Jointly Learning Grounded Task Structures from Language Instruction and Visual Demonstration

Changsong Liu[1*], **Shaohua Yang**[1*], **Sari Saba-Sadiya**[1],
Nishant Shukla[2], **Yunzhong He**[2], **Song-Chun Zhu**[2], and **Joyce Y. Chai**[1]

[1]*Department of Computer Science and Engineering*
Michigan State University, East Lansing, MI 48824
[2]*Center for Vision, Cognition, Learning, and Autonomy*
University of California, Los Angeles, CA 90095
{cliu,yangshao,sadiyasa,jchai}@cse.msu.edu
{shukla,yunzhong}@cs.ucla.edu,sczhu@stat.ucla.edu

Abstract

To enable language-based communication and collaboration with cognitive robots, this paper presents an approach where an agent can learn task models jointly from language instruction and visual demonstration using an And-Or Graph (AoG) representation. The learned AoG captures a hierarchical task structure where linguistic labels (for language communication) are grounded to corresponding state changes from the physical environment (for perception and action). Our empirical results on a cloth-folding domain have shown that, although state detection through visual processing is full of uncertainties and error prone, by a tight integration with language the agent is able to learn an effective AoG for task representation. The learned AoG can be further applied to infer and interpret on-going actions from new visual demonstration using linguistic labels at different levels of granularity.

1 Introduction

Given tremendous advances in robotics, computer vision, and natural language processing, a new generation of cognitive robots have emerged that aim to collaborate with humans in joint tasks. To facilitate natural and efficient communication with these physical agents, natural language processing will need to go beyond traditional symbolic representations, but rather ground language to sensors (e.g., visual perception) and actuators (e.g., lower-level control systems) of physical agents. The internal task representation will need to capture both higher-level concepts (for language communication) and lower-level visual features (for perception and action).

To address this need, we have developed an approach on learning *procedural tasks* jointly from language instruction and visual demonstration. In particular, we use And-Or Graph (AoG), which has been used in many computer vision tasks and robotic applications (Zhao and Zhu, 2013; Li et al., 2016; Xiong et al., 2016), to represent a hierarchical task model that not only captures symbolic concepts (extracted from language instructions) but also the corresponding visual state changes from the physical environment (detected by computer vision algorithms).

Different from previous works that ground language to perception (Liu et al., 2012; Matuszek et al., 2012; Kollar et al., 2013; Yu and Siskind, 2013; Yang et al., 2016), a key innovation in our framework is that language is no longer grounded just to perceived objects in the environment, but is further grounded to a hierarchical structure of *state changes* where the states are perceived from the environment during visual demonstration. The state of environment is an important notion in robotic systems as the change of states drives planning for lower-level robotic actions. Thus, connecting language concepts to state changes, our learned AoG provides a unified representation that integrates language and vision to not only support language-based communication but also facilitate robot action planning and execution in the future.

More specifically, within this AoG framework, we have developed and evaluated our algorithms in the

* The first two authors contributed equally to this paper.

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 1482–1492,
Austin, Texas, November 1-5, 2016.

context of learning a *cloth-folding* task. Although cloth-folding appears simple and intuitive for humans, it represents significant challenges for both vision and robotics systems. Furthermore, although symbolic language processing in this domain is easy due to limited use of vocabulary, grounded language understanding is particularly challenging. A simple phrase (e.g., "fold in half") could have different grounded meanings (e.g., lower-level representation) given different contexts. Thus, this cloth-folding domain is a good starting point to focus on grounding language to task structures.

Our empirical results have shown that, although state detection from the physical world can be extremely noisy, our learning algorithm that tightly incorporates language is capable of acquiring an effective and meaningful task model to compensate the uncertainties in visual processing. Once the AoG for the task is learned, it can be applied by our inference algorithm, for example, to infer on-going actions from new visual demonstration and generate linguistic labels at different levels of granularity to facilitate human-agent communication.

2 Related Work

Recent years have seen an increasing amount of work on grounding language to visual perception (Liu et al., 2012; Matuszek et al., 2012; Yu and Siskind, 2013; Kollar et al., 2013; Naim et al., 2015; Yang et al., 2016; Gao et al., 2016). Furthermore, the robotics community made significant efforts to utilize novel grounding techniques to facilitate task execution given natural language instructions (Chen et al., 2010; Kollar et al., 2010; Tellex et al., 2011; Misra et al., 2014) and task learning from demonstration (Saunders et al., 2006; Chernova and Veloso, 2008).

Research on Learning from Demonstration (LfD) employed various approaches to model the tasks (Argall et al., 2009), such as state-to-action mapping (Chernova and Veloso, 2009), predicate calculus (Hofmann et al., 2016), and Hierarchical Task Networks (Nejati et al., 2006; Hogg et al., 2009). However, aspiring to enable human robot communication, the framework developed in this paper focuses on task representation using language grounded to a structure of state changes detected from the physical world. As demonstrated in recent work (She et al., 2014a; Misra et al., 2015; She and Chai, 2016), explicitly modeling change of states is an important step towards interacting with robots in the physical world.

