ABSTRACT
To effectively cooperate with a human, advanced manufacturing machines are expected to execute the industrial tasks according to human natural language (NL) instructions. However, NL instructions are not explicit enough to be understood and are not complete enough to be executed, leading to incorrected executions or even execution failure. To address these problems for better execution performance, we developed a Natural-Language-Instructed Task Execution (NL-Exe) method. In NL-Exe, semantic analysis is adopted to extract task-related knowledge, based on what human NL instructions are accurately understood. In addition, logic modeling is conducted to search the missing execution-related specifications, with which incomplete human instructions are repaired. By orally instructing a humanoid robot Baxter to perform industrial tasks “drill a hole” and “clean a spot”, we proved that NL-Exe could enable an advanced manufacturing machine to accurately understand human instructions, improving machine’s performance in industrial task execution.

INTRODUCTION
To effectively cooperate with a human, advanced manufacturing machines (machine for short) are expected to execute industrial tasks by following human instructions [1][2]. Typical communication methods for translating human instructions into a machine-readable commands, include visual-clue-based method which is involving in human body pose [3], facial expression [4], and object [5], kinematics-clue-based method which is involving in action trajectory [6], action sequence [7] and movement dynamics [8], and bio-signal-clue-based method which is involving in electroencephalogram (EEG) pattern [9] and heart rate [10]. By using the above mentioned communication methods, a machine is enabled to understand human instructions and execute the human-assigned industrial task.

However, all these methods were sharing the common disadvantages. First, their communication manners are unnatural. Only the expert users who have received prior trainings can use the above methods [3][7]. Second, the methods are ineffective in information transferring. Only some simple information patterns are defined, insufficiently describing task-related knowledge such as task procedures and human special requirements [6]. Third, these methods are procedure-complex and time-consuming. Their clue patterns need to be defined for encoding and decoding the task-related information embedded in human instructions [8].

Natural language (NL) communication is natural for that even the non-expert user without any prior training could interface with the machine [11][12]. NL is effective in information transferring for that NL includes sufficient standard linguistic patterns for delivering various information from a human to a machine [13][14]. NL communication is procedure-concise. Given NL is already an encoded information structure, the NL-based communication don’t need the encoding process, largely simplifying the information delivering procedures and reducing the information-transferring time [15][16]. Given the above advantages of NL communication, we adopted the NL communication method for human-machine cooperation in industrial task execution, aiming to improve the performance of an advanced manufacturing machine in following human instructions.

With human NL instructions such as “slowly drill one hole in upper-right corner”, a machine is expected to understand the task goal such as “drill a hole” and also the execution-related specifications “location: upper right corner; requirement: slowly; action: drill”. However, some problems still exist, limiting a machine to directly understand a task from human NL instructions. First, NL instructions are various and ambiguous [17]. Determined by NL’s own characteristics such as polysemous, homophonous and divers, the expression manners of NL instructions are various. For example, “drill a hole” could also be expressed as “bore one hole”, “drilling one bore”, “create an unthreaded hole” and so on. Determined by instructors’ expression habits including referring, outlining and omitting, the NL instructions are usually ambiguous [18]. For example, in instruction “at the center point”, information such as “which
object in which place has the center point” cannot be known merely from a word ‘the’. It is challenging to accurately extract task-related knowledge in various instruction context. Second, a plan generated from human NL instructions is usually incomplete. For example, in instruction ‘clean the surface’, the task goal is “clean” while the tools such as “brush” and actions such as “sweep” have not been told explicitly [1]. It is challenging to perform a task by directly following human NL instructions.

Given the aforementioned problems, we developed a Natural-Language-Instructed Execution (NL-Exe) method. Our contributions in this method include: (1) a novel Natural-Language-Instructed Task Execution (NL-Exe) method is designed for grounding human NL instructions to executable task plan; (2) a novel semantic analysis method is developed for extracting task-related knowledge from various NL expressions; (3) a novel logic modeling method is developed for repairing the incomplete task-related knowledge into executable plan.

RELATED WORK

Object is grounded from word in natural instructions into a real object in world for guiding autonomous execution [19][20]. With semantic analysis such as “different trees” were belonging to the category “tree”, human mentioned objects such as “tree” was identified in the real world, helping a machine to adapt to unstructured environment. Even though similar semantic analysis methods were adopted in both of our works, our focuses were different. They focused on identifying the NL-instructed objects in the real world with assumptions that knowledge gaps in NL-instructed plan were not existed. While we noticed the existences of knowledge gaps in the NL-instructed plans and aimed to fill them.

