Using a Minimal Action Grammar for Activity Understanding in the Real World

Douglas Summers-Stay, Ching L. Teo, Yezhou Yang, Cornelia Fermüller and Yiannis Aloimonos

Abstract— **There is good reason to believe that humans use some kind of recursive grammatical structure when they recognize and perform complex manipulation activities. We have built a system to automatically build a tree structure from observations of an actor performing such activities. The activity trees that result form a framework for search and understanding, tying action to language. We explore and evaluate the system by performing experiments over a novel complex activity dataset taken using synchronized Kinect and SR4000 Time of Flight cameras. Processing of the combined 3D and 2D image data provides the necessary terminals and events to build the tree from the bottom-up. Experimental results highlight the contribution of the action grammar in: 1) providing a robust structure for complex activity recognition over real data and 2) disambiguating interleaved activities from within the same sequence.**

I. Introduction

How do humans come to understand, recognize, and replicate actions? Even if we have witnessed several occurrences of the same activity, each will be unique in terms of the order actions are performed, the explicit motion of the limbs involved, and the appearance of the objects involved. Somehow, the sensory data must be stored in a greatly compressed representation that captures relevant information while discarding what is irrelevant. This representation must be capable of handling actions of any complexity, where activities are composed of previously known actions and sub-actions.

This suggests that the brain uses a similar method for understanding both language and actions. This idea has support on neuroscientific, anthropological, and behavioral grounds. In the early 1950's, psychologist Karl Lashey suggested that syntax may apply to goal-directed actions as well as to language [28]. Archaeologist Leroi-Gourhan argued that tool making and use reflects a capability for compositionality of structures, linking language and action [5]. Two-year old children have been shown to have the ability to recognize and reproduce hierarchically organized actions [2], [37]. Additionally, the same parts of the brain that have long been understood to be used in language production (such as Broca's Area) have been found to be crucial to the process of action planning [8], [9].

If such a representation is taken to be an innate, central aspect of both language and activity understanding, it must be simpler and more fundamental than the grammars we learn for each individual language. It also must be a generative grammar (rather than one used purely for recognition) in order to allow an individual to learn to perform actions by example. Chomsky's minimalist program is an attempt to discover such a universal generative grammar for language, with the expectation that it plays a more general cognitive role. A generative grammar consists of a set of elements and a set of production rules that allow the formation of grammatical sentences. Context-free grammars are generative grammars which have recursive rules which allow nesting of elements in the same type of elements. Although context-free grammars are sufficiently expressive to handle the complexity of language, they cannot account for what we actually see in natural languages, such as agreement (in case, number, or gender) and reference (such as relative clauses.) These long-distance dependencies cannot be captured by context-free grammars. The Chomskyan Minimalist Program deals with this through a number of transformations on the output of context-free grammars [6].

In [25], Pastra and Aloimonos introduce a minimalist grammar of action which defines the set of terminals, features, non-terminals and production rules for such a grammar in the sensorimotor domain. However, this was a purely theoretical description. The action grammars used in our experiments are an implementation of such a grammar in a system that is capable of sensing and interpreting real-world situations under a wide range of natural conditions. Such a representation is a natural summary of the important aspects of an activity, which abstracts away such details as who is performing the action, where it is being performed, how the objects involved are spatially located, and the appearance of the objects and body parts. What is left is a tree structure that captures the order in which tools, objects and hands are brought together and separated, a structure which is easy to store, search, and compare to other such trees.

In order to make the use of the grammar practical for real robots and surveillance, we have not merely created a demonstration, but designed the system so that it will be able to handle and abstract away a wide range of realistic conditions, such as varying viewpoint, lighting, surrounding environment, object and actor appearance.

The activities we are attempting to recognize and understand are complex, concrete human activities. Actions like "stirring" or "tightening a bolt," the traditional purview of action recognition techniques, are represented by a single node in the action tree. (For this reason, we refer to what we are doing as "activity recognition" rather than "action recognition.") Abstract actions, like "doing research," or "playing soccer" contain important steps which take place

The authors are from Department of Computer Science, University of Maryland, College Park, MD 20742, USA {dss,cteo,yzyang,fer,yiannis}@umiacs.umd.edu

in terms of mental or data structures, which we have no way to detect or estimate with the current setup. Instead we are looking at multi-step activities which involve the manipulation of physical objects towards some goal state. This is basically any form of manual labor: the physical work of craftsmen, home builders, factory workers, chefs, janitors, and so forth. These are also largely the kinds of activities which we would hope for a general purpose robot to be able to perform.

"Action recognition" interacts with activity recognition in two important ways, both assisting with and being assisted by activity recognition. First, activity recognition provides important context for action recognition. One of the main difficulties in action recognition is finding when the action begins and ends. Forming an action tree provides natural endpoints for individual actions: these actions occur between the time a tool (including the hands) comes into contact with an object and the time when it breaks such contact. Knowing what the tool and object are provides significant constraints on what the action might be, reducing it to a handful of possibilities with any significant probability of occurring. Second, when action recognition is performed, the action can be used as a label on part of the activity tree, which improves our ability to match with similar activities.

