tbody>
</table>

fruit – without context information, the word senses would need to be manually annotated in the task to avoid this mistake, which would be a formidable challenge to do consistently and reliably.

Based on these results, we use the Full context model for the rest of our evaluation.

4.3 Semantic Network and Attribute Importance

We evaluated the relative importance of the semantic networks of choice, WordNet and ConceptNet, along with their corresponding similarity metrics. First, we grouped the classification features shown in Section 3.3 into two groups: WordNet, containing $WPS$ and $\Delta WPS$ features, and ConceptNet, containing $CND$, $\Delta CND$, $SSE$ and $no\_analog$ features. Table 2 shows the classification performance on each feature group. We observe that the combination of the two achieves better performance on both metrics.

Additionally, we ranked individual features by their information gain\(^5\). Table 3 shows the resulting weights, higher values indicate more importance. We observe that the baseline features are dominant (ranks 1-2), which confirms that the similarity between $t$ and $c_i$ is a strong predictor of substitution success. However, all other types of attributes contribute to a significant degree as well (ranks 3-8). We conclude that having a greater variety of similarity metrics is preferable. As expected, the delta features do not add information (ranks 9-10), however we obtained better classification results when these were included.

4.4 Context Vocabulary Sensitivity

In this experiment, we investigated how resilient our method is to task definitions that are not well populated by words. Such tasks may arise from flat representations or from automatically constructing task trees from human demonstration.

First, we trained a classifier across all tasks. Then we generated a test dataset by sampling sets of words from the full context of each task, computed

\(^5\) Using the `InfoGainAttributeEval` class from Weka.
Table 3 Rank scores for the similarity metrics, showing that ConceptNet Divisi (CND), WordNet path similarity (WPS), and SSE analogical similarity (SSE) have similar importance for classification.

<table>
<thead>
<tr>
<th>Rank</th>
<th>Similarity Measure</th>
<th>Ranker Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>WPS(t, c_i)</td>
<td>0.1633</td>
</tr>
<tr>
<td>2</td>
<td>CND(t, c_i)</td>
<td>0.1591</td>
</tr>
<tr>
<td>3</td>
<td>CND(t, V)</td>
<td>0.1095</td>
</tr>
<tr>
<td>4</td>
<td>SSE(t: V::c_i: V)</td>
<td>0.0987</td>
</tr>
<tr>
<td>5</td>
<td>no_analogy</td>
<td>0.0795</td>
</tr>
<tr>
<td>6</td>
<td>WPS(t, V)</td>
<td>0.0781</td>
</tr>
<tr>
<td>7</td>
<td>WPS(c_i, V)</td>
<td>0.0605</td>
</tr>
<tr>
<td>8</td>
<td>∆WPS(c_i, V)</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>∆CND(t, c_i, V)</td>
<td>0</td>
</tr>
</tbody>
</table>

Fig. 6 Mean and std. deviation of valid-only accuracy of full-context classifier for a varying percentage of words sampled from the context, compared to the 10-fold cross validation accuracy of the baseline.

We compare this performance to the average classification accuracy of the baseline using 10-fold cross validation. We do so because, if we were to apply the same approach for the baseline arguments, the classifier would have been trained and tested on the same dataset since the baseline attribute values do not change depending on context. By using the 10-fold cross validation performance of the baseline we get a more realistic estimate of its performance. Figures 6 shows this performance for a given percentage of the total context size of a task. These results show that prediction models trained using our method are resilient to changes in the task vocabulary, maintaining good performance on tasks which use concepts similar to the training task.
4.5 Generality of Substitution Models

We evaluated the potential for cross-task learning through two experiments: First, we trained classification models separately for each task; Figure 7 shows the 10-fold cross validation performance for using data per individual task for the full-context method and the baseline. The mean valid-only accuracy, weighted by the number of candidates per task, is 77.4%, lower than the 80% obtained by the cross-task model.

Second, we trained models on all but one of the tasks, and tested on one that was withheld (Figure 8). We observed that the two most similar tasks in our set (eat soup and cook soup) had the highest performance (26.6% and 18.8% valid-only accuracy, respectively), while for the tasks with the most distinct vocabularies performance was lower. Together, these results indicate that cross-task learning may enable a robot to make strong predictions on previously unseen tasks, particularly in cases of shared vocabulary.