In both [23][24] and our work, actions and environmental constrains were considered to help an advanced machine with decision making in the task execution. The usage manners of actions and environmental constrains were different in our methods. They conducted quantitative analysis such as “slowly” on the mentioned actions “turn” for meeting specific human requirements such as “turn slowly”, while we conducted qualitative analysis such as actions {clean, scrub, wipe, wash} are categorized into same action type “clean” for accomplishing the goals such as “clean the surface”. In their work, environmental constrains were object size and position; by considering these constrains the appropriate actions were adopted to generate proper actions. While in our work the environmental constrains were execution preconditions such as the “hole is not existed” and “surface is dirty”; by considering the constrains, appropriate sub-goal sequences such as “drill→clean” were planned.

Similar works [25][26] have been done to explore the logic relations for task execution. Many of them explored the combined logic relations such as “pick up + object” to flexibly execute tasks such as “pick + book” and “pick up + DVD”. They aimed to enable a machine to adapt to untrained situations. While we explored the sequentially logic relations such as “make the surface clean → drill one hole” and also explored the execution specifications such as “tool for clean/drill” for realizing the logic relations. We aimed to enable a machine to execute incomplete NL instructions according to their context.

NL-EXE METHODOLOGY

As Fig. 1. shows, the Natural-Language-Instructed Execution (NL-Exe) method interpret human NL instructions (input) into machine’s autonomous execution (output). The NL-Exe method includes 3 sections: Semantic Analysis, Knowledge-World Mapping and Executable Plan Generation. Semantic analysis extracts task-related knowledge including task goals and Machine Execution Specification (MES and details about MES is described in section III. B) from NL instructions. Knowledge-world mapping maps sub-goals and their corresponding MES such as tool involvement and tool location from extracted knowledge into the real world. Executable plan generation adds the missing MES to repair the incomplete task-related knowledge into an executable plan. Based on the plan an advanced machine autonomously executes human-assigned task. The following paragraphs will describe the three sections in details.

A. Semantic Analysis

When human NL instructions are given, the advanced machine should know “what to do” which is defined as task goal (such as ‘drill hole’) and “how to do” which is defined as MES (such as tool: brush, location: middle point).

(1) Initial Instruction Processing

Initially instruction processing aims to generate the sentence structure, preparing for further knowledge extraction.

First, human’s English NL instructions are interpreted into text corpus by using speech recognition tool SpeechRecognition [27]. Second, text corpus is splitted into independent sentences, words, Part-of-Speech (PoS) tags and dependences by using natural language processing (NLP) tool Stanford CoreNLP [28]. Then word normalization is conducted to normalize the interested keywords such as “drilling, drills and drill” to unified morphologies such as “drill”. Last, based on the NLP results sentence structure is analyzed, shown in Fig. 2 where {root: sentence root, dobj: direct object dependency, nomb: norm modifier dependency, amod: adjectival modifier, case: case dependency, det: determiner dependency, VB: verb, NN: noun, DT: determiner, JJ: adjective and IN: preposition}.

(2) Task-related Knowledge Extraction

Task-related knowledge includes task sub-goals and the corresponding MES. Each entity of task-related knowledge was defined by a semantic feature space, which is expressed by three types of features {keywords, PoS tag, word dependency}. Take task goal “drill a hole” for example, keywords are {drill, hole}, PoS tag are {VB, NN} and word dependency is {dobj}. The standard form for each entity is defined in the local database. By comparing the semantic features, similarity of a likely entity with the locally defined entity is calculated, shown in Fig. 3 and eq. (1). Feature weight is denoted by \( w_k \) (k=1, 2, 3). Feature similarity is denoted by \( \alpha_k \) (k=1, 2, 3). When a feature in practical situation is the same with the predefined one, \( \alpha_k \) is equal to 1; otherwise it is 0. Given a set of keywords is likely to
Figure 1. Framework of Natural-Language-Instructed Execution (NL-Exe) Method. With NL-Exe, human NL instructions is interpreted to executable plan for supporting the machine task execution. In NL-Exe method, semantic analysis module analyzes task-related knowledge; then the knowledge-world mapping module maps the knowledge into real world; last the executable plan generation module searches missing knowledge to generate an executable plan.

Figure 2. Syntax analysis for sentence “drill a hole at the right corner of the wood”. Sentence structure is analyzed by linguistic features sentence root, words, PoS tags and word dependencies.

