AAAI Fall Symposium on Artificial Intelligence and Human-Robot Interaction,
Washington, DC, November 2015.

Towards Robot Adaptability in New Situations

Adrian Boteanu*, David Kent*, Anahita Mohseni-Kabir*, Charles Rich, Sonia Chernova
Worcester Polytechnic Institute
100 Institute Road Worcester, MA 01609
aboteanu@wpi.edu, davidkent@wpi.edu, amohsenikabir@wpi.edu, rich@wpi.edu, soniac@wpi.edu

Abstract

We present a system that integrates robot task execution with user input and feedback at multiple abstraction levels in order to achieve greater adaptability in new environments. The user can specify a hierarchical task, with the system interactively proposing logical action groupings within the task. During execution, if tasks fail because objects specified in the initial task description are not found in the environment, the robot proposes substitutions autonomously in order to repair the plan and resume execution. The user can assist the robot by reviewing substitutions. Finally, the user can train the robot to recognize and manipulate novel objects, either during training or during execution. In addition to this single-user scenario, we propose extensions that leverage crowdsourced input to reduce the need for direct user feedback.

Introduction

Current learning from demonstration (LfD) work focuses on human demonstrators teaching new behaviors to robots. The end goal of this work is to have the robot replicate the demonstrated behavior reliably in as general a manner as possible. Building on this premise of human-demonstrated ground truth, we present a framework proposal that is aimed at leveraging human input to extend the applicability of the tasks a robot is capable of performing into novel environments. Our approach mitigates one of the central drawbacks of executing learned behaviors: if sufficient conditions of the environment in which the task was originally demonstrated change, the task execution will fail. Our proposal allows for incremental adjustments to the task to take place by allowing the user to define both global high-level goals as well as targeted corrections when exceptions occur during execution.

We describe a system that allows a non-expert user to define, monitor, and execute adaptive robot tasks in new environments. We focus this system on pick-and-place tasks for a mobile manipulator. Motivating examples of such tasks include making a fruit basket and packing a school bag, in which a robot must gather a set of household objects (fruit and office supplies, respectively) distributed around its environment and place them in a specific location (a basket or a school bag). We also include a table wiping task to show that the system extends to non-pick-and-place tasks such as cleaning a surface.

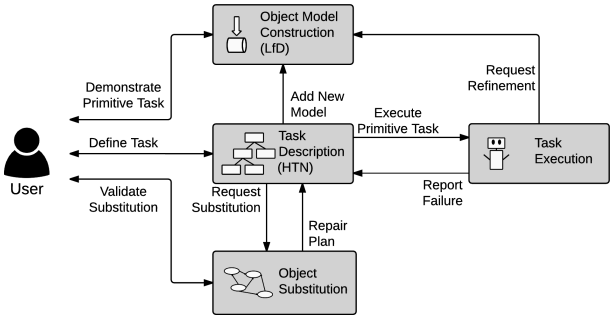

Figure 1: Block diagram for the single user system.

In its minimal implementation, our system is designed to allow a single user to provide a feasible amount of training to first define and then, should the need arise, help adapt a task through feedback and goal demonstration. The process, represented in Figure 1, begins with the user teaching a new task symbolically by combining a predefined set of primitive tasks. Whenever the user teaches a new tasks that uses objects novel to the robot, the user can also train the robot to recognize and grasp the object. The result of the training process is a hierarchical task network (HTN) that uses *primitive tasks* such as *get(object)*, *place(object, object)*, and *wipe(surface)*. The robot then attempts to execute this task and, during execution, the system can mitigate two types of failures: symbolic and execution.

Symbolic failures occur when objects in the task description are missing from the physical environment. Should such errors occur, the robot automatically proposes replacement objects that allow execution to continue. The user has the option of validating these substitutions per task, with two options for rejection (pragmatic and preferential) in addition to acceptance. Performing substitutions at the symbolic planning level has the advantage of allowing propositions that the robot has not yet observed (e.g. objects in another room

*A. B., D. K., and A. M.-K. contributed equally to this paper, working respectively on the object substitution, object model construction, and task teaching.

Copyright © 2015, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

or hidden from view), as well as allowing object propositions for which the robot does not yet have an object model trained. In this way, the user only has to demonstrate primitive task execution for approved substitutions.

Execution failures occur if the user accepts a substitution for which the robot has no model, or if a model has insufficient data to successfully manipulate that object, in that the robot does not know how to successfully execute a primitive task. In this case, the robot will request demonstrations from the user, which are used to construct and refine object models for both recognition and manipulation.

In the last part of the paper, we propose how our learning system can be framed as a cloud robotics framework and discuss the potential advantages of aggregating data across a large group of users.