4.6 Physical Robot Demonstration

We applied object substitution to the execution of two mobile manipulation tasks, pack schoolbag and wipe table. A video of the tasks is available at https://goo.gl/LI34bC. In the video, the robot uses autonomous navigation, motion planning and speech generation for prompting user feedback. The speech was produced via a text-to-speech engine which interpreted templates in which objects’ names were inserted. We simulated the robot listening for a user’s response via audio; a researcher sitting in the same room would input a corresponding option via a keyboard. However, our system is compatible with a speech recognition interface that uses distinct and unambiguous options.
Fig. 8 Classification performance for training the classifier on all but one of tasks and testing on the one left out. Each group indicates test performance on the corresponding task.

For both tasks the robot was required to collect and use various objects within a mock household environment, shown in Figure 9. To model the object’s locations, we use a semantic map. With respect to locating missing objects, semantic mapping techniques have been designed to allow the robot to identify and search most likely object locations, often based on probabilistic models of object-location correspondences [18,16]. However, within semantic mapping it is assumed that the object type is known and fixed. Semantic mapping has been extended to hierarchical representations to expand their flexibility [12]; however, using a pre-defined object taxonomy that may render this method unsuitable for a broad variety of environments. Within the context of our work, we utilize semantic mapping to identify and search likely locations for both the original objects specified in the task and the substitution candidates.

Given an HTN plan and a semantic map of the environment, the robot navigated to expected object locations (based on map information), performed object recognition and retrieved items it found. If the robot failed to find an object, it initiated object substitution and identified candidate replacement items. The robot could then search for one of the replacement items, or, if a user is present, first verify the object substitution with the user before proceeding (as shown in the video).

In the pack schoolbag task, the following substitution were performed:

- The robot suggested to substitute tape for glue, which the user accepted;
- The robot first suggested to substitute a quill for a pencil. This substitution illustrates a situation in which the semantic network may lack some contextual information relevant to the scene, in this case, that quills are obsolete for daily handwriting, despite them being functionally similar to pencils and pens. Upon the user’s pragmatic rejection, the robot proposed a pen, which the user accepted;
The robot first suggested to substitute a banana for an apple, which the user rejected due to preference. The robot then suggested substituting an orange, which the user accepted. This preferential rejection illustrates situations in which, despite the fact that the semantic network contains correct information, the user may still disagree with the robot’s output, making human interaction essential for the system to learn user preferences, for example by storing these preferential rejections and exclude candidates classified as valid for a specific user.

In the wipe table task, the robot substitutes first a sponge and then a towel for a rag. Because the most likely substitution according to the learned model, a sponge, is not available in the environment, the robot uses the second candidate selected, a towel.

5 Evaluation 2: Object Substitution in Task Learning

In the second evaluation scenario, we looked at the potential impact of object substitution on robot task learning from human instructions. User study participants were recruited to train a robot how to pack two lunches, using substitutions to adapt learned task models when some food options were not available. We designed the task around a household scenario that users would likely be familiar with, while also allowing users to express their own preferences during demonstration and also through accepted substitutions by selecting their own preferred food items.

5.1 Experimental Setup

The experimental setup, as seen in Figure 10, consisted of a Kinova Jaco2 6 degree of freedom arm mounted to a table with a Microsoft Kinect™ (not
Fig. 10 The physical experimental tabletop setup with Kinova Jaco² mounted to a table with an overhead Microsoft Kinect™ version 2 and lunch boxes, and food items for the lunch-packing task.

shown) mounted to an overhead frame to provide RGBD sensing. Two lunchboxes and a variety of food items were placed on the tabletop for each session; users were instructed to pack each lunch with three items comprising of one fruit, one drink, and one snack item.

The users provided demonstrations by interacting with a web interface, implemented using the Robot Management System (RMS) [45]. This web-based framework handles user and study management and integration with the robot control systems implemented using the Robot Operating System (ROS). The web-based interface used by participants is shown in Figure 11 and incorporates action and parameter selection via a dropdown menu system, a live camera view from the Kinect mounted overhead with annotated text labels for items, a tree view of the current task demonstration, and a text-based chat interface for communicating with an experimenter if they had questions or in cases when robot problems prevented completion of the task. Two primitive actions, \texttt{Pickup(ObjectType)} and \texttt{Store(Lunchbox)}, were available for picking up items and placing them into the specified lunchbox, respectively.

