Yale University

CPSC 490: Senior Project

Computer Science

Accounting for User Preference During a Human-Robot Collaborative Task

---

Student:
Bhavani Ananthabhotla

Project Advisors:
Corina Grigore and Prof. Brian Scassellati

Semester-end Report, May 3, 2018

Contents

1 Abstract

Strides have been made in the production of powerful industrial robotic systems for the completion of tasks that are uncomfortable, tedious and/or pose risk for humans. However, the prospect of capitalizing on both the strengths of a robot to perform precise tasks and on the expertise of a human provides significant motivation to improve systems that interact with humans in the completion of a joint task. The study of such systems is known as human-robot collaboration (HRC) and this project contributes to an ongoing study in Yale University's Social Robotics Laboratory that aims to design human-robot collaborative systems that pro-actively provide supportive behaviors to a human worker during a generalizable assembly task. The purpose of this project was to demonstrate the delivery of supportive actions by a robot to a human worker, but which accounted for variability in workers' preferences for which supportive actions would be required at various points along the progression of the work. A method of doing so, keeping in mind the desire to minimize disturbance to the human worker and the desire to have a simple and robust model, was outlined using active learning and demonstrated in simulation and using the Baxter robot.

2 Introduction

Strides have been made in the production of powerful industrial robotic systems for the completion of tasks that are uncomfortable, tedious and/or pose risk for humans, such as autonomous robotics for transmission lines-inspection [1], or the scope of commercial robotics for construction applications [2]. In such situations, robotics serve well in the completion of precise, repetitive tasks isolated from humans.

However, there is significant motivation to improve systems that interact with humans in the completion of a task. From better studies of safe operation to improvements on the versatility of robots to interact with less structured environments, recent years have seen research improvements in perception and control that would allow robotic systems to play significant roles in far less isolated environments in the near future [3, 4, 5]. Thus, the field of human-robot interaction is dedicated to the construction, study, and improvement of these integrated systems, investigating challenges unique to the introduction of robotics in human settings.

One might easily imagine a use-case that capitalizes on the strengths of the two types of agents in a human-robot system to aid in the joint completion of a task. The study of such systems is known as human-robot collaboration (HRC), and amongst the aforementioned challenges specific to this sub-field are the design decisions made in order for such interactions to be both productive and meaningful in real-world settings, where a human teammate's actions are not always predictable.

2.1 Purpose

This project contributes to an ongoing study in Yale University's Social Robotics Laboratory that aims to design human-robot collaborative systems that pro-actively provide supportive behaviors to a human worker during a generalizable assembly task. The purpose of my project was to demonstrate the delivery of supportive actions by a robot to a human worker during the completition of a joint task which accounted for variability in workers' preferences over which supportive actions would be required at various points along the progression of the work.

3 Design

3.1 Joint-Task Definition

The case task used the HRC Model Set [6], and was therefore both complex and reasonably similar to a manufacturing assembly task. The task used for this study, the model chair assembly task, is defined below using a hierachal task model.

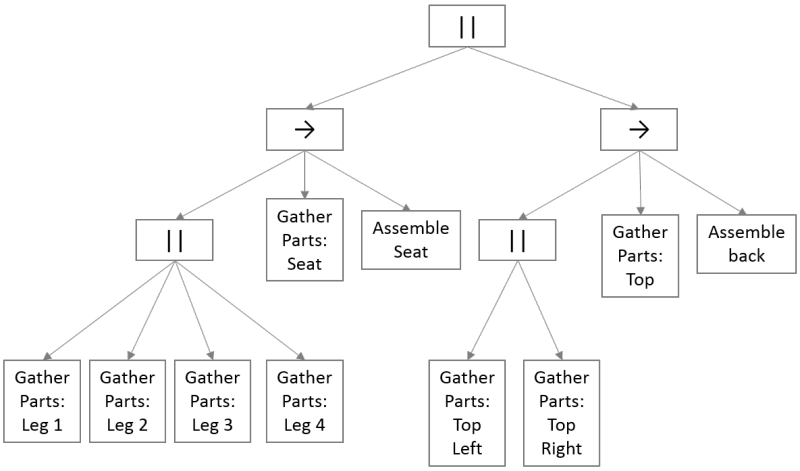

Figure 1: HTM definition of the joint-task.

A general description of each HTM-defined step in the task are defined below:

HTM Leaf Label	Description
Gather Parts: Leg1	The human uses a dowel and a front bracket
Gather Parts: Leg2	The human uses a dowel and a front bracket
Gather Parts: Leg3	The human uses a dowel and a back bracket
Gather Parts: Leg4	The human uses a dowel and a back bracket
Gather Parts: Seat	The human uses Seat
Assemble Seat	The human might need help holding the assembly
Gather Parts: Top Left	The human uses some of {top bracket, dowel, long dowel}
Gather Parts: Top Right	The human uses some of {top bracket, dowel, long dowel}
Gather Parts: Top	The human uses Back and possible some of {top bracket, dowel, long dowel}
Assemble Back	The human might need help holding the assembly

Figure 2: Descriptions of HTM-defined task steps.