Background and Related Work

Our method allows for the user to interactively teach a new task as an HTN using primitive actions. HTNs are widely used in artificial intelligence, especially in planning systems and in interactive agents (Nejati, Langley, and Konik 2006; Hinrichs and Forbus 2012). In the context of human-robot interaction, and LfD specifically, HTNs have two key advantages, as compared to flat representations. Both of these advantages flow from the fact that humans naturally tend to think about complex tasks hierarchically.

An HTN is a tree in which the fringe nodes denote *primitive* tasks, i.e., tasks that can be directly executed by the robot, and the other nodes denote *abstract* tasks. The abstract tasks, although not absolutely necessary (you could imagine representing any procedure as a sequence of primitives), provide a *vocabulary* for communication between the human and robot. This is an advantage for discussing the partial completion status of the procedure, for delegating subtasks, for discussing potential problems with execution, and more. Second, the abstract tasks can be *reused* in other situations. Previous work (Mohseni-Kabir et al. 2015) focuses on more complex tasks, such as tire rotation, and shows the advantages of using HTNs instead of a flat representation in a user study with 32 participants and a simulated robot. Current work on HTNs does not account for situations in which objects originally required are missing from the environment. Although the new object input propagates down the tree should the user specify new objects, currently there is no method to recover from execution failures due to missing objects; execution halts and the missing objects must be provided.

The first goal of our work is to learn the task description from the user. To achieve this goal, there is extensive research (involving both robots and virtual agents) on learning from demonstration (Argall et al. 2009), hierarchical task learning: (Garland, Ryall, and Rich 2001; Nicolescu and Mataric 2003; Veeraraghavan and Veloso 2008; Hayes 2013; Rybski et al. 2007), interactive task learning: (Cakmak and Thomaz 2012; Chernova and Veloso 2009; Hayes and Scassellati 2014) and learning from a single task iteration: (Huffman and Laird 1995; Mohan and Laird 2011). We used previous work on combining all these aspects to allow a user to teach new tasks (Mohseni-Kabir et al. 2015).

We suggest object substitutions using bag-of-words contexts derived from the HTN in conjunction with semantic networks. Initially used in information retrieval (Croft, Metzler, and Strohman 2010), bag-of-words models have been used in physical-world domains such as computer vision (Bolovinou, Pratikakis, and Perantonis 2013) and robot localization (Nicosevici and Garcia 2012). A semantic network represents concepts as vertices of a graph and relations between these concepts as edges. In this work, we used two such resources to derive concept similarity: WordNet and ConceptNet. WordNet represents word 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 (Pedersen, Patwardhan, and Michelizzi 2004). On the other hand, ConceptNet aggregates data from a variety of sources, including WordNet, DBPedia, and crowd-contributed information (Liu and Singh 2004). It covers a broader spectrum words, and represents a total of 48 relations such as *part-of*, *is-a*, *used-for*, *has-property*. Because it aggregates information, ConceptNet contains some noise in both the nodes (e.g. different spellings) and the edges (e.g. incorrect edges). Using the ConceptNet graph, Divisi is a measure of word similarity which uses its singular value decomposition (Speer, Arnold, and Havasi 2010). In addition to these similarity measures, we use the Semantic Similarity Engine (SSE) to evaluate relational similarity (Boteanu and Chernova 2015). SSE computes the most similar path pair between two pairs of words using ConceptNet, producing a numerical normalized value, as well as a human readable justification. SSE targets proportional analogies, which are commonly phrased as *A is to B as C is to D*. For object substitution, we can leverage this type of analogy to attempt to establish relational parallelism between a target and a candidate with respect to a context element, by forming the analogy *target:context word::candidate:context word*.

For a robot to interact with potentially novel objects provided as substitutions, we turn again to learning from demonstration. Of particular interest is goal-based learning (Chung et al. 2014; Kent and Chernova 2014; Toris, Kent, and Chernova 2015), where demonstrations provide task goals rather than the actions required to complete them. This provides more adaptability during execution in that it allows the robot's planners and controllers to determine how best to execute a task according to its present environment. Demonstrations themselves can be provided by a variety of methods, including teleoperation of the robot by joystick, mouse, or keyboard, and kinesthetic teaching, in which a user physically moves the robot (Argall et al. 2009). Finally, integrating crowdsourcing with robotics applications can provide large amounts of varied feedback and demonstration data while reducing the amount a single user needs to give (Kehoe et al. 2015). It has been used in robotics applications extensively, including the aforementioned goal-based learning, interface testing (Crick et al. 2011), learning natural language commands (Tellex et al. 2011), and learning human-robot interaction behaviors (Breazeal et al. 2013).

Based on our research, there is no previous work that has integrated all of these components into a single system, including task learning, object substitution, and recognition and grasp learning. This work aims to integrate the most useful aspects of the above work into an end-to-end system for adaptable robot task execution in new environments.

