2014 IEEE International Conference on Robotics & Automation (ICRA)
Hong Kong Convention and Exhibition Center
May 31 - June 7, 2014. Hong Kong, China

Ensuring Safety in Human-Robot Dialog – a Cost-Directed Approach

Junaed Sattar and James J. Little[1]

***Abstract*— We present an approach for detecting potentially unsafe commands in human-robot dialog, where a robotic system evaluates *task cost* in input commands to ask input-specific, directed questions to ensure safe task execution. The goal is to reduce risk, both to the robot and the environment, by asking context-appropriate questions. Given an input program, (*i.e.*, a sequence of commands) the system evaluates a set of likely alternate programs along with their likelihood and cost, and these are given as input to a *Decision Function* to decide whether to execute the task or confirm the plan from the human partner. A process called *token-risk grounding* identifies the costly commands in the programs, and specifically asks the human user to clarify those commands. We evaluate our system in two simulated robot tasks, and also on-board the Willow Garage PR2 and TurtleBot robots in an indoor task setting. In both sets of evaluations, the results show that the system is able to identify specific commands that contribute to high task cost, and present users the option to either confirm or modify those commands. In addition to ensuring task safety, this results in an overall reduction in robot reprogramming time.**

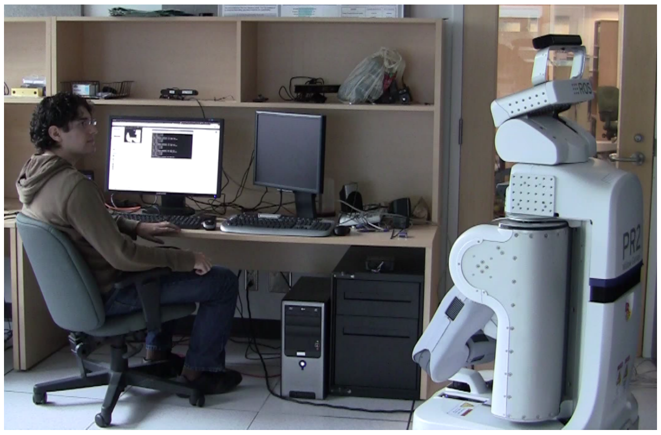
Fig. 1: A user giving instructions to a mobile robot.

I. INTRODUCTION

Task safety is an essential requirement in robot operations, particularly important in scenarios where robots and humans exist in shared spaces (*e.g.*, see Figure 1). When commanded by a human operator, the robot must take into consideration the effect of task execution on itself, on the human users, and also on the surroundings. The cumulative cost of executing any given plan thus has to take into consideration operating costs for the robot and possible risk to the robot and environment, and is an important factor in human-robot interaction. In this work, we address the problem of execution-under-risk in human-robot dialog; more specifically, we propose an approach to ensure that costly commands are not executed without confirmation from a human user, and then present an algorithm to identify elements in the command that contributed to the high-cost evaluation of the input command. In this work, human-robot interaction occurs using a structured language called RoboChat [1] [2], which has been previously used to program and communicate with an amphibious legged robot in both terrestrial and underwater environments. RoboChat enables a human partner to program a robot using an arbitrary communication medium, which may include but would not be limited to visual gestures, audio and more “traditional” input methods (such as mice and keyboards, or touch interfaces)

*This work partially supported by Fonds de Recherche du Quebéc – Nature et Technologie and CANWHEEL (the Canadian Institutes of Health Research (CIHR) Emerging Team in Wheeled Mobility for Older Adults #AMG-100925)

[1]The authors are with the Department of Computer Science, University of British Columbia, 201-2366 Main Mall, Vancouver, B.C. Canada V6T 1Z4. {junaed,little} at cs.ubc.ca

Our previous work has looked at quantitatively modeling human-robot dialog, accounting for uncertainty in input and corresponding task cost [3]. In that work, after the human operator programs a robot to perform a set of tasks, the robot assesses command execution cost and uncertainty in the input, and uses a *Decision Function* to decide whether or not to confirm this task. To that end, the system uses a set of likely programs, derived from the input using a hidden Markov model. While the cost assessment helps to prevent potentially unsafe operations from occurring without a confirmation, it does not provide to the human a cost estimate of the individual elements in the input program (which we refer to as language *tokens*). Consequently, if a program is evaluated as high-cost, the user would have to confirm (or decline) the program without a sense of the true reason as to why it was evaluated as such. In the worst case, the user may need to re-enter the program in its entirety, with corrections in case there are mistakes. For long sequence of instructions, this approach is time consuming, tedious, and increases the chance of erroneous inputs.

