The 23rd IEEE International Symposium on Robot and
Human Interactive Communication
August 25-29, 2014. Edinburgh, Scotland, UK,

Teaching Robots New Actions through Natural Language Instructions

Lanbo She[1], Yu Cheng[2], Joyce Y. Chai[1], Yunyi Jia[2], Shaohua Yang[1], and Ning Xi[2]

Abstract— **Robots often have limited knowledge and need to continuously acquire new knowledge and skills in order to collaborate with its human partners. To address this issue, this paper describes an approach which allows human partners to teach a robot (i.e., a robotic arm) new high-level actions through natural language instructions. In particular, built upon the traditional planning framework, we propose a representation of high-level actions that only consists of the desired goal states rather than step-by-step operations (although these operations may be specified by the human in their instructions). Our empirical results have shown that, given this representation, the robot can reply on automated planning and immediately apply the newly learned action knowledge to perform actions under novel situations.**

I. INTRODUCTION

A new generation of robots have emerged in recent years which serve as humans' assistants and companions [9]. However, due to significantly mismatched capabilities (e.g., perception and actions) of humans and robots, natural language based communication between them will be difficult. First, the robot's representation of its perceptions and actions are continuous and numerical in nature. But human language is discrete and symbolic. For the robot to truly understand human language and thus take corresponding actions, it needs to first ground the meanings of human language to its own sensorimotor representation of its perception and action[13]. Second, the robot may not have complete knowledge about the shared environment and the joint task. It may not be able to connect human language to its own representations given its limited knowledge. Thus it is important for the robot to continuously acquire new knowledge through interaction with humans and the environment.

To address these challenges, we have been developing techniques to bridge the gaps of perception and actions to support situated human-robot dialogue [7]. While most of our previous work have focused on the perceptual differences [16][15], in this paper we investigate the gap of action. In particular, we are developing techniques that allow humans to teach new high-level actions to the robot through natural language instructions. Different from previous work on learning through demonstrations [21], here we rely on step-by-step natural language instructions to teach robots new actions. For example, suppose the robot does not understand the action *grab* when asked to perform "grab the blue block". The human can teach the robot by step-by-step instructions for the robot to follow, for example, the human may specify "open gripper, move to the blue block, and close gripper". At the end of the teaching phase, the robot should capture what the *grab* action actually entails and more importantly should be able to apply this action under different situations.

One important issue in action learning through natural language instructions is the representation of the acquired action in the robot's knowledge base so that it can be effectively applied in novel situations. To address this issue, we developed a framework for action representation that consists of primitive actions and high-level actions. Motivated by the planning literature [11], our primitive actions capture both pre-conditions and effects which are directly linked to lower-level control functions. Our high-level actions (e.g., the action of *grab*, *stack*, etc.) are modeled by the desired goal states of the environment as a result of these actions. The goal states can be automatically acquired after following step-by-step instructions by the human. Instead of directly modeling a high-level action as a sequence of actions/steps specified by the human, capturing the desired goal states provides much flexibility to address novel situations as demonstrated in our empirical evaluations.

In the following sections, we first give an overview of the teaching/learning framework, then describe our action representation and the mechanism to facilitate action learning, and finally present empirical results of applying learned actions in novel situations.

II. RELATED WORK

Learning to follow instructions has received an increasing attention in recent years. Previous work has applied supervised learning [12][20][8] or reinforcement learning [2][3] based on large corpus of data. However, one significant challenge is that when the robot is deployed in the real world to work with people, it is not feasible to provide a large amount of training data relevant for the current situation. It is also not ideal to engage the robot in a lengthy exploration process given time constraints. What's available is the human partner co-present with the robot. Thus it is desirable for the robot to directly learn from its human partner. To address this issue, the robotics community especially in human robot interaction, has started investigation on learning by demonstration [4][21].

