Back to the Blocks World: Learning New Actions through Situated Human-Robot Dialogue

Lanbo She[1], Shaohua Yang[1], Yu Cheng[2], Yunyi Jia[2], Joyce Y. Chai[1], Ning Xi[2]
[1]Department of Computer Science and Engineering
Michigan State University
East Lansing, MI 48824, USA
{shelanbo, jchai, yangshao}@cse.msu.edu
[2]Department of Electrical and Computer Engineering
Michigan State University
East Lansing, MI 48824, USA
{chengyu9, jiayunyi, xin}@egr.msu.edu

Abstract

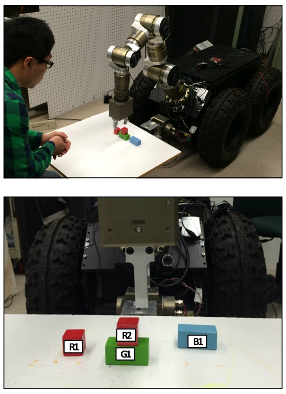
This paper describes an approach for a robotic arm to learn new actions through dialogue in a simplified blocks world. In particular, we have developed a three-tier action knowledge representation that on one hand, supports the connection between symbolic representations of language and continuous sensorimotor representations of the robot; and on the other hand, supports the application of existing planning algorithms to address novel situations. Our empirical studies have shown that, based on this representation the robot was able to learn and execute basic actions in the blocks world. When a human is engaged in a dialogue to teach the robot new actions, step-by-step instructions lead to better learning performance compared to one-shot instructions.

1 Introduction

When a new generation of robots start to work side-by-side with their human partners in joint tasks (Christensen et al., 2010), they will often encounter new objects or are required to perform new actions. It is important for the robots to automatically learn new knowledge about the environment and the tasks from their human partners. To address this issue, this paper describes our recent work on action learning through dialogue. As a first step, we limit our investigation to a simple blocks world motivated by Terry Winograd's early work (Winograd, 1972). By using an industrial robotic arm (SCHUNK) in this small world, we are interested in addressing the following questions. First, human language has a discrete and symbolic representation, but the robot arm has a continuous representation for its movements. Where should the connections between the symbolic representation and the continuous representation take place so that human language can be used to direct the robot's movements? Second, when the robot learns new tasks from its human partner, how to represent the acquired knowledge effectively so that it can be applied in novel situations? Third, during human-robot dialogue, when the robot fails to perform the expected actions due to the lack of knowledge, how should the human teach the robot new actions? through step-by-step instructions or one-shot instructions?

With these questions in mind, we have developed a three-tier action knowledge representation for the robotic arm. The lower level connects to the physical arm and defines the trajectories of executing three atomic actions supported by the arm (i.e., *open_gripper*, *close_gripper*, *move*). The middle level defines primitive operators such as *Open_Grip*, *Close_Grip* and *MoveTo* in the fashion of the traditional AI planner (Fikes and Nilsson, 1971) and directly links to the lower level. The upper-level captures the high-level actions acquired by learning from the human. These high-level actions are represented as the desired goal states of the environment as a result of these actions. This three-tier representation allows the robot to automatically come up with a sequence of lower-level actions by applying existing planning algorithms.

Based on this representation, we implemented a dialogue system for action learning and further conducted an empirical study with human subjects. In particular, we compared the dialogue



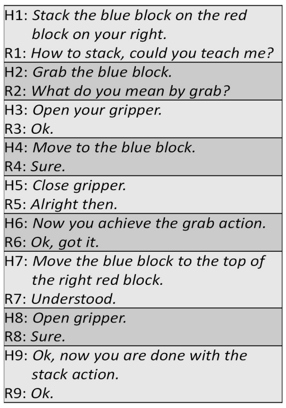

H1: *Stack the blue block on the red block on your right.*
R1: *How to stack, could you teach me?*
H2: *Grab the blue block.*
R2: *What do you mean by grab?*
H3: *Open your gripper.*
R3: *Ok.*
H4: *Move to the blue block.*
R4: *Sure.*
H5: *Close gripper.*
R5: *Alright then.*
H6: *Now you achieve the grab action.*
R6: *Ok, got it.*
H7: *Move the blue block to the top of the right red block.*
R7: *Understood.*
H8: *Open gripper.*
R8: *Sure.*
H9: *Ok, now you are done with the stack action.*
R9: *Ok.*

Figure 1: An example setup and dialogue. Objects are marked with labels only for the illustration purpose.

based on the step-by-step instructions (i.e., one step at a time and wait for the robot's response at each step before going to the next step) with the one-shot instructions (i.e., give the instruction with all steps at once). Our empirical results have shown that the three-tier knowledge representation can capture the learned new action and apply it to novel situations. Although the step-by-step instructions resulted in a lengthier teaching process compared to the one-shot instructions, they led to better learning performance for the robot.