This work addresses this specific problem, by not only confirming a costly program, but also by asking specific questions about the parts of the input that contribute to its high-cost evaluation. The contributions in this paper are the following:

1) an algorithm to identify high-cost elements in the input, through a process called *token-risk grounding*;
2) ensure user intent through multi-stage clarification queries for these specific tokens, and if required, for the final statement as a whole;
3) minimize reprogramming time by specific, targeted queries.

For each token in the program, we use a metric to evaluate the probability that the token requires clarification. While we do not perform strict grounding of tokens (to objects or actions in the physical world, for example), we use the robot's local (*i.e.*, instantaneous) and global (*i.e.*, environmental) sensor data to calculate the cost contribution of a token within a program, using a set of *cost assessors*. After executing each command in the input statement, each assessor examines the current state of the robot and produces an estimated task cost. The evaluation takes place in batch mode, and no clarifications are asked until the end of the input is reached. Once the human operator responds to all the clarification queries, the command is reevaluated, and if needed, a confirmation request is generated.

II. Related work

The research presented in this paper extends the authors' prior work on quantitatively modeling human-robot dialog considering input uncertainty and task cost. The dialog mechanism relies on having an explicit means of communication between a robot and a human partner. A rich, varied ensemble of communication modalities have been used in human-machine interfaces, both in robotics and human-computer interaction. Also, we apply a Markovian dialog model to infer intentions from dialog and to measure uncertainty. We present previous work in each of these disparate domains in this section.

Sattar et al. [1] investigated the use of visual communications, by using a gesture-based language using fiducials [4], and also looked at person-following by an underwater vehicle by tracking the swimming motion of a human "dive-buddy" [5]. By combining these two approaches, a vision-based human-robot interaction architecture has been developed [6]. In that work, while "closed-loop" robot control is achieved, an explicit dialog-based interface between the human and the robot did not exist. This paper is motivated partially from that work. Visual communication has also been used by several researchers for communication between a network of heterogeneous robotic systems, for example by Dunbabin et al. [7].

While our current work looks at interaction under uncertainty in any input modality, researchers have investigated uncertainty modeling in human-robot communication with specific input methods. For example, Pateras et al. applied fuzzy logic to reduce uncertainty to reduce high-level task descriptions into robot sensor-specific commands in a spoken-dialog HRI model [8]. Montemerlo et al. have investigated risk functions for safer navigation and environmental sampling for the Nursebot robotic nurse in the care of the elderly [9]. Bayesian risk estimates and active learning in POMDP formulations in a limited-interaction dialog model [10] and spoken language interaction models [11] have also been investigated in the past. Researchers have also applied planning cost models for efficient human–robot interaction tasks [12] [13]. Scheutz et al. [14] have looked at enhanced natural language dialogs for human-communication. In [15], the authors use information from natural language dialogs to update the robot task planner in real-time. Also, Chernova et al. [16] apply crowdsourcing to address real-world human-robot dialog through online multiplayer games.

In recent work, Tellex et. al [17] looks at asking directed questions to reduce uncertainty in human-robot dialogs, specifically for natural language communication. The authors use a graph-theoretical approach to ground tokens in the language to elements in the world to detect points of uncertainty. While somewhat similar to the problem we are addressing, our approach differs from their work in the cues being used to direct confirmations. In particular, their work looks at questions asked by a robot to reduce input language ambiguity, whereas in our case we use an evaluation of task risk (*i.e.*, cost) to direct specific questions to the user. Similarly in [18], the authors address the issue of assessing and informing non-expert users about the feasibility of the given natural language command. The particular concern in our work is to prevent the robot from executing potentially risky tasks without confirmation, yet still make it possible to do so if the user so desires. Specifically, we aim to prevent risky task execution without confirmation, whether the commanded task was intentional or not. While we do not use natural language directives in this work, such commands can be accommodated as long as a mechanism to produce possible alternates along with their likelihoods exists (*e.g.*, using the Google voice input API [19]).

III. Technical approach

The core of this work involves a dialog between a human and a robot, towards executing a set of tasks. The goal is to enable the robot to identify expensive commands and ask users to clarify them, in order to ensure that such commands are indeed intended by the user. To achieve this, we rely on generating a set of task execution plans from the most-likely programs chosen by the robot, and assess the cost of these plans by using a set of domain-specific *assessors*. Once the risk-uncertainty values are obtained, we choose the most likely of the programs, and verify if the task is high-cost. If so, the tokens in the input are analyzed by a cost-analysis component, looking at the world representation (in the form of global and local sensory information) available to the robot, and "grounding" tokens in the command to elements in this representation. This grounding step identifies potential high-cost commands by applying the assessors on the plan, pushing the 'suspect' tokens into a *clarification queue*. At the end of the cost analysis step, the robot asks the human operator for clarification for all tokens in the queue (*i.e.*, either to confirmation or modify). When all tokens from the queue are clarified by the user, the overall cost is calculated for the (potentially) modified program, and passed to the decision function. At this stage, based on the output of the decision function, the program is either carried out immediately, or a final confirmation is asked from the user. A schematic diagram of our technique can be seen in Figure 2.

