Functional Object-Oriented Network for Manipulation Learning

David Paulius, Yongqiang Huang, Roger Milton, William D. Buchanan, Jeanine Sam, and Yu Sun

Abstract— This paper presents a novel structured knowledge representation called the functional object-oriented network (FOON) to model the connectivity of the functional-related objects and their motions in manipulation tasks. The graphical model FOON is learned by observing object state change and human manipulations with the objects. Using a well-trained FOON, robots can decipher a task goal, seek the correct objects at the desired states on which to operate, and generate a sequence of proper manipulation motions. The paper describes FOON's structure and an approach to form a universal FOON with extracted knowledge from online instructional videos. A graph retrieval approach is presented to generate manipulation motion sequences from the FOON to achieve a desired goal, demonstrating the flexibility of FOON in creating a novel and adaptive means of solving a problem using knowledge gathered from multiple sources. The results are demonstrated in a simulated environment to illustrate the motion sequences generated from the FOON to carry out the desired tasks.

I. INTRODUCTION

Studies in neuroscience and cognitive science on object affordance [1] indicate that the mirror neurons in human brains congregate visual and motor responses [2], [3], [4]. Mirror neurons in the F5 sector of the macaque ventral premotor cortex fire during both observation of interacting with an object and action execution, but do not discharge in response to simply observing an object [5], [6]. Recently, Yoon et al. [7] studied the affordances associated to pairs of objects positioned for action and found an interesting so-called "paired object affordance effect." The effect was that the response time by right-handed participants was faster if the two objects were used together when the active object (supposed to be manipulated) was to the right of the other object.

Borghi et al. [8] further studied the functional relationship between paired objects and compared it with the spatial relationship and found that both the position and functional context are important and related to the motion; however, the motor action response was faster and more accurate with the functional context than with the spatial context. The study results in neuroscience and cognitive science indicate that there are strong connections between the observation of objects and the functional motions. Further, functional relationships between objects are directly associated with the motor actions. A comprehensive review of models of affordances and canonical mirror neuron system can be found in [9].

David Paulius, Yongqiang Huang, Roger Milton, William D. Buchanan, Jeanine Sam, and Yu Sun are with the Department of Computer Science and Engineering at the University of South Florida. Roger Milton, William D. Buchanan, and Jeanine Sam are undergraduate students. (Contact email: yusun@mail.usf.edu)

This interesting phenomenon can be observed in human daily life. When humans are performing tasks, they pay attention not only to objects and their states, but also to object interactions caused by manipulation. The manipulation reflecting the motor response is tightly associated with both the manipulated object and the interacted object.

Seeking an approach that can connect and model the motion and features of an object in the same framework is considered a new frontier in robotics. With the boom in learning from demonstration techniques in robotics [10], [11], [12], more and more researchers are trying to model object features, object affordance, and human action at the same time. Most of the research builds the relationship between single object features and human action or object affordance [13], [14], [15] [16], [17], [18]. Several studies obtained and used object-action relation without considering many low-level object features. In [19] and [20], concrete object recognition was not considered, and objects were categorized solely according to object interaction sequences. Objects were segmented out from a number of video sequences, and an undirected semantic graph was used to represent the space interaction relationship between objects. With a sequence of graphs, their work was able to represent temporal and spatial interactions of objects in an event. With the semantic graphs, they constructed an event table and a similarity matrix, and the similarity between two sequences of object interaction events could be obtained according to the matrix. The objects could further be categorized according to their roles in the interactions, and the obtained semantic graphs might be used to represent robotic tasks. Jain et al. [21] developed symbolic planning that coupled robot control programs with statistical relational reasoners to arrange objects such as setting a dinner table by statistical relational learning. Yang et al. [22], [23] proposed a manipulation action tree bank to represent actions of manipulations at multiple levels of abstraction.

Our recent work [24], [25] investigated object categorization and action recognition using an object-object-interaction affordance framework. We have developed an approach to capitalize on the strong relationship between paired objects and interactive motion by building an object relation model and associating it with a human action model in the human-object-object way to characterize inter-object affordance, and thus use the inter-object affordance relationship to improve object and action recognition.

Similar to the mirror neurons in human brains that congregate the visual and motor responses, a novel FOON is presented in this paper which connects interactive objects with their functional motions to represent manipulation tasks. The proposed novel FOON focuses on the core of a manipu-

lation task that is determined by both the objects' states and the objects' functional motions, which are represented in the FOON as connected nodes. The connections between them represent two-way dependencies that are the functional motions depending on the objects' states and the resulting state depending on the functional motion. The FOON provides structured knowledge about not only the objects and their states, but also about the relationship between the functional motions and the states. From a manipulation goal, the FOON can be searched to find the objects involved, their desired states, and the functional motions to achieve those states.

II. FUNCTIONAL OBJECT-ORIENTED NETWORK

The proposed FOON is a bipartite network that contains motion nodes and object state nodes. In general, an interactive manipulation motion of multiple objects would cause their states to change from so-called input objects states to outcome objects states. Therefore, we connect the input object state nodes to the outcome object state nodes through the manipulation motion node. This arrangement would only allow the object state nodes to be connected to motion nodes and the motion nodes to be connected to object nodes, which forms a bipartite network.

