Using Sequences of Movement Dependency Graphs to Form Object Categories

Shane Griffith, Vladimir Sukhoy and Alexander Stoytchev
Developmental Robotics Laboratory
Iowa State University
{shaneg, sukhoy, alexs}@iastate.edu

***Abstract*—This paper describes a new graph-based representation that captures the interaction possibilities between the robot's hand and one or more objects in the environment in terms of the dependencies between their movements or lack of movements. The nodes of the graph correspond to the tracked visual features, i.e., the robot's hand and the objects. The edges correspond to the pairwise movement dependencies between the features. As the robot performs different behaviors with the objects the structure of the graph changes, i.e., edges are added and deleted over time. This paper tests the hypothesis that sequences of such graphs can be used as a signature that captures the essence of some object categories. This framework was tested with container and non-container objects as the robot tried to insert a small block into them. The results show that the robot was able to distinguish between these two object categories based on the sequences of their corresponding movement dependency graphs.**

I. Introduction

The ability to track movement dependencies between objects can mitigate some of the perceptual issues that arise during manipulation. For example, one of the first things that infants learn about containers is that an object inside a container will move with the container when the container is moved [1]. Infants easily identify the movement dependencies between the visual features of objects, which provides a powerful cue for learning how to manipulate them [2]. Infants also learn how their own movements relate to the movements of objects. They become fascinated with making and breaking co-movement relationships as they insert blocks into containers and then shake them [3]. Clearly, a lot of information about objects can be gained by observing their co-movement patterns.

This paper describes a new representation for capturing the movement dependencies between objects during manipulation tasks. This representation uses sequences of graphs (see Fig. 1). The vertices in these graphs correspond to the tracked features of objects. The edges correspond to movement dependencies between the features. A statistical test for independence is used to determine when to insert/delete edges into the graphs. These graphs evolve over time and form sequences of graphs that reflect how the movement dependencies change as the robot manipulates different objects.

The representation was evaluated in an experiment with 20 objects and 5 blocks. The robot observed the movement patterns of the objects as it interacted with them. A sequence of movement dependency graphs was extracted from the movement patterns and used to categorize the objects. The results show that the robot could use the movement dependency graph representation to form a representation for what a human would call containers and non-containers.

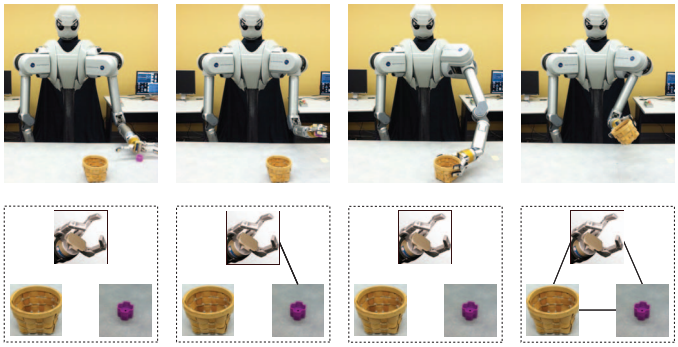

Fig. 1. **(Top Row)** Our humanoid robot, shown here grasping a small block, shaking it, dropping it inside a container, grasping the container, and shaking the container. **(Bottom Row)** The movement dependency graph as it evolved over time. The nodes correspond to the entities that are tracked visually. The edges between pairs of objects indicate movement dependencies.

II. Related Work

Approaches based on co-movement representations have been used previously for object categorization tasks [4][5]. Because these approaches focus specifically on learning about objects, their goal is to represent the overall outcome of an interaction rather than to represent the details of a behavioral interaction. Thus, these approaches lack the ability to identify exactly when co-movement or separate movement occurs during manipulation. This paper, in contrast, proposes a representation that has this ability.

Movement representations have also been used in previous work to detect visual features attached to the robot's body [6][7][8][9]. Because these approaches focus on detecting the body, however, they require proprioceptive data: timestamps of motor commands [6][7] or vectors of joint angles [9]. Using proprioceptive data some robots were also able to learn from the movement patterns of objects, but only if the robot directly controlled the object [8][9]. In contrast, this paper describes a representation that does not require proprioceptive data and can be applied using vision alone.