For interactive task learning we use an existing framework for learning HTNs from demonstration [30]. In addition to learning the task structure, the framework provides users with suggestions on how to organize the task hierarchy through action grouping suggestions. In the context of the lunch packing task, the grouping suggestion based on data flow (one action’s output being used by a subsequent action’s input) would suggest users to combine the \texttt{Pickup} action with the \texttt{Store} action resulting in a learned action called \texttt{Pickup/Store} that more efficiently enables the storing of actions with a single command.

During the study, participants were first presented with an easy to use registration page (no email required) in which they entered a username and password. Next, they were redirected to a tutorial describing the lunch-packing task in detail as well as how to use the interface to make the robot execute tasks. This information was accessible throughout the experiment via a help
Fig. 11 The web-based interface used by participants. The interface includes dropdown menus for action and parameter selection, a live camera view with annotated text labels from the Kinect, and a tree view of the current hierarchical task network.

button. After completing the tutorial, users were redirected to the interface as described above where they could execute primitive or learned actions, create new learned actions, and end the experiment after they had packed both lunches.

A total of 21 user study participants were recruited by word of mouth to take part in the study. The experimenters contacted potential participants via email and social media resulting in the majority of subjects having an affiliation with Georgia Tech and participating from the Atlanta area, with a handful of users participating from other national and international locations, including as far away as India. 10 participants were assigned to the suggestions condition and 11 to the no-suggestions condition. No financial compensation was made to participants.

The two experimental conditions were designed to assess the impact of substitution suggestions on the interactive teaching task rather than to assess the quality of the suggestions in the lunch-packing scenario. Suggestions were generated using the approach presented in Section 3. In the base case each time a Pickup primitive action was executed, either by itself or as part of a learned action, the user had to select an object from a drop-down list of 14 possible objects. Users could select items that were not physically present when creating a learned task. This could happen if all copies of the item had already been packed or if it was not correctly detected on the table. When an item was missing in the base case an error message was output in a popup dialog box telling the user which item was not available for pickup.

The suggestion interface was identical except for the missing item case. Instead of an error message users were presented with a dialog box suggesting a substitute item, e.g., "An orange could not be found. Would you instead like to try an apple?" with yes and no buttons. Since substitutions only occur
for learned actions where we have saved the previously selected input, both
conditions were identical until a user created and executed a learned action.
We only placed one of each item on the table except for beverages, which
meant that the second lunchbox would necessarily have some items that were
not available since they were packed in the first lunch. This also means that
number and timing of suggestions was dependent on the users creation and use
of learned actions in the hierarchical task and was not necessarily the same for
each user. Although the suggestion condition requires selection between just
two buttons and the base case requires selection from a list of 14 items in four
categories: (beverages, fruit, main items, and sides). For this experiment, we
structured the suggestions to only allow candidates from the objects available
on the table.

For each participant we recorded the output of the interactive learning
system, a hierarchical task network, as well as timing data including the start
and stop time of the teaching period from page load to the user clicking the
completion button. In addition, we recorded the number and types of items
stored by each participant and additional metadata about the task completion
including the number of primitive actions executed by the robot, the number of
primitive and learned actions executed by the user, and a log from the control
system detailing requested actions and their results. We also computed a score
for each user’s demonstration awarding points for packing the correct number
of items of each type as per the instructions as well as half credit for packing
items that did not meet the recommended allocation, e.g., packing more items
than requested or all items of one type (Formula (2)).

\[
Score(c) = \begin{cases} 
  \text{packed}(c), & \text{packed}_c \leq \text{target}_c \\
  \max(0, \text{target}_c - 0.5 \times (\text{packed}_c - \text{target}_c)), & \text{packed}_c > \text{target}_c 
\end{cases} 
\]

\[
\text{FinalScore} = \sum_{c \in \text{Categories}} Score(c) 
\]

The category score above is computed for each category of items, i.e.,
drinks, fruit, snacks, where \( \text{target}_c \) is the requested number of that type of item
from the instructions and \( \text{packed}_c \) is the number the user packed successfully.
The final score is a summation of each category score (Formula (3)), which we
then normalized for analysis. This score is a means of assessing how well the
final task taught by a user (in this case, a resulting lunch) matches the desired
specification. Our hypotheses in this study were that:

1. the use of item substitution suggestions would reduce the time required to
   train the robot, and
2. the use of item substitution suggestions would not affect task outcome
   quality.
5.2 Results

We observed 100% acceptance rate for object substitution suggestions among users within the suggestions condition. In this section, we evaluate the impact that the suggestions have on task execution time and quality of task outcomes.