A. Nodes

The nodes in a bipartite FOON have two types: object state *O* or motion *M*. In a manipulation task, an object state node N_O represents an object in a certain state, which is either manipulated by a manipulator or is passively interacting with another object. For example, in a cooking task, a person chops a tomato with a knife. Here, both the tomato and the knife are objects. Initially, the tomato has a state "whole" and the knife has a state "clean". After a chopping motion, which is a motion node, the outcome object states are chopped tomato and dirty knife. A motion node N_M contains the type of the manipulation. From this point, we use "object node" that is short for "object state node." Objects may also be seen as containers of other objects, typically ingredients. These would cover objects such as bowls, pans or ovens which are manipulated with objects within it.

In a FOON, no two object nodes are the same and each object node in the graph is unique in terms of its name and attributes. However, a motion node of the exact same type could appear at multiple locations in the graph, allowing FOON to contain more information than a regular bipartite network.

B. Edges

A FOON is identified as a directed graph, as some nodes are the outcomes of the interaction between other nodes. An edge is denoted by *E* that connects two nodes. Edges are drawn from either an object node to a motion node, or vice-versa, but *no two objects or two motions are usually connected to each other*. FOON can also be represented as a *bipartite graph* which can be transformed into a *one-mode projected graph* [26]. These representations are required for network analysis which will be discussed further in the paper.

In addition, if several object nodes have edges connected to a motion node, it indicates that the objects are interacting with the motion. If a motion node has edges directed to object nodes, it indicates that the objects are the outcomes of the motion.

A FOON can be called a *directed semi-acyclic graph*; this means that there may be some instances of loops where a motion does not necessarily cause a change in an object, as certain objects will remain in the same state.
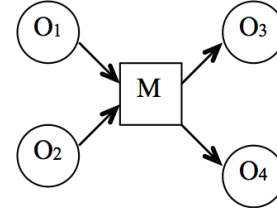

Fig. 1: A basic functional unit with two input nodes and two output nodes connected by an intermediary single motion node.

C. Functional Unit

A *functional unit* is considered as the minimum learning unit in a FOON. It represents the relationship between one or several objects and *one* functional motion associated to the objects. Each unit represents one action which is part of an activity. As shown in Figure 1, the object nodes connected with the edges pointing to the functional motion node are called input object nodes, while the object nodes connected with the edges pointing from the functional motion node are called output object nodes.

D. Network Data Structure

A FOON is represented by conventional graph representations, namely *adjacency matrices* and *adjacency lists*. We use an adjacency matrix to represent the network for its simplicity in representing a digraph and for performing network analysis. Each node is represented by a row, and its relation to other nodes is given by the columns of the matrix. An edge from a node N_i to N_j is denoted by a value of 1, preserving directionality of edges; if two nodes are not connected, then an index has a value of 0.

Accompanying the adjacency matrix is a *node list* which keeps track of all object and motion nodes found in the graph. This list is needed to map each node to its row/column representation. Each node also maintains its own list which provides directional information to other nodes that it is connected to.

III. LEARNING FOON

Ideally, a FOON can be automatically trained from observing human activities. However, due to the complexity of object, state, and motion recognition, we currently construct many small sets of functional units by labeling instructional videos. We manually input these functional units by hand which are then merged together automatically into a single

Fig. 2: A FOON subgraph based on an instructional video on making a watermelon-strawberry smoothie. The green solid circles are object nodes and the red solid squares are motion nodes. The object nodes are labeled with object name and their states in parentheses. The motion nodes are labeled with their manipulation motion types.

Fig. 3: Our current universal FOON that is constructed from 60 videos.

subgraph for each video. All subgraphs from each video gathered are then merged into a large universal FOON.

Although the functional units are constructed by humans, the process of combining the knowledge together is done algorithmically. The creation of a FOON can thus be seen as a *semi-automatic* process in this sense.

A. Creating Subgraphs

We have recruited five volunteers to manually label the input object states, manipulation motion, and output object states in instructional videos through an annotating interface that we have developed. This interface displays the potential graph which would be made from a given set of functional units. The annotations are then converted into functional units with time sequence labels. These functional units are then connected and combined into a subgraph automatically using the time sequence labels. For each video, its FOON subgraph is visualized and verified manually.

Each subgraph provides the essential structured knowledge needed to prepare the dish including objects (ingredients and utensils), their states, and their interactive motions. Figure 2 illustrates the FOON subgraph obtained from an online instructional video.