Aksoy *et al.* [10] have shown that a graph representation works well for activity recognition and object categorization. They introduced semantic scene graphs, which can capture the

spatiotemporal relationships between many different objects in an image. Each node in the graph represents an object and each edge represents one of four different spatial relationships between two objects: absence, no connection, touching, and overlapping. A sequence of semantic scene graphs can capture enough information for activity recognition or object categorization. This representation, however, does not capture co-movement relationships between the objects.

Sridhar *et al.* [11] also proposed an activity graph representation. A single lattice structure was used to encode the evolution of spatiotemporal relationships between objects over an entire video sequence. The graph captured different spatial relationships between the features of objects: disconnects, surrounds, and touches. The graph also captured the times during which different spatial relationships between object features persisted in the video. The representation was used to compare objects from different videos in order to form a hierarchical categorization of the objects [11].

In our previous work [4], we used visual co-movement to perform object categorization. Co-movement was represented using two features. The first feature captured whether the two objects moved at the same time. The second feature captured whether the two objects moved in the same direction. In these experiments, the robot interacted with a block and several objects by grasping the block, dropping the block above the object placed on a table, and then pushing the object. The co-movement features for each trial were clustered into outcome classes. The frequency with which different outcome classes occurred with each object was sufficient to separate the containers from the non-containers.

In a follow-up study [5], we introduced a second approach for learning from the movement sequences of objects. Instead of extracting specific features for co-movement, a single string was used to represent both the movement sequence for a block and the movement sequence for an object. Strings from different interaction trials were compared and clustered in order to identify the different co-movement patterns between the objects. The frequency with which different outcome classes occurred with each object was, again, sufficient to separate the containers from the non-containers.

This paper introduces a new graph representation that captures the movement dependencies between objects during an activity. The representation is an improvement over our previous work [4][5] because it can capture the specific times during an activity when different movement dependencies occur between objects. Because interactive learning about objects is one of the main applications for representations of movement [4][5][8][12][13], the representation described in this paper is evaluated using an interactive learning task to categorize containers from non-containers.

III. Experimental Setup

A. Robot

All experiments were performed with the upper-torso humanoid robot shown in Fig. 1. The robot's arms are two 7-dof Whole Arm Manipulators (WAMs) by Barrett Technology,

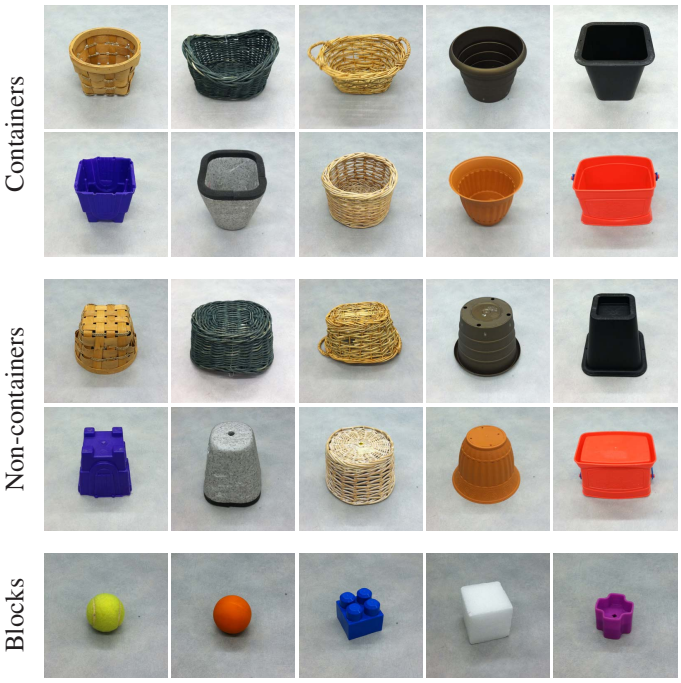

Fig. 2. The objects and the blocks used in the experiments. (**Containers**) The first two rows show the 10 containers: wicker basket, plant basket, potpourri basket, flower pot, bed riser, purple bucket, styrofoam bucket, candy basket, brown bucket, and red bucket. (**Non-containers**) The second two rows show the same 10 objects as before, but flipped upside down, which makes them non-containers for this particular robot with this particular set of behaviors. (**Blocks**) The last row shows the 5 blocks: tennis ball, rubber ball, mega block, foam cube, and purple block.

each equipped with a Barrett Hand as its end effector. The arms are controlled in real time from a Linux PC at 500 Hz over a CAN bus interface. The robot is also equipped with two cameras (Quickcams from Logitech) that capture 640x480 color images at 15 fps. Rubber finger tips were stretched over each of the three fingers on the robot's left hand in order to increase friction and improve the quality of grasps.