Single User System

In this section, we present the full single user system, shown in Figure 1. The system has four major components, with varying levels of user interaction: *task description*, *task execution*, *object model construction*, and *object substitution*. Each component is described in more detail later in this section, but the general flow is as follows. The workflow of using the system is shown in Figure 2. The user first teaches a task, represented as a hierarchical task network stored in the *task description* component. For any primitive tasks specified in the task description which the robot does not already know how to complete, for example if a user adds *get* on a novel object, the *task description* component requests a new model be added by the *object model construction* component, which in turn asks the user for demonstrations of the new primitive task.

Once the task has been learned, the *task description* component executes it by sending a sequence of primitive tasks to the robot's execution system, represented by the *task execution* component. The *task execution* component reports any failure to complete a primitive task back to the *task description* component, which in turn requests a substitution from the *object substitution* component. Upon finding a suitable substitution, the *object substitution* component repairs the plan with the new object. If the repaired plan contains primitive tasks that are new to the robot, *task description* again requests the *object model construction* component to add any new object models required by the new primitive tasks. Finally, the *task description* component sends a new primitive task from the repaired plan to *task execution*. Execution will continue in this manner until the task is successfully completed. The *task execution* component can also request refinements to existing object models if they are not providing sufficient data for manipulation, in which case the user will be asked to provide further demonstrations.

To illustrate the single user scenario, we implemented the task of making a fruit basket on a mobile robot manipulator. We use this task as the primary example for the single user scenario because it provides a both simple and typical case to demonstrate our approach. We represent the task as an HTN in which retrieving a piece of fruit is a repeatable learned subtask, which is applied to successively. During execution, object substitutions occur when one of the required fruit is not available. The video included [1] with this paper shows the user teaching the task, followed by the robot autonomously executing it. The robot prompts the user for feedback when a substitution requires attention and for demonstrations when it encounters a *get* task on a novel object.

[1] https://www.youtube.com/watch?v=Ry3QtbSoOfM

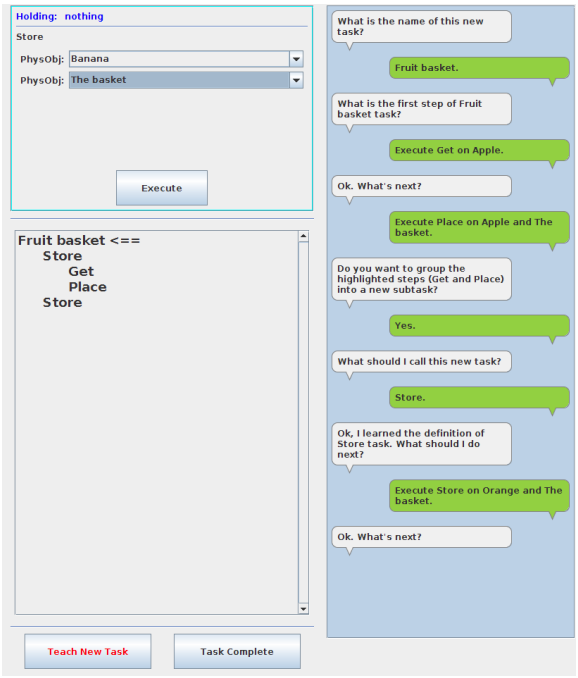

Figure 4: Graphical user interface for teaching a task.

Task Description

First, we focus on the problem of how a robot can efficiently learn a hierarchical task model from a human teacher who is an expert in the domain, but not in robot programming. Our approach integrates learning from demonstration with hierarchical task networks, and our approach consists of viewing LfD as a collaborative discourse. In a collaborative discourse, both participants, the human teacher and the robot learner, are committed to the shared goal of the interaction (in this case, for the robot to learn a new task) and both actively contribute towards achieving that goal. Collaborative discourse theory (Grosz and Sidner 1986) provides a foundation for both the algorithms and the implementation of our system. We are using an open-source tool, called Disco (Rich and Sidner 2012), based on collaborative discourse theory in our implementation. We focus on learning from a single demonstration. This is possible because of the bi-directional communication between the teacher and the learner. For more details, please refer to (Mohseni-Kabir et al. 2015).

We show an example HTN for making a fruit basket in Figure 3. The fruit basket HTN uses two primitive task types: *Get*, and *Place*, with their respective input and output types shown. For example *Get* takes an object as input and returns the object as output. The fruit basket task has only a single abstract task, *Store*, which decomposes into *Get* and *Place* primitives with a temporal constraint between them. We have used data flow in HTNs to learn temporal constraints from a single demonstration. Our method for learning temporal constraints is discussed in details in (Mohseni-Kabir, Rich, and Chernova 2014).