B. Merging Subgraphs

The universal FOON can be expanded by merging new subgraphs when many videos are processed. The merging algorithm is described in Algorithm 1.
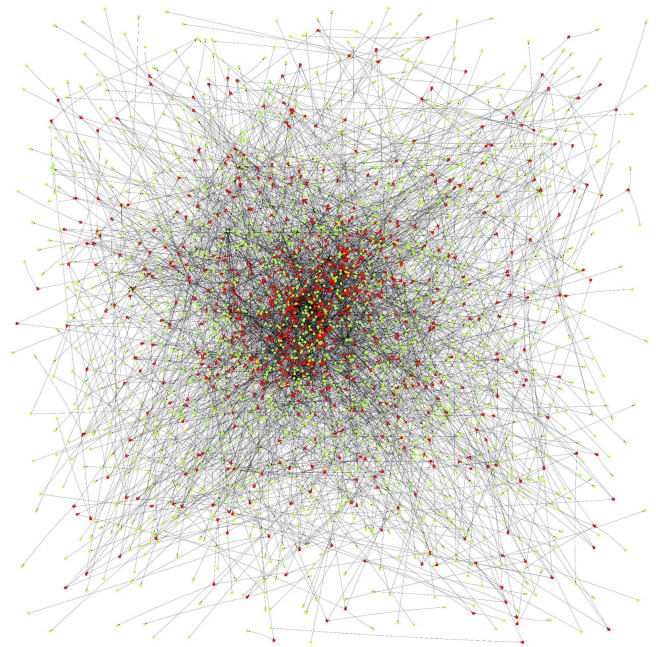
Since subgraphs are created separately by different volunteers, they are prone to inconsistencies from different labels. Therefore, a parser was developed to preprocess all labeled subgraph files to keep all labeling consistent. The parser has three main functions: create a main index with a list of all the objects, updating the input file by relabeling all of the objects so they are consistent throughout, and creating a records file that records all changes in any modified files. To keep track of all data elements, we create a records file that contains the object name, its old identifier, its new identifier, initial state, final state, file name, and motion. The parser also finds possible duplicates in objects or motions through the use of the WordNet lexical database [27] by comparing the stem word with the current object index.

Algorithm 1 Merging functional unit FU_{new} with G_{FOON}

---

    let FU_{new} be functional unit to merge
    found = false;
    for all functional unit FU_i in G_{FOON} **do**
      if $FU_i == FU_{new}$ **then**
        found = true
      end if
    end for
    if found is $true$ **then**
      Add FU_{new} to G_{FOON}
      Add input nodes N_{Input} to node list
      Add output nodes N_{Output} to node list
      Add motion node N_{Motion} to node list
    end if

After the nodes are made consistent within all the subgraphs, we run the union operation to merge all subgraphs

into a universal FOON graph, one at a time. The universal FOON is initially empty. The union operator first checks if the functional unit is already present in FOON. Functional units are kept unique, and similar units will have almost the same objects and states found within them. After this process, objects in the new unit will be added as if they did not exist in the universal node list; if they exist, a reference is made to those existing nodes and then the edges are connected to a new motion node.

So far, we have processed 60 instructional videos on food preparation. Our universal FOON presently has 2169 nodes (broken down into 1229 object nodes made unique by states and 792 instances of 57 possible motion nodes) with 3223 edges after the merging process. These numbers gradually increase as more subgraphs are continuously being generated. The low resolution compressed visualization of the generated universal FOON is shown in Figure 3. The high resolution image of the FOON is exceptionally large with a size of more than 200 MB. The full list of videos and their functional unit subgraphs along with the universal FOON graph are all available for download at [28].

IV. MOTION LEARNING

For motion generating purpose, the motion type in the FOON is represented using motion harmonics [29], which are extracted from demonstrated data using functional principal component analysis (fPCA) [30].

We collected the manipulation data used for motion learning purpose using an OptiTrack 3D motion capture system in our lab (Figure 4). Although processing an online 2D video can also produce motion data, those data are 2D and do not contain enough information for generating executable motions in the 3D world. With our 3D motion capture system, we collected the position and orientation of the objects.

Let $X = \{x_1, x_2, ..., x_N\}$ represent the data that includes N trials, where $x_i(d, t) \in \mathbb{R}$ denotes the value of degree d of trial i at time step t, $d = 1, 2, \ldots, D$, $t = 1, 2, \ldots, T_i$. We assume that six degrees are used: $\{x, y, z, \phi, \theta, \psi\}$, in which x, y, z refer to location coordinates and ϕ, θ, ψ refer to Euler angles.

Unlike location coordinates, Euler angles invariably have ranges, such as $[-\pi, \pi]$. For simplicity, we assume the range is $[-1, 1]$. Thus, to facilitate optimization for motion generation, we apply inverse hyperbolic tangent on the angle trajectory data:

$$x_i(d) \leftarrow \text{atanh}(x_i(d)), \quad d \in D_q \tag{1}$$

where D_q represents the set of angular degrees.