2 Related Work

Over forty years ago, Terry Winograd developed SHRDLU (Winograd, 1972) to demonstrate natural language understanding using a simulated block-moving arm. One aspect he did not address, but mentioned in his thesis (Winograd, 1972) as an important aspect, was learning new actions through natural language. Motivated by Winograd's early work, we start our initial investigation on action learning in a physical blocks world and with a physical robotic arm. The blocks world is the most famous domain used for planning in artificial intelligence. Thus it allows us to focus on mechanisms that, on one hand, connect symbolic representations of language with lower-level continuous sensorimotor representations of the robot; and on the other hand, support the use of the planning algorithms to address novel situations.

Most previous work on following human instructions are based on supervised learning (Kollar et al., 2010; Tellex et al., 2011; Chen et al., 2010) or reinforcement learning (Branavan et al., 2012; Branavan et al., 2010). These types of learning may not be adequate in time-critical situations where only resources available to the robot is its human partners. Thus it is desirable that humans can engage in a natural language dialogue to teach robots new skills. Using natural language dialogue to learn new skills have been explored previously by (Allen et al., 2007) where an artificial agent was developed to acquire skills through natural language instructions (i.e., find restaurant). But this work only grounds language to symbolic interface widgets on web pages.

In the robotics community, previous work has applied learning by demonstration to teach robots new skills (Cakmak et al., 2010). To potentially allow natural language instructions, previous work has also explored connecting language with lower-level control systems (Kress-Gazit et al., 2008; Siskind, 1999; Matuszek et al., 2012). Different from these previous works, here we investigate the use of natural language dialogue for learning actions. Previous work described in (Cantrell et al., 2012; Mohan et al., 2013) is most similar to our work. Here we focus on both grounded learning and the use of planning for action learning.

3 Dialogue System

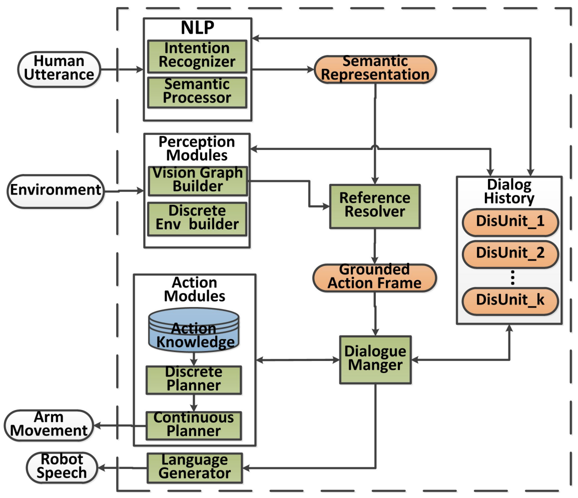


Figure 2: System Architecture

We developed a dialogue system to support learning new actions. An example setup is shown in Figure 1, in which a SCHUNK arm is used to manipulate blocks placed on a surface. In H_1, the human starts to ask the robot to stack the blue block (i.e., B_1) on top of the red block (i.e., R_1). The robot does not understand the action "stack", so it asks the human for instructions. Then the hu-

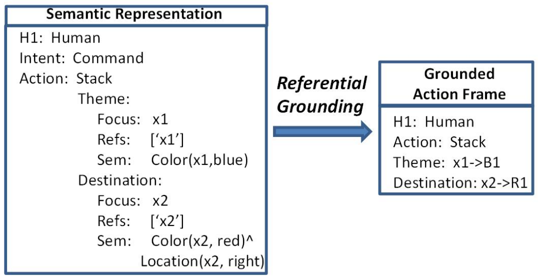


Figure 3: Example semantic representation and action frame for the human utterance "*stack the blue block on the red block on your right.*"

man provides detailed steps to accomplish this action (i.e., H_2 to H_8) and also observes the robot's response in each step. Note that during this process, another unknown action (i.e., "grab" as in H_2) is encountered. The robot thus needs to learn this action first. The robot is able to keep track of the dialogue structure so that actions and subactions can be learned accordingly. Once the robot receives a confirmation from the human that the corresponding action is successfully performed (i.e., H_6 and H_9), it acquires the new action and explicitly represents it in its knowledge base for future use. Instead of representing the acquired knowledge as specific steps as illustrated by the human, the acquired action is represented by the expected final state, which represents the changes of environment as a result of the action. The new action can be directly applied to novel situations by applying planning algorithms. Figure 2 shows the system structure. Next we explain main system modules in detail.