II. Recent Works

The problem of action recognition and human activity has been an active research area in Computer Vision, motivated by several promising applications, such as human-computer interface, video indexing and retrieval and video surveillance, etc. Several excellent surveys on the topic of visual recognition are available [22], [34]. But non-visual descriptions, using motion capture systems, have also been of interest in Computer Vision and Graphics. Many of those studies are concerned with dimensionality reduction techniques, that provide a good characterization for classification [3], [36], [20]. Most of the focus in visual action analysis was on the study of human actions that were characterized by movement and change of posture, such as walking, running, jumping etc. The dominant approaches to the recognition of single actions compute statistics of spatio-temporal interest points [18], [38], [7] and flow in video volumes as descriptors, or represent short actions by stacks of silhouettes [11], [39]. Approaches to more complex, longer actions employ parametric approaches, such as Hidden Markov Models [15], Linear Dynamical Systems [30] or Non-linear Dynamical Systems [4], which are defined on tracked features or optic flow presentations.

To capture the semantics of complex activities, higher level reasoning methods are required. A number of approaches use stochastic context free grammars with the primitives beings body parts [29] or trajectories [14], and some also include the interaction with objects [23]. To model the temporal constraints, several approaches have used Hidden Markov Models, which allow to exploit the relation between specific objects and actions [13], [24]. A related class of approaches use dynamic Bayesian networks to divide the temporal sequence into sub-sequence and define relative temporal relations [27], [10], [19].

Most closely related to our work are a few recent studies on hand manipulation actions. In [35] manipulation actions are represented as a sequences of motion primitives. The process is modeled using a combination of discriminative support vector machines and generative hidden Markov models. In [16] hands and objects segmented from the video and shape-based hand/object features and manipulation features are defined to provide a sequence of interrelated manipulations and object features. Semantic manipulation object dependencies are extracted using conditional random fields. In [33] manipulations in a breakfast scenario are analyzed. The image sequence is represented by an activity graph that codes spatiotemporal object interactions. Event classes are extracted from the activity graphs, where each event class encodes a similar pattern of spatiotemporal relations between corresponding objects, but the objects are known beforehand. While all these approaches use task-dependent primitives, our approach is general; its basics are simply the merging and parting of objects. A similar idea was pursued by [1] for the analysis of short stereo video sequences of of hand motions manipulating a number of objects. Relations between object at decisive time points during manipulation, such as when two objects touch or overlap, are stored in a transition matrix. Using simple sub-string search algorithms different matrices are compared for recognition. The objects in these sequences are however easily visually recognized, and the approach was only applied to short activities, such as putting two objects on a plate.

III. Approach

We describe the overall approach of using the action grammar for activity understanding (see Fig. 1) by first introducing the experimental dataset in sec. III-A. Next, we define the action grammar and how it is created in sec. III-B. We then detail how the important subcomponents: hand state determination and object recognition are achieved in secs. III-C and III-D respectively. With these detections, we illustrate how an *activity* tree can be built (sec. III-E) and be used for comparing the similarity between different trees (sec. III-F) and how the action grammar is useful for separating complex interleaved activities in sec. III-G.

A. Kinect+SR4000 Complex Activity Dataset

We introduce a novel dataset that contains 5 complex hand manipulation activities performed by 4 different human actors. Each *activity* is defined by the completion of a complex object or entity: for example, making a sandwich. The task of creating this entity is further comprised of 9 specific *actions* which may involve the use of different kinds of hand-tools and objects. Different actions could be concatenated to form novel activities: Cooking vegetables + making a sandwich. Other well known datasets such as the KTH, Weizmann or Human-EVA datasets [31], [11], [32] do not involve hand-tools. The human-object interaction dataset by Gupta et al. [12] has only 4 objects with extremely

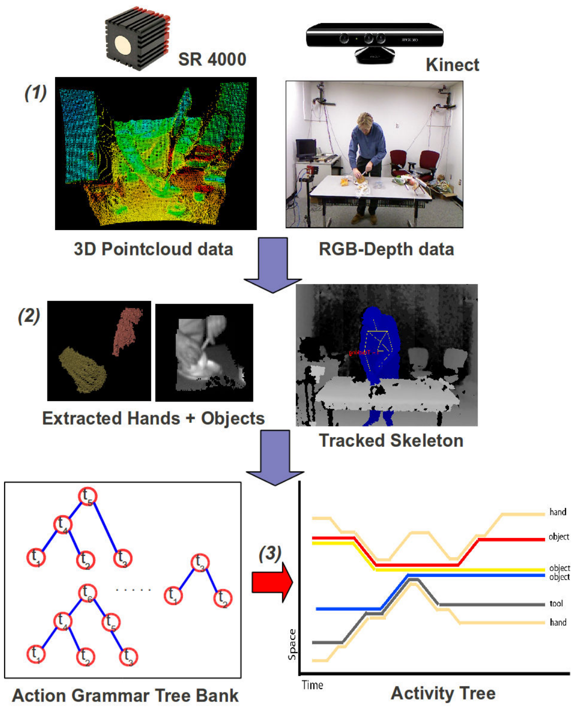


Fig. 1. Overview of the approach: (1) Pointcloud data and RGB-Depth data are extracted and processed from the SR4000 and Kinect cameras respectively. (2) Hands and objects are detected from pointclouds and the human pose is extracted from Kinect. (3) The detected objects are then combined using an action grammar treebank to produce an activity tree.

simple actions. The dataset by Messing et al. [21] has only 4 simple actions with tool use. The CMU Kitchen Dataset [17] has several actions performed by 18 subjects for 5 recipes, but many of the actions are blocked from view due to the placements of the 4 static cameras.