We use batch Dynamic Time Warping (DTW) [31] to align the trials and obtain $Y = \{y_1, y_2, ..., y_N\}$, where $y_i(d, t) \in \mathbb{R}$ corresponds to x_i, $t = 1, 2, \ldots, T$, and T is the common trajectory length. Applying fPCA to data Y, we obtain the mean trajectory $\bar{g}(d, t) \in \mathbb{R}$, the motion harmonics $g(d, t) \in \mathbb{R}^M$ where M is the number of motion harmonics, and the weights $c_i(d) \in \mathbb{R}^M$, $i = 1, 2, \ldots, N$. We assume the
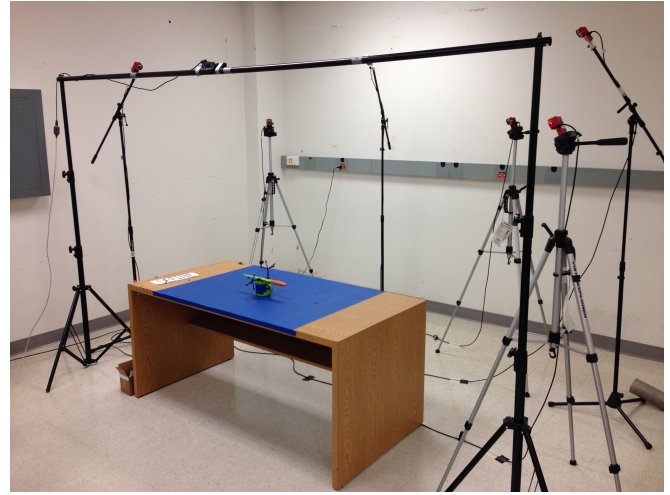

Fig. 4: The OptiTrack motion capture system with which we collect data for motion learning. The system consists of six motion capture cameras on tripods. Within the blue area on the desk are two objects with reflective markers attached to them.

weights are produced from a Gaussian distribution, whose maximum likelihood parameters equal the sample mean $\mu(d) \in \mathbb{R}^M$ and sample covariance $\Sigma(d) \in R^{M \times M}$ of the weights. Thus the Gaussian is denoted by $\mathcal{N}(\mu(d), \Sigma(d))$.

To summarize, the motion demonstrated in the data is represented by

$$\Theta = (g(d, t), \bar{g}(d, t), \mathcal{N}(\mu(d), \Sigma(d))), \quad d = 1, 2, \ldots, D \tag{2}$$

We direct readers to [29] for more detail of motion harmonics as a motion representation.

V. ANALYSIS OF FOON

We primarily focus on determining the most central or important nodes in our network. The importance of the node will lie in the frequency of this node in the interaction of one node with many other nodes. This measure of importance in networks is referred to as *centrality*, and this is a computed value that is assigned to each node. There are many ways of computing the centrality, and the measures we have applied to FOON were *degree* centrality, *eigenvector* centrality and *Katz* centrality [26]. The one-mode projected network is used specifically for centrality analysis on objects to observe the relationship between tools and ingredients used in a FOON.

We can apply the information obtained to our specific application, where we can determine from the object nodes which objects need to be frequently used by the robot, and from the motion nodes which manipulation skills are the most important for the robot to learn well.

A. Object Centrality

We chose to look at the relationship between objects by converting it into a one-mode projected network [26]. The one-mode projection removes all intermediary motion nodes and so all object nodes are directly connected to one
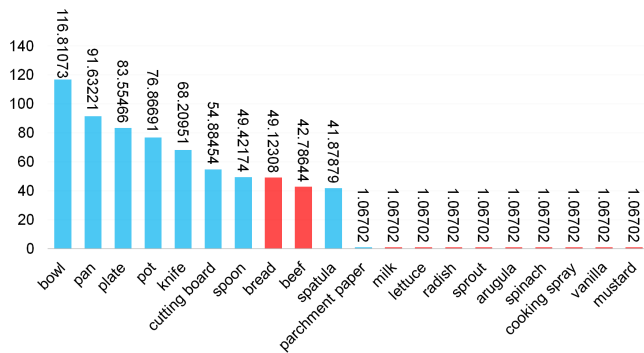
Fig. 5: Graph showing the objects found with the ten (10) highest and lowest centrality values. The higher the value, the more important a node is. Objects are also classified as utensils (shown in blue) and ingredients (shown in red).

Fig. 6: Graph showing the top 10 motions observed in our universal FOON (out of 798 motion instances).

another; in this way, we can also investigate object-object relationships. Objects are connected to each other based on the direction of edges in all functional units in FOON.

These values are not integer values, as the computations involve more than simply counting the node's degree. We should not only be concerned with the degree of each node, but we also ought to consider the influence of all nodes connected to every other node.

Our findings for the most important nodes when using these three measures were the same when not considering states; the bowl object was found to be the most important object with a total of 72 edges. Other objects along with their Katz centrality values are illustrated in Figure 5.

We can use this information in determining the objects which require the most attention in mastery and skill in manipulating them. The centrality values also let us know which objects are in high demand in recipes across the entire network; this is important for us to know so that we ensure that these objects are available in our working environment.

B. Motion Frequency

We also consider the frequency at which objects and motions appear in our network, which we can use for determining the most likely action to occur at a given time and with a given object. We do this by counting the number of instances of each motion belonging to a functional unit that were found in the network.

The most frequent motion observed out of 57 possible motion types is the *pick-and-place* motion. This makes sense as there is much translation and movement of objects when preparing meals. For us, this means that robots used for cooking tasks should have mastered the pick-and-place motion for different objects.

