Functional Object-Oriented Network: Considering Robot’s Capability in Human-Robot Collaboration

David Paulius, Kelvin Sheng Pei Dong, and Yu Sun

Abstract—In this work, we explore human-robot collaborative planning using the functional object-oriented network (FOON), a graphical knowledge representation for manipulations that can be performed by domestic robots. The knowledge retrieval procedure, used for acquiring the necessary steps (as a task tree) to solve a given problem, is modified to account for weights that reflect the difficulty of performing motions in a universal FOON. These weights are given as success rates, which describe the likelihood of a robot successfully completing the action(s) on its own. However, certain manipulations may be too difficult for it to perform on its own based on its own physical limitations. To make it easier for the robot, a human can assist to the minimal extent needed to perform the activity to completion by identifying those actions with low success rates for the human to do. From our experiments, it is shown that tasks can be executed successfully with the aid of the assistant. Our results show that the best task tree can be found with the adequate chance of success in completing three activities while minimizing the effort needed from the human assistant.

I. INTRODUCTION

In the ideal world, we want to build a robot that is capable of performing all tasks for a person who is unable to do the task themselves. However, to perfectly design such a robot is an exceptionally daunting task. For one, the variability of the environment in which robots work is very dynamic and is likely to feature objects of different shapes and sizes, while also varying in the position of objects. Secondly, robot motions are not guaranteed to be 100% reliable and can fail occasionally. A robot’s capability to perform human-like manipulations heavily depends on how it is made; features such as the type of end-effector it has (e.g. what type of gripper it uses, how many fingers it has, etc.), the number of degrees of freedom and joints it has for its appendages, and the freedom (or lack of) to navigate the environment in search for the items it requires for problem solving. We can leverage the available resources or capabilities of the robot by introducing collaboration with a human assistant. Human-robot collaboration is an ongoing research area that focuses on robot and human interaction [1], [2], [3], [4] to solve a common goal and has been extensively studied for areas such as social interaction [5], [6], [7], [8], [9], coordinated tasks [10], [11], [12] rehabilitation [13], [14], and care for the elderly or disabled [15], [16], [17]. Motivated by this idea, we explore human-robot collaboration using our knowledge representation called the functional object-oriented network (FOON) [18], which we briefly review in Section II to demonstrate how task trees can be executed.

In this paper, we aim to address the question of how a robotic system can use the FOON knowledge representation for task planning through knowledge retrieval from this network. Previously, our representation was considered as a strictly procedural representation, and FOON was free from information related to task planning; also, motions were treated equally and assumed to all be 100% reliable in execution, but this does not match the reality of real robots. Therefore, weights are better able to capture uncertainty. Furthermore, weights would also be set for robots with different architectures to reflect their ability to perform certain manipulations. Therefore, we now introduce success rates as weights to identify a task sequence that is best suited to the current situation. In addition to this weighted approach, we also consider problem solving as a collaborative effort between a robot and an assistant. Ideally, a robot can be programmed with the necessary skills to solve these problems on its own, based on sequences in FOON; however, there may be instances in which robots are not fully equipped to perform such manipulations on their own due to
Because of this, we have posed this as a problem of human-robot collaboration, where a human can work with a robot to solve manipulation problems together using the knowledge retrieved from a FOON. In this case, the human acts as an assistant to the robot who has all of the knowledge needed to perform the tasks; given a goal, the robot determines the best course of action through task tree retrieval and collaborates with the human to solve the problem posed to the robotic entity. This not only makes things easier for the human person in reducing the complexity of solving the task (in comparison to doing it on his/her own), but it also improves the chances of the robot succeeding in task tree execution. To the best of our knowledge, our representation is the only one that considers success rates as a means for capturing uncertainty for task planning. In Section IV we introduce a variant knowledge retrieval algorithm that considers the likelihood of successfully performing all the actions (again, as functional units) in a task tree with varying levels of involvement of a human assistant. We discuss our experiments and results and show that such a system effectively works at making tasks easier and more successful overall, even with a robot of limited capabilities.