Since language provides the most natural means to communicate, recent work has also explored the use of natural language and dialogue to teach agents actions. For example, [1] applied natural language dialogue technology to

[1]Lanbo She, Joyce Y. Chai, and Shaohua Yang are with the Department of Computer Science and Engineering at Michigan State University, East Lansing, MI 48824, USA; {shelanbo, jchai, yangshao}@cse.msu.edu

[2]Yu Cheng, Yunyi Jia and Ning Xi are with the Department of Electrical and Computer Engineering at Michigan State University, East Lansing, MI 48824, USA; {chengyu9, jiayunyi, xin}@egr.msu.edu

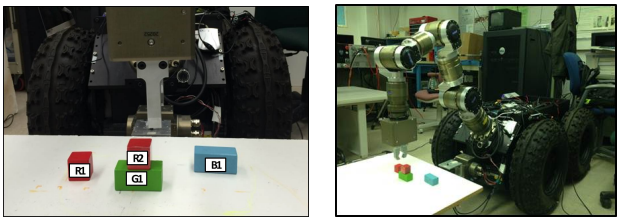
teach an artificial agent (web-based agent) actions (e.g., order books, find restaurant, etc.). However this previous work only needs to ground language to interface widgets (i.e., symbolic representation) without considering lower-level continuous actions as in robots. Related to human-robot interaction, natural language has been applied for the robot to learn new actions [5][6][18], parsing strategies[17], learning symbols[14]. Different from these earlier works, this paper focuses on action learning that connects high-level language with lower-level robotic actions using a three-tier knowledge representation and planning.

III. System Overview

We use SCHUNK arm manipulator [23] in our investigation. Our goal is to teach the manipulator high-level actions through natural language instructions. Figure 1(a) shows an example setup where several blocks are placed on a table which can be manipulated by the robotic arm. Figure 2 shows the overall system architecture.

Suppose the human says "stack the blue block on the red block to your right". This utterance will first go through a `semantic processor` and key semantic information from the utterance will be extracted and represented. Besides interpreting language input, the robot continuously perceives the shared environment using its camera and represents the perceived world as a `vision graph`. The robot also assesses its own internal state such as the status of the gripper, whether it's open or closed and where the gripper is. Given the semantic information from the utterance and current vision graph, the robot will apply a `reference resolver` to ground referring expressions in the utterance to the objects in the physical world. In this case, "the blue block" will be grounded to the block *B1* in Figure 1(a) and "the red block to your right" will be grounded to the block *R1*. Since the robot does not understand how to perform the action *stack*, the human will then specify a sequence of actions/steps for the robot to follow, for example, "open gripper, move to the blue block, close gripper, move to the red block on your right, open gripper". Currently, we assume each of the actions in the sequence is known to the robot. The known actions are captured in the `action knowledge base` which specifies the desired goal state for each action. Based on the goal state and the current environment state, the `discrete planner` will come up with a sequence of operators which are directly connected to the `continuous planner` to perform the lower-level arm movements. By following the step-by-step instructions, the robot will come to a final state when the actions are completed. This final state will be captured and connected with the original action *stack* to serve as the representation for the action *stack(x,y)*. The new knowledge about *stack* is thus acquired and stored in the `action knowledge base` for future use.

Next we give a detailed description of some key modules in the system.



(a) (b)

Fig. 1. An example setup. Objects are marked with labels only for illustration purposes.

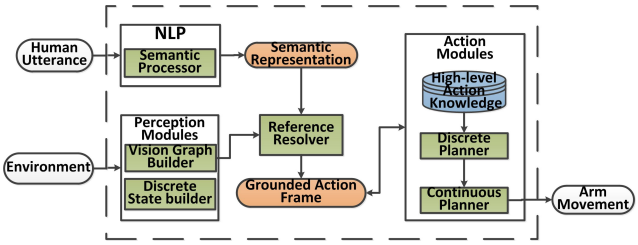


Fig. 2. System Architecture

IV. Natural Language Processing and Referential Grounding

The language understanding process consists of two sub-procedures: *Natural Language Processing (NLP)* and *Referential Grounding*.