Natural Languge Processing: Natural language processing modules capture semantic information from human language inputs. In particular, the `Intention Recognizer` is used to recognize human intent (e.g., *Command* and *Confirmation*). The `Semantic Processor`, implemented as *Combinatory Categorial Grammar (CCG)* [1], is used to generate semantic representation. Current semantic information includes the actions (e.g., stack) and their roles (e.g., *Theme* and *Destination*). The roles are further represented by objects' properties (*Color*, *Location* and *Spatial Relation*). An example semantic representation of "H1: *Stack the blue block on the red block on your right.*" is shown in Figure 3.

[1]We utilized *OpenCCG*, which could be found at: http://openccg.sourceforge.net/

Perception Modules: Besides interpreting human language, the robot also continuously perceives the shared environment with its camera. Objects in video frames are recognized through vision system (Collet et al., 2011), and further represented as a *Vision Graph* (computed by `Vision Graph Builder`), which captures objects and their properties (in the numerical form). The robot can also access to its own internal status, such as the location of the gripper and whether it's open or closed. Combining the robot's state and environment information, the `Discrete State Builder` can represent the entire environment as a conjunction of predicates, which will be later used for action planning.

Referential Grounding: To make the semantic representation meaningful, it must be grounded to the robot's representation of perception. We use the graph-based approach for referential grounding as described in (Liu et al., 2012)(Liu et al., 2013). Once the references are grounded, the semantic representation becomes a *Grounded Action Frame*. For example, as shown in Figure 3, "*the blue block*" refers to *B1* and "*the red block on your right*" refers to *R1*.

Dialogue Manager: The `Dialogue Manager` is used to decide what dialog acts the system should perform give a situation. It is composed by: a representation of dialogue state, a space of system activity and a dialogue policy. The dialogue status is computed based on the human intention a dialogue state captures (from semantic representation) and the `Grounded Action Frame`. The current space of system activities includes asking for instructions, confirming, executing actions and updating its action knowledge base with new actions. The dialogue policy stores the (dialogue state, system activities) pairs. During interaction, the `Dialogue Manager` will first identify the current dialogue state and then apply the dialogue acts associated with that state as specified in the dialogue policy.

Action Modules: The `Action Modules` are used to realize a high-level action from the `Grounded Action Frame` with the physical arm and to learn new actions. For realizing high-level actions, if the action in the `Grounded Action Frame` has a record in the `Action Knowledge`, which keeps track of all the knowledge about various actions, the

Discrete Planner will do planning to find a sequence of primitive actions to achieve the high-level action. Then these primitive actions will sequentially go through Continuous Planner and be translated to the trajectories of arm motors. By following these trajectories, the arm can perform the high-level action. For learning new actions, these modules will calculate state changes before and after applying the action on the focus object. Such changes of the state are generalized and stored as knowledge representation of the new action.

Response Generator: Currently, the Response Generator is responsible for language generation to realize the detail sentence. In our current investigation, the speech feedback is simple, so we just used a set of pre-defined templates to do language generation. And the parameters in the templates will be realized during run time.

4 Action Learning through Dialogue

To realize the action learning functionality we have developed a set of action related processes including an action knowledge base, action execution processes and action learning processes. Next we give detailed explanations.

4.1 Action Modules

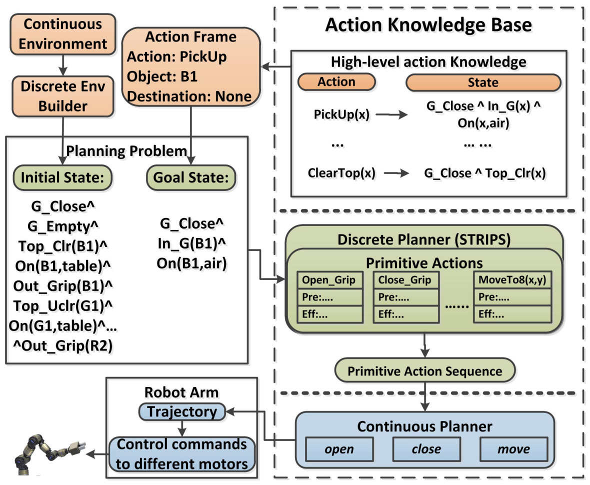


Figure 4: Execution example for "*Pick up the blue block*".

As shown in Figure 4, the action knowledge base is a three-level structure, which consists of *High-level action Knowledge*, *Discrete Planner* and *Continuous Planner*.