The Complex Activity Dataset extends beyond these datasets by considering the compositional aspect of the activity in terms of the entities created by the actions involved in each step. The activities are classified into two general categories: *Kitchen* and *Crafts*, each with 8 separate video sequences captured from two externally synced and calibrated active sensors: 1) the Kinect which provides RGB-Depth and 2) a Swissranger SR4000 Time of Flight camera which provides Intensity(Grayscale)-Depth. The Kinect camera is positioned frontal-parallel at a distance of $\approx$ 6m from the actor so that we can track the entire body motion, while the SR4000 is positioned $\approx$ 1.5m from the actor on the side so that hand-actions and objects can be clearly seen (see Fig. 2). In total, there are 16 video sequences made from different combinations of activities and objects. The sequences are fully annotated with of the names of relevant objects and manipulative actions for evaluation and training purposes. The list of activities, actions and objects considered are summarized in Table I. Sample sequences from the dataset are available in the supplementary material[1].

[1]More information on the dataset and how the data is collected can be found at: `http://www.umiacs.umd.edu/research/POETICON/umd_complex_activities/`

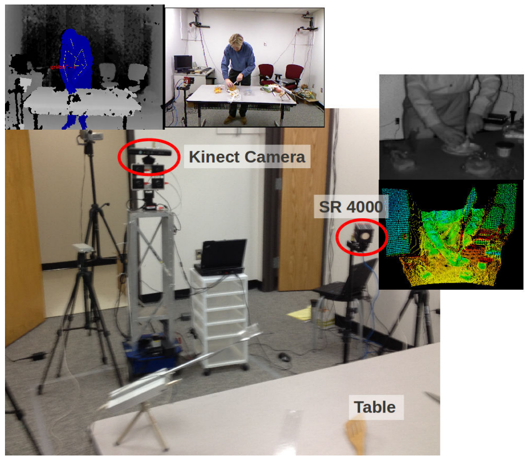


Fig. 2. Data collection setup. The Kinect is mounted on an Erratic mobile robot base, the SR4000 is mounted on the side nearer to the table where the actions are performed.

B. The Action Grammar

The grammar has one simple rule, which is applied repeatedly:

An activity consists of

1) Using a tool to bring two objects together, resulting in a new object or a tool or
2) Using a tool together with a single object, resulting in a transformed object or tool

These tools or objects can themselves be the result of an activity, which gives rise to the tree structure of activities. Hands can be thought of as tools which are not made of other objects or tools. The new ”object” can be something like ”a piece of bread on a plate” formed by bringing together a slice of bread and a plate. The point is that after they have been brought together, they are treated as one combined object, temporarily, as the bread moves together with the plate.

There are two ways to use the grammar. In this paper we parse the actions that take place, starting with recognizing simple actions (of type 1 or 2, above) and building them up into an activity tree, an example is shown in Fig. 5 (right). Every non-terminal node of this tree is an action. The other way to use the grammar would be in a generative way: starting with an activity one wanted to perform, and working out how to do it. One would simply look for an activity one has observed resulting in the final object one wanted to have, find out what the input objects and actions to attain that are needed, and what activities result in *those* objects, and so on, breaking it down to the point that the objects and tools needed are the ones available.

C. Extracting Hand Locations from 3D Pointclouds

Since we are concerned with *manipulative* actions, the terminals in the action grammar trees are the objects/tools that are currently manipulated by the hands. An approach that passively searches for objects and tools in the video will not be sufficient as many objects are visible on the table but are not participating in the activity. Instead, we actively

Activity Class	Name	Actions	Objects/tools
Kitchen	Cooking Vegetables	{*slice, cut*}	{*cucumbers, carrots, tomatoes, apple, chopper*}
	Making Sandwich	{*spread, slice*}	{*bagel, ham, cheese, knife*}
Crafts	Sewing a Toy	{*cut, pin, thread, sew*}	{*cloth, paper, needle, thread, scissors*}
	Card Making	{*cut, paste, write, fold*}	{*paper, glue, scissors, marker*}
	Assemble a Machine	{*screw, push, twist*}	{*wrench, nut, bolt, frames*}

TABLE I

LIST OF MANIPULATION ACTIVITIES CONSIDERED.

search for hands in each frame, and determine if the hand is currently occupied with an object or free directly from 3D pointclouds – a binary hand state $H_s = \{occ, free\}$. Once we know the approximate location of each hand and its state, a trained object classifier can then be used only on these regions, which reduces processing time and false positives. Note that we process pointclouds from the SR4000 since it is nearer to the actor than the Kinect and provides a clearer view for object recognition.