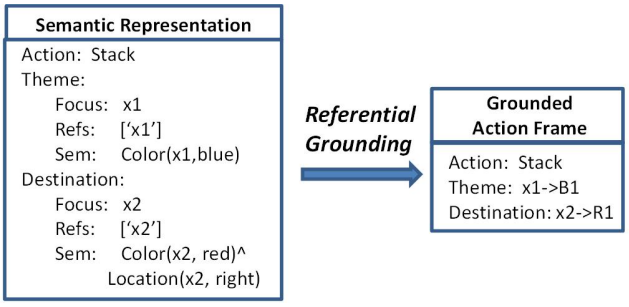


Fig. 3. Example semantic representation and action frame of "*Stack the blue block on the red block to your right.*"

1) Natural Language Processing: Given a raw utterance, the first step is extracting semantic information. Our *NLP* module is mainly composed by a `Semantic Processor`, which is used to extract action related information from the utterances, implemented as a set of *Combinatory Categorial Grammar (CCG)* lexicons and a semantic parser [1]. Currently, the semantic representation captures the actions (e.g., *stack*) and their relevant roles (e.g., *Theme* and *Destination*). The augment for the roles are further represented by the properties of the object (the *Color*, *Location* and *Spatial Relation*). For example, the *Semantic Representation* of "*Stack the blue block on the red block to your right.*" is shown in Figure 3.

2) Referential Grounding: The semantic representation by itself is still not understandable by the robot. It must be grounded to the robot's representation of perception and actions. Extending our previous work[15], we represent the

[1]We used *OpenCCG* as the parser and write the lexicons. *OpenCCG* could be found at: http://openccg.sourceforge.net/

robot's perceived world as a *Vision Graph*, which captures objects and their properties (in the numerical form) recognized by the computer vision algorithm. We further represent linguistic entities and their relations (captured in the semantic representation) as a language graph. Given these graph-based representations, referential grounding (i.e., `Reference Resolver`) becomes a graph matching problem that matches the language graph to the vision graph. Once the references are grounded, the semantic representation of an action becomes a `Grounded Action Frame`. For example, as shown in Figure 3. *B1* and *R1* in the grounded action frame are the referents to "the blue block" and "the red block to your right" respectively.

V. Action Learning through Language Instructions

A. Action Knowledge Representation

The `Action Knowledge Base` represents action knowledge in a three level structure, as shown in Figure 5. At the top level is the *High-level action knowledge*, which stores mappings from high-level action frames captured in human sentences to desired goal states. The middle level is a *Discrete Planner*, like *STRIPS* planner, maintaining domain information and operators. The bottom level is a *Continuous Planner* which calculates the trajectory to realize each of the atomic actions performed by the robotic arm.

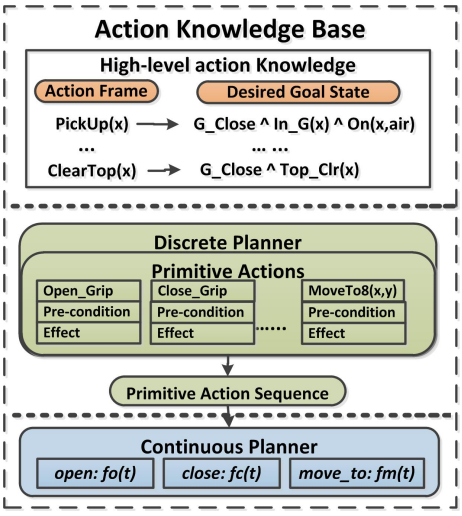


Fig. 5. The three-tier action knowledge representation.

1) Continuous Planner: The lowest level of our knowledge representation is a *Continuous Planner*. It is directly connected with the robotic control system and categorizes the basic operations the robot can perform. In general, this low level representation can include any atomic robotic operations, considering different types of physical robots. For the SCHUNK arm we used, the basic capability can be categorized as three atomic operations: *open* (i.e., open gripper), *close* (i.e., close gripper) and *move_to* (i.e., move to certain location). Each of these three actions corresponds to a pre-defined end-effector trajectory parameterized by time *t*. Specifically, the *open/close* is a function *fo(t)/fc(t)*, which only controls the distance between two gripper fingers attached to the arm. For the *move_to*, given a destination coordinate, function *fm(t)* calculates the trajectory for the end-effector to reach the location but does not change the distance of the gripper fingers. These functions are further translated to control commands through inverse-kinematics and sent to each motor to track the trajectory.