4.1.1 Continuous Planner

This lowest level planner defines three primitive actions: *open* (i.e., open gripper), *close* (i.e., close gripper) and *move* (i.e., move to the destination). Each primitive action is defined as a trajectory computing function, implemented as inverse kinematics. The outputs of these functions are control commands sendt to each arm motor to keep the arm following the trajectory.

4.1.2 Discrete Planner

The *Discrete Planner* is used to decompose a high-level action into a sequence of primitive actions. In our system, it is implemented as a *STRIPS* (Fikes and Nilsson, 1971) planner, which is defined as a quadruple $\langle P, O, I, G \rangle$:

- P: Set of predicates describing a domain.
- O: Set of operators. Each is specified by a set of preconditions and effects. An operator is applicable only when its preconditions could be entailed in a state.
- I: Initial state, the starting point of a problem.
- G: Goal state, which should be achieved if the problem is solved.

In our system, O set includes *Open_Gripper*, *Close_Gripper* and 8 different kinds of *MoveTo* (She et al., 2014). And the P set consists of two dimensions of the environment:

- *Arm States*: *G_Open/Close* (i.e., whether the gripper is open or closed), *G_Full/Empty* (i.e., whether the gripper has an object in it) and *G_At(x)* (i.e, location of the arm).
- *Object States*: *Top_Uclr/Clr(o)* (i.e., whether the block *o* has another block on its top), *In/Out_G(o)* (i.e., whether *o* is within the gripper fingers or not) and *On(o,x)* (i.e., *o* is supported by *x*).

The I and G are captured real-time during the dialogue interaction.

4.1.3 High-level action Knowledge

The high-level actions represent actions specified by the human partner. They are modeled as desired goal states rather than the action sequence taught by human. For example, the "*Stack(x,y)*" could be represented as "*On(x,y)*$\wedge$*G_Open*". If the human specifies a high-level action out of the action knowledge base, the dialogue manager will verbally request for instructions to learn the action.