B. Objects

Ten objects were used in the experiments (see Fig. 2). They were selected to have a variety of shapes and material properties. Given the small behavioral repertoire of the robot, the objects were containers in one orientation and non-containers when flipped over. All objects had approximately the same height. Five blocks were also used in the experiments (see Fig. 2). The blocks also varied in shape and material properties. All blocks were small enough to fit inside each of the containers, but large enough to be graspable by the robot.

C. Behaviors

For each object–block combination, the robot performed a short sequence of exploratory behaviors (see Fig. 1) in order to learn from their co-movement patterns, if any. The exploration sequence started with the robot grasping the object and waving it back and forth four times. The robot then dropped the block above the object. The drop point was sampled uniformly from a 10 cm x 10 cm area above the object. Next, the robot grasped

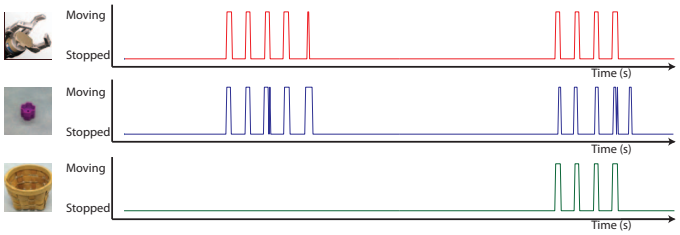

Fig. 3. Detected movements for the robot's hand, the purple block, and the wicker basket as the robot interacted with them during one trial. The first set of movement spikes was observed when the robot was shaking the block. The second set of spikes was observed when the robot was shaking the container with the block inside it.

the object and waved it back and forth four times. Finally, the robot dropped the object on the table.

Before the start of each trial the block and the object were manually placed at marked locations on the table. Double-sided tape was used to mark the location of the block and to keep it from rolling out of place before the robot could grasp it. Even though the objects were placed in the robot's field of view at the start of each trial, they sometimes left the frame during the process of manipulation.

IV. Methodology

A. Data Collection

During each trial the robot captured a sequence of 640x480 color images, recorded at 15 fps. Each trial lasted approximately 40 seconds. Thus, the robot collected roughly 600 images per trial (40 seconds $\times$ 15 frames per second). There were 100 trials in the dataset (20 objects $\times$ 5 blocks). The same data set was also used to obtain the results in [14] except [14] used another 100 trials in which a human performed the same manipulation activities as the robot.

B. Movement Detection

The movements of the features were detected using a combination of color tracking and optical flow. Color tracking was used to identify the centroid of each tracked feature. The optical flow was used to eliminate small vacillations in the positions of the features for each frame during periods of no movement. The optical flow was also used to limit the change in the tracking position of the blobs. More specifically, the displacement of the blob from frame to frame was capped at 1.5 times the magnitude of the largest vector in the segment of the optical flow field corresponding to the color blob. For this paper, the dense optical flow vectors were computed off-line using the Matlab implementation of Sun *et al.*'s state-of-the-art algorithm [15].

The output for each trial was a movement detection sequence for the block, the object, and the robot's hand (see Fig. 3). A feature was treated as moving when its position changed by more than 5 pixels between two consecutive frames. The movement detection sequence for each feature was further filtered using a box filter of width 5.

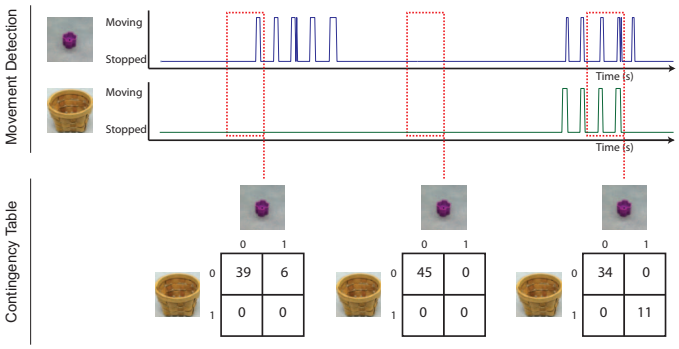

Fig. 4. The process of extracting contingency tables from the detected movements of the block and the wicker basket. Each contingency table is computed from the movement detection data within a three-second-long sliding window. Three different contingency tables are shown. The first table was generated for a temporal window during which the robot was waving the block. The second one was generated when the robot was grasping the wicker basket. Finally, the third one was calculated when the robot was waving the basket with the block inside it.