Note that we make an important distinction between 'safety' and 'accuracy' in this approach. While the purpose

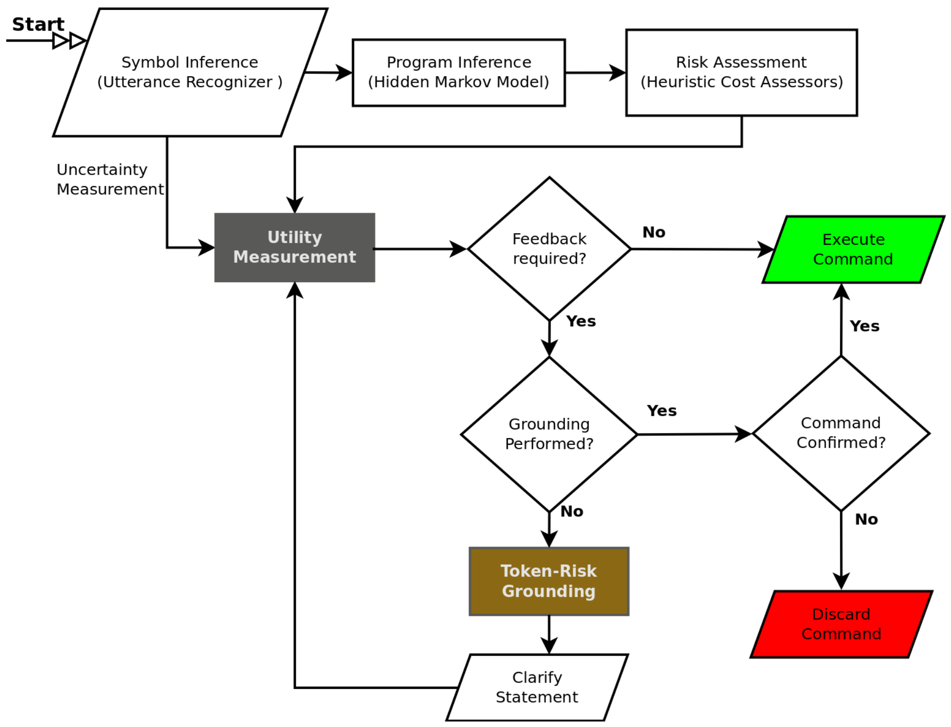

Fig. 2: Control flow in the proposed context-dependent dialog scheme.

of the proposed algorithm is to ensure costly tasks are executed only after a confirmation, it will in no way ensure that the program as intended by the user has been input. In other words, as long as the program is evaluated to be low-cost, the system will allow it to execute. Thus it is quite possible for an unintended but low-cost program to be executed, without feedback from the system. Also, the cost of the given task is estimated immediately after the program is input; currently, we do not perform any on-the-fly sensing to update this estimate once the robot begins task execution. The following sections take a detailed look at the various steps of this algorithm.

A. Language structure

The robot language used in this work is based on the RoboChat syntax. For the sake of brevity, we assume that a mechanism exists to only allow syntactically valid and semantically consistent commands to be sent to the robot. Details of the language structure can be found in [3], but for the sake of completeness a summary is given here.

1) A *program*, G, is described as consisting of a finite number of tokens, g_i.
2) A program must start with a *command* token, g_c, and is optionally followed by a *parameter*, g_p.
3) We assume each token is independently uttered by the operator, and thus the probability of a sequence of tokens chained together to form a "compound statement" simply becomes:

$$p(S) \equiv p(g_{c1}, [g_{p1}], g_{c2} \cdots g_{cn}[, g_{pn}]) = \prod_{i=1}^{n} p(g_i) \quad (1)$$

where $p(g_i)$ is the probability of the token g_i being used (*i.e.*, uttered) by the human operator, for both commands or parameters. Thus, if a table of values for all $p(g_i)$ is available, it is straightforward to compute the probability of a command (*i.e.*, a sequence of tokens).
4) Each program G_i has a likelihood of occurrence l_i and a cost c_i associated with it. It is worth noting however, given the input language, the set of all possible programs will be reduced, as the inconsistent ones, both syntactically and semantically, are going to be expunged.