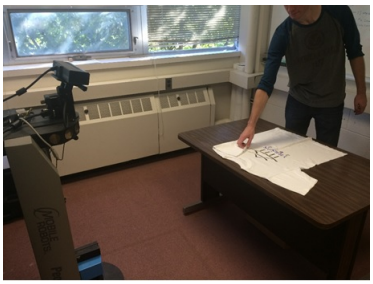

Figure 1: The setting of our situated task learning where a human teacher teaches the robot how to fold a T-shirt through both task demonstrations and language instructions.

Additionally, there has also been an increasing amount of work that learns new tasks either using methods like supervised learning on large corpus of data (Branavan et al., 2010; Branavan et al., 2012; Tellex et al., 2014; Misra et al., 2014), or by learning from humans through dialogue (Cantrell et al., 2012; Mohan et al., 2013; Kirk and Laird, 2013; She et al., 2014b; Mohseni-Kabir et al., 2015). In this paper, we focus on jointly learning new tasks through visual demonstration and language instruction. The learned task model is explicitly represented by an AoG, a hierarchical structure consisting of both linguistic labels and corresponding changes of states from the physical world. This rich task model will facilitate not only language-based communication, but also lower-level action planning and execution.

3 Task and Data

In this paper, we use cloth-folding (e.g., teaching a robot how to fold a T-shirt) as the task to demonstrate and evaluate our joint task learning approach. As mentioned earlier, cloth-folding, although simple for humans, represents a challenging task for the robotics community due to the complex state and action space.

Figure 1 illustrates the setting for our situated task learning. A human teacher can teach a robot how to fold a T-shirt through simultaneous verbal instructions and visual demonstrations. A Microsoft Kinect2 camera is mounted on the robot to record

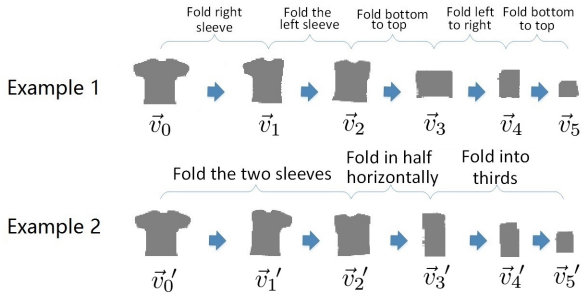

Figure 2: Examples of our parallel data where language instructions are paired with a sequence of visual states detected from the video.

the human's visual demonstration, and the human's corresponding verbal instructions are recorded by Kinect2's embedded microphone array.

A recorded video of task demonstration and its corresponding verbal instruction become one training example for our task learning system. Figure 2 shows two examples of such "parallel data". The visual demonstration is processed into a sequence of visual states, where each state is a numeric vector ($\vec{v}_i$) capturing the visual features of the T-shirt at a particular time (see later Section 5.1 for details). The recorded verbal instructions are then aligned with the sequence of visual states based on the timing information.

During teaching the task, we specifically requested the demonstrator to describe and do each step at roughly the same time. This greatly simplified the alignment problem. Since our ultimate goal is to enable humans to teach the robot through natural language dialogue and demonstration, our hypothesis is that the alignment issue can be alleviated by certain dialogue mechanism (e.g., ask to repeat the action, ask for step-by-step aligned instructions, etc.). As it is human's best interest that the robot gets the clearest instructions, we also anticipate during dialogue human teachers will be collaborative and provide mostly aligned instructions. Certainly, these hypotheses will need to be validated in the dialogue setting in our future work.

In our collected data, each *change of state*, i.e., a transition between two visual states, is caused by one or more physical actions. Some language descriptions align with only a single-step change of state. For instance, "*fold right sleeve*" is aligned with the change ($\vec{v}_0 \rightarrow \vec{v}_1$) and "*fold left sleeve*" is aligned with ($\vec{v}_1 \rightarrow \vec{v}_2$) in Example 1. This kind of single-step change of state is considered as a **primitive action**. Other language descriptions are aligned with a sequence of multiple state changes. For instance, "*fold the two sleeves*" in Example 2 is aligned with two consecutive changes: ($\vec{v}_0', \vec{v}_1', \vec{v}_2'$). This kind of sequence of state changes is considered as a **complex action**, which can be decomposed into partially ordered primitive actions. A complex action can also be concisely represented by the change from the initial state to the end state in the sequence, such as ($\vec{v}_0' \rightarrow \vec{v}_2'$) in Example 2.

These parallel data are used to train and test our learning and inference algorithms presented later.

4 And-Or Graph Representation

We use AoG as the formal model of a procedural task. Figure 3 shows an example AoG for the cloth-folding task. It is a hierarchical structure that explicitly captures the compositionality and reconfigurability of a procedural task. The terminal nodes capture state changes associated with primitive actions of this task, and non-terminal nodes capture state changes associated with complex actions which are further composed by lower-level actions.

In addition to state changes, the learned AoG is also labeled with linguistic information (e.g., verb frames) capturing the causes of the corresponding state changes. The state changes are also considered as grounded meanings of these verb frames. For example, Figure 3 shows two "*fold the t-shirt*" labels at the top layer. Note that although symbolically, these two phrases have the same meaning (e.g., same verb frames), their grounded meanings are different as they correspond to different changes of state. Being able to represent differences or ambiguities in grounded meanings is crucial to connect language to perception and action.