C. Extracting Movement Dependency Graphs

Sequences of movement dependency graphs were used to represent the movement patterns between different features during trials. The vertices in each of these graphs represent the three visual features that were tracked (the robot's hand, the block, and the object). The edges represent movement dependencies between pairs of features. In other words, an edge connects two vertices in the graph if and only if the movement of one feature determines the movement of another feature, at least partially. Dependencies can be observed for lack of movement as well. For example, if two features suddenly stop moving at the same time, then there will be a corresponding edge in the sequence of movement dependency graphs for a short time.

During trials, features can become attached and move together or they can become detached and move independently. For example, the robot can drop the block so that subsequent movements of the block become unrelated to the robot's hand movements. To represent these possibilities, we extracted a sequence of graphs using a sliding temporal window of size 3 seconds (i.e., 45 frames). For each window we constructed a contingency table for each of the three possible pairs of tracked features (i.e., hand–block, hand–object, and block–object). Each of these contingency tables contains four cells that correspond to the four possible combinations of two binary movement variables. Fig. 4 shows several example contingency tables for the block–object pair. The elements of a contingency table sum up to 45 because it contains data from 45 frames (i.e., 3 seconds $\times$ 15 frames per second).

The contingency table summarizes how often the features were moving together or not moving together (diagonal entries) or how often one feature was moving and the other one was not moving (off-diagonal entries). The contingency tables were recalculated incrementally as the temporal window advanced. The next section describes how a contingency table can be used to decide if the corresponding edge for a pair of objects should be present in the graph or not.

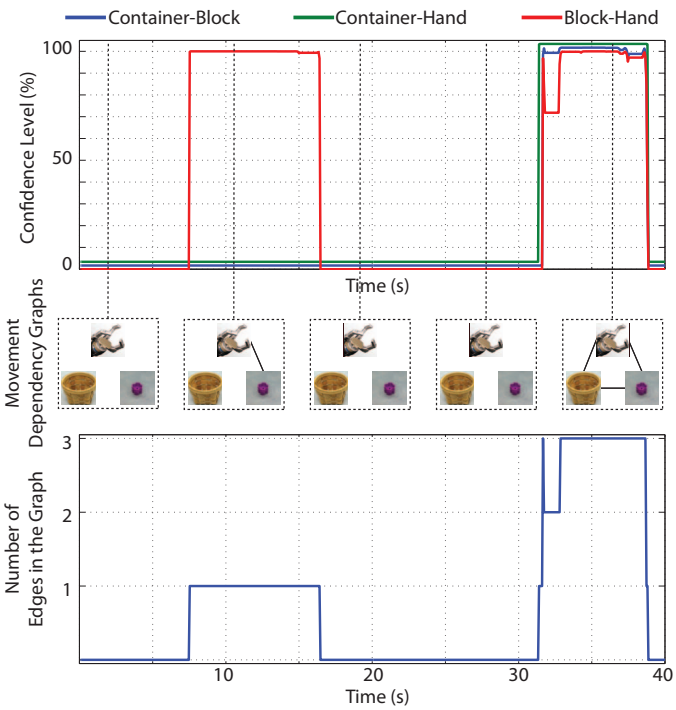

Fig. 5. The process of extracting a sequence of temporally evolving movement dependency graphs for one trial performed by the robot. An edge between a pair of features in the movement dependency graph is created if the confidence level for that pair is greater than 95%. The result is a temporally evolving movement dependency graph, which shows what the robot controls at different points in time during the trial. The bottom figure shows the number of edges in the movement dependency graphs over time.

D. Statistical Test for Movement Dependencies

Given the contingency table for two features A and B, the goal is to decide if the movements of A and B are dependent or independent. This decision is made using the *G-test of independence*, which is a statistical test of independence [16].

The G-test selects between the *null* hypothesis and the *alternative* hypothesis. In this case, the null hypothesis is that the movements of the two variables are independent. The alternative hypothesis is that they are dependent.