The robot was defined to have the following capabilities to aid the human in the completion of the chair assembly task:

Supportive Action	Description
bring dowel	bring a short dowel to the workspace, get feedback for whether the human accepted and take back if not
bring long dowel	bring a long dowel to the workspace, get feedback for whether the human accepted and take back if not
bring screwdriver	bring the screwdriver to the workspace, get feedback for whether the human accepted and take back if not
bring top bracket	bring top bracket to the workspace, get feedback for whether the human accepted and take back if not
bring back bracket	bring back bracket to the workspace, get feedback for whether the human accepted and take back if not
bring front bracket	bring front bracket to the workspace, get feedback for whether the human accepted and take back if not
hold	allow for a the human to use the robot to hold a part already in the workspace, get feedback for whether the human accepted the help
seat	bring seat to the workspace, get feedback for whether the human accepted and take back if not
back	bring seat to the workspace, get feedback for whether the human accepted and take back if not

Figure 3: Descriptions of possible supportive behaviors the robot should be able to provide.

The robot's knowledge of the environment was based on the human-given feedback for delivered supportive behaviors.

3.2 Design Decisions

The primary aim in designing this robot interaction was to reasonably minimize the disturbance to the expert worker's practice. A secondary goal was to have a reasonably simple and robust model for delivering supportive behavior. The following design was conceived with these aims.

First, while a hierarchal task model is a convenient way to organize a task into discrete steps, I didn't want to place the burden of labeling abstract 'task states' with desired supportive behaviors. The increased cognitive load seemed to conflict with the first goal, and the need to provide the robotic system with any model of the task seemed to conflict with the second goal. Instead, an assumption was made: that tasks this system will be applied to had some sort of structural similarity across users that could be taken advantage of.

Therein, rather than training a new model per user by having them provide labels to a high-level task model, it was thought that three workers (refered to here as "robot-trainers") could interact once-each with the robot. For each interaction, in sequence as the robot-trainer worked, the robot would query the robot-trainer for the next set of supportive behaviors desired. An example of one training session is represented below:

(timestep, environment features)	supportive action labels
0, ()	('top_bracket', 'dowel', 'screwdriver')
1, ('dowel_1taken', 'screwdriver_taken', 'topb_1taken')	('dowel', 'top_bracket')
2, ('dowel_2taken', 'screwdriver_taken', 'topb_2taken')	('long_dowel',)
3, ('dowel_2taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('hold',)
4, ('dowel_2taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('dowel', 'back_bracket')
5, ('backb_1taken', 'dowel_3taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('dowel', 'front_bracket')
6, ('backb_1taken', 'dowel_4taken', 'frontb_1taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('dowel', 'back_bracket')
7, ('backb_2taken', 'dowel_5taken', 'frontb_1taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('dowel', 'front_bracket')
8, ('backb_2taken', 'dowel_6taken', 'frontb_2taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'topb_2taken')	('seat',)
9, ('backb_2taken', 'dowel_6taken', 'frontb_2taken', 'hold_taken', 'longdowel_taken', 'screwdriver_taken', 'seat_taken', 'topb_2taken')	()

Figure 4: Simulated labels provided by a robot-trainer during their interaction with the robot. Environment features are abbreviated with the count of objects in the workspace that fit a title. Supportive actions are abbreviated as the name of the object to bring, and 'hold' for holding supportive behavior.

The data gathered from these three workers would go into a model like the following, where at each (timestep, environment features) tuple, possible lists of supportive actions would be given a vote by each of the three robot-trainers. As one can imagine, certain (timestep, environment features) keys would have agreement on supportive actions, while others would have discord, while there would also likely be differences in the exact (timestep, environment features) keys present in each robot-trainer's model.

(timestep, environment features)	Supportive Actions Label Votes
(0, ())	Counter({('dowel', 'back_bracket', 'screwdriver'): 1, ('dowel', 'top_bracket', 'screwdriver'): 1, ('front_bracket', 'dowel', 'screwdriver'): 1})
(1, ('backb_1taken', 'dowel_1taken', 'screwdriver_taken'))	Counter({('dowel', 'front_bracket'): 1})
(1, ('dowel_1taken', 'frontb_1taken', 'screwdriver_taken'))	Counter({('back_bracket', 'dowel'): 1})
(1, ('dowel_1taken', 'screwdriver_taken', 'topb_1taken'))	Counter({('dowel', 'top_bracket'): 1})
(2, ('backb_1taken', 'dowel_2taken', 'frontb_1taken', 'screwdriver_taken'))	Counter({('back_bracket', 'dowel'): 1, ('dowel', 'front_bracket'): 1})
(2, ('dowel_1taken', 'dowel_2taken', 'screwdriver_taken', 'topb_2taken'))	Counter({('long_dowel',): 1})
...	...