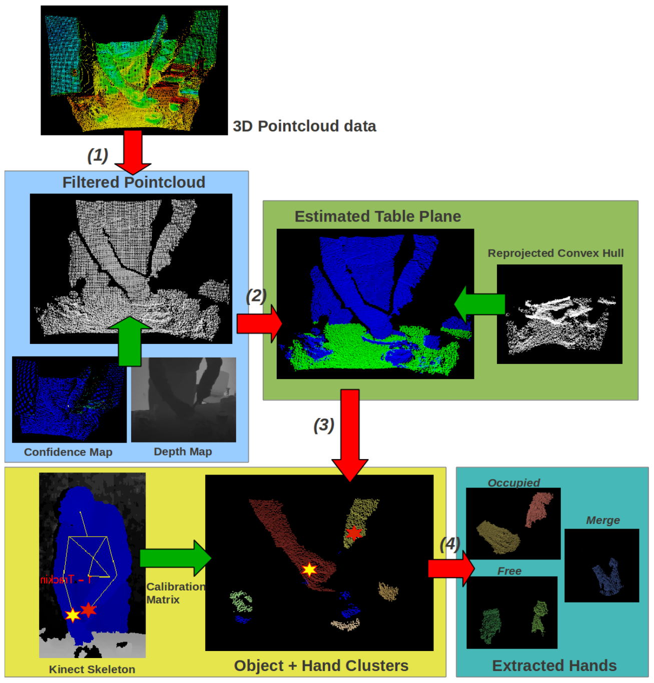


Fig. 3. Detecting hand locations from SR4000 pointclouds. (1) Outliers are first removed, (2) Table surface is then estimated, (3) Objects and hands are extruded and clustered from reprojected convex hull, and (4) Predicted hand pointcloud locations.

The procedure is summarized on Fig. 3. The inputs are the pointclouds obtained from the SR4000 and the tracked skeleton from the Kinect[2]. Since both cameras are calibrated, the approximate 3D locations of the tracked hands are known in the SR4000 camera coordinates. However, relying solely on the Kinect for hand tracking is unreliable since it may fail, especially when half the body is blocked by a table (see Fig. 2). Our proposed approach is to 1) robustly extract potential hand regions from SR4000 pointclouds, and 2) combine it with the predicted locations from Kinect, so as to determine the final locations of the hands in SR4000 and its hand state: occupied or free. We use PCL 1.4[3] as the main pointcloud processing library to first remove obvious outliers by filtering out points that have low confidence values or those does not belong to any obvious cluster. A plane estimation procedure using RANSAC is then applied to estimate a planar model that represents the table surface. The estimated coefficients are then used to reproject the remaining points so that a convex hull is created from which points in the original cloud that are within the convex hull (table points) are removed. A 3D Euclidean clustering is then applied to obtain reasonable pointcloud clusters. The predicted hand locations from Kinect are then used to extract the hand pointclouds if the location is within a fixed distance threshold of the cluster's centroid. Finally, the extrema of each hand pointcloud is computed from which we use a region growing segmentation algorithm using nearest neighbors to extract the hand and any associated objects/tools that are in contact with the hand (Fig. 3(d)). The current hand state H_s is obtained from the difference in the pointcloud sizes against a running average of previous pointcloud sizes. A significant deviation beyond a ratio threshold will indicate that the hand is occupied (ratio > 1) or empty (ratio < 1). In addition, if only a single hand pointcloud is detected, and its current size is approximately equal to the combine sizes of the the left and right hand in previous frames, a *merge* event is raised. This will be important for building the activity tree (sec. III-E).

D. *Object Recognition*

The activity tree is built when there are changes H_s for each hand (left and right), and the terminals are the objects (if any) on each hand when H_s changes. Using the results of the predicted hand locations and states described in the previous section, we crop out a rectangular region slightly larger than the hand point cloud size to obtain an intensity image of the potential objects/tools should $H_s = occ$ (Fig. 4). We also extract the cropped region whenever a merge event starts or ends.

[2]PrimeSense OpenNI implementation was used to obtain the skeleton.

[3]`http://www.pointclouds.org`

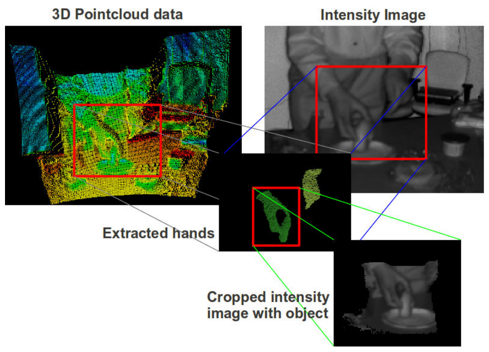


Fig. 4. Merging occupied hand pointclouds with intensity image for object recognition.

We extract Histogram of Gradient (HoG) features from the cropped image from which an object classifier is then used to predict the object label. Classifiers for each object/tool class are trained over a separate training set of labeled data using a degree three polynomial SVM. We select the object labels from the classifier that gives the highest response for the case when $H_s = occ$. Due to the large amounts of occlusions when a merge event occurs, we select from the top $N = 4$ detection responses the most consistent object label from the previous frames (since it is unlikely that an object label changes when a merge occurs).

E. Building the Activity Tree

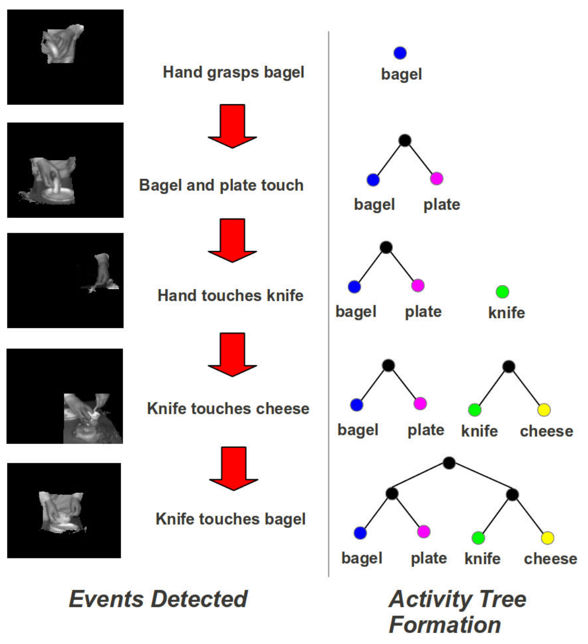


Fig. 5. Creating an Activity Tree: (Left) Events and objects detected from SR4000 intensity images. (Right) Formation of an activity tree that parallels the events and objects occurrence, based on the defined action grammar.