B. Cost Analysis

Once the system has resolved the uncertainty and chosen a set of likely programs, the next step is to assess the cost in each of these programs. We approach the cost factor from two different perspectives; namely, the risk associated from the human operator's perspective, and the cost involved in terms of operational overhead of the robot while attempting to perform the task. Risk encompasses many factors, including domain–specific ones. In our case, the risk model reflects the difficulty of recovering the robot in the event of a total systems failure, in addition to the risk to the human operators. Risk to the robot is a particular concern for those operating outdoors, for examples in UAVs and underwater vehicles. Operational overhead includes robot operation costs over the duration of the task to be executed. The overhead measures are a function of factors such as power consumption, battery condition, system temperature, computational load, total distance traveled, etc. If f is the risk measurement function, and φ denote the overhead measurement function. Then, overall operational cost, C becomes,

$$C = f(\alpha_1, \alpha_2, \ldots, \alpha_n) + \varphi(\beta_1, \beta_2, \ldots, \beta_n) \qquad (2)$$

Given the mostly-likely input, the algorithm simulates the program and the cost of execution is evaluated by a set of assessors. After executing each individual token in the input, these assessors examine the current state of the robot and compute a *conservative* value of cost (*i.e.*, not less than the true cost). The final program cost is a sum of all the assessors' outputs over the duration of the simulated program.

C. Token-Risk grounding

Once a likely program has been chosen and is evaluated as high-cost, the system attempts to analyze the program to identify possible tokens that contributed to the overall cost of the system. Owing to the inherent structure of RoboChat, the components can contribute to the cost in one of two possible ways, given a (semantically and syntactically) valid command. In the first, a command token, irrespective of the parameter for the command, would cause the plan to take a potentially unsafe path, and thus increase the cost factor. Secondly, the command may not be evaluated as costly on its own (*i.e.*, independent of a parameter), but a misinterpretation of the parameter (caused either by a mistake by the programmer or uncertainty in the communication medium) would raise the execution cost significantly.

To illustrate the point, consider the following two program segments given to an unmanned aerial vehicle in RoboChat syntax:

1) `TAKEOFF, GEARUP, ALTITUDE 100, , RECORD_IMAGE_DATA, ALTITUDE` **2**`, ...`
2) **GEARUP**`, TAKEOFF, ALTITUDE 100, RECORD_IM AGE_DATA, ALTITUDE 200 , ...`

In the first program, the last `ALTITUDE` command is given a parameter of 2 meters. The system would evaluate the parameter value as too low an altitude for the aircraft to operate safely, and thus place a high-cost value at that token. On the other hand, in the second program, the plane is commanded to lift up its landing gear before it has taken off. Though no parameters are required, since lifting the landing gear before take-off would cause significant damage to the aircraft, the assessors would consider this command unsafe.

Once a program has been evaluated as high-cost, the system attempts to analyze the program to identify possible tokens that contributed to the overall cost of the system. The process can be summarized as follows: along with the total estimated cost, the system computes the expected cost contribution per input token. Then, for each token, the cost contribution for that particular token is set to zero, and overall cost is computed, keeping cost estimates for other tokens fixed. If the difference between this new cost estimate is greater than the mean cost estimate per token, it is placed in the clarification queue. The process continues until all tokens have been processed in this manner. At the end of the cost assessment step, the tokens from the clarification queue are extracted and clarification is requested from the user, as explained in the following section. Algorithm 1 presents the complete pseudo-code for this algorithm.

Algorithm 1 Token-risk grounding and dialog clarification.

```
1:  $\lambda \leftarrow$ Input Command
2:  if $\lambda \neq \phi$ then
3:      for all $t_j \in \lambda$ do
4:          $Risk_j \leftarrow ComputeTokenRisk(t_j)$
5:      end for
6:      $Risk_\lambda \leftarrow$ Assess($\lambda$)
7:      $Risk_{mean} \leftarrow ComputeMeanRisk(Risk_\lambda)$
8:      for all Tokens, $t_j \in \lambda$ do
9:          repeat
10:             $Risk_{temp} \leftarrow Risk_j$
11:             $Risk_j \leftarrow 0$
12:             $Risk_{new} \leftarrow$ Assess($\hat{\lambda}$)
13:             if $|Risk_{new} - Risk_\lambda| \geq Risk_{mean}$ then
14:                 $Push(t_j, Q_C)$
15:             end if
16:             $Risk_j \leftarrow Risk_{temp}$
17:         until Risk Assessment Complete
18:     end for
19:     if $Q_C \neq \phi$ then
20:         while $Q_C \neq \phi$ do
21:             $T \leftarrow Pop(Q_C)$
22:             $Clarify(T)$
23:         end while
24:     end if
25: end if
```