II. Functional Object-Oriented Network

The functional object-oriented network (FOON) represents manipulations as seen in cooking activities (and can possibly be extended to other manipulation tasks) by capturing the objects and the activity’s motions within a graphical structure. Originally proposed in [18], we introduced FOON as a graphical knowledge representation that represents high-level concepts related to human manipulations for service robotics tasks. This representation is motivated by the theory of affordance [20], wherein it describes the underlying uses and effects of objects afforded to the robot, which are innately depicted though edges connecting objects to actions. As we have introduced before, the purpose of this knowledge representation is to serve as a source of knowledge for a robot to determine how it can go about solving a problem.

To represent activities, a FOON contains two types of nodes: object nodes and motion nodes. Object nodes symbolize any object that is manipulated passively or actively within the activities in FOON, while motion nodes symbolize the type of manipulation that connected object nodes are participating in at a given period of time. These motion nodes can be actions commonly performed in cooking such as pouring, cutting, or stirring, but they can also be extended to manipulations in other domains. As an example shown in Figure 2 describing the task of stirring a cup of tea using a spoon as a functional unit, the active object in this case would be a spoon object that acts upon a passive object tea cup which contains the ingredients tea and sugar. The stirring manipulation is represented here with a motion node with the label stir. The joint representation of both object and motion nodes make FOON a bipartite network. As with typical bipartite networks, object nodes can only connect to motion nodes, and motion nodes can only connect to object nodes. Edges are directed to inherently indicate an order or sequence of actions within the network.

A. Creating a FOON

To suitably capture the essence of actions within a FOON, we denote a collection of object nodes and motion nodes that describe a single action within an activity as a functional unit. A functional unit describes the change in the states of objects used in a manipulation action before and after execution; it is important to consider the change in an object’s state to identify when an action has been completed [21]. Each functional unit contains a single motion node describing the action. Typically, an activity is represented by a series of functional units that are connected by common object nodes. Input object nodes describe the required state(s) of objects needed to perform the task, and output object nodes describe the outcome of performing the action on those input object nodes. Some actions do not necessarily cause a change in all input objects’ states, and so there may be instances where there are fewer output object nodes than inputs.

A FOON is constructed from annotating human demonstrations from videos and converting them into the FOON graph structure; in this annotation process, we note the actions, objects, and state changes (as functional units) that occur to produce a specific meal or product. At this present moment, we do this by manually annotating the videos by hand, but efforts have been made to investigate how we can do this in a semi-automatic process [22]. A FOON that represents a single activity is referred to as a subgraph; a subgraph contains functional units in sequence to describe the objects’ states before and after each action occurs, the time at which the action happens within the activity, and what objects are actively or passively being manipulated. Two or more subgraphs can be merged together to form a universal FOON, which is simply a FOON that contains information from several sources of knowledge for any type of manipulation; this universal FOON could propose variations of methods to recipes. The merging procedure is simply a union operation done on all functional units from each subgraph we wish to combine; as a result, duplicate functional units are
eliminated. Duplicates among functional units are indicated by overlap, suggested by: 1) the same number of input/output objects, 2) the commonality of object-state types, and 3) the same motion node type.

**B. Integrating Weights into a FOON**

Up to this point, we have yet to evaluate the innate capability of a robot in task planning with FOON. Previously in [18], [23], all motions were considered to have equal weights in a FOON, implying that all motions can be executed by any robot. In other words, the assumption was that any robot should be able to perform the manipulations as well as any other robot or even humans. However, this does not match the reality of current technology since robots come in different shapes and sizes, meaning that they may not all perform the same manipulations equally in terms of precision. As much as we would like any robot to perform every and any motion, it is difficult to achieve human-like dexterity as observed in demonstrations. For these reasons, we introduce weights into the FOON representation to indicate how challenging a manipulation is to perform. The values are based on: 1) physical capabilities of the robot, 2) past experiences and ability in performing the action, and 3) the tools or objects that the robot needs to manipulate.