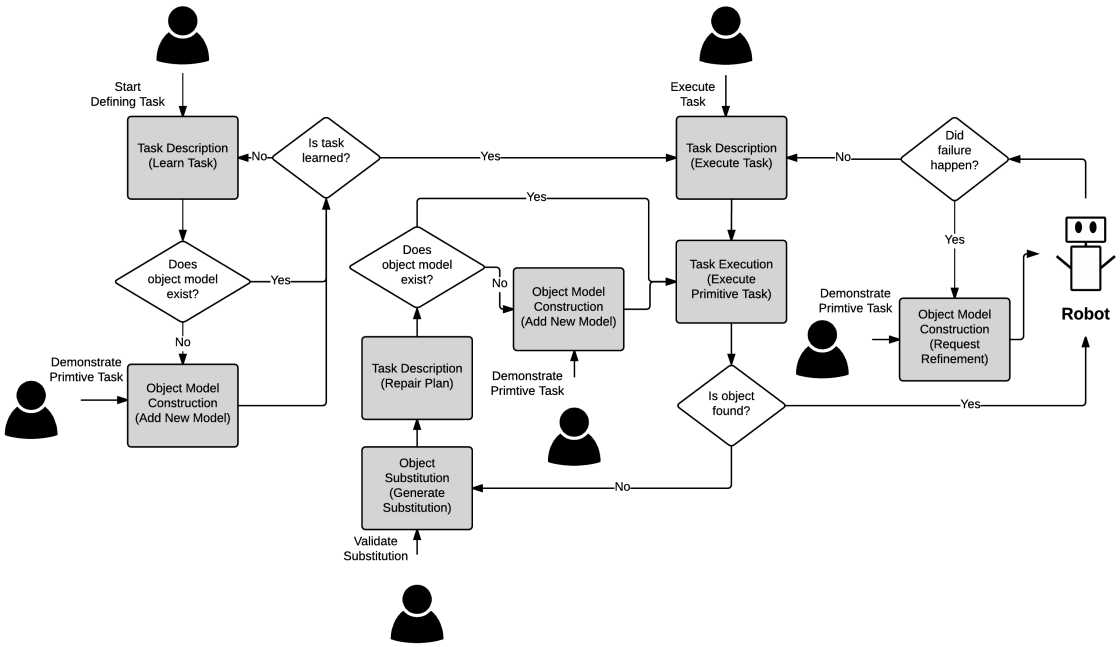

Figure 2: System flow diagram.

Figure 5 illustrates an example scenario of the interaction between a user and the *task description* component which learns the HTN of the fruit basket task depicted in Figure 3. The user starts by teaching a new abstract task called *Fruit-Basket*, adding its two primitive subtasks: *Get* (applied to the apple) and *Place* (applied to the apple and the basket). Then, in lines 8–12, the robot suggests the user to group the *Get* and *Place* tasks, and the user accepts the suggestion, leading to the creation of a new abstract task type called *Store*. Continuing in lines 15–19, the user similarly applies this new abstract task, *Store*, to the other three fruits. The interaction is ended by the user in lines 20–23. Other prior work (Mohseni-Kabir et al. 2015) discusses the algorithms developed for generating grouping and repetition suggestions and shows that these suggestions significantly assist the users in building HTNs.

Figure 4 shows a partial view of the graphical user interface we have developed for our preliminary experiments. This user interface has the same functionality as the illustrated scenario in Figure 5. The left side of the GUI (omitted from figure) contains buttons that select a primitive or abstract task to execute or add to the definition of a new abstract task. The primitive task types are fixed. New abstract task buttons are added whenever a new abstract task type is learned. The top middle area of the GUI is mainly for specifying inputs, if any, as needed before executing/adding a task. Each input can be selected from a drop-down list of objects in the environment of the correct type. When the user presses the Execute button, the robot adds or executes the selected primitive or abstract task. Please note that we distinguish between executing and adding the selected task but we use the same Execute button for both cases. If the user is in the middle of teaching a new abstract task, pressing the Execute button will just add the selected task to the definition of the abstract task; however if the user has not started teaching a new abstract task, pressing the Execute button results in the robot immediately performing the selected task. The middle area of the GUI below the input selection area is the main informational display, which shows the current hierarchical structure of the learned tasks.

Finally, the right side of the GUI contains a "chat" window in which the robot can ask the user questions and the user can reply. This is where the robots helpful suggestions appear and where the user can accept or reject them. Based on the lessons learned in our prior work (Mohseni-Kabir et al. 2015), the robot engages more actively with the user in the interactions presented here; in addition to giving repetition and grouping suggestions, the robot also gives more feedback on the completion of the tasks, the definition of already known tasks, and the state of the execution.