Figure 5: First few entries of a simulated default model built by aggregated data from three robot-trainers to be used as a starting place with each new user.

The above aggregated model for the robot-trainers would become the default model. It was thought that using this as a starting point for delivering supportive actions to any new user, in conjunction to querying the user explicitly for their desired supportive behaviors whenever "uncertain" at a given (timestep, environment features) would be a good way to support adaptability to user preferences while not having to receive all supportive behavior labels from each user and then learn over hundred of users. In a robust way, the model is updated for each user over each interaction with him or her using the following algorithm:

```
main(): #for one interaction with a new worker, given a trained default model
    for timestep in range(len(task)): #10 for the chair assembly task
        if should_query() given state_of_the_world:
            user_resp = query()
            supp_acts = user_resp
            proactive queries ++
        else:
            supp_acts = prediction() given state_of_the_world

        fail_check = 0
        successful_actions = []

        while supp_acts is not empty:
            feedback = do_action(supp_acts.next)
            if success:
                succesful_actions += supp_acts.next
                supp_acts.pop, supp_acts.next()
                fail_check = 0
            elif wrong
                user_resp = query()
                supp_acts = user_resp
                supp_acts.next()
                incorrect acts ++
                fail_check = 0
            else (mech fail):
                fail_check ++
                if fail_check < 90
                    continue
                else _stop(), raise Exception('Robot Failure')

should_query(): if for the (timestep, environment feature) there is no majority
vote for a particular list of supportive actions, query

prediction(): return the list of supportive actions that got the most votes (
only gets called if should_query is false)
```

Figure 6: Pseudocode for how the model updates for an interaction with a new user.

More generally, this method of querying for labels for features the model is least sure about, most useful when labeled training data is limited, is called active learning.

3.3 Implementation

HTM

The HTM class was only used to generated simulated user prefence label data. HTMs were defined to be trees, where inner nodes were 'parallel' or 'sequential' directives for the children, and the leaves are steps in the task.

Chair Assembly Task

This includes a class UserPrefEnvDemo which represents the state of the environment which can be updated given past successful actions. This file also includes sim user labels() which produces labels for given (timestep, environment features) based on some random variables and some reasonable heuristics.

Model Framework

This contains the UserPrefModel class that contains a dict of (timestep, environment features) keys and Counter objects keeping track of votes for supportive action lists.

Other

Utils.py contains an implementation of a prettyprint function I wrote useful for debugging.

4 Results and Discussion

Two metrics were recorded over each interaction with the robot. Pro-active queries is defined to be the number of queries raised without starting an action sequence at a timestep. Incorrect action queries is defined to be the number of queries made in response to negative feedback for an attempted supportive action.

4.1 Simulation

The results for 10 simulated workers over six simulated interactions with the robot where the workers' preferences were consistent across their own interactions with the robot are summarized below. Both the number of proactive queries (where the maximum possible number per interaction is 10) and the number of incorrect action queries decreased and converged to 0 over the six interactions.

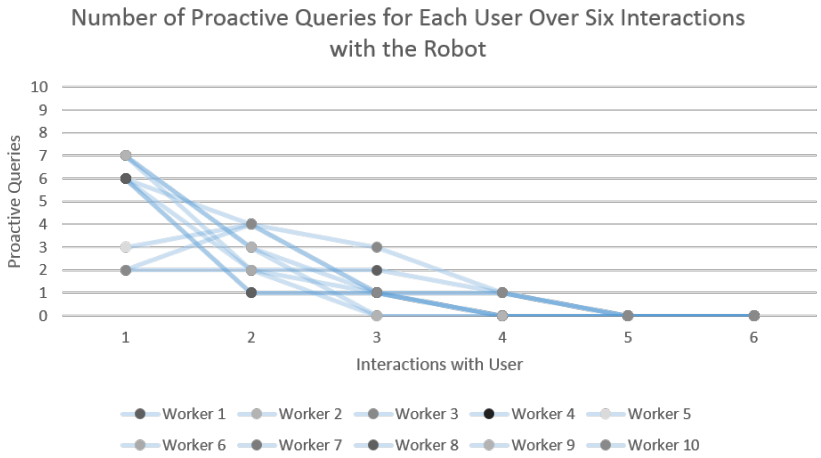

Figure 7: Proactive queries for each simulated user over six interactions with the robot. (Maximum possible queries per interaction= 10)