The weights in this paper reflect the robot’s success rate of performing a given action. Success rate weights (as percentages) are assigned to each functional unit’s motion node and are based not only on the manipulation type, but also on the objects contained within the functional unit. In Figure 3, success rate weights are assigned to each functional unit with values ranging between 0 and 1. To guarantee that a robot can perform such motions, weights can be used as heuristics for knowledge retrieval; even though several robots will be equipped with the same universal FOON (meaning they will all have knowledge of the same sequence of actions for all activities), different weights will be assigned to them based on the robot’s attributes, which can ultimately result in potentially very different task trees. Hence, it is important to note that these weights must first be defined for each type of robot. For instance, a small robot like Aldebaran’s NAO would not be able to handle a knife well enough to chop vegetables since it cannot exert the force needed to cut them as well as lacking the dexterity to do so properly.

To determine representative weights for a robot, we can do so empirically, where, given a manipulation task, we measure the frequency of successful manipulation trials. It is also important to note that when conducting these experiments, one should vary the attributes of the tools or ingredients the robot is manipulating to better capture the conditions in which a robot can sufficiently perform those motions. However, this is not a trivial matter, as motions are likely to have a large number of variables to tune and learn; for example, when learning to scoop with a spoon, several parameters can be tuned such as the point at which the tool is grasped, the weight of the contents in/on the spoon, and the matter or substance that is being scooped. Therefore, in the experiments discussed later within this paper, we assign weights to motions based on our experiences in teaching the robot to perform certain motions. Motions that cannot be executed by a robot were assigned a success rate of 0.01 (or 1%), while other motions would be assigned higher values which can vary between 0.8 and 0.95 (80 - 95%). Overall, evaluations of the capability of a robot performing the tasks represented in FOON should be based on the robot’s perception, strength, dexterity, and reach within its workspace.

**III. USING FOON FOR MANIPULATION PROBLEMS**

A FOON can not only be used for representing knowledge, but it can also be used by a robot for problem solving. Given a problem defined as a goal, a robot can perform knowledge retrieval to obtain a subgraph that contains functional units outlining the steps it needs to follow to solve it. The searching procedure is driven by a list of items available to the robot in its environment (i.e. the kitchen), which is used to determine the functional units that can be executed in the given scenario due to the availability of inputs to these units. This algorithm is motivated by typical graph-based depth-first search (DFS) and breadth-first search (BFS): starting from the goal node, we search for candidate functional units in a depth-wise manner, while for each candidate, we search among its input nodes in a breadth-wise manner to determine whether or not they are available in our kitchen. A subgraph that is obtained from knowledge retrieval is referred to as
a task tree. A task tree differs from a regular subgraph, as it will not necessarily reflect the complete procedure from a single human demonstration. Rather, it will leverage the knowledge obtained from multiple sources to produce a novel task sequence. For a more detailed explanation on the algorithm, we refer readers to [13].

However, this algorithm does not consider the weights we have added to FOON. In this section, we introduce a different approach to finding the ideal task tree based on success rates, which accounts for every combination of functional units that can be used to solve the problem.

A. A Weighted Knowledge Retrieval

A robot can use task tree retrieval to find the sequence of functional unit steps needed to complete a task. The algorithm originally proposed in [18] considers the availability of objects in the robot’s environment to determine the best course of action to take in achieving a goal. The knowledge of what is in the robot’s environment allows us to select those steps that can be executed without having to worry about acquiring missing items. However, as with all greedy algorithms, this algorithm is not likely to find the ideal or optimal course of action. In order to find the task tree with optimality in mind, we would need to explore all possible paths to a given goal node through a similar procedure. This problem is similar to sequence generation problems in natural language processing, where a trade-off needs to be made between exploring all possible paths and finding a solution in real-time. The objective of the algorithm is to build a tree whose nodes can be explored in a depth-wise manner to find all possible combinations of functional units that lead to a specific goal, which is described by the tree’s root node. We describe the steps as Algorithm [1]