Object Model Construction

The primitive tasks defined in the *task description* component may or may not be known by the robot. In the event that a primitive task unknown to the robot is added, either while the user is initially teaching the task or by the *object substitution* component's plan repair (see the Object Substitution section below), the *object model construction* component will request demonstrations of the new primitive task. In the pick-and-place domain, if the robot must pick up an

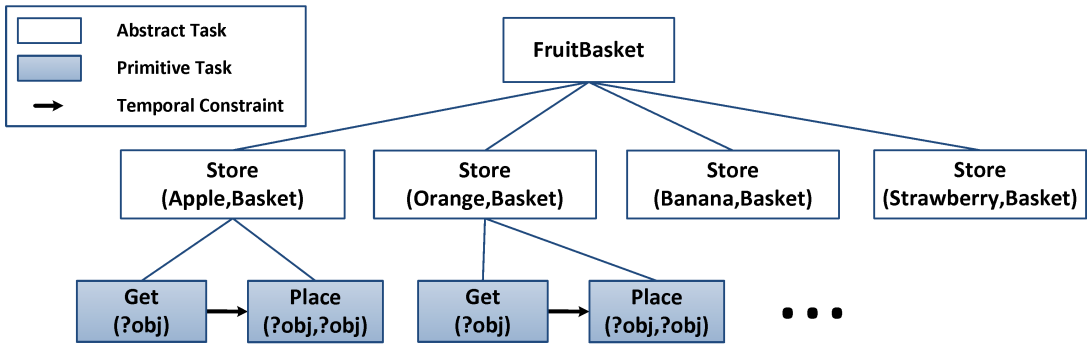

Figure 3: An example hierarchical task network for making a fruit basket.

User: "Teach a new task."
Robot: What is the name of this new task?
User: "Fruit basket."
Robot: What is the first step of fruit basket task?
User: "Execute Get on the apple."
Robot: Ok. What's next?
User: "Execute Place on the apple and the basket."
Robot: Ok. Do you want to group Get and Place into a new subtask?"
User: "Yes."
Robot: What should I call this new task?
User: "Store."
Robot: I learned the definition of store task. What should I do next?
User: "Execute store on the orange and the basket."
Robot: Ok. What's next?
...
...
...
User: "I'm done teaching the fruit basket task."
Robot: I learned the definition of fruit basket task. Are you going to teach me another task?
User: "No."

Figure 5: A typical interaction example teaching the fruit basket task.

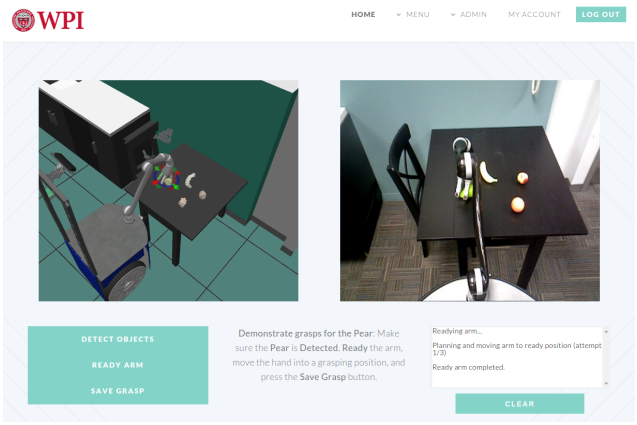

Figure 6: Grasp demonstration interface. The left window provides interactive marker control of the robot's end effector position and orientation, the right window shows a camera feed from the robot, the set of buttons in the bottom left allows the user to save new demonstrations, and the bottom right text box provides feedback from the robot.

object that it does not have an object model for, it requests that the user provide a set of grasp demonstrations for that object.

Depending on the capabilities of the robot, one of two demonstration approaches can be used. The first is a web interface designed for non-expert users, shown in Figure 6. The interface is implemented using the Robot Management System[2], and can be run on any computer, tablet, or mobile device with a web browser. Using a mouse or touchscreen, the user can teleoperate the robot's end effector via interactive markers provided in the left half of the interface. Once the end effector is in an appropriate pose for grasping, the user can save the demonstration, storing the segmented point cloud of the grasped object, the object name, and the end effector pose. Alternatively, if the robot has a compliant arm, the user can perform demonstrations through kinesthetic teaching by physically moving the end effector into a grasp pose. This eliminates the need for an interface and provides a natural method of collecting grasp data.

The data collected from the demonstrations provide input to the object model generation algorithm. Models are generated using a graph-based point cloud registration algorithm, which iteratively selects and merges pairs of point clouds based on their pairwise registration score. The registration algorithm also transforms the associated grasp poses stored during the demonstration process, resulting in 3D object models suitable for recognizing and grasping previously unknown objects. The object model generation algorithm is described in more detail in (Kent, Behrooz, and Chernova 2014) and (Kent and Chernova 2014).