2) Discrete Planner: The middle level action knowledge is used to automatically generate a plan (e.g., an action sequence) to achieve the goals specified by the high-level actions. Specifically, the *Discrete Planner* here is implemented as a *STRIPS* planner[2][11], which is a well-known AI planner. In the *STRIPS*, a domain is defined as a 2-tuple $\mathcal{D} =< \mathcal{P}, \mathcal{O} >$, where $\mathcal{P}$ is a finite set of predicates to symbolically represent the world and $\mathcal{O}$ is a set of legal operations distinguishing changes of the domain world. A planning problem is defined as another 2-tuple $\mathcal{Q} =< \mathcal{I}, \mathcal{G} >$, where $\mathcal{I}$ represents the world status at the beginning of the problem and $\mathcal{G}$ is the desired goal state if the problem can be solved.

Each predicate corresponds to a fluent $p(c_1, c_2, ..., c_n)$, where $p \in \mathcal{P}$ and c_i is an object in the domain. All the possible domain states can be represented by conjunctions of fluents. In our domain, the $\mathcal{P}$ set captures two dimensions of the domain:

- Robotic arm states: *G_Open/Close* stands for whether the gripper is open or closed, *G_Full/Empty* represents whether the gripper has an object in it; and *G_At(x)* describes where the arm locates.
- Object states: *Top_Uclr/Clr(x)* means whether the block x has another block on its top, *In/Out_G(x)* stands for whether it's within the gripper fingers or not; and *On(x,y)* means object x locates on y.

Each operator $o \in \mathcal{O}$ is defined as a set of pre-conditions and effects. Both pre-conditions and effects can be any conjunctions of fluents or the negated fluents. An instance of operator o is only applicable when the pre-conditions can be entailed in the world state. And after accomplishing the operation, the world will change by adding the positive fluents and deleting the negated fluents in effects. In our blocks world domain, the operators include *Open_Grip*, *Close_Grip* and 8 types of *MoveTo*. The 8 *MoveTo*s are used to distinguish different situations of arm position changes, which can be categorized as two sets:

- When the gripper is open: 1. The gripper has no object between fingers (empty) and move destination is an object (m_{oe}_obj); 2. The gripper is empty and moves to locations other than objects (e.g., table and air.) (m_{oe}_loc); 3. The gripper is open but there's an object between the fingers (full) which can only occur when the object is supported, the destination is an object (m_{of}_obj); 4. The gripper is full and destination is locations other than objects (m_{of}_loc).
- When the gripper is closed: 1. The gripper is closed and full, it moves the in-hand object o from the top of object

[2]For the planner implementation we used pyperplan, which can be found under https://bitbucket.org/malte/pyperplan.

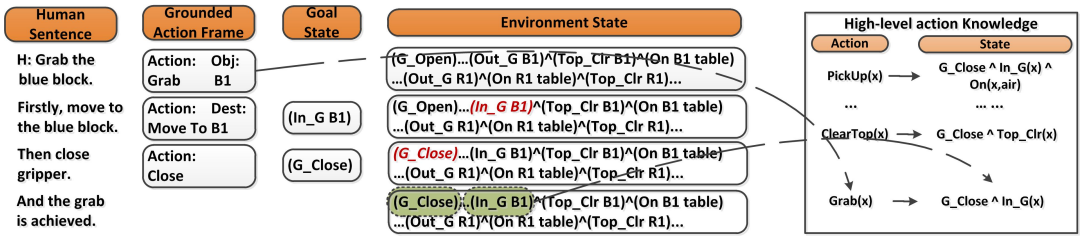


Fig. 4. Learning process illustration. The robot cannot perform after hearing "*Grab the blue block*". So human gives instructions "*Firstly, move to the blue block. Then close gripper. And the grab is achieved.*". By the end of instructions, the robot updates the knowledge by adding in the new *Grab(x)* item.

obj_1 to the top of another object obj_2 ($m_{cf}\_obj_1\_obj_2$); 2. Similar to the $m_{cf}\_obj_1\_obj_2$ but the destination is a location ($m_{cf}\_obj_1\_loc$); 3. The gripper moves the in-hand object from one location loc_1 to another location loc_2 ($m_{cf}\_loc_1\_loc_2$); 4. The gripper moves the object from one location to the top of an object obj_2 ($m_{cf}\_loc\_obj_2$).