In detail, the algorithm works as follows. First, we define a goal node \( N_{\text{Goal}} \) that pertains to the object that the robot is tasked to prepare. All paths to making a specific object will be given as a tree data structure with tree nodes that define a combination of functional units that are needed to make its parent node. For the sake of discussion, we refer to these trees as path trees. Each path tree’s root node (given in \( R \)) is a single functional unit whose output object nodes contain the goal node the robot is trying to make. We construct varying number of path trees for each functional unit that contains the goal node as an output. Initially, these path tree root nodes are appended to a list of path tree nodes \( T \). Once these root nodes have been identified, we proceed with nodes \( t \) in \( T \) to build new connections to newer path tree nodes, which we iteratively create and add to the list \( T \) until we have covered all levels of dependency (i.e., there is a functional unit that precedes those in \( R \)) between objects needed to make \( N_{\text{Goal}} \). Equally important, we iterate for each of its input object nodes \( N_{\text{Input}} \) and identify the functional units \( FU_{\text{candidate}} \) that produce these input nodes (i.e., functional units that contain the input nodes as output nodes \( N_{\text{Output}} \)). For each of these inputs, we build a list \( L_{\text{candidates}} \) that contains a set of candidate units, which are then appended to a list \( L_{\text{prelim}} \) that contains each set of \( L_{\text{candidates}} \). Here, we will encounter two cases of operations: there may be more than one functional unit that needs to be executed along with other units to create all necessary input objects (non-mutually exclusive events), or there may be multiple candidate functional units where we can pick either one to execute to create each of the necessary input objects (mutually exclusive events). These can be likened to the “AND” and “OR” conditions. Therefore, we can assume all path tree nodes of depth 1 or higher may not necessarily have one single functional unit.

Once we have finalized appending to \( L_{\text{prelim}} \), we then compute the Cartesian product using \( L_{\text{prelim}} \) to create new path tree nodes for each product set of functional units \( S \) in \( S_{\text{Cartesian}} \) needed for each input object of the current

```
Algorithm 1: Retrieval of All Possible Task Trees

1: Let \( N_{\text{Goal}} \) be the goal object node
2: Let \( T \) be list of path tree nodes, \( R \) be list of roots of \( T \)
3: {Find the root functional units for all paths:}
4: for all functional units \( FU_i \) in \( G_{\text{FOON}} \) do
5:   if \( N_{\text{Goal}} \) in \( N_{\text{Output}} \) of \( FU_i \) then
6:     Add \( FU_i \) to \( R \) and \( T \) as path tree node
7:   end if
8: end for
9: {For all path tree roots, build its dependency tree:}
10: for all path tree nodes \( t \) in \( T \) do
11:   \( L_{\text{prelim}} = \{ \} \)
12:   for all \( FU_i \) in \( t \) do
13:     for all nodes \( N_{\text{Input}} \) in \( FU_i \) do
14:       \( L_{\text{candidates}} = \{ \} \)
15:       for all functional units \( FU_i \) in \( G_{\text{FOON}} \) do
16:         if \( N_{\text{Input}} \) in \( N_{\text{Output}} \) of \( FU_i \) then
17:           \( FU_{\text{candidate}} = FU_i \)
18:         end if
19:         if \( FU_{\text{candidate}} \) not ancestor(\( t \)) then
20:           Add \( FU_{\text{candidate}} \) to \( L_{\text{candidates}} \)
21:         end if
22:       end for
23:     Add \( L_{\text{candidates}} \) to \( L_{\text{prelim}} \)
24:   end for
25: end for
26: {Build path tree nodes for all unit combinations:}
27: \( S_{\text{Cartesian}} = \text{cartesian product}(L_{\text{prelim}}) \)
28: for all ordered sets \( S \) in \( S_{\text{Cartesian}} \) do
29:   Create new path tree node \( t_{\text{new}} \) containing \( S \)
30:   Set parent of \( t_{\text{new}} \) as current path tree node \( t \)
31:   Add path tree node \( t_{\text{new}} \) to \( T \)
32: end for
33: Remove node \( t \) from the list \( T \)
34: end for
35: {Perform DFS on \( R \) to find all task trees:}
36: for all path tree nodes \( t \) in \( R \) do
37:   for all paths \( P \) from DFS(\( t \)) do
38:     Print functional units in \( P \)
39: end for
40: end for
```
node of focus and add them as children to the current path tree node \( t \). With these new path tree nodes, we add them to \( T \) and we proceed with the searching process using the child nodes we have just created. The connection between a parent and child node lies in the overlapping of input objects of the parent with the outputs of the child. This procedure of propagating and extending the tree continues until we have identified all of the objects needed to solve the manipulation problem (or simply, until we can no longer add new leaf nodes). Once the trees have been finalized, we then perform a simple depth-first search, down to all leaf nodes, to find each individual path \( P \) from the root nodes (kept in \( R \)) to the leaves. Each path in turn will describe all possible functional unit steps that can be followed to solve the given goal. The algorithm described in [18] will likely give one of these paths, but as emphasized before, it is not likely to be the optimal path in terms of success rates. We can use every path uncovered from performing Algorithm 1 to reduce the search space for searching using the available items. However, there may be instances in which a certain path cannot be executed; even though there are connections between all functional units, certain object-state transitions may not make sense. It is therefore crucial to properly define objects and their states to minimize these occurrences.