IV. Evaluation

To validate our approach and quantify the performance of the proposed algorithm, we conduct a set of experiments, both on-board and off-board. In the off-board experiments, we perform evaluations with a set of tasks for two simulated robots – an autonomous underwater vehicle and an indoor robot. In the on-board version, two robots, one PR2 and the other a TurtleBot, are instructed by a human operator to perform a set of navigation, manipulation and surveillance tasks (*i.e.*, record images and videos), given an annotated map of the robot's environment. In both cases, the input language chosen conforms to the RoboChat grammar. For both on-board and off-board experiments, we use a RoboChat vocabulary consisting of motion commands, surveillance (*i.e.*, recording photos and videos) and manipulation commands (see Tables I and II for a sample of language tokens used), with the `EXECUTE` command marking the end of input. In the off-board experiments, input modality is limited to mouse input, whereas the robot trials are performed using a ROS [20] module as a back-end to the mouse input system, to communicate commands over an IP network. The ROS navigation stack is used by the robots to localize and navigate in the environment.

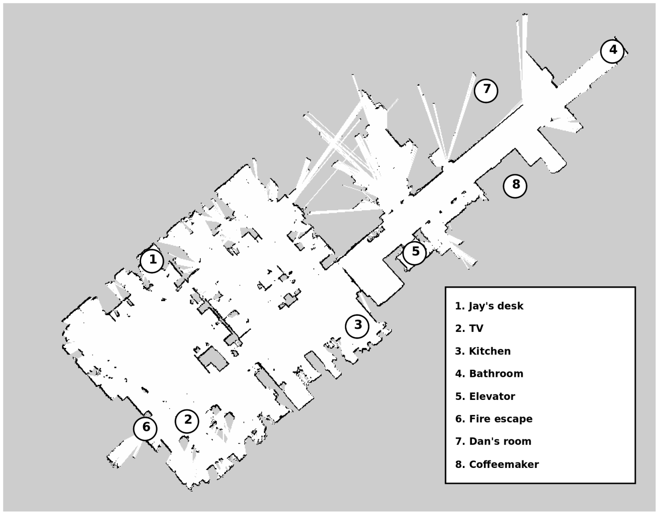

Fig. 3: Annotated map of the real indoor environment created by the PR2 which was used for the experiments.

To compute uncertainty in input for each of the simulated robots, a Hidden Markov Model is trained with estimated parameters for the given vocabulary. To estimate task costs, we use two custom simulation models for each type of robot. For the underwater vehicle, we use a set of assessors that takes into account the operating contexts of an autonomous underwater vehicle, and use the simulator from [3]. The simulator has been designed to take into account the robot's velocity, maneuverability and propulsion characteristics to accurately and realistically simulate trajectories taken by the robot. For the terrestrial robots, the simulator design is based upon the ROS simulation packages for the PR2 and the TurtleBot robots.

Since the goal of the off-board evaluations are to quantify the performance of the proposed algorithm in identifying potentially costly parts of a given command, we do not require the simulated robots to execute any tasks, but only perform token-risk grounding and ask clarifications if judged necessary. Results from both sets of experiments are presented in the following sections, preceded by a brief description of the input assessors and the actual tasks that were given.

A. Assessors

For both simulated and on-board robot trials for the indoor environment, we use the following assessors to evaluate cost:

1) **Total distance**: The operating cost and risk factors both increase with total distance traveled by the robot.
2) **Average Distance**: While the farthest and total distance metrics consider extremes in range and travel, respectively, the average distance looks at the distance of the robot (from start location) where most of the task execution time is spent.
3) **Execution Time**: An extremely long execution time also carries the overhead of elevated operational and external risk.
4) **Preconditions**: Certain functions need to have parameters initialized before they can be executed, and also depend on other functions having already executed.
5) **Navigability**: If the robot cannot safely navigate to a given location, the chance of damage to the robot from collision increases, as does the probability of task failure.

Also, for the simulated underwater robot, in place of the Navigability assessor, we use the following two additional assessors:

1) **Farthest distance**: The farther the robot goes from the initial position, the higher the chance of losing the robot.
2) **Depth**: The deeper the robot dives under the surface, the higher the chances are of exceeding maximum operational depth, thus irreparably damaging the vehicle. Also, certain operations, such as getting a GPS fix or bearing is not possible at depth, and thus would be flagged if attempted.