The G-test rejects the null hypothesis and accepts the alternative hypothesis if the p-value is below a chosen significance level. In this work, the significance level was set to 0.05, which is a standard significance level value in statistics. More formally, the rule for deciding if the p-value indicates that the two variables are independent or not is shown below:

$$\text{Independent}(p) = \begin{cases} \text{Yes}, & \text{if } p \geq 0.05; \\ \text{No}, & \text{if } p < 0.05. \end{cases} \quad (1)$$

To compute the p-value, the G-test uses the G statistic, which is defined using the following formula:

$$G = 2\sum_{i=0}^{1}\sum_{j=0}^{1} N_{ij} \ln\left(\frac{N_{ij}(N_{00}+N_{01}+N_{10}+N_{11})}{(N_{0j}+N_{1j})(N_{i0}+N_{i1})}\right),$$

where N_{00}, N_{01}, N_{10}, and N_{11} are the four elements of a contingency table (see also Fig. 4):

N_{00}	N_{01}
N_{10}	N_{11}

Note that the value of the G statistic is proportional to the mutual information between the movements of the two features.

If the null hypothesis is true, i.e., the variables are independent, then the G statistic is approximately distributed according to a χ^2 distribution with 1 degree of freedom. Thus, the p-value can be computed from the value of the G statistic using the following formula:

$$p = 1 - F_{\chi_1^2}(G),$$

where $F_{\chi_1^2}(G) = \Pr(\chi_1^2 < G)$ is the cumulative distribution function of the χ^2 distribution with 1 degree of freedom.

The confidence level, which is measured in percent, is computed from the p-value as shown below:

$$\text{Confidence Level} = (1-p)\cdot 100\%.$$

The confidence level quantifies the confidence of the G-test in rejecting the null hypothesis. According to the decision rule (1), if the confidence level exceeds 95%, which corresponds to a p-value below 0.05, then the two variables are considered dependent and the corresponding edge is added to the graph. Otherwise, the two variables are considered independent and the corresponding edge is removed from the graph. In practice, the confidence level increases exponentially as the window traverses a section of the movement data in which the movements are dependent (see Fig. 5).

V. Results

A. Analyzing the Movement Dependency Graphs

A sequence of movement dependency graphs was calculated for each of the 100 trials as described in Section IV. To show how these sequences differ between containers and non-containers, the number of edges were calculated for each movement dependency graph and for each of the 100 sequences. This results in 100 sequences of integers between 0 and 3. These integer sequences were partitioned into two groups: 1) the group of sequences for all trials with containers and 2) the group of sequences for all trials with non-containers. For each of these two groups, the median and the median absolute deviation were computed over sequence elements with the same temporal index.

The two resulting sequences, which summarize all trials with containers and non-containers, are shown in Fig. 6. The graphs show two peaks, which represent the period of time when the robot was shaking the block (first peak) and the period of time when the robot was shaking the object (second peak). A block was inside a container during 30 out of 50 trials with the containers, i.e., three links were present in the movement dependency graphs when the robot was shaking the object only in 30 of the trials. The containers were empty during the other 20 trials, which meant that the movement dependency graphs had at most one link in these trials. Fig. 6 shows that the movement dependency graph representation

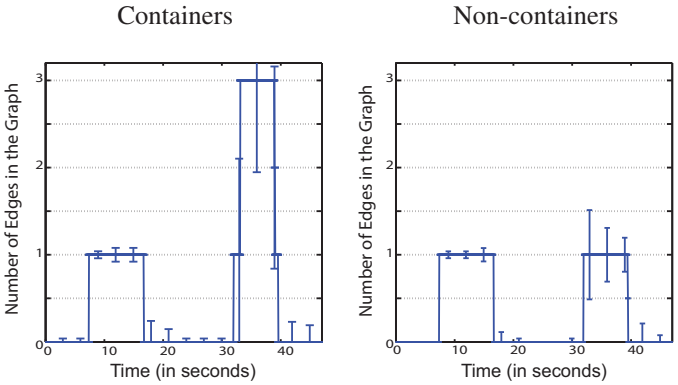

Fig. 6. The median and the median absolute deviation of the number of movement dependencies, represented by the number of graph edges, for all trials with containers and non-containers. The number of edges in the movement dependency graphs is greater for containers when the robot is shaking them (second peak) because a block can be inside a container, which adds two more edges to the graph.

clearly distinguishes between cases of containment and cases of non-containment in this experiment.