The optimal task tree is defined based on whatever the criteria is; initially, the optimal task tree is that which leverages all items available to perform all necessary functional units (or, like Petri Nets, causing the necessary transitions to fire). Another criterion which can be used as constraints for the search is finding a task tree with the fewest number of functional units. With the inclusion of weights as success rates for each functional unit in FOON, the optimal task tree would simply be the one with the best overall success rate. This is simply determined by multiplying the robot’s success rate for each action (i.e. functional unit) outlined in the candidate task tree. For example, the total success rate for a given robot based on Figure 3 would be equal to 0.006859%. Although this is very low, we can improve the chance of a robot successfully performing a given task through the assistance of another robot or human.

### IV. Human-Robot Collaboration

The power of using FOON is attributed to the merging of a large number of demonstrations, which are initially gathered as individual subgraphs, into a single source of knowledge. To fully benefit from a FOON, however, it is important for us to gather knowledge from several sources spanning a wide array of activities. With the addition of weights reflecting the difficulty in executing a motion, we can plan while keeping the robot’s capabilities in focus. However, because of the overall complexity of human motions as seen in demonstrations, a robot is not guaranteed to perform the same manipulations as well on its own; it would be difficult to program certain manipulations into robots or perhaps the robot is not built to the task. Instead of allowing the robot to act on its own at the risk of failing, it would be best for a robot to collaborate with another entity to raise its chances of successfully solving the problem. This entity can either be another robot or a human assistant who can step in to perform certain actions in its stead. In this section, we will talk about the considerations needed to execute manipulations in a collaborative way, starting with task planning using FOON.

#### A. Human-assisted Manipulations

With the alternative retrieval algorithm, we can obtain novel task trees for different combinations of methods as we have in a universal FOON. However, certain trees must be eliminated due to the robot’s inability to accomplish the required manipulations for all actions described in those task trees; even the execution of the best task tree can still result in failure. A NAO robot (which is used in our experiments) for instance can only manipulate small and light objects; when compared to larger robots such as the PR2 or Baxter, it is not able to perform very complex manipulations due to its limited workspace and body configuration. Equally important is its limited locomotion to navigate its surroundings since its workspace is very small. To remedy this, we can involve a human assistant in manipulation problems. The human assistant, depending on his/her ability to contribute in the task, can identify the number of steps out of the total number of steps (as functional units) in a task tree that he/she is able to perform with the robot to cooperatively solve the problem.