ID	Sequence	Risky?
1	FORWARD, 30, PICTURE, LEFT, 30, PICTURE, SURFACE, GPSFIX, GPSBEARING, EXECUTE	No
2	FORWARD, 180, LEFT, 20, FORWARD, 180, MOVIE, 300, RIGHT, 15, GPSFIX, SURFACE, STOP, EXECUTE	Yes
3	LEFT, 30, RIGHT, 10, MOVIE, 120, FOLLOW, 60, SURFACE, GPSFIX, EXECUTE	Yes

TABLE I: Commands sent to the simulated underwater robot.

ID	Sequence	Risky?
4	GOTO, JAYS_DESK, PICKUP, CUP, GOTO, KITCHEN, PUTDOWN, CUP, GOTO, FIRE_ESCAPE, TAKE_PICTURE, STOP, EXECUTE	No
5	GOTO, JAYS_DESK, PICKUP, BOOK, GOTO, DANS_ROOM, PUTDOWN, BOOK, GOTO, JAYS_DESK, EXECUTE	Yes
6	GOTO, COFFEEMAKER, PICKUP, MUG, GOTO, DANS_ROOM, PUTDOWN, MUG, PICKUP, DVD, GOTO, TV, PUTDOWN, DVD, GOTO, BATHROOM, EXECUTE	Yes

TABLE II: Commands sent to the simulated terrestrial robot.

B. Simulated tasks

The programs used for the simulated underwater robot are shown in Table I, along with the decision of confirmation. The numerical arguments are either distance measures (in meters, for the motion commands) or time (in seconds, for the non-motion commands, such as in `PICTURE`). Table II shows the programs sent to the indoor robot, operating in the annotated map shown in Figure 3.

C. Clarification Queries

Our focus on this paper is the identification of costly commands and specific high-cost components within them. While the output generated by the token-risk grounding algorithm can be used in an arbitrary fashion to provide detailed feedback and further planning, for the experiments, we limit the feedback query to take the following form:

$$\begin{aligned}\text{CLARIFICATION\_QUERY} \leftarrow \text{“Did you mean ”}\\ + \text{HIGH\_RISK\_TOKEN} + \text{“ ?”}\end{aligned} \quad (3)$$

ID	Assessors Flags	**Clarification Queue**
2	FARTHEST_DISTANCE, TOTAL_DISTANCE, DEPTH, PRECONDITION	180, 180, GPSFIX
3	PRECONDITION	FOLLOW
5	NAVIGABILITY	DANS_ROOM
6	TOTAL_DISTANCE, NAVIGABILITY	COFFEEMAKER, DANS_ROOM, BATHROOM

TABLE III: Contents of the clarification queue, and assessors that evaluated the tokens as high-cost.

where the + symbol denotes string concatenation. The users only have to either confirm the identified high-cost token, or to change it by editing that one token only.

D. Results & Discussion

In the simulated tasks from Tables I and II, the algorithm correctly evaluated commands $2, 3, 5$ and 6 as 'costly'. Commands 1 and 4 were evaluated as 'safe', and accordingly, no clarifications were requested for these programs. The contents of the clarification queue for the 'costly' programs are shown in Table III, along with the assessors that contributed to the high-cost evaluation. The user was given an option to change the tokens in commands in-place, if they so desired.

In command 2, the distances given to the simulated underwater robot was large, and would cause the robot to travel a great distance from the initial position, thereby increasing the chances of losing the robot. Thus the system requested clarification on the distances. Furthermore, the `GPSFIX` command was performed before the `SURFACE` command, causing the precondition assessor to put a high cost on this token, and placing it in the clarification queue.

In command 3, the `FOLLOW` instruction is given to visually track and follow a target. But since the tracker has not been initialized (in this case, with a `TUNETRACKER` command) the precondition assessor flags the instruction as high-cost.

In command 5, the location referred to as "DANS_ROOM" is not reachable by the robot, as no paths exist in the map (seen in Figure 3). A failure of the navigation task will occur if the task is executed, and thus the target location is marked for clarification.

In command 6, a navigability issue exists, similar to command 5, to reach the "COFFEEMAKER" location. Moreover, the overall distance traveled by the robot to navigate between the location is assessed to be too large, and thus a clarification is requested for the final location, "BATHROOM".

E. Timing

By asking for targeted feedback, it was possible to achieve gains in programming times. Specifically, there were three possible outcomes after a program was given:

1) *No feedback required*: in such cases, the overall programming time was unchanged.
2) *Feedback required, tokens unchanged*: in such cases, the overall programming time increased because of the additional clarification queries asked by the system, plus the time taken by the users to answer those queries.
3) *Feedback required, one or more token changed*: in such cases, we achieved a reduction in programming time. In particular, for long programs (*i.e.*, many tokens), the cost of re-programming without targeted queries is higher, as the user needs to input the entire sequence (plus the changes) again.