Figure 3. Semantic Similarity Measurement of Task-related Entity “goal: drill”

be involved in multiple entities $e_i$ ($i=1, 2, \ldots, M$), we designed a classifier with which the entity with largest overall similarity value is identified as the actual human-meant entity $e_{max}$, described in eq. (2).

$$e_i = \sum_{k=1}^{3} w_k \alpha_k$$  \hspace{1cm} (1)

$$e_{max} = \arg \max_{e_i \in \{e_1, e_2, \ldots, e_M\}} e_i$$  \hspace{1cm} (2)

B. Knowledge-World Mapping
First to set standards for task execution and also for the knowledge-world mapping, we defined a machine execution specification (MES) language, which is constructed by the Backus-Naur Form (BNF) [19][29]. MES specified the execution-related critical parameters, including task goal (goal), precondition of goal realization (precondition), actions of task performing (actions), tools of task performing (tools), working spot location (location) and human special requirements towards task performing (requirements). Sample MES for goal “clean a spot” is shown in Fig. 4. Second, by comparing the NL-mentioned MES with the actually existed MES, task-related knowledge is mapped into real world. MES mentioned in NL is usually inconsistent with the existed MES. With semantic similarity measuring, some of the mentioned MES could be replaced by the existed MES. For example, “brush” is replaceable by “rag” and “center point” is replaceable by “wood middle”.

C. Executable Plan Generation
To detect and repair the incomplete task-related knowledge into executable plan, the entire plan is presented in logic manner with which the logic relations among task goal, sub-goals and the corresponding MES were explored. With the logic-manner
representation, task-related entity will be detected when it is missing. Given first-order logic has an unambiguous syntax, precisely describing likely situations in a world [30], we use it for modeling the mutual relations among task-related knowledge.

(1) Task Formulization

A task-goal is defined as the logic conjunctions of the related sub-goals. Only when all the sub-goals have been accomplished, the task goal could be achieved. The formulization for task goal is shown in Fig. 5 and eq. (3) where \( n \) denotes the total sub-goal number for the overall task goal, \( l_i \) denote the \( i \)-th working location \((i = 1, 2, \ldots, n)\) in a working surface and \( \text{subgoal}_i \) denotes the \( i \)-th sub-goal. MES is expressed as task goal \( \text{Goal} \), task sub-goal \( \text{subgoal} \), preconditions \( \text{PreCond} \), actions \( \text{Action} \), tools \( \text{Tool} \) and requirements \( \text{Req} \). All the MES features were conjunct for that only all the critical variables have been specified, a sub-goal could be executed, shown Fig. 5 and eq. (4).

\[
\forall l_i \left[ \bigwedge_{i=1}^{n} \text{subgoal}_i(l_i) \rightarrow \text{Goal}(l_i) \right] \tag{3}
\]

\[
\forall \forall \forall \forall a \left[ \text{PreCond}(l) \land \text{Action}(a, l, t) \land \text{Tool}(t, l) \land \text{Req}(a, l, t) \rightarrow \text{subgoal}(l) \right] \tag{4}
\]

![Figure 5. Executable plan representation with a MES-based first-order logic network](image)

(2) Executable Plan Generation

A backward-chaining logic method [31] is adopted to check the execution feasibility of each higher-level formula based on the lower-level formulas and then repair the infeasible formula by adding back the missing knowledge, shown in Fig. 5 arrow. Given the paper length limitation, we ignored the sub-goal missing problem and only focused on the MES missing problem. Based on the assigned task goal (higher-level formula), the lower-level sub-goals were queried from the knowledge database; then for each sub-goal, the feasibility of execution is assessed and the missing MES variables were detected. Detailed model learning process is shown in Algorithm I. The core idea of backward-chaining is downward searching which first set up a goal and then search the required MES to realize the goal.

![Algorithm I Executable Task Generation](image)

**EVALUATION**

A humanoid Baxter robot is orally instructed to execute three tasks \{drill, clean, drill+clean\} with two different locations \{center spot, upper-right corner\} in controlled lab environment, shown in Fig. 6. The three tasks are mutually similar and dependent. When they are described, the instructions manners are mutually similar and confusing. Therefore, the three tasks were selected to evaluate the effectiveness of NL-Exe in task understanding and generation.

Two main aspects of our NL-Exe method were evaluated. First, the accuracy of task understanding was evaluated by the accuracy of extracting sub-goal sequence and sub-goals’ MES from various natural descriptions. Second, the effectiveness of executable task execution was evaluated by calculating the correctness of mapping knowledge into real world and the feasibility of executing the final generated tasks in real environment.

**A. Knowledge Representation**

To generate the knowledge supporting robot’s task executions, 50 volunteers were surveyed that each of them is required to give completed instructions for executing the three tasks in the two locations. The instructions for each task include task goal, sub-goals and MES. Then with two experts, volunteers’ instructions were processed into first-order logic networks for formulizing the tasks.
Figure 6. Task execution with a Baxter Robot. Three tasks \{drill, clean, drill+clean\} were performed in two different points \{center spot, upper-right corner\}.