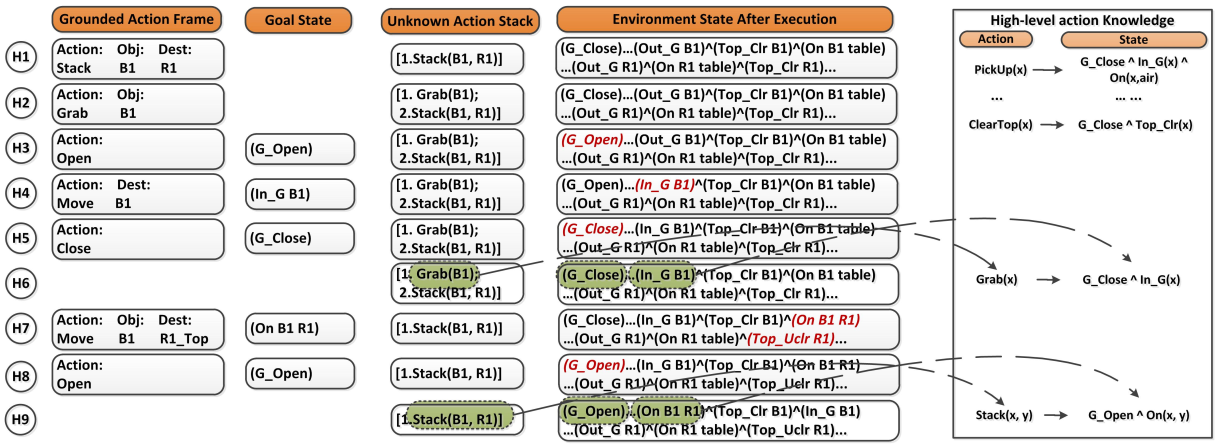

Grounded Action Frame
Goal State
Unknown Action Stack
Environment State After Execution
High-level action Knowledge
Action
State
PickUp(x) → G_Close ^ In_G(x) ^ On(x,air)
ClearTop(x) → G_Close ^ Top_Clr(x)
Grab(x) → G_Close ^ In_G(x)
Stack(x, y) → G_Open ^ On(x, y)
H1
Action: Stack Obj: B1 Dest: R1
[1.Stack(B1, R1)]
(G_Close)...(Out_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H2
Action: Grab Obj: B1
[1. Grab(B1); 2.Stack(B1, R1)]
(G_Close)...(Out_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H3
Action: Open
(G_Open)
[1. Grab(B1); 2.Stack(B1, R1)]
(G_Open)...(Out_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H4
Action: Move Dest: B1
(In_G B1)
[1. Grab(B1); 2.Stack(B1, R1)]
(G_Open)...(In_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H5
Action: Close
(G_Close)
[1. Grab(B1); 2.Stack(B1, R1)]
(G_Close)...(In_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H6
[1. Grab(B1); 2.Stack(B1, R1)]
(G_Close)...(In_G B1)^(Top_Clr B1)^(On B1 table) ...(Out_G R1)^(On R1 table)^(Top_Clr R1)...
H7
Action: Move Obj: B1 Dest: R1_Top
(On B1 R1)
[1.Stack(B1, R1)]
(G_Close)...(In_G B1)^(Top_Clr B1)^(On B1 R1) ...(Out_G R1)^(On R1 table)^(Top_Uclr R1)...
H8
Action: Open
(G_Open)
[1.Stack(B1, R1)]
(G_Open)...(In_G B1)^(Top_Clr B1)^(On B1 R1) ...(Out_G R1)^(On R1 table)^(Top_Uclr R1)...
H9
[1.Stack(B1, R1)]
(G_Open)...(On B1 R1)^(Top_Clr B1)^(In_G B1) ...(Out_G R1)^(On R1 table)^(Top_Uclr R1)...

Figure 5: Learning process illustration. After hearing the stack action, the robot cannot perform. So the human gives step by step instruction. When the instruction is completed, new knowledge of *Grab(x)* and *Stack(x,y)* are learned in the high-level action knowledge base as the combination of the goal state of the robotic arm and the changes of the state for the involved objects.

4.2 Action Execution

Given a `Grounded Action Frame`, it is firstly checked with the high-level action knowledge base. If the knowledge base has its record (e.g., the *Pickup* and *ClearTop* in Figure 4.), a goal state describing the action effect will be retrieved. This goal state, together with the initial state captured from the current environment, will be sent to the `Discrete Planner`. And, through automated planning, a sequence of primitive actions will be generated to complete the task, which can be immediately executed by the arm.

Take the "*Pick up*" action frame in Figure 4 as an example. By checking the grounded action frame with the high-level action knowledge, a related goal state (i.e., "$G\_Close \wedge Top\_Clr(B1) \wedge In\_G(B1) \wedge On(B1,air)$") can be retrieved. At the same time, the `Discrete Evn Builder` translates the real world environment as a conjunction of predicates, which serves as the initial state. Given the combination of initial state and goal state, the *STRIPS* planner can search for a path of primitive actions to solve the problem. For example, the *PickUp(B1)* in Figure 4 can be solved by *Open_Grip*, *MoveTo(B1)*, *Close_Grip* and *MoveTo(air)*.

The primitive actions are executed by the continuous planner and control process in the lower robotic system. For the "*open*" and "*close*", they are executed by controlling the position of the gripper fingers. For the "*move*", a task-space trajectory is first planned based on the minimum-time motion planning algorithm to move the robot end-effector from the current position to the final position. A kinematic controller with redundancy resolution (Zhang et al., 2012) is then used to generate the joint movements for the robot to track the planned trajectory. Achieving the end of the trajectory indicates the action completion.

4.3 Action Learning

Figure 5 illustrates the system internal process of acquiring action knowledge from the dialogue in Figure 1.

At the beginning of the dialogue, the grounded action frame *Stack(B1, R1)* captured from the first human utterance is not in the action knowledge, so it will be pushed to the top of the unknown action stack as a new action waiting to be learned. The environment state at this point is calculated as shown in the figure. Then the robot will verbally request instructions. During the instruction, it's possible that another unknown action *Grab(B1)* is referred. The same as the *Stack* action, it will be pushed to the top of unknown action stack waiting to be learned.

In the next instruction, the human says "*Open your gripper*". This sentence can be translated as action frame *Open* and the goal state "*G_Open*" can be retrieved from the action knowledge base. After executing the action sequence, the gripper state will be changed from "*G_Close*" to "*G_Open*", as shown in Figure 5. In the following two instructions, the human says "*Move to the blue block*" and "*Close gripper*". Similarly, these two instructions are translated as action frames *Move(B1)* and *Close*, then are executed accordingly. After executing these two steps, the state of *B1* is changed from "*Out_G(B1)*" to "*In_G(B1)*".

At this point, the previous unknown action *Grab(B1)* is achieved, so the human says "*Now you achieve the grab action*" as a signal of teaching completion. After acknowledging the teaching completion, the action learning module will learn the new action representation by combining the arm state with the state changes of the argument objects in the unknown action frame. For example, the argument object of unknown action *Grab(B1)* is *B1*. By comparing the original state of *B1*, [*(Out_G B1)*$\wedge$*(Top_Clr B1)*$\wedge$*(On B1 table)*] with the final state, [*(In_G B1)*$\wedge$*(Top_Clr B1)*$\wedge$*(On B1 table)*], *B1* is changed from *(Out_G B1)* to *(In_G B1)*. So, the learning module will generalize such state changes and acquire the knowledge representation of the new action *Grab(x)* as *G_Close*$\wedge$*In_G(x)*.