B. Object Categorization

Each sequence of movement dependency graphs (one sequence per trial) was converted into a string, for a total of 100 different strings. Because an edge was either present or not present in a movement dependency graph, each graph was mapped to a specific number, whose binary equivalent represented the existence of specific edges in the graph. A string represented the evolution of the existence of the different edges in the graph. The similarity between two strings was computed using the string edit distance between them. A 100×100 similarity matrix was constructed to represent the similarity between all pairs of strings. The similarity matrix was clustered using Spectral Clustering in order to obtain outcome classes. Finally, the objects were categorized based on the frequency with which different outcomes occurred with each object. See [5] for more details.

The object categorization results are shown in Fig. 7. The brown bucket was the only misclassified object. None of the blocks fell into that container during the trials with it, which is why it was not distinguished from the non-containers. Only a single instance of containment was necessary to distinguish the other containers from the non-containers. Thus, the robot could have easily performed a few more trials per object in order to accurately categorize all objects. In summary, the results suggest that the movement dependency graph representation can be used to distinguish between different object categories.

VI. Discussion

The representation in this paper is a significant improvement over previous work. The robot performed only 5 trials per object in this set of experiments, which was much less than the number of trials performed in our previous work (i.e., 100 trials per object in [4] and in [5]). Previously, about 40 interactions per object were required to accurately identify its type [5] and more than a single instance of containment

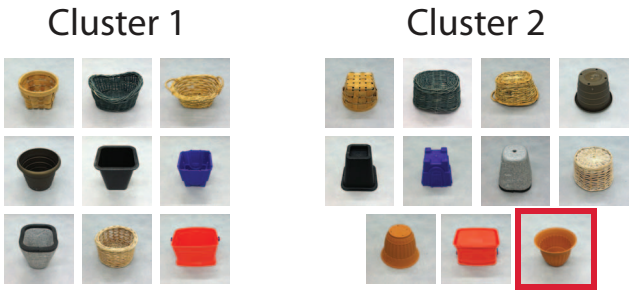

Fig. 7. The object categorization formed by the robot. The brown bucket was the only misclassified object.

was necessary to distinguish the containers from the non-containers. In this paper, however, an object was identified as a container even if a containment event was observed during only one trial with that object.

The representation described in this paper would become analogous to the representation described in [4] if the window size was increased to be equal to the number of video frames in one trial instead of being fixed at 45 frames. In this case, each sequence of movement dependency graphs would contain only one graph. That representation of movement dependencies would work well if each trial consisted of only one behavior (e.g., shake). If a trial spans more than one behavior, which is typically the case for manipulation activities, then reducing the co-movement of objects to a single value completely misses the temporal structure of the activity.

Similarly, the representation described in this paper would be analogous to the representation described in [5] if the raw visual tracking data was used instead of the sequences of movement dependency graphs. This would replace the movement dependencies between pairs of objects with traces that individual objects produce in visual space. This representation would also work well if a trial consisted of only one behavior [5]. However, this approach would not extract the exact moments when objects start or stop moving together. This information is present in the tracking data only implicitly. Thus, the ability to detect events that correspond to the beginning or the end of co-movement or separate movement, which is crucial for understanding the structure of manipulation tasks, would be missing.

Because the temporal window that was used to construct the graphs spans three seconds, the movement dependencies that persist for only a few frames are ignored. Only statistically significant movement dependencies are extracted. This is important when using the representation in this paper for comparing human and robot manipulations. The robot may move at a more consistent and slower speed than a human does. Still, a preliminary study showed that the graph–based representation can be used to compare manipulation tasks performed by the robot with manipulation tasks performed by a human [14]. In that case, however, it was necessary to use dynamic time warping to align the two sequences before they could be compared [14].

VII. Conclusion and Future Work

This paper introduced a new representation that a robot can use to form object categories. The proposed representation captures the movement dependencies between objects as a temporal sequence of graphs. The vertices in these graphs correspond to visual features tracked by the robot. The edges correspond to movement dependencies between these features. An edge between two vertices is present in a graph if and only if a statistical test indicates that the movements of the two corresponding features are dependent. Graphs are combined into sequences of graphs, which reflect how the movement dependencies between features change over time as the robot performs actions with different objects.

The representation was tested on a categorization task with container and non-container objects. The graph sequences were constructed from 100 trials during which the robot attempted to insert a small block into the objects and shake them. The results showed that the sequences of movement dependency graphs captured the differences between containers and non-containers.