Formally, an AoG is defined as a 5-tuple $\mathcal{G} = (S, \Sigma, N, R, \Theta)$, where

- S is a root node (or a start symbol) representing a complete task.
- Σ is a set of terminal nodes, each of which represents a change of state associated with a primitive action.
- $N = N^{AND} \cup N^{OR}$ is a set of non-terminal nodes, which is divided into two disjoint sub-

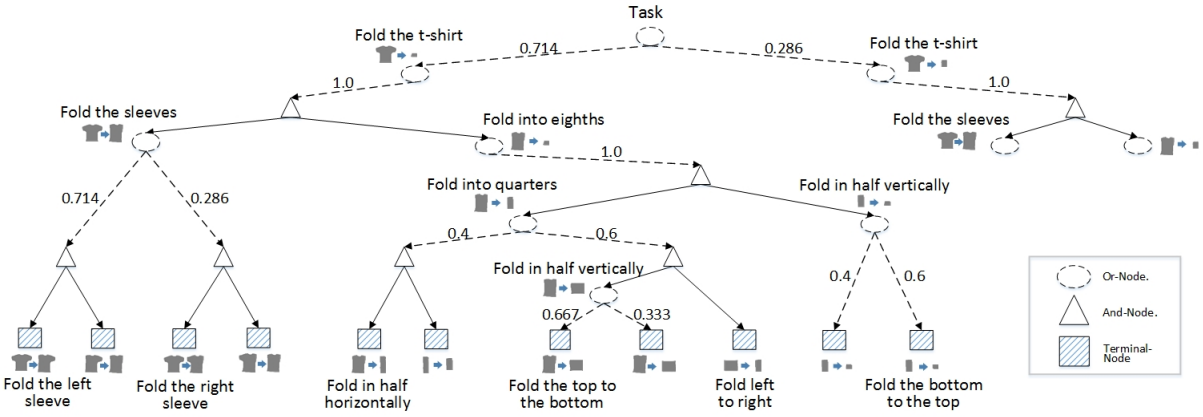

Figure 3: An example of the learned AoG

sets of And-nodes and Or-nodes.

- R is a "child-parent" relation (many-to-one mapping), i.e., $R(n^{ch}) = n^{pa}$ (meaning n^{pa} is the parent node of n^{ch}), where $n^{ch} \in \Sigma \cup N$ and $n^{pa} \in N \cup \{S\}$.
- Θ is a set of conditional probabilities $p(n^{ch}|n^{pa})$, where $n^{pa} \in N^{OR}$, $n^{ch} \in \{n \mid R(n) = n^{pa}\}$. Namely, for each Or-node, Θ defines a probability distribution over the set of all its children nodes.

In essence, our AoG model is equivalent to Probabilistic Context-Free Grammar (PCFG). An AoG can be converted into a PCFG:

- Each And-node and its children form a production rule

$$n^{AND} \rightarrow n_1^{ch} \wedge n_2^{ch} \wedge \dots$$

that represents the decomposition of a complex action into sequentially ordered sub-actions.

- Each Or-node and its children form a production rule

$$n^{OR} \rightarrow n_1^{ch} \mid n_2^{ch} \mid \dots$$

that represents all the alternative ways of accomplishing an action. Each alternative also comes with a probability as specified in Θ.

5 Method

5.1 Vision and Language Processing

The input data to our AoG learning algorithm consist of co-occurring visual demonstrations and language instructions as described in Section 3. Based on the RGB-D information provided by the Kinect2 sensor, we developed a vision processing system to keep track of human's actions and statuses of the T-shirt object.

To learn a meaningful task structure, the most important visual information are those key statuses that the object goes through. Therefore, our vision system processes each visual demonstration into a sequence of *states*. Each state $\vec{v}$ is a multi-dimensional numeric vector that encodes the geometric information of the detected T-shirt, such as its smallest bounding rectangle and largest inscribed contour-fitting rectangle. These key states are detected by tracking the human's folding actions. Namely, whenever a folding action is detected[1], we append the new state caused by the action to the sequence of observed states, till the end of the demonstration.

The verbal instructions given by the demonstrators were mainly verb phrases such as "fold *which-part*", "fold to *which-position*", or "fold *in-what-manner*". A semantic parser[2] is applied to parse each instruction text into a canonical verb-frame representation, such as

$$FOLD: \quad [PART: left\ sleeve] \\ [POSITION: middle].$$

Through the vision and language processing, each task demonstration becomes two parallel sequences, i.e., a sequence of extracted visual states and a sequence of parsed language instructions. The align-

[1]The vision system keeps track of human's hands, and detects a folding action as a gripping action followed by moving and releasing the hand(s).

[2]We use the CMU's Phoenix parser: http://wiki.speech.cs.cmu.edu/olympus/index.php/Phoenix

ment between these two sequences is also extracted from their co-occurrence timing information. Thus, an instance of a task demonstration is formally represented as a 3-tuple $x = (D, L, \Delta)$, where $D = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_M\}$ is the sequence of visual states, $L = \{l_1, l_2, \ldots, l_K\}$ is the sequence of linguistic verb-frames, and $\Delta(k) = (i, j)$ is an "alignment function" specifying the correspondence between a linguistic verb-frame l_k and a single or a sub-sequence of visual state(s) $\{\vec{v}_i, \ldots, \vec{v}_j\}$ $(i \leq j)$.