3) High-level Actions: This level captures high-level actions known to the robot. High-level actions are those specified by the human partners (e.g., *PickUp* and *ClearTop*). Currently, we model them by their desired goal states:

$$\mathcal{A}_i(c_1...c_{k_i}) = p_1(c_1...c_{k_i}) \wedge ... \wedge p_{n_i}(c_1...c_{k_i}) \wedge p_{grip}$$

$\mathcal{A}_i \in A$ represents high-level action known to the robot. $p_i \in \mathcal{P}$, which is a predicate defined in the domain. c_i is an object (e.g., block) in the robot's view. The $p_{grip} \in \{G\_Open, G\_Close\}$. For example, the "*ClearTop(x)*" will be represented as a goal state "$Top\_Clr(x) \wedge G\_Close$" rather than a sequence of action taught by the human. The intuition behind this is, when humans describe an action, they already have certain goal states in mind. And they want these goal states to be attained after robot performing the action.

B. Action Acquisition

In our action knowledge representation, the middle level and the low level are domain knowledge, which are predefined in the system. The high level actions are specified by humans, which are more flexible and have more possibilities. So, the *Action Acquisition* means learning how to represent the human specified actions as high level action knowledge.

For a new unknown action to the robot, the human will typically go through a sequence of actions. Specifically, suppose there is an unknown action frame $\mathcal{A}_i(c_1...c_k)$ specified by human utterance. $\mathcal{A}_i$ is the name of unknown action, and $c_1...c_k$ are the arguments which are objects in the scene. At the beginning of the instruction, the states of $c_1...c_k$ are $\mathcal{S}_b(c_1...c_k) = p_{b1}(c_1...c_k) \wedge ... \wedge p_{bn}(c_1...c_k)$, where p_{bj} is a fluent whose arguments include all the objects $c_1...c_k$. And at the end of the instruction, we calculate another object states $\mathcal{S}_e(c_1...c_k) = p_{e1}(c_1...c_k) \wedge ... \wedge p_{en}(c_1...c_k)$. This $\mathcal{S}_e$ represents the end states of argument objects. Then, the high level representation of $\mathcal{A}_i(c_1...c_k)$ is calculated as:

$$\mathcal{A}_i(c_1...c_k) = (\mathcal{S}_e - \mathcal{S}_e \cap \mathcal{S}_b) \cup p_{grip} \quad (1)$$

where $p_{grip} \in \{G\_Open, G\_Close\}$ depends on the arm status at the end of instruction.

An example illustration of learning process is shown in Figure 4. Suppose under the setup in Figure 1(a), we teach the robot how to do "grab the blue block", which has action frame $Grab(B1)$. The environment state at this point is captured as shown in Figure 4. And the state of $B1$ is $\mathcal{S}_b(B1) = Out\_G(B1) \wedge Top\_Clr(B1) \wedge On(B1, table)$.

Then the first instruction is "*Firstly, move to the blue block*". As the *MoveTo* action is in the initial system, the goal state *In_G (B1)* is retrieved. After executing the action sequence, the state of block *B1* is changed from *Out_G(B1)* to *In_G(B1)*, which means currently the block *B1* is within gripper fingers. This is reflected in the environment after the first step instruction, shown in Figure 4. After executing next instruction "*Then open gripper.*", the environment changed from *(G_Open)* to *(G_Close)*.

At last, the human informs robot the teaching completion by signaling "*And the grab is achieved.*". The states of $B1$ after the teaching is $\mathcal{S}_e(B1) = In\_G(B1) \wedge Top\_Clr(B1) \wedge On(B1, table)$. By Equation 1, the acquired knowledge of $Grab(B1)$ is inferred as $Grab(x) = In\_G(x) \wedge G\_Close$. Then this new knowledge will be stored in the high level action knowledge base.

VI. EMPIRICAL STUDY

Motivated by Terry Winograd's previous work [22], we designed an empirical study to evaluate our system performance. In the study, we would like to check the system ability to learn new actions through natural language instructions and the ability to apply the learned actions in various novel situations.