Task Execution

The *task execution* component of the system contains all of the necessary robot controllers to execute the primitive tasks sent by the *task description* component. While executing primitive tasks is often straightforward, this component also

[2]Documentation and source code are available at http://www.ros.org/wiki/rms

makes use of user feedback to recover from both encountering a new type of primitive task (e.g. picking up a novel object) and general execution failure of a primitive task.

Similar to adding new object models, the robot can interrupt task execution by using the *object model construction* component's demonstration system to recover from failed executions of a known primitive task. For example, if the robot repeatedly fails to pick up an object for which it already has a trained object model, it can request that the *object model construction* component gather additional grasp demonstrations from the user. These new grasp demonstrations along with the previously generated model can then be used as input to the object model generation algorithm, resulting in a refined object model.

Object Substitution

Humans are able to improvise and adapt to unforeseen changes in their environment when carrying out a task, for example using different tools for the same purpose. Robots executing tasks, however, lack sufficient flexibility to improvise. The field of plan repair addresses the problem of changing an existing plan locally such that execution can resume should a halting problem be encountered (Gerevini and Serina 2000; Van Der Krogt and De Weerdt 2005; Fox et al. 2006; Koenig and Likhachev 2002). However, repairing plans assumes that the domain is unchanged. In addition to allowing execution in entirely different environments, such as cross-user execution, adapting existing plans can compensate for less drastic changes such as unusable (e.g. dirty dishes) or lost items when executing tasks in the same environment.

We present *object substitution* as an extension to plan repair, which allows for the domain to be extended to include new objects. This is necessary if a task is executed in a new environment, or if the objects in the environment changed from those used to teach the task. Assuming that the robot cannot find an object specified in the task (*target*), our method leverages semantic similarity as derived from general purpose semantic networks to propose and evaluate alternatives (*candidates*). Since the robot may have partial knowledge of the objects present in the environment, the substitution is performed at a symbolic level.

Our method consists of three steps: generating candidates for the target, extracting context from the HTN, and evaluating the fitness of each candidate within said context. We generate candidates by selecting words that are connected through similar affordance-like edges in ConceptNet. The edges we focus on are *Used-For*, *Has-Property* and *Capable-Of*. For example the concepts *apple* and *pear* are both linked to *good to eat* by *Has-Property* edges, while *apple* and *cookie* are linked to *edible* via the same edge type. For the context, we parse and lemmatize all node and input tags from the HTN, for example the task label *GetApple* produces the words *get, apple*, and the location input name *Object:Basket* produces *basket*.

Using semantic networks, we evaluate substitution using a number of measures: ConceptNet Divisi similarity, WordNet path length similarity and SSE analogical similarity. We model substitutions using two criteria: (1) the similarity between a candidate and the target (using Divisi and WordNet path similarity); (2) the similarity between each context word are the target and the candidate (using, in addition to Divisi and WordNet path similarity, the SSE similarity score).

To evaluate our model we conducted a supervised classification experiment in which we constructed a model using the similarity metrics described above, with the substitution annotation as the class attribute. We generated candidates for the *fruit basket*, *wipe table* and *pack schoolbag* tasks. We annotated these candidates as either suitable or unsuitable substitutions using two experts. The final annotation was reached by consensus or through discussion.

We next constructed a dataset from the metrics and annotations. Each instance in the dataset contained the average similarity metric values per context and the substitution quality annotation; we removed all word-related information such as the task, target, or candidate. We trained a random forest classifier using ten-fold cross validation[3]. The classifier achieved a performance of 98.93%. Of substitutions selected as valid by the classifier, 88.73% were correct; we computed this percentage using the following formula:

$$\frac{valid\ substitutions}{valid\ substitutions + false\ positives + false\ negatives} * 100.$$

Once a model has been trained for predicting substitutions, it can be used in future task executions. Since there is a chance of erroneous substitutions, we propose prompting the user when either new substitutions or low-confidence substitutions are attempted for each task. Our framework accommodates three user responses:

- *Acceptance*: the user accepts the substitution and the system continues execution;
- *Pragmatic Rejection*: the user rejects the substitution citing objective reasons;
- *Preferential Rejection*: the user rejects the substitution in order to express a personal preference. In this case, although the substitution is possible from a pragmatic standpoint, the user does not allow it due to personal preference.

We divide rejection into two categories because pragmatic rejection generalizes across users, and preferential rejection should be used for only a single user. As the system gains feedback on each *(task, target, candidate)* substitution proposal, it can learn a confidence value for both the general case as well as for personalizing substitutions to user preference. This behavior leaves the feedback open for use in a multi-user environment (see the Cloud Robotics Framework section). Using feedback in conjunction with the SSE-derived explanations, we project the potential of the system deriving substitution fitness based on edge information in addition to the similarity metrics.