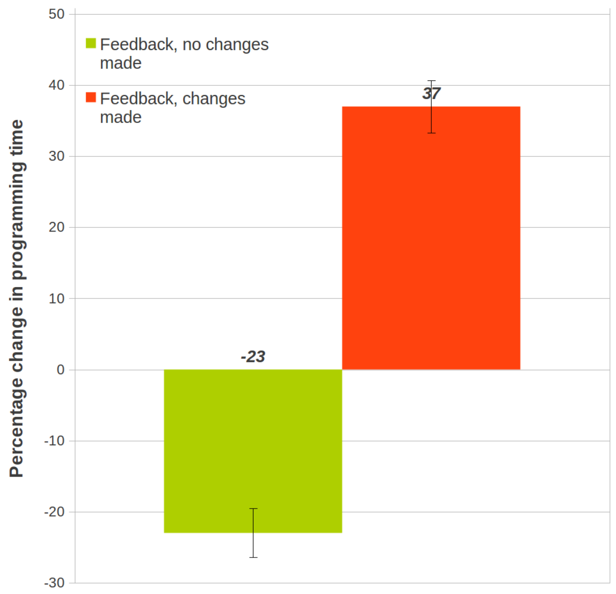

Fig. 4: Relative change in programming time with token-risk grounding. The zero-percent line is the baseline case without token-risk grounding.

To measure the changes in timing, we performed 20 trials each of the three different feedback cases discussed above, with and without token-risk grounding, resulting in 120 total experiments. Figure 4 shows the percentage changes in programming time obtained from these trials as compared to a system without token-risk grounding (*i.e.*, the baseline case obtained from exactly the same tasks with no token-risk grounding performed, but with overall cost computed). The results show approximately 40 *per cent* saving in programming time from a system without token-risk grounding when the user is using the feedback process to make changes to the original input program.

F. Robot trials

Two separate robot trials were performed to qualitatively assess the performance of the algorithm on-board a robotic system. Both robots, one a PR2 and the other a TurtleBot, were given a set the three commands comprised of navigation and surveillance tasks (see Table II) in the same environment (see Figure 3), and thus the map data was shared between the robots. Note that the map data was obtained prior to the experiments by using the ROS *gmapping* method using the PR2 robot. The manipulation tasks were manually performed on the PR2. Also, as the TurtleBot is not equipped with a manipulator, no manipulation tasks were performed on that robot. Input was given to the robots on a desktop PC, using a mouse gestures-driven interface. through a ROS module developed using ROS messaging stack. A gesture- or speech-based interface would be equally applicable, but since the focus of our work was on the problem of asking targeted questions, we chose an arbitrary input modality. The only difference in using another modality would lie in uncertainty evaluation.

Clarification of the tokens were asked of the user over the same module, on the a desktop PC. The user had an option of either repeating the same command/parameter, or entering a new token in place of the previous one. After the clarification stage was complete, the robot either asked a final task confirmation (in case of a high-risk task) or executed the given command. The trial performances were similar to the simulation runs, with the difference that the robots actually carried out the task once they were confirmed by the operator.

V. Conclusions

We present an algorithm for a robotic system to identify and ask confirmations for high-cost instructions to the human operator and improve safety in task execution. The system assesses uncertainty and cost in human-robot dialog, and finds 'costly' language tokens from the input command by grounding evaluated cost with language tokens. Specifically, we associate tokens in the input command with actions in the world representation of the robot to detect tokens that cause a significant increase in overall cost. The system has been evaluated with a set of simulated tasks for an indoor robot, and also evaluated on-board the PR2 and Turtlebot platforms.

As an immediate next step, we are planning large-scale human interface trials on real and simulated robots (and environments), and measure the impact of this approach in terms of time savings, ease of communication, and feedback relevance. Future work is planned to enhance the grounding of the task set to both actions and objects in the real-world representation of a robot. We intend to investigate the effects of modeling cost as a distribution, dynamically updating cost assessment using on-the-fly sensing, and thus updating the token-risk grounding mid-task. Also of interest is a shared estimation of cost-grounding, where a network of mobile robots and static sensors, including portable devices (*e.g.*, smartphones) build and share a representation of the world, and assist in distributed and coordinated human-machine interaction. Particularly in the domain of assistive robotics (*e.g.*, autonomous wheelchairs for the mobility impaired), a shared HRI network with a strong grounding of risk to objects and actions would help to establish a safe and secure domain of operations, yet maintain a natural, intuitive modality in the human-machine interface.