Then, given a dataset $\mathcal{X}$ of such task demonstrations, our AoG learning algorithm learns an AoG $\mathcal{G}$ as defined in Section 4. The next section describes our learning algorithm in detail.

5.2 AoG Learning Algorithm

Learning an AoG $\mathcal{G} = (S, \Sigma, N, R, \Theta)$ is carried out in two stages. Firstly, we learn a set of terminal nodes Σ to represent the primitive actions (i.e., the actions that can be preformed in a single step). This is done through clustering the observed visual states. Secondly, the hierarchical structure (i.e., N and R) and parameters Θ of the AoG is learned using an iterative grammar induction algorithm.

5.2.1 Learning Terminal Nodes

A terminal node in the AoG represents a primitive action, which causes the object to directly change from one state to another. Thus we represent a terminal node as a 2-tuple of states (or a "change of state"). Since the visual states detected by computer vision are numeric vectors with continuous values, we first apply a clustering algorithm to form a finite set of discrete state representations. Each cluster then represents a unique situation that one can encounter in a task. Since when learning a new task we usually do not know how many unique situations exist, here we employ a greedy clustering algorithm (Ng and Cardie, 2002), which does not assume a fixed number of clusters.

As the greedy clustering algorithm relies on the pairwise similarities of all the visual states, we also train an SVM classifier on a separate dataset of 22 T-shirt folding videos and use its classification output to measure the similarity between two visual states. The SVM classifier takes two numeric vectors as an input, and predicts whether these two vectors represent the same status of a T-shirt. We then apply this SVM classifier on each pair of detected visual states in our new dataset (i.e., the dataset for learning the AoG), and use the SVM's output class label (1 or -1) multiplies its classification confidence score as the similarity measurement between two visual states.

After clustering all the observed visual states in the data, we then replace each numeric vector state representation with the cluster "ID" it belongs to. Thus each visual demonstration now becomes a sequence of symbolic values, denoted as $D' = \{s_1, s_2, \ldots, s_M\}$. And we further transform it into an equivalent *change of state* sequence $C = \{(s_1, s_2), (s_2, s_3), \ldots, (s_{M-1}, s_M)\}$, in which each change of state essentially represents a primitive action in this task. These change of state pairs then form the set of terminal nodes Σ.

5.2.2 Learning the Structure and Parameters

With the sequences of numeric vector states replaced by the "symbolic" change of state sequences in the first stage, we can further learn the structure and parameters of an AoG. Namely, to learn N, R, and Θ that maximize the posterior probability:

$$\begin{aligned} & \arg\max_{N,R,\Theta} P(N, R, \Theta | \mathcal{X}, \Sigma) \\ = & \arg\max_{N,R,\Theta} P(N, R | \mathcal{X}, \Sigma)\, P(\Theta | \mathcal{X}, \Sigma, N, R). \end{aligned}$$

Following the iterative grammar induction paradigm (Tu et al., 2013; Xiong et al., 2016), we employ an iterative procedure that alternatively solves $\arg\max_{N,R} P(N, R | \mathcal{X}, \Sigma)$ and $\arg\max_{\Theta} P(\Theta | \mathcal{X}, \Sigma, N, R)$.

To solve the first term, we use greedy or beam search with a heuristic function similar to (Solan et al., 2005). To solve the second term, we estimate the probability of each branch of an Or-node by computing the frequency of that branch, which is essentially a maximum likelihood estimation similar to (Pei et al., 2013).

In detail, the learning procedure first initializes empty N, R, and Θ, then iterates through the following two steps until no further update can be made.

Step (1): search for new And-nodes.

This step searches for new And-node candidates from $\Sigma \cup N$, and update N and R with the top-ranked candidates. Specifically, we denote an And-node candidate to be searched as $A = (s_l \rightarrow s_m \rightarrow s_r)$.

Here s_l is the initial state of an existing node, whose end state is s_m. And s_r is the end state of another existing node, whose initial state is s_m. Thus an And-node candidate always has two child nodes, and represents a pattern of sub-sequences which starts from state s_l, ends at s_r, and has s_m occurred somewhere in the middle.

Using the above notation, the heuristic function for ranking And-node candidates is defined as

$$h(A) = (1 - \lambda)P_{state}(A) + \lambda P_{label}(A)$$

where $P_{state}(A)$ captures the prevalence of a particular And-node candidate based on the observed state change sequences:

$$P_{state}(A) = \frac{P_R(A) + P_L(A)}{2}$$

and $P_R(A)$ is the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears and the number of times $(s_l \rightarrow s_m)$ appears, and $P_L(A)$ is the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears and the number of times $(s_m \rightarrow s_r)$ appears.

The component $P_{label}(A)$ captures the prevalence of linguistic labels associated with the sequential state change patterns. It is computed as the ratio between the number of times $(s_l \rightarrow s_m \rightarrow s_r)$ co-occurs with a linguistic instruction[3] and the total number of times $(s_l \rightarrow s_m \rightarrow s_r)$ appears.