5 Empirical Studies

The objectives of our empirical studies are two folds. First, we aim to exam whether the current representation can support planning algorithms and execute the learned actions in novel situations. Second, we aim to evaluate how extra effort from the human partner through step-by-step instructions may affect the robot's learning performance.

5.1 Instruction Effort

Previous work on mediating perceptual differences between humans and robots have shown that a high collaborative effort from the robot leads to better referential grounding (Chai et al., 2014). Motivated by this previous work, we are interested in examining how different levels of effort from human partners may affect the robot's learning performance. More specifically, we model two levels of variations:

- **Collaborative Interaction:** In this setting, a human partner provides step-by-step instructions. At each step, the human will observe the the robot's response (i.e., arm movement) before moving to the next step. For example, to teach "stack", the human would issue "pick up the blue block", observe the robot's movement, then issue "put it on the red block" and observe the robot movement. By this fashion, the human makes extra effort to make sure the robot follows every step correctly before moving on. The human partner can detect potential problems and respond to immediate feedback from the robot.
- **Non-Collaborative Interaction:** In this setting, the human only provides a one-shot instruction. For example, to teach "stack", the human first issues a complete instruction "pick up the blue block and put it on top of the red block" and then observes the robot's responses. Compared to the collaborative setting, the non-collaborative setting is potentially more efficient.

5.2 Experimental Tasks

Similar to the setup shown in Figure 1, in the study, we have multiple blocks with different colors and sizes placed on a flat surface, with a SCHUNK arm positioned on one side of the surface and the human subject seated on the opposite side. The video stream of the environment is sent to the vision system (Collet et al., 2011). With the pre-trained object model of each block, the vision system could capture blocks' 3D positions from each frame. Five human subjects participated in our experiments [2]. During the study, each subject was informed about the basic actions the robot can perform (i.e., open gripper, close gripper, and move to) and was instructed to teach the robot several new actions through dialogue. Each subject would go through the following two phases:

5.2.1 Teaching/Learning Phase

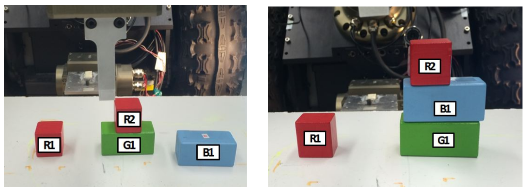
Each subject was asked to teach the following five new actions under the two strategies (i.e., step-by-step instructions vs. one-shot instructions): {Pickup, Grab, Drop, ClearTop, Stack} Each time, the subject can choose any blocks they think are useful to teach the action. After finishing teaching one action (either under step-by-step instructions or under one-shot instructions), we would survey the subject whether he/she thinks the teaching is completed and the corresponding action is successfully performed by the robot. We record the teaching duration and then re-arrange the table top setting to move to the next action.

For the teaching/learning phase, we use two metrics for evaluation: 1) *Teaching Completion Rate*(R_t) which stands for the number of actions successfully taught and performed by the robot; 2)*Teaching Completion Duration* (D_t which measures the amount of time taken to teach an action.

[2]More human subjects will be recruited to participate in our studies.

5.2.2 Execution Phase

The goal of learning is to be able to apply the learned knowledge in novel situations. To evaluate such capability, for each action, we designed 10 additional setups of the environment which are different from the environment where the action was learned. For example, as illustrated in Figure 6, the human teaches the *pick Up* action by instructing the robot how to perform "*pick up the blue block(i.e., B1)*" under the environment in 6(a). Once the knowledge is acquired about the action "pick up", we will test the acquired knowledge in a novel situation by instructing the robot to execute "*pick up the green block(i.e., G1)*" in the environment shown in 6(b).



(a) Learning: the human teaches the robot how to "*pick up the blue block (i.e., B1)*" during the learning phase (b) Execution: the human asks the robot to "*pick up the green block (i.e., G1)*" after the robot acquires the knowledge about "pick up"

Figure 6: Examples of a learning and an execution setup.

For the execution phase, we also used two factors to evaluate: 1) *Action Sequence Generation*(R_g) which measures how many high-level actions among the 10 execution scenarios where the corresponding lower-level action sequences are correctly generated; 2) *Action Sequence Execution*(R_{ge}) which measures the number of high level actions that are correctly executed based on the lower level action sequences.