In the experiment, we have multiple blocks with different size and color randomly placed on a flat surface. The block positions could be either on the table or on the other blocks, as shown in Figure 1. A SCHUNK arm mounted on a mobile base was used to execute specified actions. To capture the environment in real time, a high resolution web camera is mounted on the mobile base to send video stream to the vision system [10]. With the pre-trained object model of each block, the vision system is able to calculate objects' 3D space location from each video frame. After translating the location in the camera coordinate to the robot's coordinate system, our robot can localize all the objects recognized by the vision server.

A. Experimental Tasks

The experiments mainly consists two stages: a *Learning stage* and a *Testing stage*. In the *Learning stage*, the focus

is to acquire multiple new actions through instructions. The *Testing stage* is designed to evaluate the adequacy of the representation of the learned actions, and the system ability to apply actions in various scenarios.

1) Learning Phase: In this stage, the system high-level action knowledge is initialized with only three actions *Open Gripper*, *Close Gripper* and *Move To*. A human teaches the robot six new high-level actions shown in TABLE I:

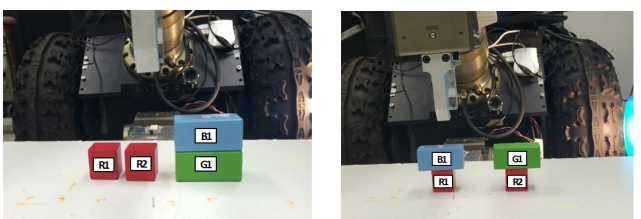
Grab:	To learn: Grab the blue block. Instruction: First, move to the blue block. Then close gripper.
PickUp:	To learn: Pick up the blue block. Instruction: Grab the blue block. And move the blue block to the air.
Drop:	To learn: Drop the blue block in your hand. Instruction: Firstly move the blue block to the table. Then open gripper.
EmptyArm:	To learn: Empty your arm. Instruction: Drop the blue block in your hand. Then go back to the air.
ClearTop:	To learn: Clear the top of the green block. Instruction: Move the blue block on the green block to the table.
Stack:	To learn: Stack the blue block on the red block to your right. Instruction: Move to the blue block. And close gripper. Then move to the top of the red block on your right. And open gripper.

TABLE I

INSTANCES OF UNSEEN ACTIONS AND THEIR INSTRUCTIONS.

For each new action, an action instance is used for the robot to learn, e.g. the "To learn" sentences in the table. After hearing the instruction [3], the robot would follow it step by step until the action completion. And the learning stage will finish until all the 6 actions are learned.

2) Testing Phase: In the testing stage, each of the six learned complex actions was applied in 20 different scenarios. These 20 scenarios would include operating on different objects, changing object locations, and more complex settings. For example, Figure 6 illustrates the learning setup of action *Grab* and one of its 20 testing setups. In the learning process, the *Grab* action was taught through primitive action sequence: *MoveTo(B1)* and *Close_Grip*. However, in the testing phase, the robot arm must remove *G1* first, and then grab the *R2* to complete the action. We would like to evaluate system ability to generate different primitive action sequences in different situations like this example.



(a) Learning: "*Grab the blue block (B1)*" (b) Testing: "*Grab the red block under the green block (R2)*"

Fig. 6. Examples of an learning and a testing setup.

[3]We have the Sphinx integrated in the system. As speech recognition is not our focus, we only used typed utterances in our experiments.

B. Empirical Results

1) Learning Phase: During the *Learning* process, all the six new actions were correctly performed after step-by-step instructions. The *Learning* results are stored in the *High-level action Knowledge* as mappings between action predicates and goal states. Acquired mappings could be found in TABLE II:

$$
\begin{aligned}
EmptyArm &\rightarrow G\_Open \wedge G\_Empty \\
PickUp(x) &\rightarrow G\_Close \wedge In\_G(x) \wedge On(x, air) \\
ClearTop(x) &\rightarrow G\_Close \wedge Top\_Clear(x) \\
Grab(x) &\rightarrow G\_Close \wedge In\_G(x) \\
Drop(x) &\rightarrow G\_Open \wedge On(x, table) \\
Stack(x, y) &\rightarrow G\_Open \wedge On(x, y)
\end{aligned}
$$

TABLE II

LEARNING RESULT OF 6 ACTIONS.