We specially define two AoG learning settings based on the role that language plays:

- *Tight* language integration: incorporate heuristics on linguistic labels (i.e., $\lambda = 0.5$). In this setting, the learned AoG prefers And-nodes that not only happen frequently, but also can be described by a linguistic label.
- *Loose* language integration: without incorporating the heuristics on linguistic labels ($\lambda = 0$). Each And-node is learned only based on the frequency of its state change pattern. The learned node can still acquire a linguistic label if there happen to be a co-occurring one, but the chance is lower than the "tight" setting.

Step (2): search for new Or-nodes or new branches of existing Or-nodes, then update Θ.

Once new And-nodes are added by the previous step, the next step is to search for Or-nodes that can be created or updated. An Or-node in the AoG essentially represents the set of all And-nodes that share the same initial and end states, denoted as $(s_l \rightarrow s_r)$ here (s_l and s_r are the common initial and end states, respectively). Suppose $(s_l \rightarrow s'_m \rightarrow s_r)$ is a newly added And-node, it is then assigned as a child of the Or-node $(s_l \rightarrow s_r)$. To further update Θ, the branching probability is computes as the ratio between the number of times $(s_l \rightarrow s'_m \rightarrow s_r)$ appears and the number of times $(s_l \rightarrow s_r)$ appears.

5.3 Inference Using AoG

Once a task AoG is learned, it can be applied to interpret and explain new visual demonstrations using linguistic labels. Due to the noises and uncertainties from computer vision processing, one key challenge in interpreting the visual demonstration is to reliably identify the different states of the T-shirt.

To tackle this issue, we formulate a joint inference problem. Namely, given a task demonstration video, we first process it into a sequence of numeric vector states $D = \{\vec{v}_1, \vec{v}_2, \ldots, \vec{v}_M\}$ as described in Section 5.1. Then the goal of inference is to find the most-likely parse tree T and a sequence of "symbolic states" $D' = \{s_1, s_2, \ldots, s_M\}$ based on the AoG $\mathcal{G}$ and the input D:

$$(T^*, D'^*) = \arg\max_{T, D'} P(T, D' \mid \mathcal{G}, D)$$

We apply a chart parsing algorithm (Klein and Manning, 2001) to efficiently solve this problem. Furthermore, to accommodate the ambiguities in mapping a numeric vector state $\vec{v}_m$ to a symbolic state, we take into consideration the top-k hypotheses measured by the similarity between $\vec{v}_m$ and a symbolic state s_k.[4] For each state mapping hypothesis, we add a completed edge between indices m and $m + 1$ in the chart, with s_k as its symbol and a probability p based on the similarity between $\vec{v}_m$ and s_k. Based on the given AoG, the chart parsing algorithm then uses Dynamic Programming to search the best parse tree that maximizes the joint probability of $P(T, D'|\mathcal{G}, D)$.

Figure 4 illustrates the input and output of our inference algorithm. As illustrated by this example,

[3] Such information is encoded in the Δ function as mentioned in Section 5.1.

[4] A symbolic state is represented by a cluster of numeric vector states learned from the training data.

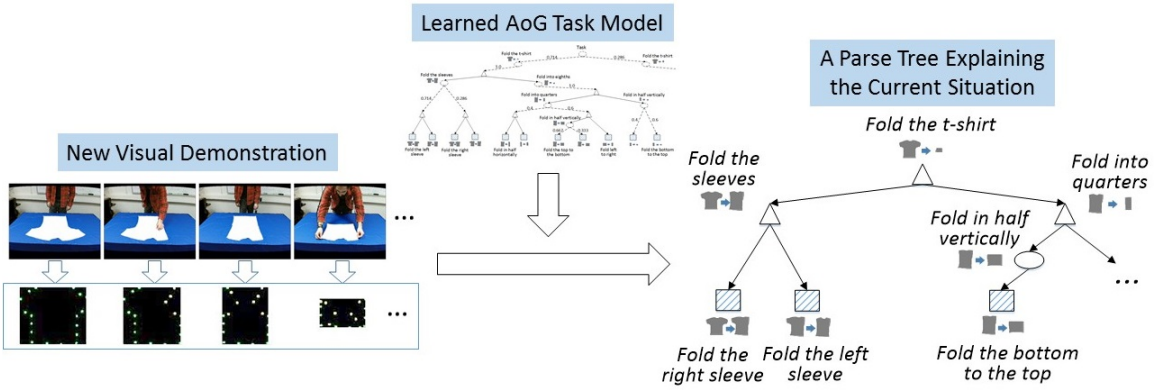

Figure 4: An illustration of the input and output of our AoG-based inference algorithm.

the parse tree represents a hierarchical structure underlying the observed task procedure, and the linguistic labels associated with the nodes can be used to describe the primitive and complex actions involved in the procedure.

6 Evaluation

Using the setting as described in Section 3, we collected 45 T-shirt folding demonstrations from 6 people to evaluate our AoG learning and inference methods. More specifically, we conducted a 5-fold cross validation. In each fold, 36 demonstrations were used for training to learn a task AoG. Then the remaining 9 demonstrations were used for testing, in which the learned AoG is further applied to process each of the testing visual demonstrations.