References

[1] G. Dudek, J. Sattar, and A. Xu, "A Visual Language for Robot Control and Programming: A Human-Interface Study," in *Proceedings of the International Conference on Robotics and Automation ICRA*, Rome, Italy, 4 2007, pp. 2507–2513.

[2] A. Xu, G. Dudek, and J. Sattar, "A Natural Gesture Interface for Operating Robotic Systems," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Pasadena, California, 5 2008, pp. 3557–3563.

[3] J. Sattar and G. Dudek, "Towards Quantitative Modeling of Task Confirmations in Human–Robot Dialog," in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA*, Shanghai, China, May 2011, pp. 1957–1963.

[4] M. Fiala, "ARTag, a Fiducial Marker System Using Digital Techniques," in *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2*. Washington, DC, USA: IEEE Computer Society, 2005, pp. 590–596.

[5] J. Sattar and G. Dudek, "Underwater Human-Robot Interaction via Biological Motion Identification," in *Proceedings of the International Conference on Robotics: Science and Systems V, RSS*. Seattle, Washington, USA: MIT Press, 6 2009, pp. 185–192.

[6] ——, "A Vision-based Control and Interaction Framework for a Legged Underwater Robot," in *Proceedings of the Sixth Canadian Conference on Robot Vision (CRV)*, Kelowna, British Columbia, 5 2009, pp. 329–336.

[7] M. Dunbabin, I. Vasilescu, P. Corke, and D. Rus, "Data Muling over Underwater Wireless Sensor Networks using an Autonomous Underwater Vehicle," in *International Conference on Robotics and Automation, ICRA 2006*, Orlando, Florida, May 2006.

[8] C. Pateras, G. Dudek, and R. D. Mori, "Understanding Referring Expressions in a Person-Machine Spoken Dialogue," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, 1995. (ICASSP '95)*, vol. 1, May 1995, pp. 197–200.

[9] M. Montemerlo, J. Pineau, N. Roy, S. Thrun, and V. Verma, "Experiences with a Mobile Robotic Guide for the Elderly," in *Proceedings of the 18th National Conference on Artificial Intelligence AAAI*, 2002, pp. 587–592.

[10] F. Doshi, J. Pineau, and N. Roy, "Reinforcement Learning with Limited Reinforcement: Using Bayes risk for Active Learning in POMDPs," in *Proceedings of the 25th international conference on Machine learning*. ACM New York, NY, USA, 2008, pp. 256–263.

[11] F. Doshi and N. Roy, "Spoken Language Interaction with Model Uncertainty: An Adaptive Human-Robot Interaction System," *Connection Science*, vol. 20, no. 4, pp. 299–318, 2008.

[12] K. Krebsbach, D. Olawsky, and M. Gini, "An Empirical Study of Sensing and Defaulting in Planning," in *Artificial intelligence planning systems: proceedings of the first international conference, June 15-17, 1992, College Park, Maryland*. Morgan Kaufmann Publishers, 1992, p. 136.

[13] D. Kulic and E. Croft, "Safe Planning for Human-Robot Interaction," in *Proceedings of the 2004 IEEE International Conference on Robotics and Automation (ICRA)*, vol. 2, 2004.

[14] M. Scheutz, R. Cantrell, and P. Schermerhorn, "Toward Humanlike Task-Based Dialogue Processing for Human Robot Interaction," *AI Magazine*, vol. 32, no. 4, pp. 77–84, 2011.

[15] 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 *Proceedings of the 2012 Human-Robot Interaction Conference*, Boston, MA, March 2012.

[16] S. Chernova, N. DePalma, and C. Breazeal, "Crowdsourcing Real World Human-Robot Dialog and Teamwork through Online Multiplayer Games," *AI Magazine*, vol. 32, no. 4, pp. 100–111, 2011.

[17] S. Tellex, P. Thaker, R. Deits, D. Simeonov, T. Kollar, and N. Roy, "Toward Information Theoretic Human-Robot Dialog," in *Proceedings of Robotics: Science and Systems VII*, Sydney, New South Wales, Australia, June 2012.

[18] V. Raman, C. Lignos, C. Finucane, K. C. T. Lee, M. Marcus, and H. Kress-Gazit, "Sorry Dave, I'm Afraid I Can't Do That: Explaining Unachievable Robot Tasks Using Natural Language," in *Proceedings of Robotics: Science and Systems*, Berlin, Germany, June 2013.

[19] "Google Text to Speech API," Developer Reference.

[20] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: An Open-Source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.