2) Testing Phase: The system capability of applying learned actions to novel situations is reflected in the wide range of the length of generated action sequences during testing. The length of action sequence specified in the learning stage and the statistics of the number of primitive actions generated for the same high-level action in the testing stage (20 scenarios) are shown in Figure 7. As shown in the figure, in the learning stage, we used simpler scenarios. Each high-level action typically could be achieved by 2 to 4 primitive actions. But when applying the learned high-level actions in the 20 testing trials, the number of primitive actions to accomplish these high-level actions are different. This is because the sequence of primitive actions is automatically obtained for each high-level action which depends on the environment states. For certain complex scenarios, e.g., for the *ClearTop* action, it could take up to 15 primitive actions to accomplish the high-level action.

	PickUp	EmptyArm	Grab	Drop	ClearTop	Stack
Teaching length	3	3	2	2	2	4
Mean	5.12	1.44	5.71	1.83	7.21	6.86
Std	1.97	0.50	3.67	0.37	3.93	2.72
Min	4	1	2	1	2	4
Max	12	2	15	2	15	12

Fig. 7. Statistical result for number of primitive actions in generated action sequences.

To evaluate the quality of generated sequence of primitive actions, we use two metrics: 1) R_g, number of high-level actions where correct sequence of primitive actions are identified 2) R_{ge}, number of high-level actions correctly executed. R_{ge} is always smaller than or equal to R_g since even though the action sequence is correct, exceptions could happen during execution (e.g., inaccurate localization) to prevent successful execution of the action sequence. For the R_g, the action sequences for the 120 trails were manually inspected. While for the R_{ge}, it can be directly observed during robot execution. The results of R_g and R_{ge} are shown in Figure 8.

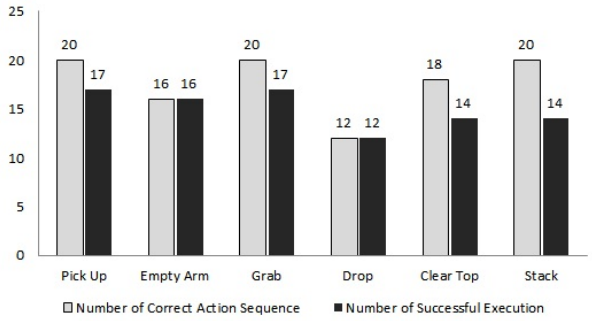


Fig. 8. Testing result

As shown in Figure 8, the *PickUp*, *Grab* and *Stack* actions resulted in 100% correct action sequences. For other actions, our robot was not able to generate correct sequence of primitive actions for 14 trials in total during the testing. The *Drop* action contributes to most of these failures. This is caused by the failure of detecting the object in-hand. Because once an object is between the gripper fingers, it is often blocked by the fingers, which makes it difficult for the vision system to detect the object. But this system flaw could be solved by combining the real-time detected states with recent state history. Another problem is inaccurate localization, which further leads to the incorrect state representation. Our future work will look into these errors to further improve our models.

VII. CONCLUSION

This paper describes a mechanism for humans to teach a robotic arm new high-level actions through natural language instructions. Our current focus is on how to represent high-level actions so that the learned actions can be applied in different situations. Nevertheless, the teaching/learning process is rather complex and many different kinds of exceptions could occur. For example, when teaching a new action, the robot could encounter another unknown action. Thus, it is important to support dialogue and keep track of dialogue discourse to understand relations between different actions [19]. Furthermore, besides actions, the robot may not perceive the environment perfectly. The human and the robot will need to maintain a common ground in order for teaching/learning to be successful. To address these issues, our future work will extend to collaborative dialogue to support automated learning of new actions.

VIII. ACKNOWLEDGEMENT

This work was supported by IIS-1208390 from the National Science Foundation and N00014-11-1-0410 from the Office of Naval Research.

REFERENCES

[1] J. Allen, N. Chambers, G. Ferguson, L. Galescu, H. Jung, M. Swift, and W. Taysom. Plow: A collaborative task learning agent. In *Proceedings of the 22Nd National Conference on Artificial Intelligence - Volume 2*, AAAI'07, pages 1514–1519.