Figure 7 shows the semantic feature maps for basic sub-goal “clean/drill”. For ‘clean a spot’, the important keywords were categorized into 3 types \{clean, dust, spot\}. The typical word pattern is \{clean, dust, spot\}, the corresponding PoS tag pattern is \{VB, NN, NN\} and the corresponding dependency pattern is \{dobj, nmod, dobj\}. The feature map for “drill a hole” could be interpreted in the same way. With Fig. 7, we could see the NL instructions which were semantically similar also shares the similar semantic feature patterns. Based on this, semantic similarity measuring is conducted.

Figure 8 shows the logic relations and the MES values. For example, sub-goal clean is usually happened after the drilling and they usually happened in a same location. Precondition usually decides the involvements of location, action and tools. Both the semantic feature maps and the task logic representation are consistent with daily experience, evaluated knowledge correctness.

B. Evaluation for Task Understanding

Another 20 volunteers were involved to give NL instructions for executing the three tasks in the two locations. Accuracy of sub-goal sequence extraction and MES extraction were used to evaluate the reliability of the NL-Exe method in generating plan framework and plan details respectively. The recall is calculated by the ratio of the number of robot extracted items (sub-goal sequences or MES) to the number of overall items (sub-goal sequences or MES), which were extracted by one new volunteer and considered as ground truth baseline. The precision was calculated by the ration of the number of correctly predicted items to the number of all the predicted items.

As Table I shows, the average recall is higher than 0.83 and average precision is 1.00, proving that the NL-Exe method is reliable in extracting task-related knowledge. The good performance in the three different tasks shows NL-Exe method has good adaptability in adapting different instruction context.

C. Evaluation for Executable Plan Generation

After the task understanding, Baxter was required to repair the incomplete extracted knowledge for executable plan generation. Knowledge-world mapping correctness is for assessing NL-Exe’s reliability of mapping the NL-described MES into the actual MES in real world. Correctness recall calculated by the proportion of successfully mapped MES among all the actual MES. Correctness precision calculated by the proportion of correctly mapped MES among all the mapped MES. Task execution feasibility is used for evaluating effectiveness of the NL-Exe method in generating executable plan. It is calculated by the proportion of executable plans among all the generated plans. Plan executability is calculated by comparing the generated plan with the actually executable plan.

As Table II shows, the average mapping correctness is with recall higher than 0.94 and precision higher than 0.98, proving the NL-Exe method is accurate in mapping the mentioned MES into the real world. The final average task execution feasibility is higher than 0.78, which means more than 78% of generated plans are actually executable. High execution feasibility proves that the NL-Exe method is effective in repairing the incomplete plan into executable plan. With effectively executable plan generation, machines’ performance in understanding human instructions and autonomous task execution is largely improved.

CONCLUSION & FUTURE WORK

In this paper, we presented a novel method, named NL-Exe, which interpreted human ambiguous and incomplete NL instructions into machine executable plan. By performing three different tasks \{drill, clean, drill+clean\} in controlled lab environment, the effectiveness of NL-Exe method in understanding task and generating executable plan was demonstrated. In the future, we will use the computer vision technics and sensor network to perceive the world for complicated task performing.

ACKNOWLEDGEMENT

We would like to thank Dr. Hao Zhang for providing Baxter robot to help us with the experiment.

REFERENCES

(a) a map for “clean the spot”  
(b) a map for “drill a hole”

**Figure 7.** Semantic feature maps. A basic task is represented by task goal such as “clean” and “drill”, task specification such as “dust” and working area location such as “spot”.

**Figure 8.** Executable Task Modeling by First-Order Logic Network. onlyDrill means the machine is only required to drill a hole without cleaning. onlyClean means the machine is only required to clean a spot without drilling. drillClean means the machine is required to drill a hole and then clean it.

<table>
<thead>
<tr>
<th>Table I. Accuracy of Task Understanding</th>
<th>Table II. Evaluation of Executable Plan Generation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Tasks</td>
<td>Knowledge-World Mapping Correctness</td>
</tr>
<tr>
<td>subgoal Extraction</td>
<td>Recall</td>
</tr>
<tr>
<td>only drill</td>
<td>0.83</td>
</tr>
<tr>
<td>only clean</td>
<td>0.95</td>
</tr>
<tr>
<td>drill+clean</td>
<td>0.73</td>
</tr>
<tr>
<td>Average</td>
<td>0.84</td>
</tr>
</tbody>
</table>

*subgoal Extraction: average accuracy of sub-goal sequence extraction; MES Extraction: average accuracy of MES extraction