The previous steps tell us what objects or tools are grasped by each hand at each frame of the recording, and when objects are brought together and begin to be treated as a single object (a *merge* event). This provides enough information to build the activity tree, as shown in Fig. 5. The parser creates a new leaf node whenever one of the actor's hands (or tools held in the hands) come into contact with a new object. These nodes keep track of the time the object was first and last seen, and what the object was recognized as (using the HoG object recognition described in sec. III-D.) If these objects are brought together, a new node is created with each of the original object nodes as children. This process gradually builds up tree structures.

Detecting when objects are brought together is not a foolproof method of recognizing when the objects begin to be treated as one combined object. One may, for instance, pick up two unrelated objects in one hand just to clear away a working space. In videos where this kind of event happens frequently, a better means of recognizing a significant, meaningful contact would be needed, or a better way of tracking where the objects ended up during a merge event.

To build a robust system, it would be useful to have several examples of each activity one wanted to recognize, and measure whether the activity tree from an unknown activity fell into this cluster. There are many ways to perform any object manipulation activity. Certain aspects proved to be highly variable. Objects were frequently handled, set down, and picked up again, or passed from one hand to another, in an irregular way. A cutting action might be followed by a second cutting action if the results of the first cut were deemed unacceptable. The order of some of the actions differed from one trial to another. Certain aspects of the activities however, were much more consistent. In order to correctly perform the tasks, certain objects needed to come together and not be separated again before the end of the activity. In the *Sewing a Toy* activity, for example, the pieces of felt needed to be joined together with the thread. Recognizing these critical merge events is crucial for understanding the activities.

F. Comparing Activity Trees

In order to compare the trees, we use a measure of tree *edit distance*. The edit distance between two trees is the minimal-cost sequence of edit operations on labeled nodes that transforms one tree into the other. The following operations are possible, each with its own cost:

- inserting a node (between a node and a subset of its children)
- deleting a node (and connecting its children to its parent)
- renaming a node (changing the label)

Efficient algorithms exist to find the minimal edit distance. The most recent advancement was made by Pawlik et. al [26] which has $O(n^3)$ time with $n > m$ and $O(mn)$ space complexity for the worst case. For the small sizes of trees we encounter, solving this takes negligible time and memory. Using the tree edit distance is critical for discriminating situations where the same objects are used in different ways. For example, if one were to put drafting instruments into a case to bring to school, the tree would consist of each instrument being merged with the case one by one. However, if one were to use the instruments for drafting, a more complex tree would be created, where the triangle is used with the pencil and paper, then moved out of the way, and so forth. Forming activity trees allows us to capture the structure of this interaction.

G. Separating Interleaved Activities

In many cases, an activity is an uninterrupted sequence of related events. In this case, segmenting the activity in a recording means simply finding the beginning and end point of the activity. However, there may be interruptions, in which the actor is trying to deal with more than one goal simultaneously. This results in actions that are mixed together. By parsing these actions, we are able to get a fine grained segmentation of the recording, identifying which actions belong to each activity, even when they are thoroughly mixed. To demonstrate this, several activities in the Complex Activity Dataset contain interleaved actions of *different* activities combined together. For example, the actor was asked to perform a cutting and pasting task (*Making Card*) and the *Assemble a Machine* task, interleaving the actions for each activity. Because the activities involved separate objects, we were able to use the grammar to successfully separate out the actions for each task. This is shown in Fig. 6. As we will see in the experiments (sec. IV-C), the strength of this approach is highlighted when we are able to recognize such complex interleaved activities much better than a simpler approach when no action grammar is imposed.

IV. Experiments

We report the results of two experiments that evaluate the performance of the action grammar in recognizing complex manipulation activities. We first derive theoretical bounds of the expected performance by inducing artificial noise in the terminals (sec. IV-B) and then evaluate the performance of recognizing activity trees over real data from the Kinect+SR4000 Complex Activity Dataset (sec. IV-C).

A. Experimental Procedure

For the experiment that explores the theoretical performance of the action grammar, we manually induced corruption in the input terminals of each activity tree from the Complex Activity Dataset in 2 ways: 1) by sampling from a uniform distribution of all possible object labels considered (except the ground truth) and 2) by consistently selecting the object labels from only one but a *different* activity tree for each associated object: e.g, if the activity was *Card Marking*, we will replace object labels consistently from another activity such as *Cooking Vegetables*. We considered corruption of the input ranging from 10% (almost correct) to 90% (almost all wrong) and report the accuracy scores in interpreting the activity using the corrupted activity tree using the following procedure: for each level of corruption, we compute the edit distances for each tree, and take the ground truth identity of the smallest edit distance. We then count how many trees are correctly matched and report the accuracy score per level.