![Fig. 4. Illustration of a weighted subgraph for the activity of making tea with \( M = 3 \). The overall success rate for this subgraph with a human assistant increases to 68.59\%, high enough for the robot to succeed in the task with the human’s help.](image)
As input to the task tree retrieval, the human can indicate the number of steps as a value $M$, which cannot exceed the length of the task tree $N$ minus 1 step (as an involvement where $N$ is equal to $M$ means that the human will perform the entire task with no robot assistance in its manipulations). If $M$ is 0, there will be no human involvement in achieving his/her desired goal but at the chance of not being able to perform the entirety of the activity’s manipulations. The output of the algorithm can be modified to produce the best task tree based on different values of $M$, as certain trees may be better to execute due to a higher likelihood of success (assuming that the human assistant can perform the manipulation flawlessly). The total success rate of a given path $P$ is denoted by the multiplication of all success rate weights among all functional unit within the tree, which can be likened to the joint probability that all actions are successfully performed. In these human-assisted steps, the success rate would change to 100% by default for the sake of this paper, unless the human assistant’s ability to perform the action is impaired in any way. It is up to the user to determine the degree of involvement he/she is willing to put into an activity, which realistically varies according to the person’s health/condition, mood, age, and other factors. Once the human identifies $M$, the algorithm is run to find the suitable task tree for the given amount of participation. If the human user does not provide a value for $M$, the optimal value of $M$ can also be determined by the robot; this is done by finding the tree whose success rate at some value of $M$ does not significantly improve over the prior value $M - 1$. In Figure 3, the task of tea-making increases in success rate with the introduction of human-assisted steps; the success rate increases from 0.006859% to 68.59%, high enough to execute to its entirety. The robot may still fail its manipulations, but it will not have to worry about performing those that it does not have programmed in its primitives. The $M$ steps would then be modified to indicate that a human assistant should execute those steps when the robot executes the task tree. In the task tree execution phase, the robot will perform its delegated actions, and the remaining $M$ steps are given as instructions to the assistant on how to perform actions on the robot’s behalf.

We also illustrate an example in Figure 5 that shows how candidate task trees are weighed against one another and how the total success rate can change between a pair of trees when there is human involvement. As the value of $M$ becomes higher, the ideal task tree changed within trees and caused a significant improvement in the overall success rate of the task (from 28.5% to 95%). However, we can probably make a reasonable trade-off with $M = 1$ rather than $M = 2$ since it should demand less effort from the human assistant.

V. EXPERIMENTAL RESULTS

In our experiments, the aim is to show that we can significantly improve robot task manipulation performance through human-robot collaboration within the task planning and execution phases. To demonstrate this, we show that a robot can acquire the ideal task tree for execution, delegate commands to the human assistant, and successfully obtain the goal product for varying levels of involvement. We use Aldebaran’s NAO robot to execute manipulations needed to complete the tasks of making tea, mashed potatoes, and ramen noodles. Different variations of preparing each dish were merged together into a single, universal FOON, which was then provided to the algorithm to identify different candidate paths for preparing these items and to illustrate how functional units are selected based on success rates. Because the NAO robot itself is very small, its physical capabilities are limited to using smaller versions of items, and furthermore, certain manipulations are very difficult to capture and replicate. Under these circumstances, the robot can greatly benefit from human participation in the task tree execution phase. Certain parts of the tasks, such as heating
Fig. 6. Graph showing the gradual improvement in success rates (y-axis) as $M$ (x-axis) increases. Sudden drops between $M$ signifies that other paths are considered that exceed the length of $M$, resulting in a completely human tree (e.g. for values $M = 10$ and $M = 11$, the best potential path trees are different in length). Bars are omitted for values of $M$ that exceed the length of a task tree. The values in red indicate the path tree used in Section V.2.

containers to obtain hot water, cannot be left to the robot to perform; for such motions, their nodes were assigned a very low success rate of 1% to reflect how impossible they are for the robot to do on its own. However, for those motions executable by the robot, we assign higher rates based on our confidence in the robot performing the programmed motion primitives. The task trees obtained through the weighted retrieval approach, along with demonstrations of the robot performing each of these trees, can be viewed within the supplementary material provided here.

1) Finding the Optimal Task Tree for NAO: First, we show that we can obtain optimal task trees suitable for the NAO robot to prepare tea, mashed potatoes, and ramen noodles. In order to improve the overall success rate of each activity, the task tree algorithm is expected to iterate through several values of $M$ to then determine the optimal $M$ that balances the effort performed by the robot as well as the human assistant. We show the best overall success rates in the graph shown in Figure 6 to show how success rates increased as we increased $M$. As observed from the numbers, the chances of success significantly improve as more steps are delegated to the human assistant. Based on the success rates assigned to the NAO robot’s universal FOON, the values of $M$ that were ideal for balanced human-robot manipulations were $M = 1$, $M = 2$, and $M = 3$ for the tasks of mashed potatoes, ramen noodles, and tea-making respectively, as even though some of the robot’s primitives have questionably low success rates, it will still be able to execute the task tree on its own. Within the supplementary material, the task trees contain the same number of units labelled as “human-executable” as $M$.