Motivated by earlier work on plan/activity recognition using CFG-based models (Carberry, 1990; Pynadath and Wellman, 2000), we use an extrinsic task that automatically assigns linguistic labels to new demonstrations to evaluate the quality of the learned AoG and the effectiveness of the inference algorithm. This involves three steps: (1) parse the video using the learned AoG; (2) identify linguistic labels associated with terminal or nonterminal nodes in the parse tree; and (3) compare the identified linguistic labels with the manually annotated labels.

We conduct the evaluation at two levels:

- **Primitive actions:** use linguistic labels associated with terminal nodes to describe the primitive actions in each video. This level provides detailed descriptions on how the observed task procedure is performed step-by-step.
- **Complex actions:** use linguistic labels associated with nonterminal nodes to describe complex actions. This provides a high-level "summary" of the detailed low-level actions.

The capability to recognize fine-grained primitive actions as well as high-level complex actions in a task procedure and to communicate those in language is important for many real-world AI applications such as human-robot collaboration (Mohseni-Kabir et al., 2015) and visual question answering (Tu et al., 2014).

6.1 Primitive Actions

We first compare the performance of interpreting primitive actions using the learned AoG with a baseline. The baseline applies a memory-based (or similarity-base) approach. Given a testing video, it extracts all the different visual states and maps each state to the nearest cluster learned from the training data (see Section 5.2.1). It then pairs each two consecutive states as a change of state instance, and uses the linguistic label corresponding to the identical change of state found in the training data as the label of a primitive action.

We measure the primitive action recognition performance in terms of the *normalized Minimal Edit Distance (MED)*. Namely, for each testing demonstration we calculate the MED between the ground-truth sequence of primitive action labels and the automatically generated sequence of labels, and divide the MED value by the length of the ground-truth sequence to produce a normalized score (a smaller score indicates better performance in recognizing the primitive actions).

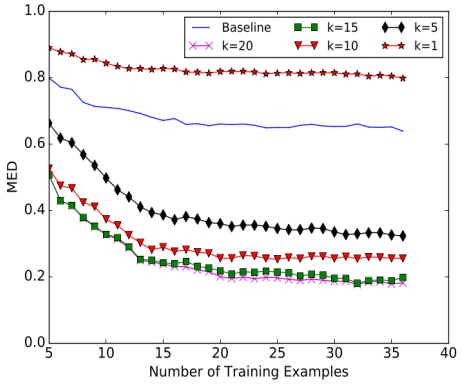

Figure 5: Performance of interpreting primitive actions. Different number of state mapping hypotheses (k) are used in the inference algorithm. The x-axis is the number of training examples used for learning the AoG.

The performances of the baseline and our AoG-based approach are shown in Figure 5. For the AoG-based approach, Figure 5 also shows the performances of incorporating different number of state mapping hypotheses (i.e., $k = 1, 5, 10, 15, 20$) into the inference algorithm (Section 5.3). Here we only report the performance of using AoG learned with the *tight* language integration (see Section 5.2.2), since there is no difference in performance between the tight and loose language integration settings in recognizing primitive actions[5].

As Figure 5 shows, the baseline performance is rather weak (i.e., high MED scores). This is largely due to the noise in state clustering and mapping from vision. After manually inspecting the collected demonstration videos, we found 18 unique statuses associated with folding a T-shirt. However the computer vision based clustering on average produces more than 30 clusters when all the 36 training examples are used. This makes it difficult to directly match the state changes as in the baseline. For our AoG-based method, when the inference algorithm only takes the single best state mapping hypothesis into consideration (i.e., $k = 1$), it yields a very weak performance because the observed state change sequence often cannot be parsed using the learned AoG.

However, the performance of the AoG-based method is significantly improved when multiple state mapping hypotheses are incorporated into the inference process. When the top-5 ($k = 5$) state mapping hypotheses are incorporated into the AoG-based inference, its MED score has already outperformed the baseline by a 0.3 gap ($p < 0.001$ using the Wilcoxon signed-rank test). When $k = 20$, the MED score has dropped by more than 0.6 compared to $k = 1$ ($p < 0.001$).

These results indicate that our AoG-based method is capable of learning useful task structure from small data. When multiple hypotheses of visual state mapping are incorporated, the learned AoG can compensate the uncertainties in vision processing and identify highly reliable primitive actions from unseen demonstrations.

6.2 Complex Actions

We further evaluate the performance of interpreting complex actions using the learned AoG. The baseline for comparison is similar to the one used in the previous section. It first converts a test video into a sequence of "symbolic" states by mapping each detected visual state to its nearest cluster. It then enumerates all the possible segments that consist of more than two consecutive states and search for the identical segments in the training data. If a matching segment is found, then the corresponding linguistic label (if any) is used as the label for a complex action. Since complex actions correspond to nonterminal nodes in the parse tree generated by AoG-based inference, and some of them may have linguistic labels while others may not. We use *precision*, *recall*, and *F-score* to measure how well the generated linguistic labels match the manually segmented and annotated complex actions in testing videos.

Figure 6 shows the F-scores of recognizing complex actions using the AoG learned from the *loose* and the *tight* language integration, respectively. In this figure, results are based on $k = 20$ state mapping hypotheses incorporated into the inference algorithm. As shown here, performances from both settings are significantly better than the baseline ($p < 0.001$). The AoG learned based on the tight integration with language yields significantly better performance than the loose integration (over 0.2 gain on F-score, $p < 0.001$).