Cloud Robotics Framework

In the above section we described a scenario in which only a single user contributes tasks, feedback, and demonstra-

[3]The classification model was obtained using Weka 3.7.12, with the default configuration parameters

tions. Now we propose a number of incremental learning approaches that will allow the robot to expand its knowledge by leveraging input from a large group of separate users, while also reducing the burden placed on the single user to provide feedback and demonstrations. For this scenario, we expand the system design by adding a database to which the crowd can contribute.We use crowdsourced input to improve substitution feedback and expand the set of object models for recognition and manipulation.

Requesting demonstrations for new objects can be time consuming for a single user, particularly if many object models are missing. The object model construction algorithm is designed to work with unordered demonstrations, thus allowing models to be constructed from data collected from multiple users. Data collection is easily extended to the crowdsourcing domain as well. Since it's implemented in the Robot Management System, the grasp demonstration interface (Figure 6) can allow connections from remote online users, allowing anyone to provide grasp demonstrations for the robot. The interface also includes elements to promote situational awareness for remote users, such as the camera feed, the feedback window, and the task instructions. This allows the crowd to contribute to a large database of grasp demonstrations, which is then used to inform object model creation.

The single user may still provide grasp demonstrations and refinements to supplement the crowdsourced data, both for online error recovery and if they have personal preferences for how their objects are manipulated. By crowdsourcing model construction for a known set of objects in an environment before the robot attempts any task execution, though, the amount of online training required from the user can be significantly reduced.

In the case of object substitution, the crowd can contribute pragmatic quality judgments that evaluate a specific substitution for a specific task, providing judgements for *(task, target, candidate)* tuples. As in the case of a single-user scenario, the system can build confidence towards a decision as the number of judgments increases. To evaluate how suitable the crowd's responses are for this purpose, we conducted a survey[4] on the substitution candidate set we used in Object Substitution section. The survey consisted of a brief description of the general setting, the task name with no other details about the task, the target object name, and the substitution name. There was an agreement of 80% between the crowd and the expert annotations. This shows that the crowd is a reasonable source of substitution annotations, even with minimal information about the task. We note that this prediction power is comparable to our substitution system's classification performance. By using substitution evaluations derived both algorithmically and from the crowd, we are able to limit the need for direct user feedback only to edge cases and for personalization.

[4]The survey was conducted on the Crowdflower platform (www.crowdflower.com) using a minimum of five respondents per candidate. Questions were available to only participants from English-speaking countries.

Conclusion

We described a system which allows a non-expert user to specify and refine robot tasks in an abstract-to-concrete manner. Starting from the global task definition, the robot requests feedback as needed, reducing the initial and global amount of training data the user has to give. The robot implements corrective behaviors that dynamically request refinements from the user as needed. First, in defining the initial task, the system proposes structures that improve the task's modularity, which in turn improves the robot's autonomy. Second, we mitigate discrepancies between the task definition and the environment by allowing the robot to propose object substitutions in order to adapt the plan as needed. Third, the robot requests demonstrations if the tasks require the manipulation of objects for which no models exist, or if the existing models need refinement. The complete system maintains adaptability in task performance for new environments, while requesting user feedback and demonstrations only when necessary. Additionally, we further minimize the amount of user input required by expanding the single-user scenario with a series of improvements that leverage input from the crowd.

Acknowledgments

This work is supported in part by the Office of Naval Research grants N00014-13-1-0735 and N00014-14-1-0795, and National Science Foundation award number 1149876.

References

Argall, B. D.; Chernova, S.; Veloso, M.; and Browning, B. 2009. A survey of robot learning from demonstration. *Robotics and Autonomous Systems* 57(5):469 – 483.

Bolovinou, A.; Pratikakis, I.; and Perantonis, S. 2013. Bag of spatio-visual words for context inference in scene classification. *Pattern Recognition* 46(3):1039–1053.

Boteanu, A., and Chernova, S. 2015. Solving and explaining analogy questions using semantic networks. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*.

Breazeal, C.; DePalma, N.; Orkin, J.; Chernova, S.; and Jung, M. 2013. Crowdsourcing human-robot interaction: New methods and system evaluation in a public environment. *Journal of Human-Robot Interaction* 2(1):82–111.

Cakmak, M., and Thomaz, A. L. 2012. Designing robot learners that ask good questions. In *ACM/IEEE International Conference on Human-Robot Interaction*, 17–24. ACM.

Chernova, S., and Veloso, M. 2009. Interactive policy learning through confidence-based autonomy. *Journal of Artificial Intelligence Research* 34(1):1.

Chung, M.; Forbes, M.; Cakmak, M.; and Rao, R. 2014. Accelerating imitation learning through crowdsourcing. In *Robotics and Automation (ICRA), IEEE International Conference on*, 4777–4784.

Crick, C.; Osentoski, S.; Jay, G.; and Jenkins, O. 2011. Human and robot perception in large-scale learning from demonstration. In *ACM/IEEE International Conference on Human-Robot Interaction (HRI 2011)*.