2) Executing the Optimal Task Trees: Secondly, we show that we can perform these actions successfully using human-robot collaboration. The NAO robot is programmed to execute certain motions as described in a task tree’s motion nodes. Since the objective of this work is to demonstrate the use of a universal FOON in task planning, each motion skill/primitive that can be taught to the robot (such as pouring, scooping, or stirring) are learned by manually recording trajectories to simplify the process of programming the robot and to reduce the complexity of the problem space. We also do not use any sensors nor vision systems for manipulation, as there is no need for object detection. Nevertheless, the execution of the entire sequence is determined by the order in which the actions are sequenced in the acquired task tree, meaning that the NAO robot was programmed to perform the activities modularly. In the supplementary material, we provide video demonstrations of the execution of those actions shown in each tree and show how they are carried out with respect to the ideal value of $M$. Without human

1 Video demonstrations can be found at the following link: [http://www.foonets.com/human-robot.html](http://www.foonets.com/human-robot.html)
Involvement, the NAO robot attempts to execute the task tree but ends up failing once it encounters the motion it does not know how to perform (which is reflected by a success rate of 1%); however, with human involvement, the robot can finish all of the tasks and produce the final product. In some cases, we did observe that the motion primitives of the robot can fail, rendering the entire sequence as a failure. As future work, we would like to include sensors or behaviour that allow the robot to determine when it has failed a particular action and to determine what it needs to do to recover from the failed action. Even without its own notion of failure, the robot can supplement this through human interaction by communicating with the assistant to determine whether it should perform the action again.

VI. CONCLUSION AND FUTURE WORK

To summarize, in this paper, we introduce human-robot collaborative task planning using the graphical knowledge representation known as the functional object-oriented network (FOON). Previously, we have shown that a FOON can be used for obtaining the steps needed to achieve a given goal through task tree retrieval, and that these task trees can be novel and flexible to the given scenario. We introduced a modified retrieval procedure that takes the robot’s physical capabilities into account for task planning through the integration of robot success rates. These success rates determine whether the robot can successfully execute the task tree on its own or whether it needs some assistance. To improve the performance of the robot in execution, a human assistant can perform the difficult motions for the robot. We discussed the modified task tree retrieval to acquire the ideal task tree based on the amount of involvement that can be given by the human assistant, and in our experiments, we show that we can obtain suitable task trees that leverage both the robot’s and human’s capabilities without requiring too much effort from the human assistant.

In the future, we would like to explore task tree execution for manipulations done by multiple robots, thereby creating a multi-robot collaborative effort to solving problems. This would require identifying difficulties in performing various types of manipulations so that an optimal task tree can be produced that maximizes the performance of the participating robots. We will demonstrate the interaction between two or more robots, even of different types, to illustrate that FOON can be used for task tree retrieval and execution for any given robot and that plans can be made to synchronize efforts made by the robots to solve the given problem. In addition, we would like to focus more on the robot’s recovery from failure to perform a specific action in a FOON task tree since this is also important to successfully execute its given task.

ACKNOWLEDGEMENT

This material is based upon work supported by the National Science Foundation under Grant No. 1421418.