The next motion found to be frequent is the *pour* motion. We believe this is due to the nature of cooking, where items are usually mixed together or put into containers from other objects. The top 10 motion frequencies can be found in Figure 6. These values were found after the merging process, and so these are the most frequent nodes after compression
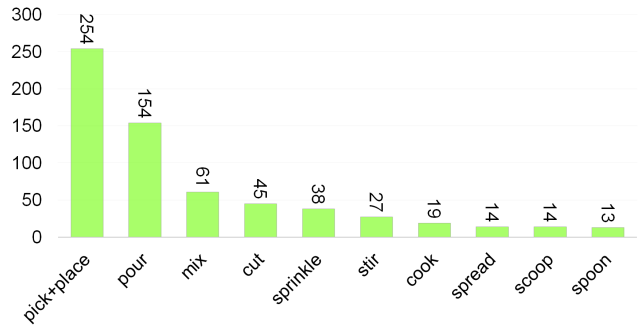
of the network. We can say that these do reflect the reality of cooking in the kitchen.

With these probabilities, we hope to improve our structure to behave more like a typical probabilistic graphical model within the next phases of our project. The frequencies can be used for compressing FOON even further by possibly removing the need for duplicate motion nodes. When paired with the objects, our system would be able to determine the next likely outcome for each object and thus making robot manipulations easier to perform.

VI. MANIPULATION KNOWLEDGE RETRIEVAL

The universal FOON can be considered as a structured knowledge representation that can be used for solving manipulation tasks. Given a desired goal and a set of available objects, formally, there are two steps in generating manipulations from the FOON: retrieving a task tree and generating the motions needed to accomplish the task. The approach is a combination of the breadth-first search and the depth-first search as a specialized application of the branch-and-bound algorithm.

A. Retrieving Task Tree

First, a goal node N_{Goal} in the FOON is identified as a desired goal. The initial task tree T_{task} will be empty. T_{task} will be populated with functional units which make N_{Goal}. To guide our searching process, we require a set of lists: a list of all items in the environment (L_{scene}), a list of items we do not know how to make (L_{search}), and a list of functional units which produce nodes in L_{search} (S).

We begin by adding N_{Goal} to our list of items to search L_{search}. We then search the universal FOON G_{FOON} for all procedures which make the head of L_{search} which is denoted as $N_{Current}$; we would add these units to list S. Once this list is fully populated, we then search for the functional unit FU_i in S which we can execute in its entirety. This is determined by the availability of objects in each functional unit as listed in L_{scene}. If we find all input objects N_{Input} in any unit, we can add this unit to T_{task} and mark $N_{Current}$ as seen. However, if we cannot find any functional unit that

can be executed fully due to missing objects, then we add the items N_{Input} to L_{search} so we can figure out how to make them.

We repeat the search for functional units that make $N_{Current}$ instances as long as there are items in L_{search} that we need to know how to make, especially N_{Goal}. We do this process entirely until it has been determined that there is no possible way of solving the manipulation task (due to shortage of items) or we have found an executable task tree. We will know that a task tree sequence is found when N_{Goal} has been marked as being "solveable".

Algorithm 2 Retrieval of task tree using branch-and-bound

---

let N_{Goal} be the desired output node, L_{scene} be list of objects in scene, L_{search} be list of objects to search, T_{task} be final task tree, $N_{Current}$ be head of L_{search}, S be list of functional units that make $N_{Current}$
$L_{search} = \{N_{Goal}\}$
while N_{Goal} not found **do**
  $N_{Current} = L_{search}[0]$
  for all FU_i in G_{FOON} **do**
    if $N_{Current}$ in N_{Output} of FU_i **then**
      Add FU_i to S
    end if
  end for
  for all FU_i in S **do**
    for all N_{Input} in FU_i **do**
      if N_{Input} not in L **then**
        $L_{search} = \{L_{search}, N_{Input}\}$
      end if
    end for
    if all N_{Input} of FU_i in L **then**
      $T_{tree} = \{T_{tree}, FU_i\}$
      remove all FU in S that make $N_{Current}$
    end if
  end for
end while

This searching procedure can also be adjusted to make use of weights which act as heuristics and constraints on the creation of a task tree. These heuristics can be a *cost* value that is associated with each motion, influencing the selection of functional units which are added to T_{task}. We hope to improve the quality of our search through the use of cost values reflecting the complexity of motions or finding the shortest path to accomplishing a goal.

We now demonstrate an example of task tree searching with an example FOON shown in Figure 7. One example task tree we obtained by searching this case is shown in Figure 8. For emphasis, the graph has been color-coded to distinguish between functional units of two subgraphs; video 1 is denoted by the lime-green object nodes and the red motion nodes, video 2 is denoted by the dark-green object nodes and the indigo motion nodes, and the overlapped functional units are denoted by the yellow object nodes and orange motion nodes. The goal is to make cooked ribs
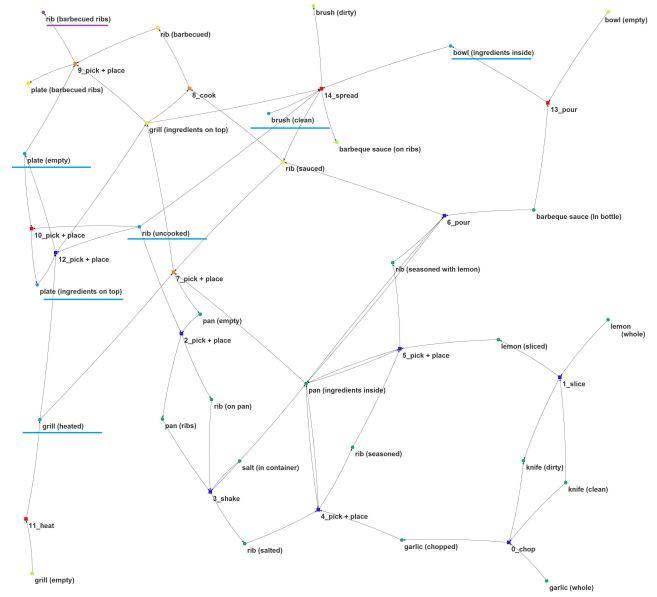