Croft, W. B.; Metzler, D.; and Strohman, T. 2010. *Search engines: Information retrieval in practice*. Addison-Wesley Reading.

Fox, M.; Gerevini, A.; Long, D.; and Serina, I. 2006. Plan stability: Replanning versus plan repair. In *ICAPS*, volume 6, 212–221.

Garland, A.; Ryall, K.; and Rich, C. 2001. Learning hierarchical task models by defining and refining examples. In *International Conference on Knowledge Capture*, 44–51.

Gerevini, A., and Serina, I. 2000. Fast plan adaptation through planning graphs: Local and systematic search techniques. In *AIPS*, 112–121.

Grosz, B. J., and Sidner, C. L. 1986. Attention, intentions, and the structure of discourse. *Comput. Linguist.* 12(3):175–204.

Hayes, B., and Scassellati, B. 2014. Discovering task constraints through observation and active learning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*.

Hayes, B. 2013. Social hierarchical learning. In *Proceedings of the 8th ACM/IEEE International Conference on Human-Robot Interaction (HRI 2013) Pioneers Workshop*.

Hinrichs, T. R., and Forbus, K. D. 2012. Learning qualitative models by demonstration. In *AAAI*.

Huffman, S. B., and Laird, J. E. 1995. Flexibly instructable agents. *Journal of Artificial Intelligence Research* 3:271–324.

Kehoe, B.; Patil, S.; Abbeel, P.; and Goldberg, K. 2015. A survey of research on cloud robotics and automation. *Automation Science and Engineering, IEEE Transactions on* 12(2):398–409.

Kent, D., and Chernova, S. 2014. Construction of an object manipulation database from grasp demonstrations. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, 3347–3352.

Kent, D.; Behrooz, M.; and Chernova, S. 2014. Crowdsourcing the construction of a 3d object recognition database for robotic grasping. In *Robotics and Automation (ICRA), 2014 IEEE International Conference on*, 4526–4531.

Koenig, S., and Likhachev, M. 2002. Improved fast replanning for robot navigation in unknown terrain. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 1, 968–975. IEEE.

Liu, H., and Singh, P. 2004. Conceptnet - a practical commonsense reasoning tool-kit. *BT technology journal* 22(4):211–226.

Mohan, S., and Laird, J. E. 2011. Towards situated, interactive, instructable agents in a cognitive architecture. In *AAAI Fall Symposium Series*.

Mohseni-Kabir, A.; Rich, C.; Chernova, S.; Sidner, C. L.; and Miller, D. 2015. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, 205–212. ACM.

Mohseni-Kabir, A.; Rich, C.; and Chernova, S. 2014. Learning partial ordering constraints from a single demonstration. In *Proceedings of the 2014 ACM/IEEE international conference on Human-robot interaction*, 248–249. ACM.

Nejati, N.; Langley, P.; and Konik, T. 2006. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning*, 665–672. ACM.

Nicolescu, M. N., and Mataric, M. J. 2003. Natural methods for robot task learning: Instructive demonstrations, generalization and practice. In *AAMAS*, 241–248.

Nicosevici, T., and Garcia, R. 2012. Automatic visual bag-of-words for online robot navigation and mapping. *Robotics, IEEE Transactions on* 28(4):886–898.

Pedersen, T.; Patwardhan, S.; and Michelizzi, J. 2004. Wordnet:: Similarity: measuring the relatedness of concepts. In *Demonstration papers at HLT-NAACL 2004*, 38–41. Association for Computational Linguistics.

Rich, C., and Sidner, C. L. 2012. Using collaborative discourse theory to partially automate dialogue tree authoring. In *Proc. Int. Conf. on Intelligent Virtual Agents*, 327–340.

Rybski, P. E.; Yoon, K.; Stolarz, J.; and Veloso, M. M. 2007. Interactive robot task training through dialog and demonstration. In *ACM/IEEE Int. Conf. on Human-Robot Interaction*, 49–56.

Speer, R.; Arnold, K.; and Havasi, C. 2010. Divisi: Learning from semantic networks and sparse svd. In *Proc. 9th Python in Science Conf.(SCIPY 2010)*.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M.; Banerjee, A.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*.

Toris, R.; Kent, D.; and Chernova, S. 2015. Unsupervised learning of multi-hypothesized pick-and-place task templates via crowdsourcing. In *Robotics and Automation (ICRA), 2015 IEEE International Conference on*, 4504–4510.

Van Der Krogt, R., and De Weerdt, M. 2005. Plan repair as an extension of planning. In *ICAPS*, volume 5, 161–170.

Veeraraghavan, H., and Veloso, M. 2008. Learning task specific plans through sound and visually interpretable demonstrations. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2599–2604.