The next experiment evaluates the action grammar over 12 activities from the Complex Activity Dataset. In this part, we used a leave-one-out training procedure to train the object classifiers – for each test sequence, the remaining 11 sequences were used for training. Note that 4 sequences involving *Sewing* and *Assembling a Machine* are left out of the evaluation due to the fact that the object recognition simply failed as the objects of interests: pins, bolts, etc. are too small[4]. We then report the normalized tree edit distances of the resulting activity trees when they are compared with the ground truth, together with the amount of terminal corruption per sequence. As a comparison to highlight the contribution of the action grammar in building the activity tree, we also report the activity recognition performance when only the terminals are used to build a degenerate tree of depth 1 only (a flattened tree).

B. Results over Artificial Noisy Data

The accuracy scores over increasing degree of terminal corruption are summarized in Fig. 7.

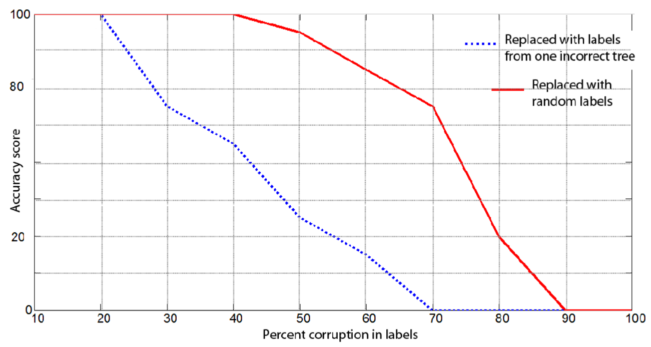


Fig. 7. Accuracy scores with varying degrees of terminal corruption: 1) Randomly replaced object labels (red solid line) and 2) Replaced object labels consistently from another (incorrect) tree (blue dotted line).

The activity trees are robust enough to handle the fact that no object detection method is completely accurate. In an attempt to characterize the behavior of tree matching in the presence of noise, we considered two possible causes of terminal corruption as described in the previous section. In the first case where the missed detections are completely random (the red solid line), the trees perform fairly well, accurately matching the true tree to the partially mislabeled tree in all cases until 40% of the labels have been replaced. In the second case (the blue dotted line), all the incorrect labels come from a single incorrect tree and so are consistent with each other. In this worst case scenario, the performance does worse, and errors in recognition show up when 20% of the labels are incorrect.

C. Results over Complex Activity Dataset

We summarize the matching performance for the 12 test activity trees in Fig. 8 and compare it against the baseline method of using terminals alone (Fig. 9).

[4]the specific sequences used and left out can be found at `http://www.umiacs.umd.edu/research/POETICON/umd_complex_activities/`

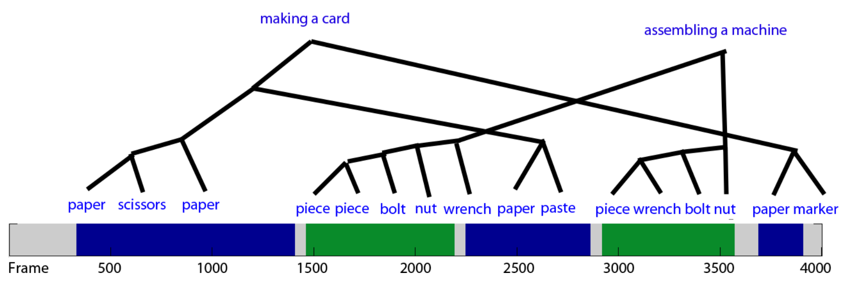
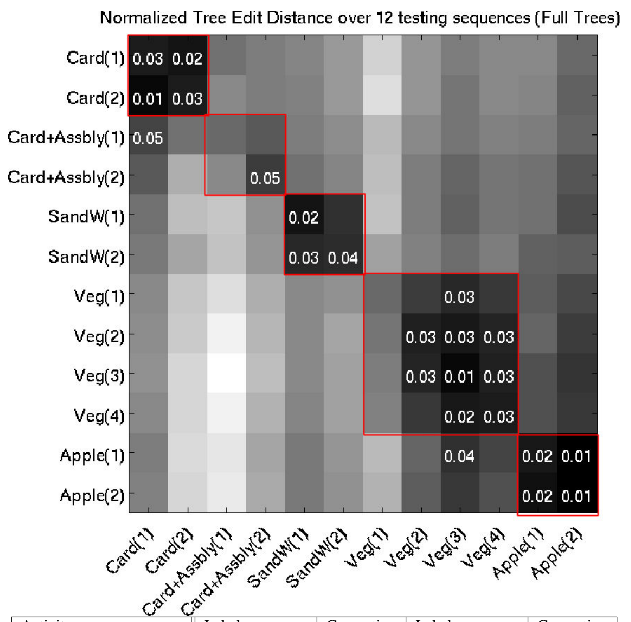


Fig. 6. A complex interleaved sequence *Making Card* + *Assemble a Machine* can be cleanly separated into its component activity trees using the action grammar.



Activity	Label	Corruption	Label	Corruption
Card Making	Card(1)	0.48	Card(2)	0.49
Card Making+Assemble Machine	Card+Assbly(1)	0.55	Card+Assbly(2)	0.48
Making Sandwich	SandW(1)	0.55	SandW(2)	0.36
Cooking Vegetables	Veg(1)	0.42	Veg(2)	0.62
	Veg(3)	0.54	Veg(4)	0.47
Cutting Apples[a]	Apple(1)	0.55	Apple(2)	0.42

[a]A subset of the *Cooking Vegetables* activities

Fig. 8. (Above) Confusion matrix of normalized tree edit distances for each of the 12 test sequences. Lower values along the diagonals are better. Boxes indicate the diagonal blocks of interest for each set. (Below) Amount of corrupted terminals [0, 1] per testing sequence. A value closer to 1 means more corruption.