Fig. 7: Example of a FOON merging two ways of preparing cooked ribs barbecued ribs (node in purple) using available objects (in blue).

(highlighted in purple) given a set of objects observed in the scene (highlighted in blue). We can start at these blue nodes to arrive to our goal. These nodes can be viewed as root nodes in basic tree structures, except that a tree does not typically have multiple roots. The path we take is entirely dependent on the availability of the objects in the robot's environment.

The ability to merge and combine knowledge into one single network makes our network very powerful and useful. Within a universal FOON lies many possible task trees for different scenarios. These possibilities can be an entirely novel way of executing a task, as there may be several ways of creating a particular meal. Our task sequences therefore will not necessarily follow the entire procedure from a single video source. For example, there are many ways to prepare a sauce for meat, and by using the knowledge on how to prepare sauces with a variety of ingredients we can compensate for the unavailability of certain items needed if we instead followed one recipe. The novelty not only comes from the possibility of different task sequences, but also in the flexibility in how we prepare the meals.

B. Motion Generating

The task tree is then used to generate a task sequence that contains a series of motions, objects, and their states, which provides step-by-step commands executable by a robot. After a functional unit in the task tree is provided and the involved objects are identified in an environment, a new trajectory of the motion needs to be generated using the locations of those objects as constraints. The new trajectory is generated using motion harmonics, given by:
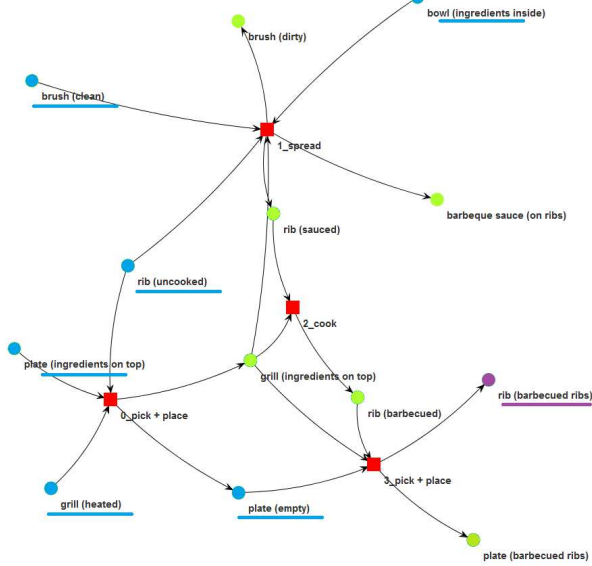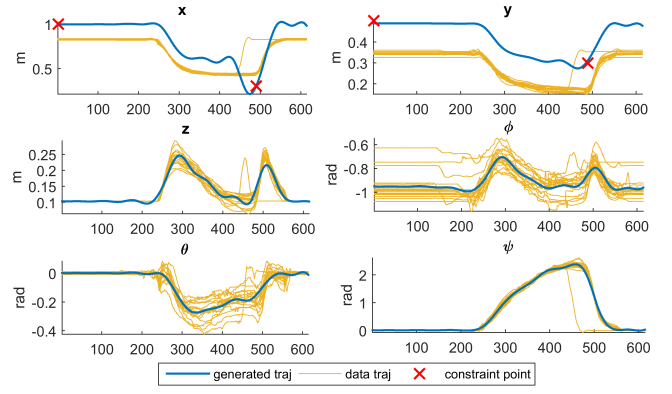
Fig. 9: In degree x and y, the new trajectory meets the constraints well. Without constraints, the rest degrees of the new trajectory equal the mean of the data. The 'data traj' (in dark yellow) have been aligned using DTW.

Figure 9 shows one example: a generated pouring motion trajectory in a new environment with a new relative start and target positions between a cup and a teapot. The pouring motion was learned from twenty trials of pouring instances. According to the new relative position of pair, we applied two constraints to x and y, which represent the start and target of the new pouring motion: $e_1 = [1, 0.5]^\top$, $e_2 = [0.3, 0.3]^\top$, $t_1 = 1$, $t_2 = 490$. We set $\lambda = 1e - 6$. From the results in Figure 9, we can see that the generated trajectory resembles the learning and satisfies the new constraints. Currently our motion generation process does not consider dynamic or kinematic constraints of the actual robot.

We have developed a simulated kitchen environment using Unity to demonstrate the results of the task sequences and the motion generating approach. We have generated several manipulation simulations and attached two simulation videos with this paper.