[2] S. Branavan, H. Chen, L. Zettlemoyer, and R. Barzilay. Reinforcement learning for mapping instructions to actions. In *ACL/IJCNLP*, pages 82–90, Suntec, Singapore, August 2009. Association for Computational Linguistics.

[3] S. Branavan, N. Kushman, T. Lei, and R. Barzilay. Learning high-level planning from text. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, pages 126–135, July 2012.

[4] M. Cakmak, C. Chao, and A. L. Thomaz. Designing interactions for robot active learners. *IEEE T. Autonomous Mental Development*, 2(2):108–118, 2010.

[5] R. Cantrell, P. W. Schermerhorn, and M. Scheutz. Learning actions from human-robot dialogues. In H. I. Christensen, editor, *RO-MAN*, pages 125–130. IEEE, 2011.

[6] R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. Tell me when and why to do it! run-time planner model updates via natural language instruction. In *Human-Robot Interaction (HRI), 2012 7th ACM/IEEE International Conference on*, pages 471–478, March 2012.

[7] J. Y. Chai, L. She, R. Fang, S. Ottarson, C. Littley, C. Liu, and K. Hanson. Collaborative effort towards common ground in situated human robot dialogue. In *Proceedings of 9th ACM/IEEE International Conference on Human-Robot Interaction*, Bielefeld, Germany, 2014.

[8] D. L. Chen, J. Kim, and R. J. Mooney. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Int. Res.*, 37(1):397–436, Jan. 2010.

[9] H. I. Christensen, G. M. Kruijff, and J. Wyatt, editors. *Cognitive Systems*. Springer, 2010.

[10] A. Collet, M. Martinez, and S. S. Srinivasa. The MOPED framework: Object Recognition and Pose Estimation for Manipulation. 2011.

[11] R. E. Fikes and N. J. Nilsson. Strips: A new approach to the application of theorem proving to problem solving. In *Proceedings of the 2Nd International Joint Conference on Artificial Intelligence*, IJCAI'71, pages 608–620, 1971.

[12] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *Proceedings of the 5th ACM/IEEE International Conference on Human-robot Interaction*, HRI '10, pages 259–266, Piscataway, NJ, USA, 2010. IEEE Press.

[13] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359, 2008.

[14] S. Lauria, G. Bugmann, T. Kyriacou, and E. Klein. Mobile robot programming using natural language. *Robotics and Autonomous Systems*, 38(3-4):171–181, 2002.

[15] C. Liu, R. Fang, and J. Chai. 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, Seoul, South Korea, July 2012.

[16] C. Liu, R. Fang, L. She, and J. Chai. Modeling collaborative referring for situated referential grounding. In *Proceedings of the SIGDIAL 2013 Conference*, pages 78–86, Metz, France, August 2013.

[17] C. Matuszek, E. Herbst, L. S. Zettlemoyer, and D. Fox. Learning to parse natural language commands to a robot control system. In J. P. Desai, G. Dudek, O. Khatib, and V. Kumar, editors, *ISER*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 403–415, 2012.

[18] S. Mohan, J. Kirk, and J. Laird. A computational model for situated task learning with interactive instruction. In *Proceedings of ICCM 2013 - 12th International Conference on Cognitive Modeling*, 2013.

[19] L. She, S. Yang, Y. Cheng, Y. Jia, J. Chai, and N. Xi. Back to the blocks world: Learning new actions through situated human-robot dialogue. In *Proceedings of the SIGDIAL 2014 Conference*, Philadelphia, US, June 2014.

[20] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. J. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In W. Burgard and D. Roth, editors, *AAAI*. AAAI Press, 2011.

[21] A. L. Thomaz and M. Cakmak. Learning about objects with human teachers. In *Proceedings of the 4th ACM/IEEE International Conference on Human Robot Interaction*, HRI '09, pages 15–22, New York, NY, USA, 2009. ACM.

[22] T. Winograd. Procedures as a representation for data in a computer program for understanding natural language. *Cognitive Psychology*, 3(1):1–191, 1972.

[23] H. Zhang, Y. Jia, and N. Xi. Sensor-based redundancy resolution for a nonholonomic mobile manipulator. In *IROS*, pages 5327–5332, 2012.