Our first analysis is of total task time — the total time it took for participants to complete the study in each of the conditions. We performed an analysis of variance (ANOVA) with type II sum of squares to assess the effect of substitutions on total task time as a response variable. We found a marginally significant effect of substitution suggestions $F(1, 35) = 3.78, p = 0.06$, on the basis of total time, with participants who received substitution suggestions completing the experience in a shorter amount of time ($M = 840.0$ seconds, $SD = 338.5$) than those who did not receive the suggestions ($M = 1121.9$ seconds, $SD = 396.3$) as shown in Figure 12(a).

Total task time consists of three activities: the time the user takes to instruct the robot, the time the robot takes to execute the specified action, and errors in robot task execution. In this study, we are most interested in the first factor, user teaching time. Before we analyze this factor, however, we look at the other two components, execution time and errors. Robot execution time per action remains constant across both conditions, the same object recognition and planning algorithms are used in both cases. Robot errors are task- and user-independent, consisting of missed grasps, replanned grasps due to perceived potential collisions, dropping items between pick and place. Some sources of these errors include miscalibration, misfitting of 3D object models to segmented objects point clouds, as well as other sources. In error cases, users are notified that an error occurred and are asked to re-execute the failed action. By post-processing the execution logs to annotate robot failures we
were able to assess the number and duration of errors affecting each user’s demonstration. We performed another two-way ANOVA with type II sum of squares and found no significant differences between the number of errors for any of the conditions ($M = 3.26, SD = 3.36$), indicating that errors were randomly distributed across the study conditions.

Removing execution and error time from our analysis enables us to more accurately measure the actual time users spent teaching the robot and the impact of suggestions. After performing an ANOVA with user idle time, we found a significant effect in the substitution condition ($F(1,35) = 6.95, p = 0.01$) as shown in Figure 12(b), showing a resulting reduction of mean total time by approximately 191 seconds for the groups receiving substitutions ($M = 344.4, SD = 131.4$) versus the groups not receiving suggestions ($M = 535.4, SD = 230.6$). In summary, object substitution suggestions reduced the time and effort of the users during task training.

Second, we assessed the quality of the taught tasks based on the lunch quality score that gives users credit for matching the requested target allocation and partial credit for packing unrelated items. A comparison of this quality score across conditions enables us to evaluate whether the suggestions altered the quality of the task outcome. As in other analyses we performed an ANOVA on user scores and found no significant differences across suggestion conditions ($F(1,35) = 0.77, p = 0.38$) as shown in Figure 13.

Given these results, we can observe that the total demonstration time and robot idle time results support our first hypothesis, that suggestions can reduce the overall time to provide a demonstration. One qualitative observation of users’ task demonstrations is that not all users teach the task as an experimenter familiar with the hierarchical task network system would. That is, some users would execute the series of Pickup/Store tasks sequentially rather

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{score.png}
\caption{Score of the quality of user demonstrations at adhering to the target allocation of item types for each lunch given in the instructions.}
\end{figure}
than teaching one lunch and then applying the learned task from the first lunch to the second. This results in lower reuse of actions (instead of packing the second lunch with one action it instead takes three Pickup/Store executions) and requires more interface time possibly affecting our measure of robot idle time. The lack of major differences in quality scores as noted in Figure 13 supports the second hypothesis, that the suggestion mechanism would not negatively impact the quality of demonstrations provided by users.

5.3 Additional Evaluation with Crowd Workers

Since our study utilizes a web interface for user interaction, it provides an opportunity to explore nontraditional approaches to study participant recruitment. In particular, we ran a supplemental study in which we recreated the above evaluation with participants recruited from the micro-task marketplace CrowdFlower. Crowdsourcing has facilitated the creation of big data resources for many other fields, and in this study we wanted to explore the feasibility of utilizing micro-task markets as recruiting pools for real-time data collection with real robots. In particular, we were concerned that the longer duration and greater training time associated with most robotics tasks would turn away or impact the performance of many crowd workers, even when payment was adjusted to compensate workers fairly for the additional time spent.

CrowdFlower users were redirected to our study website for the experiment, during which they controlled our robot over the web in real time. After completion of the task, users were asked to enter their CrowdFlower ID and were given a code to enter into the CrowdFlower interface to prove completion. Participants were compensated $0.50 for completing the task and provided an additional $0.50 bonus if they packed items in both lunches. This pay structure was used to discourage users from quitting early with an incomplete demonstration, as well as down-rating the results of users who did not complete the task. In testing, international users located across the globe from our Atlanta-based server were able to successfully complete the task, so we configured the study to allow English-speaking users without geographic restrictions. Except for the study completion code shown to the CrowdFlower users, participants from each recruitment source used the same interface.