In order to measure how well the activity trees could be used for activity recognition in real data, we computed the tree edit distance distance between each test tree and the ground truth for each of the activities. Each activity comes as a set containing at least 2 similar sequences. For example, *Card Making* has two sequences: *Card(1)* and *Card(2)*, performed by 2 different actors which introduces a small amount of variation within each activity set itself. In the confusion matrix above, the blocks of low edit distances along the diagonal for each activity set and higher distances elsewhere indicate that the activity trees are finding fairly good matches among the correct set of activities (Fig. 8 (above)). This performance is achieved in spite of the high

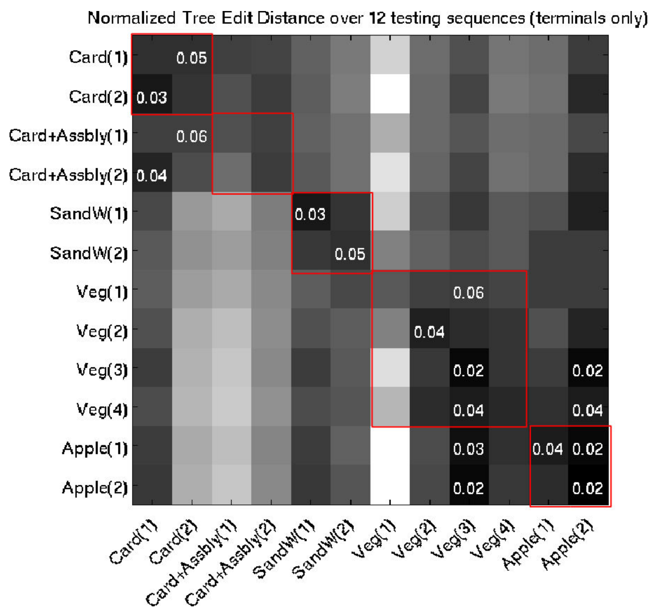


Fig. 9. Confusion matrix of normalized tree edit distances when terminals are used alone. Lower values along the diagonals are better. Boxes indicate the diagonal blocks of interest for each set.

levels of corruption in the terminals (Fig. 8 (below)) of between 36% to 62% that are sufficient to degrade performance (shown in the first experiments), which is indicative of the robustness of the approach in noisy real data.

By way of comparison, we flattened the trees so that all the nodes were at the same level (depth 1) and repeated the same experiment (Fig. 9). This effectively eliminates the effect of using the action grammar. In this case, the diagonal structure is much less evident, highlighting that the absence of the tree structure derived from the action grammar greatly reduces the ability of the system to find the right matches. This is especially true for activity sets that contains complex interleaved activities such as *Card Making* + *Assemble a Machine* and *Cooking Vegetables*. As was explained in sec. III-G and illustrated in Fig. 6, the ability of the action grammar in disambiguating complex interleaved activities is shown by the fact that the block diagonals for such activities display lowered performance when flattened trees are used (the tree edit distances are much higher within each block) compared to the ones when the full action grammar is used in the previous experiment (Fig. 8).

V. Conclusion and Future Work

Using a grammar of action to build activity trees appears to be a practical way to begin to parse complex activities. We are considering many possibilities for how to build on the current work. The grammar described here could easily be extended to include more specific patterns to be matched, creating a richer grammar that provides immediate information about actions and subactions. More traditional action recognition techniques could also be incorporated. For example, when a tool touches an object, we could determine whether the tool is actually performing its function on the object and transforming it, or just coming into contact with it. Object recognition could be improved by feedback from the tree structure, increasing the probability of detection for an object consistent with the rest of the tree. The activity trees could also be used by a robot to emulate activities it has seen performed previously by humans, or even generated based on goals the robot is trying to attain.

References

[1] E. E. Aksoy, A. Abramov, J. Dörr, K. Ning, B. Dellen, and F. Wörgötter. Learning the semantics of object-action relations by observation. *International Journal of Robotics Research*, 30(10):1229–1249, 2011.

[2] P. Bauer. Recalling past events: from infancy to early childhood. *Annals of Child Development*, 11:25–71, 1995.

[3] R. Chalodhorn, D. Grimes, R. R. Gabriel, and M. Asada. Learning humanoid motion dynamics through sensory-motor mapping in reduced dimensional spaces. In *ICRA*, 2006.

[4] R. Chaudhry, A. Ravichandran, G. Hager, and R. Vidal. Histograms of oriented optical flow and binet-cauchy kernels on nonlinear dynamical systems for the recognition of human actions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[5] J. Chavaillon. Andr leroi-gourhan, le geste et la parole. *L'Homme*, 7(3):122–124, 1967.

[6] N. Chomsky. *Lectures on Government and Binding: The Pisa Lectures*. Mouton de Gruyter, 1993.

[7] P. Dollár, V. Rabaud, G. Cottrell, and S. Belongie. Behavior recognition via sparse spatio-temporal features. In *VS-PETS*, October 2005.