VII. CONCLUSION AND FUTURE WORK

In this paper, we present a FOON representation of manipulation tasks, which connects interactive objects with their functional motions. The FOON provides structured knowledge about the relationship between the object states and functional object motions, which is valuable for not only learning manipulation tasks, but also understanding human activities.

We have developed an approach to construct functional units using abstracted knowledge of online instructional videos, mainly cooking videos; this information is extracted by human users tasked with annotating these functional units manually. The functional units are then connected into subgraphs and then merged into a universal FOON through an automatic process. A large universal FOON is constructed from 60 online videos which has been made available online. It has been analyzed to obtain insights of the structure of the network using centrality measures.

Manipulation knowledge can be retrieved from the FOON given a manipulation goal using our searching algorithm. The

Fig. 8: Task tree showing the steps needed to prepare barbecued ribs (highlighted in purple) using available objects (in blue) to create objects of other states.

$$y_{\text{new}}(d, t) = \begin{cases} \bar{g}(d, t) + c(d)^\top g(d, t) + c_0(d) & d \notin D_q \\ \tanh\left(\bar{g}(d, t) + c(d)^\top g(d, t) + c_0(d)\right) & d \in D_q \end{cases}$$
(3)

where weights $c(d)$ and $c_0(d)$ are the variables, and $\top$ denotes transpose. Let D_e denote the set of degrees on which constraints are imposed. Let $e_s(d)$ represent the s-th constraint on degree $d \in D_e$, and t_s be the time stamp of $e_s(d)$, $s = 1, 2, \ldots, S$. The new trajectory tries to resemble the demonstrated data as well as meeting the constraints. For $d \notin D_e$, $c(d) = \mu(d)$ and $c_0(d) = 0$. For $d \in D_e$, we define the loss function as:

$$L = \frac{1}{2} \sum_s \left(e_s(d) - \bar{g}(d, t_s) - c_0(d) - c(d)^\top g(d, t_s)\right)^2$$
$$+ \frac{\lambda}{2} \left(c(d) - \mu(d)\right)^\top \Sigma(d)^{-1} \left(c(d) - \mu(d)\right),$$
(4)

where λ is a hyperparameter chosen by the user. The optimal weights that minimize L are obtained by first computing $c(d)$ by solving $Ac(d) = b$, where

$$A = \sum_s g'(d, t_s) g'(d, t_s)^\top + \lambda \Sigma(d)^{-1}$$
(5)

$$b = \sum_s f_s'(d) g'(d, t_s) + \lambda \Sigma(d)^{-1} \mu(d),$$
(6)

$g'(d, t_s) \overset{\text{def}}{=} g(d, t_s) - \frac{1}{S} \sum_s g(d, t_s)$,
$f_s'(d) \overset{\text{def}}{=} f_s(d) - \frac{1}{S} \sum_s f_s(d)$, and $f_s(d) \overset{\text{def}}{=} e_s(d) - \bar{g}(d, t_s)$; and then use $c(d)$ to compute $c_0(d)$:

$$c_0(d) = \frac{1}{S} \left(\sum_s f_s(d) - c(d)^\top \sum_s g(d, t_s)\right).$$
(7)

manipulation knowledge is stored in a task tree sequence with a series of involved objects, manipulation motions, and immediate goals. These task trees will not necessarily follow the same exact procedure as described by a single recipe or video, making them a flexible and novel way of manipulating objects based on the knowledge acquired from several sources.

The motion nodes in FOON are described as a combination of motion harmonics. With the parameterized representation, a new motion of a learned type can be generated to accommodate new constraints in different environments and motion contexts. Based on the obtained task tree, a sequence of manipulation motions is generated properly to perform the desired task. In addition to the examples illustrated in the paper, generated manipulation motion sequences are demonstrated in a simulated kitchen environment and two demo videos are attached with this paper and additional demo videos are available at [28].

In the future, we plan to perform more network analyses, such as computing connection strength and efficiency, on the universal FOON to better understand the dynamics of the network. Additionally, we hope to integrate probabilities into our representation using the findings we obtained from the analyses we have done as well. We are also exploring means of making FOON more "intelligent" by making inferences based on object similarity. For instance, even though we have not seen how certain objects are used or manipulated, we can instead use the knowledge we know and apply it to an unknown problem. Missing information such as quantities will also be considered. We are also seeking methods to which we can solve the problem of automatically generating a FOON from instructional videos.

References

[1] J.J. Gibson. The theory of affordances. In R. Shaw and J. Bransford, editors, *Perceiving, Acting and Knowing*. Hillsdale, NJ: Erlbaum, 1977.

[2] G. Rizzolatti and L. Craighero. The mirror neuron system. *Ann. Rev. Neurosci.*, 27:169–192, 2004.

[3] G. Rizzolatti and Craighero L. Mirror neuron: A neurological approach to empathy. In Jean-Pierre Changeux, Antonio R. Damasio, Wolf Singer, and Yves Christen, editors, *Neurobiology of Human Values*. Springer, Berlin and Heidelberg, 2005.

[4] E. Oztop, M. Kawato, and M. Arbib. Mirror neurons and imitation: a computationally guided review. *Epub Neural Networks*, 19:254–271, 2006.