In total, 129 unique users were recruited from CrowdFlower. However, despite the large number of crowdsourced users who accepted the job on CrowdFlower, the vast majority did not complete the job, quitting either before the signup page, during the tutorial, or after the first item was packed. The breakdown of participants is shown in Figure 14; only 17 crowd workers completed enough of the task to meet inclusion criteria for the study\(^6\). Of these, 9 were assigned to the suggestions condition and 8 to the no suggestions condition.

We reran both the study time and task outcome quality evaluations on the additional data, performing a two-way ANOVA with type II sum of squares.

\(^6\) In contrast, all 21 word of mouth participants that were recruited completed the task as instructed.
Fig. 14 Crowdsourced user completion rates for the 129 users who accepted the job on CrowdFlower.

We found no significant differences across either the recruitment source condition ($F(1,35) = 0.58, p = 0.45$), Figure 15, or outcome quality ($F(1,35) = 1.93, p = 0.17$). In conclusion, we found that dedicated crowd workers performed largely on par with traditionally recruited participants—crowdsourced users spent slightly less time on the task and also showed modestly lower scores than word of mouth participants, but neither difference is statistically significant. This trade-off between speed of demonstration and quality is an interesting outcome that will require additional research to verify in other task settings and with larger samples of word of mouth and crowdsourced populations. Perhaps word of mouth users, despite being unpaid, are more invested...
in following the instructions as provided, whereas crowdsourced users are more
time-sensitive since they must complete a certain number of jobs per hour to
make a reasonable wage. This is also evident in the number of crowdsourced
users who began or partially completed the study and may have an impact in
applying interactive learning with these users in task settings that are lengthy
or would require significant effort to reset after each user.

One additional consideration when working with crowdsourced users is lan-
guage differences. Although our study was set to English-speaking contributors
only, we found several people using in-browser translation and other languages
in the text chat. This may have had an affect on the ability for the participants
to read and understand the instructions.

6 Conclusion and Future Work

In this paper, we presented a technique for leveraging large-scale semantic
networks to propose object substitutions for robot tasks, allowing for objects
unknown to the robot to be incorporated into the repaired plan by leverag-
ing human input. Object substitution is complementary to affordance-based
reasoning, targeting execution failure cases in which the robot does not al-
ready have sufficient information to perform substitutions autonomously, such
as object models, semantic maps and object affordances. Our method allows
the robot to initiate user interaction by proposing substitutions. Our approach
uses similarity metrics to contrast the missing object with substitution candi-
dates against contexts derived from HTNs.

We evaluated our object substitution algorithm in two scenarios: (1) pro-
viding substitutions during execution, in which case the user can teach the
robot new object models as needed to continue execution, and (2) providing
substitutions in order to enhance task teaching by leveraging existing task
models. Our first evaluation has shown that using concepts extracted from
the task definition is key for achieving reliable substitution candidates; our
method outperforms a context-agnostic baseline. Through the user study in
our second evaluation we were able to demonstrate that object substitution
can increase the effectiveness and speed at which users teach new tasks to a
robot.

One limitation of our work is the assumption that the substitute object has
the same affordances as the original. Relaxing this assumption and allowing
for substitutions at a higher level in the task tree would enable the robot to
achieve the same task in different ways [4], referred to as “climbing a flexibility
ladder” [2]; for example, boiling water using a microwave oven instead of a gas
stove. We anticipate a need for dedicated interfaces which would allow teaching
multiple new affordances prior to the plan repair stage. Existing work has
shown how grasping models can be demonstrated, but these approaches do
not encode specific affordances [21]. Our current work is the building block
for such behavior, enabling the repair process to start by proposing viable
substitutions to the user and repairing under same-affordance assumptions.
An additional aspect of this work that merits further study is the impact of suggestions on the variance of demonstrations from user to user. Since providing suggestions is also a means of guiding users’ demonstrations, making the same substitution suggestions for multiple users will likely bias their demonstrations and result in less variance in users’ input selections, especially since we saw that users unanimously accepted our suggestions in this task scenario. This could be accomplished by suggesting item substitutions at random or providing a choice of substitutions that meet a relevance threshold.

Acknowledgements We would like to thank David Kent and Russell Toris for their help in implementing our work in the Robot Management System and with producing the demonstration video.

References