5.3 Empirical Results

Our experiments resulted in a total of 50 action teaching dialogues. Half of these are under the step-by-step instructions (i.e., collaborative interaction) and half are under one-shot instructions (i.e., non-collaborative). As shown in Figure 7, 5 out of the 50 teaching dialogues were considered as incomplete by the human subjects and all of them are from the *Non-Collaborative* setting. For each of the 45 successful dialogues, an action would be learned and acquired. For each of these acquired actions, we further tested its execution under 10 different setups.

Action learning successful rate	User 1		User 2		User 3		User 4		User 5	
	Non	Col	Non	Col	Non	Col	Non	Col	Non	Col
Pickup	1	1	1	1	1	1	1	1	0	1
Grab	1	1	1	1	1	1	1	1	1	1
Drop	1	1	1	1	1	1	1	1	0	1
ClearTop	1	1	1	1	1	1	1	1	1	1
Stack	1	1	0	1	0	1	0	1	1	1
Total	5	5	4	5	4	5	4	5	3	5

Figure 7: The teaching completion result of the 50 teaching dialogues. "1" stands for the dialogue where the subject considers the teaching/learning as complete since the robot performs the corresponding action correctly; and "0" indicates a failure in learning. The total numbers of teaching completion are listed in the bottom row.

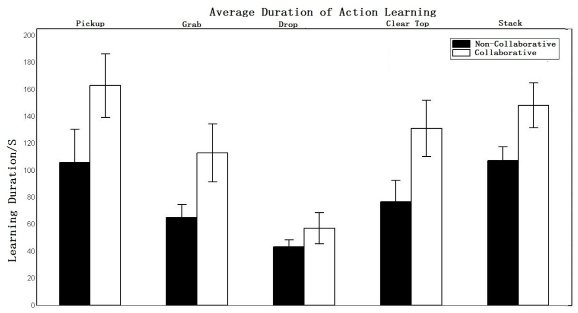


Figure 8: The teaching completion duration results. The durations under the non-collaborative strategy are smaller than the collaborative strategy in most cases.

5.3.1 Teaching Performance

The result of teaching completion is shown in Figure 7. Each subject contributes two columns: the "*Non*" stands for the *Non-Collaborative* strategy and the "*Col*" column refers to the *Collaborative* strategy. As the table shows, all the 5 uncompleted teaching are from the *Non-Collaborative* strategy. In most of these 5 cases, the subjects thought the actual performed actions were different from their expectations. For example, in one of the "*stack*" failures, the human one-shot instruction was "*move the blue block to the red block on the left.*". She thought the arm would put the blue block on the top of red block, open gripper and then move away. However, based on the robot's knowledge, it just moved the blue block above the red block and stopped there. So the subject considered this teaching as incomplete. On the other hand, in the *Collaborative* interactions, the robot's actual actions could also be different from the subject's expectation. But, as the instruction

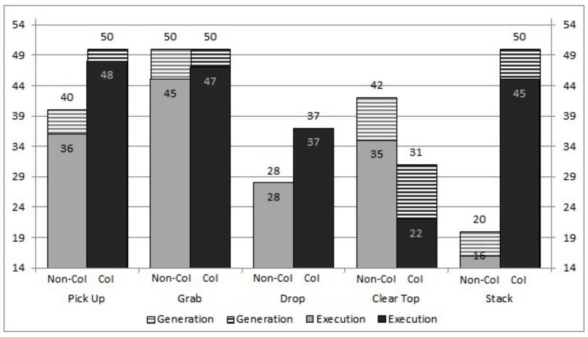


Figure 9: Each bar represents the number of successfully generated action sequences during testing. The solid portion of each bar represents the number of successfully executed action sequences. The number of successfully execution is always smaller than or equal to the generation. This is because we are dealing with dynamic environment, and the inaccurate real-time localization will make some correct action sequence fail to be executed.

was given step-by-step, the instructors could notice the difference from the immediate feedback and adjust their follow-up steps, which contributed to a higher completion rate.

The duration of each teaching task is shown in Figure 8. Bar heights represent average teaching duration, the ranges stand for standard error of the mean (SEM). The 5 actions are represented by different groups. As shown in the figure, the teaching duration under the *Collaborative* strategy tends to take more time. Because in the *Collaborative* case, the human needs to plan next step after observing the robot's response to a previous step. If an exception happens, a sub-dialogue is often arranged to do correction. But in the *Non-Collaborative* case, the human comes up with an entire instruction at the beginning, which appears more efficient.

5.3.2 Execution Performance

Figure 9 illustrates the action sequence generation and execution results in the execution phase.