Several interesting extensions of this representation are left for future work. Currently, the movement dependency graphs have unweighted edges, which assumes equal movement dependency relationships between all pairs of objects. This is not completely accurate, though. For example, when the robot is controlling a container that has a loose–fitting block inside it, the robot has less control over the block's movements. Thus, the movements of the block are not as synchronized with the robot's hand as are the movements of the container. It may be possible to extend the proposed representation to account for different types of dependencies by assigning weights to the edges of the movement dependency graphs.

Another interesting extension that is left for future work is to augment the representation with spatial information. Currently, the representation captures only movement dependencies between the objects. For different activities, however, a movement dependency between two objects could occur in different spatial relationships. For example, a block can comove with a container when it is inside the container while the container is shaken or when it is next to the container while the container is pushed. An improved representation that also captures spatial information between objects could distinguish between these two different types of movement dependencies.

References

[1] S. Hespos and E. Spelke, "Precursors to spatial language: The case of containment," in *The Categorization of Spatial Entities in Language and Cognition*, M. Aurnague, M. Hickmann, and L. Vieu, Eds. Amserdam, Netherlands: Benjamins Publishers, 2007, pp. 233–245.

[2] E. Spelke, "Initial knowledge: six suggestions," *Cognition*, vol. 50, no. 1, pp. 431–445, 1994.

[3] R. Largo and J. Howard, "Developmental progression in play behavior of children between nine and thirty months. II: Spontaneous play and language development," *Developmental Medicine and Child Neurology*, vol. 21, no. 4, pp. 492–503, 1979.

[4] S. Griffith, J. Sinapov, M. Miller, and A. Stoytchev, "Toward interactive learning of object categories by a robot: A case study with container and non-container objects," in *Proc. of the 8th IEEE Intl. Conf. on Development and Learning (ICDL)*, Shanghai, China, June 2009.

[5] S. Griffith, J. Sinapov, V. Sukhoy, and A. Stoytchev, "A behavior–grounded approach to forming object categories: Separating containers from non-containers," *To appear in the IEEE Transactions on Autonomous Mental Development*, 2011.

[6] K. Gold and B. Scassellati, "Using probabilistic reasoning over time to self-recognize," *Robotics and Autonomous Systems*, vol. 57, no. 4, pp. 384–392, 2009.

[7] A. Stoytchev, "Self-detection in robots: A method based on detecting temporal contingencies," *Robotica*, vol. 29, pp. 1–21, 2011.

[8] G. Metta and P. Fitzpatrick, "Early integration of vision and manipulation," *Adaptive Behavior*, vol. 11, no. 2, pp. 109–128, June 2003.

[9] C. Kemp and A. Edsinger, "What can I control? The development of visual categories for a robot's body and the world that it influences," in *Proc. of the 5th Intl. Conf. on Development and Learning (ICDL), Special Session on Autonomous Mental Development*, 2006.

[10] E. Aksoy, A. Abramov, F. Worgotter, and B. Dellen, "Categorizing object-action relations from semantic scene graphs," in *Proc. of the IEEE Intl. Conf. on Robotics and Automation (ICRA)*, Anchorage, AK, July 2010, pp. 398–405.

[11] M. Sridhar, A. Cohn, and D. Hogg, "Learning functional object categories from a relational spatio-temporal representation," in *Proc. of the 2008 conf. on ECAI*, 2008.

[12] E. Ugur, M. Dogar, M. Cakmak, and E. Sahin, "The learning and use of traversability affordance using range images on a mobile robot," in *Proc. of the IEEE Intl Conf. on Robotics and Automation*, 2007, pp. 1721–1726.

[13] L. Montesano, M. Lopes, A. Bernardino, and J. Santos-Victor, "Learning object affordances: From sensory-motor coordination to imitation," *IEEE Trans. on Robotics*, vol. 24, no. 1, pp. 15–26, 2008.

[14] V. Sukhoy, S. Griffith, and A. Stoytchev, "Toward imitating object manipulation tasks using sequences of movement dependency graphs," in *the RSS 2011 Workshop on the State of Imitation Learning (to appear)*, Los Angeles, CA, June 27, 2011.

[15] D. Sun, S. Roth, and M. Black, "Secrets of optical flow estimation and their principles," in *Proc of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, San Francisco, CA, August 2010, pp. 2432–2439.

[16] R. Sokal and F. Rohlf, *Biometry: the principles and practice of statistics in biological research*, 3rd ed. Freeman, New York, 1994.