This result indicates that the tight integration of

[5]Because the linguistic labels generated for primitive actions are all from terminal nodes, and the two different AoG learning settings only affect nonterminal nodes.

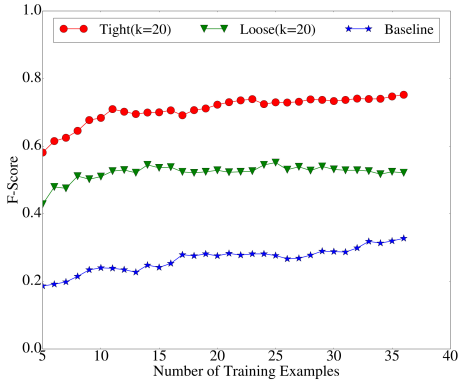

Figure 6: Performances (F-score) of recognizing complex actions. The lowest curve shows the performance from the baseline. Two other curves represent the performance using the AoG learned from the *loose* integration and the *tight* integration with language respectively (where $k = 20$ is used in inference).

language during AoG learning favors And-node patterns that are more likely to be described by natural language (or more consistent with human conceptualization of the task structure)[6]. Such an AoG representation can lead to recognition of video segments that can be better explained or summarized by human language. This capability of learning explicit and language-oriented task representations is important to link language and vision for enabling situated human-agent communication/collaboration.

Table 6.2 further shows the results from different numbers of state mapping hypotheses that are incorporated into the inference algorithm. As shown here, the trend of performance improvement with the increase in k is again observed. When multiple state mapping hypotheses are incorporated in inference, the learned AoG is capable of compensating uncertainties in vision processing and producing better parses for unseen visual demonstrations.

7 Conclusion and Future Work

This paper presents an approach on task learning where an agent can learn a grounded task model from human demonstrations and language instructions. A key innovation of this work is grounding language to a perceived structure of state changes based on AoG representation. Once the task model is acquired, it can be used as a basis to support collaboration and communication between humans and agents/robots. Using cloth-folding as an example, our empirical results have demonstrated that tightly integrating language with vision can effectively produce task structures in AoG that can generalize well to new demonstrations.

Although we have only made an initial attempt on a small task, our approach can be naturally extended to more complex tasks such like assembling and cooking. Both the AoG representation and the task learning approach are general and applicable to different domains. What needs to be adapted is the representation of the visual states and computer vision algorithms to detect these states for a specific task.

Grounding language to a structure of perceived state changes will provide an important stepping stone towards integrating language, perception, and action for human-robot communication and collaboration. Currently, our algorithms learn the task structures based on offline parallel data. Our future work will explore incremental learning through human-agent dialogue to acquire grounded task structures.

Table 1: Performance of recognizing complex actions using the AoG learned from the loose and tight integration of language as described in Section 5.2. Different (k) number of state mapping hypotheses are used in the inference algorithm.

		k=1	k=5	k=10	k=15	k=20
Precision	Loose	0.34	0.76	0.79	0.84	0.84
	Tight	0.34	0.8	0.86	0.89	0.9
Recall	Loose	0.12	0.33	0.35	0.38	0.38
	Tight	0.12	0.51	0.59	0.64	0.65
F-Score	Loose	0.17	0.46	0.49	0.52	0.52
	Tight	0.18	0.63	0.70	0.74	0.75

Acknowledgments

The authors are grateful to Sarah Fillwock and James Finch for their help on data collection and processing, to Mun Wai Lee for his helpful discussions, and to anonymous reviewers for their valuable comments and suggestions. This work was supported in part by N66001-15-C-4035 from the DARPA SIMPLEX program, and IIS-1208390 and IIS-1617682 from the National Science Foundation.

[6]By further investigating the learned AoG under the two different settings, we found that the nonterminal nodes learned from the tight language integration setting is more likely to acquire a linguistic label (33%) than the nonterminal nodes learned from the loose setting (18%).

References

Brenna D. Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. 2009. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483.

S. R. K. Branavan, Luke S. Zettlemoyer, and Regina Barzilay. 2010. Reading between the lines: Learning to map high-level instructions to commands. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1268–1277.

S. R. K. Branavan, Nate Kushman, Tao Lei, and Regina Barzilay. 2012. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 126–135.

Rehj Cantrell, J. Benton, Kartik Talamadupula, Subbarao Kambhampati, Paul Schermerhorn, and Matthias Scheutz. 2012. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *7th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 471–478.

Sandra Carberry. 1990. *Plan recognition in natural language dialogue*. MIT press.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *Journal of Artificial Intelligence Research*, 37(1):397–436.

Sonia Chernova and Manuela Veloso. 2008. Teaching multi-robot coordination using demonstration of communication and state sharing. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems-Volume 3*, pages 1183–1186.

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

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Y. Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*, Volume 1: Long Papers, pages 1814–1824.

Till Hofmann, Tim Niemueller, Jens Claßen, and Gerhard Lakemeyer. 2016. Continual planning in golog. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Chad Hogg, Ugur Kuter, and Héctor Muñoz-Avila. 2009. Learning hierarchical task networks for nondeterministic planning domains. In *Proceedings of the Twenty-First International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1708–1714.