As shown in Figure 9, testing results of actions learned under the Collaborative strategy are higher than the ones using *Non-Collaborative*, this is because teaching under the *Collaborative* strategy is more likely to be successful. One exception is the *Clear Top* action, which has lower generation rate under the *Col* setting. By examining the collected data, we noticed that our system failed to learn the knowledge of *Clear Top* in one of the 5 teaching phases using *Col* setting, although the human subject labeled it as successful. Another phenomenon shown in Figure 9 is that the generation results are always larger than or equal with the corresponding execution results. This is caused by inaccurate localization and camera calibration, which introduced exceptions during executing the action sequence.

6 Conclusion

This paper describes an approach to robot action learning in a simplified blocks world. The simplifications of the environment and the tasks allow us to explore connections between symbolic representations of natural language and continuous sensorimotor representations of the robot which can support automated planning for novel situations. This investigation is only our first step. Many issues have not been addressed. For example, the world is full of uncertainties. Our current approach can only either succeed or fail executing an action based on the acquired knowledge. There is no approximation or reasoning of the uncertain states which may affect potential execution. Also, when the robot fails to execute an action, there is no explanation why it fails. If the robot can articulate its internal representations regarding where the problem occurs, the human can provide better help or targeted teaching. These are the directions we will pursue in our future work.

7 Acknowledgment

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

References

James F. Allen, Nathanael Chambers, George Ferguson, Lucian Galescu, Hyuckchul Jung, Mary D. Swift, and William Taysom. 2007. Plow: A collaborative task learning agent. In *AAAI*, pages 1514–1519. AAAI Press.

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 '10, pages 1268–1277, Stroudsburg, PA, USA. Association for Computational Linguistics.

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 (Volume 1: Long Papers)*, pages 126–135, Jeju Island, Korea, July. Association for Computational Linguistics.

Maya Cakmak, Crystal Chao, and Andrea Lockerd Thomaz. 2010. Designing interactions for robot active learners. *IEEE T. Autonomous Mental Development*, 2(2):108–118.

R. Cantrell, K. Talamadupula, P. Schermerhorn, J. Benton, S. Kambhampati, and M. Scheutz. 2012. 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.

Joyce Y. Chai, Lanbo She, Rui Fang, Spencer Ottarson, Cody Littley, Changsong Liu, and Kenneth Hanson. 2014. Collaborative effort towards common ground in situated human robot dialogue. In *Proceedings of 9th ACM/IEEE International Conference on Human-Robot Interaction*, Bielefeld, Germany.

David L. Chen, Joohyun Kim, and Raymond J. Mooney. 2010. Training a multilingual sportscaster: Using perceptual context to learn language. *J. Artif. Int. Res.*, 37(1):397–436, January.

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

Alvaro Collet, Manuel Martinez, and Siddhartha S. Srinivasa. 2011. The MOPED framework: Object Recognition and Pose Estimation for Manipulation.

Richard E. Fikes and Nils J. Nilsson. 1971. 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, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

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 '10, pages 259–266, Piscataway, NJ, USA. IEEE Press.

Hadas Kress-Gazit, Georgios E. Fainekos, and George J. Pappas. 2008. Translating structured english to robot controllers. *Advanced Robotics*, 22(12):1343–1359.

Changsong Liu, Rui Fang, and Joyce 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, Seoul, South Korea.

Changsong Liu, Rui Fang, Lanbo She, and Joyce Chai. 2013. Modeling collaborative referring for situated referential grounding. In *Proceedings of the SIGDIAL 2013 Conference*, pages 78–86, Metz, France.

Cynthia Matuszek, Evan Herbst, Luke S. Zettlemoyer, and Dieter Fox. 2012. Learning to parse natural language commands to a robot control system. In Jaydev P. Desai, Gregory Dudek, Oussama Khatib, and Vijay Kumar, editors, *ISER*, volume 88 of *Springer Tracts in Advanced Robotics*, pages 403–415. Springer.

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

Lanbo She, Yu Cheng, Joyce Chai, Yunyi Jia, Shaohua Yang, and Ning Xi. 2014. Teaching robots new actions through natural language instructions. In *RO-MAN*.

Jeffrey Mark Siskind. 1999. Grounding the lexical semantics of verbs in visual perception using force dynamics and event logic. *J. Artif. Int. Res.*, 15(1):31–90, February.

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 Wolfram Burgard and Dan Roth, editors, *AAAI*. AAAI Press.

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

Huatao Zhang, Yunyi Jia, and Ning Xi. 2012. Sensor-based redundancy resolution for a nonholonomic mobile manipulator. In *IROS*, pages 5327–5332.