REFERENCES

[1] Terrence Fong, Iliah Nourbakhsh, and Kerstin Dautenhahn. A survey of socially interactive robots. Robotics and autonomous systems, 42(3-4):143–166, 2003.
[2] Holly A Yanco and Jill Drury. Classifying human-robot interaction: an updated taxonomy. In systems, man and cybernetics, 2004 IEEE International Conference on, volume 3, pages 2841–2846. IEEE, 2004.
[3] Michael A Goodrich, Alan C Schultz, et al. Human–robot interaction: a survey. Foundations and Trends® in Human–Computer Interaction, 1(3):203–275, 2008.
[4] Balasubramaniyan Chandrasekaran and James M Conrad. Human–robot collaboration: A survey. In SoutheastCon 2015, pages 1–8. IEEE, 2015.
[5] Ouassama Khatib. Mobile manipulation: The robotic assistant. Robotics and Autonomous Systems, 26(2-3):175–183, 1999.
[6] Michael Zinn, Ouassama Khatib, Bernard Roth, and J Kenneth Salisbury. Playing it safe [human-friendly robots]. IEEE Robotics & Automation Magazine, 11(2):12–21, 2004.
[7] A. Edsinger and C. C. Kemp. Human-robot interaction for cooperative manipulation: Handing objects to one another. In Robot and Human interactive Communication, 2007. RO-MAN 2007. The 16th IEEE International Symposium on, pages 1167–1172. IEEE, 2007.
[8] Kerstin Dautenhahn. Socially intelligent robots: Toward human–robot interaction. Philosophical Transactions of the Royal Society of London B: Biological Sciences, 362(1480):679–704, 2007.
[9] Paul E Rybski, Jeremy Stolarz, Kevin Yoon, and Manuela Veloso. Using dialog and human observations to dictate tasks to a learning robot assistant. Intelligent Service Robotics, 1(2):159–167, 2008.
[10] ridge Mutlu and Jodi Forlizzi. Robots in organizations: the role of workflow, social, and environmental factors in human-robot interaction. In Proceedings of the 3rd ACM/IEEE international conference on Human robot interaction, pages 287–294. ACM, 2008.
[11] Gabriele Randelli, Taigo Maria Bonanni, Luca Iocchi, and Daniele Nardi. Knowledge acquisition through human–robot multimodal interaction. Intelligent Service Robotics, 6(1):19–31, 2013.
[12] Matthew C Gombolay, Cindy Huang, and Julie A Shah. Coordination of human–robot teaming with human task preferences. In AAAI Fall Symposium Series on AI-HRI, volume 11, page 2015, 2015.
[13] Ben Robins, Paul Dickerson, Penny Stibring, and Kerstin Dautenhahn. Robot-mediated joint attention in children with autism: A case study in robot-human interaction. Interaction studies, 5(2):161–198, 2004.
[14] Andre Billard, Ben Robins, Jacqueline Nadel, and Kerstin Dautenhahn. Building a robot, a mini-humanoid robot for the rehabilitation of children with autism. Assistive Technology, 19(1):37–49, 2007.
[15] Henry Kautz, Larry Arstein, Gaetano Borriello, Oren Etzioni, and Dieter Fox. An overview of the assisted cognition project. In AAAI-2002 Workshop on Automation as Caregiver: The Role of Intelligent Technology in Elder Care, number 2002, page 6065. 2002.
[16] Kerstin Dautenhahn, Michael Walters, Sarah Woods, Kheng Lee Koay, Christophe L Nehaniv, A Sisbot, Rachid Alami, and Thierry Simeon. How may i serve you?: a robot companion approaching a seated person in a helping context. In Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction, pages 172–179. ACM, 2006.
[17] RS Rao, K Conn, Sang-Hack Jung, Jeyanthi Katupitiya, Terry Kientz, Vijay Kumar, J Ostrowski, Sarangi Patel, and Camillo J Taylor. Human robot interaction: application to smart wheelchairs. In Robotics and Automation, 2002. Proceedings. ICRA’02. IEEE International Conference on, volume 4, pages 3583–3588. IEEE, 2002.
[18] D. Paulius, Y. Huang, R. Milton, W. D. Buchanan, J. Sam, and Y. Sun. Functional Object-oriented network for manipulation learning. In Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on, pages 2655–2662. IEEE, 2016.
[19] FOON Website: Graph Viewer and Videos. FOONets.com Accessed: 2018-09-10.
[20] J.J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, Perceiving, Acting and Knowing. Hillsdale, NJ: Erlbaum, 1977.
[21] A. B. Jelodar, M. S. Salekin, and Y. Sun. Identifying object states in cooking-related images. arXiv preprint arXiv:1805.06956, May 2018.
[22] Ahmad Babaeian Jelodar, David Paulius, and Yu Sun. Long activity video understanding using functional object-oriented network. IEEE Transactions on Multimedia, 2018.
[23] David Paulius, Ahmad B Jelodar, and Yu Sun. Functional Object-oriented Network: Construction & Expansion. pages 5935–5941, 2018.