[8] L. Fadiga, L. Fogassi, V. Gallese, and G. Rizzolatti. Visuomotor neurons: ambiguity of the discharge or motor perception? *International Journal of Psychophysiology*, 35:165–177, 2000.

[9] L. Fogassi, P. F. Ferrari, B. Gesierich, S. Rozzi, F. Chersi, and G. Rizzolatti. Parietal lobe: From action organization to intention understanding. *Science*, 308:662 – 667, 2005.

[10] S. Gong and T. Xiang. Recognition of group activities using dynamic probabilistic networks. In *Proc. International Conference on Computer Vision*, 2003.

[11] L. Gorelick, M. Blank, E. Shechtman, M. Irani, and R. Basri. Actions as space-time shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2247–2253, 2007.

[12] A. Gupta and L. S. Davis. Objects in action: An approach for combining action understanding and object perception. In *CVPR*. IEEE Computer Society, 2007.

[13] S. Hongeng and R. Nevatia. Large-scale event detection using semi-hidden markov models. In *Proc. International Conference on Computer Vision*, 2003.

[14] Y. Ivanov and A. Bobick. Recognition of visual activities and interactions by stochastic parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2000.

[15] A. Kale, A. Sundaresan, A. N. Rajagopalan, N. P. Cuntoor, A. K. Roy-Chowdhury, V. Kruger, and R. Chellappa. Identification of humans using gait. *IEEE Transactions on Image Processing*, 13(9):1163–1173, 2004.

[16] H. Kjellstrom, J. Romero, and D. Kragic. Simultaneous visual recognition of manipulation actions and manipulated objects. In *Proc. European Conference on Computer Vision*, 2008.

[17] F. D. la Torre, J. Hodgins, J. Montano, S. Valcarcel, R. Forcada, and J. Macey. Guide to the carnegie mellon university multimodal activity (cmu-mmac) database. Technical report, CMU-RI-TR-08-22, Robotics Institute, Carnegie Mellon University, July 2009.

[18] I. Laptev. On space-time interest points. *International Journal of Computer Vision*, 64(2–3):107–123, 2005.

[19] B. Laxton, J. Lim, and D. Kriegman. Leveraging temporal, contextual and ordering constraints for recognizing complex activities in video. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2007.

[20] Y. Li, C. Fermuller, Y. Aloimonos, and H. Ji. Learning shift-invariant sparse representation of actions. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2010.

[21] R. Messing, C. Pal, and H. Kautz. Activity recognition using the velocity histories of tracked keypoints. In *ICCV '09: Proceedings of the Twelfth IEEE International Conference on Computer Vision*, Washington, DC, USA, 2009. IEEE Computer Society.

[22] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding*, 104:90–126, 2006.

[23] D. Moore and I. Essa. Recognizing multitasked activities using stochastic context-free grammar from video. In *Proceedings of AAAI Conference,*, 2002.

[24] N. Oliver, E. Horvitz, and A. Garg. Layered representations for human activity recognition. In *ICMI*, 2003.

[25] K. Pastra and Y. Aloimonos. The minimalist grammar of action. *Phil. Trans. R Soc. B*, 367(1585):103–117, 2012.

[26] M. Pawlik and N. Augsten. Rted: a robust algorithm for the tree edit distance. *Proc. VLDB Endow.*, 5(4):334–345, 2011.

[27] C. Pinhanez and A. Bobick. Human action detection using pnf propagation of temporal constraints. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 1998.

[28] D. A. Rosenbaum, R. G. Cohen, S. A. Jax, D. J. Weiss, and R. Van Der Wel. The problem of serial order in behavior: Lashleys legacy. *Human Movement Science*, 26(4):525–554, 2007.

[29] M. Ryoo and J. Aggarwal. Recognition of composite human activities through context-free grammar based representation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2006.

[30] P. Saisan, G. Doretto, Y. N. Wu, and S. Soatto. Dynamic texture recognition. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[31] C. Schuldt, I. Laptev, and B. Caputo. Recognizing human actions: A local svm approach. In *ICPR*, 2004.

[32] L. Sigal, A. O. Balan, and M. J. Black. Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion. *International Journal of Computer Vision*, 87(1-2):4–27, 2010.

[33] M. Sridhar, A. G. Cohn, and D. C. Hogg. Learning functional object-categories from a relational spatio-temporal representation. In *Proc. 18th European Conference on Artificial Intelligence,*, pages 606–610, 2008.

[34] P. Turaga, R. Chellappa, V. S. Subrahmanian, and O. Udrea. Machine recognition of human activities: A survey. *IEEE Transactions on Circuits and Systems for Video Technology*, 18(11):1473–1488, 2008.

[35] I. Vicente, V. Kyrki, and D. Kragic. Action recognition and understanding through motor primitives. *Advanced Robotics*, 21:1687–1707, 2007.

[36] J. Wang, D. Fleet, and A. Hertzmann. Gaussian process dynamical models for human motion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2008.

[37] A. Whiten, E. Flynn, K. Brown, and T. Lee. Imitation of hierarchical action structure by young children. *Developmental Science*, 9:574–582, 2006.

[38] G. Willems, T. Tuytelaars, and L. J. V. Gool. An efficient dense and scaleinvariantspatio-temporal interest point detector. In *Proc. European Conference on Computer Vision*, 2008.

[39] A. Yilmaz and M. Shah. Actions sketch: A novel action representation. In *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pages 984–989, 2005.