[5] G. Di Pellegrino, L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Understanding motor events: A neurophysiological study. *Exp Brain Res*, 91:176–80, 1992.

[6] V. Gallese, L. Fogassi, L. Fadiga, and G. Rizzolatti. Action representation and the inferior parietal lobule. In W. Prinz and B. Hommel, editors, *Attention and Performance XIX. Common mechanisms in perception and action*. Oxford University Press, Oxford, 2002.

[7] E.Y. Yoon, W.W. Humphreys, and M.J. Riddoch. The paired-object affordance effect. *J. Exp. Psychol. Human*, 36:812–824, 2010.

[8] A.M. Borghi, A. Flumini, N. Natraj, and L.A. Wheaton. One hand, two objects: emergence of affordance in contexts. *Brain and Cognition*, 80(1):64–73, 2012.

[9] S. Thill, D. Caligiore, A.M. Borghi, T. Ziemke, and G. Baldassarre. Theories and computational models of affordance and mirror systems: An integrative review. *Neuroscience and Biobehavioral Reviews*, 37(3):491–521, 2013.

[10] G. D. Konidaris, S.R. Kuindersma, R.A. Grupen, and A.G Barto. Robot learning from demonstration by constructing skill trees. *Intl J Robotics Research*, 31(3):360–375, 2012.

[11] B. D. Argall, S. Chernova, and et al. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[12] S. Schaal, S. Ijspeert, and et al. Computational approaches to motor learning by imitation. *Philosophical Transactions of the Royal Society of London Series B- Biological Sciences*, 358(1431):537–547, 2003.

[13] A. Gupta and L. Davis. Objects in action: An approach for combining action understanding and object perception. In *Conference on Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[14] H. Kjellstrom, J. Romero, and D. Kragic. Visual object-action recognition: Inferring object affordances from human demonstration. *Computer Vision and Image Understanding*, 115:81–90, 2010.

[15] J. Gall, A. Fossati, and L Gool. Functional categorization of objects using real-time markerless motion capture. In *Conference on Computer Vision and Pattern Recognition*, pages 1969–1976, 2011.

[16] Cornelia Fermüller Yezhou Yang, Yiannis Aloimonos and Eren Erdal Aksoy. Learning the semantics of manipulation action. *CoRR*, abs/1512.01525, 2015.

[17] Alessandro Pieropan, Carl Henrik Ek, and Hedvig Kjellström. Recognizing object affordances in terms of spatio-temporal object-object relationships. In *2014 IEEE-RAS International Conference on Humanoid Robots*, pages 52–58. IEEE, 2014.

[18] Yifan Shi, Yan Huang, D. Minnen, A. Bobick, and I. Essa. Propagation networks for recognition of partially ordered sequential action. In *Computer Vision and Pattern Recognition, 2004. CVPR 2004. Proceedings of the 2004 IEEE Computer Society Conference on*, volume 2, pages II–862–II–869 Vol.2, June 2004.

[19] E. Aksoy, A. Abramov, F. Worgotter, and B. Dellen. Categorizing object-action relations from semantic scene graphs. In *IEEE Intl. Conference on Robotics and Automation*, pages 398–405, 2010.

[20] Eren Erdal Aksoy, Alexey Abramov, Johannes Dörr, Kejun Ning, Babette Dellen, and Florentin Wörgötter. Learning the semantics of object-action relations by observation. *Int. J. Rob. Res.*, 30(10):1229–1249, September 2011.

[21] Dominik Jain, Lorenz Mosenlechner, and Michael Beetz. Equipping robot control programs with first-order probabilistic reasoning capabilities. In *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*, pages 3626–3631. IEEE, 2009.

[22] Yezhou Yang, Anupam Guha, Cornelia Fermuller, and Yiannis Aloimonos. Manipulation action tree bank: A knowledge resource for humanoids. In *Humanoid Robots (Humanoids), 2014 14th IEEE-RAS International Conference on*, pages 987–992. IEEE, 2014.

[23] Yezhou Yang, Cornelia Fermuller, and Yiannis Aloimonos. Detection of manipulation action consequences (mac). In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2013.

[24] Y. Sun, S. Ren, and Y. Lin. Object-object interaction affordance learning. *Robotics and Autonomous Systems*, 2013.

[25] Shaogang Ren and Yu Sun. Human-object-object-interaction affordance. In *Workshop on Robot Vision*, 2013.

[26] M. E. J. Newman. *Networks: An Introduction*. Oxford University Press, USA, 2010.

[27] Christiane Fellbaum. *WordNet: An Electronic Lexical Database*. Bradford Books, 1998.

[28] Online FOON graphs and videos. http://www.foonets.com. Accessed: 2016-07-07.

[29] Y. Huang and Y. Sun. Generating manipulation trajectories using motion harmonics. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4949–4954, 2015.

[30] J.O. Ramsay and B.W. Silverman. *Functional Data Analysis*. Springer, 2005.

[31] Athanassios Kassidas, John F. MacGregor, and Paul A. Taylor. Synchronization of Batch Trajectories Using Dynamic Time Warping. *AIChe*, 44(4):864–875, April 1998.