James R. Kirk and John E. Laird. 2013. Learning task formulations through situated interactive instruction. In *Proceedings of the Second Annual Conference on Advances in Cognitive Systems (ACS)*, volume 219, page 236.

Dan Klein and Christopher D. Manning. 2001. An o(n^3) agenda-based chart parser for arbitrary probabilistic context-free grammars. *Stanford Technical Report*.

Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. 2010. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction (HRI)*, pages 259–266.

Thomas Kollar, Jayant Krishnamurthy, and Grant P. Strimel. 2013. Toward interactive grounded language acqusition. In *Robotics: Science and Systems*.

Bo Li, Tianfu Wu, Caiming Xiong, and Song-Chun Zhu. 2016. Recognizing car fluents from video. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Changsong Liu, Rui Fang, and Joyce Y. Chai. 2012. Towards mediating shared perceptual basis in situated dialogue. In *Proceedings of the 13th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 140–149.

Cynthia Matuszek, Nicholas FitzGerald, Luke Zettlemoyer, Liefeng Bo, and Dieter Fox. 2012. A joint model of language and perception for grounded attribute learning. *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Dipendra K. Misra, Jaeyong Sung, Kevin Lee, and Ashutosh Saxena. 2014. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. In *Proceedings of Robotics: Science and Systems (RSS)*.

Dipendra K. Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics*, pages 992–1002.

Shiwali Mohan, James Kirk, and John Laird. 2013. A computational model for situated task learning with interactive instruction. In *Proceedings of the 12th International Conference on Cognitive Modeling (ICCM)*.

Anahita Mohseni-Kabir, Charles Rich, Sonia Chernova, Candace L Sidner, and Daniel Miller. 2015. Interactive hierarchical task learning from a single demonstration. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 205–212.

Iftekhar Naim, Young Chol Song, Qiguang Liu, Liang Huang, Henry Kautz, Jiebo Luo, and Daniel Gildea. 2015. Discriminative unsupervised alignment of natural language instructions with corresponding video

segments. In *North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT)*.

Negin Nejati, Pat Langley, and Tolga Konik. 2006. Learning hierarchical task networks by observation. In *Proceedings of the 23rd international conference on Machine learning (ICML)*, pages 665–672.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics (ACL)*, pages 104–111.

Mingtao Pei, Zhangzhang Si, Benjamin Z Yao, and Song-Chun Zhu. 2013. Learning and parsing video events with goal and intent prediction. *Computer Vision and Image Understanding*, 117(10):1369–1383.

David V. Pynadath and Michael P. Wellman. 2000. Probabilistic state-dependent grammars for plan recognition. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 507–514. Morgan Kaufmann Publishers Inc.

Joe Saunders, Chrystopher L. Nehaniv, and Kerstin Dautenhahn. 2006. Teaching robots by moulding behavior and scaffolding the environment. In *Proceedings of the 1st ACM SIGCHI/SIGART conference on Human-robot interaction (HRI)*, pages 118–125.

Lanbo She and Joyce Y. Chai. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Lanbo She, Yu Cheng, Joyce Y. Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014a. Teaching robots new actions through natural language instructions. In *Proceedings of the 23rd IEEE International Symposium on Robot and Human Interactive Communication*, pages 868–873.

Lanbo She, Shaohua Yang, Yu Cheng, Yunyi Jia, Joyce Y. Chai, and Ning Xi. 2014b. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue*.

Zach Solan, David Horn, Eytan Ruppin, and Shimon Edelman. 2005. Unsupervised learning of natural languages. *Proceedings of the National Academy of Sciences of the United States of America*, 102(33):11629–11634.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew R. Walter, Ashis Gopal Banerjee, Seth J. Teller, and Nicholas Roy. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Stefanie Tellex, Pratiksha Thaker, Joshua Joseph, and Nicholas Roy. 2014. Learning perceptually grounded word meanings from unaligned parallel data. *Machine Learning*, 94(2):151–167.

Kewei Tu, Maria Pavlovskaia, and Song-Chun Zhu. 2013. Unsupervised structure learning of stochastic and-or grammars. In *Advances in Neural Information Processing Systems (NIPS)*, pages 1322–1330.

Kewei Tu, Meng Meng, Mun Wai Lee, Tae Eun Choe, and Song-Chun Zhu. 2014. Joint video and text parsing for understanding events and answering queries. *IEEE MultiMedia*, 21(2):42–70.

Caiming Xiong, Nishant Shukla, Wenlong Xiong, and Song-Chun Zhu. 2016. Robot learning with a spatial, temporal, and causal and-or graph. In *Proceedings of the 2016 IEEE International Conference on Robotics and Automation (ICRA)*.

Shaohua Yang, Qiaozi Gao, Changsong Liu, Caiming Xiong, Song-Chun Zhu, and Joyce Y. Chai. 2016. Grounded semantic role labeling. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 149–159.

Haonan Yu and Jeffrey M. Siskind. 2013. Grounded language learning from video described with sentences. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 53–63.

Yibiao Zhao and Song-Chun Zhu. 2013. Scene parsing by integrating function, geometry and appearance models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3119–3126.