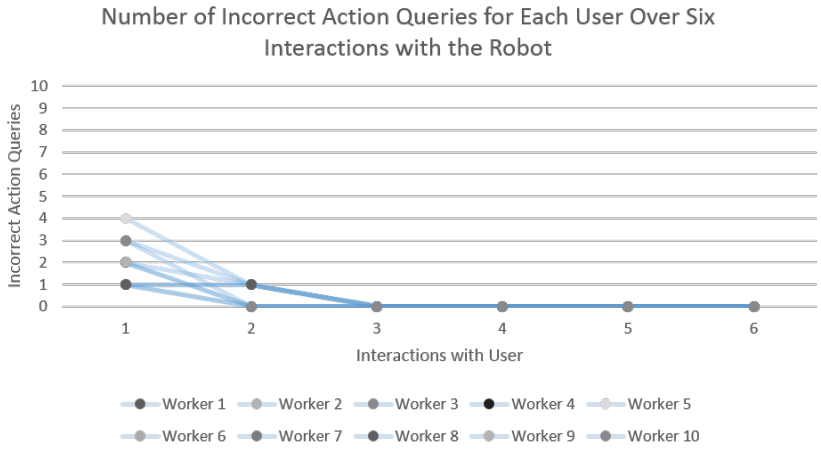

Figure 8: Incorrect action queries for each simulated user over six interactions with the robot.

4.2 Robot Demo

The results for a trial with the robot over for one user over 6 interactions is summarized below. Here also, a set of user preferences (that happened to not be the same as any of the three simulated robot-trainer's preferences) was adhered to, but for one change over the first two interactions (the taskstep for which the screwdriver was asked for changed

during the second interaction). Both the number of proactive queries (where the maximum possible number per interaction is 10) and the number of incorrect action queries decreased and converged to 0 over the six interactions.

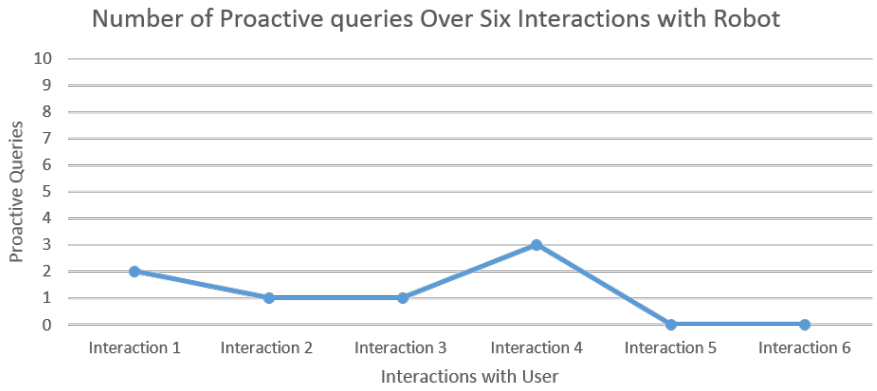

Figure 9: Proactive queries to the user over six interactions with the robot. (Maximum possible queries per interaction= 10)

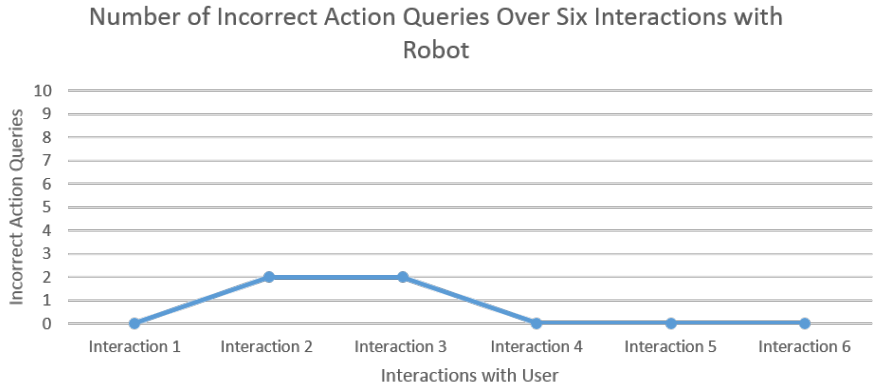

Figure 10: Incorrect action queries to the user over six interactions with the robot.

4.3 Discussion

The average number of proactive queries per interaction with the robot over all simulated interactions was for "the six-day week" was 1.55. The robot demonstration seemed to follow this result, even with slightly varied labels between interactions.

However, there are a few limitations to this simple model that need to be addressed. Primarily, the definition of a task-phase or task-step for which a set of supportive behaviors need to be mapped to has the ability to be quite fluid. While the model is robust in a way to handle such cases (there would just be many queries made), (1)the usefulness of the default model definitely depends on the task and (2) this may place unnecessary cognitive load on the human worker. While one option is to update the UserPrefModel following every supportive action delivered/every update to the state of the environment, rather than an arbitrarily defined task phase, I would argue that this would significantly worsen the user experience. The ability to specify multiple next supportive actions desired, but in small groups, strikes a balance between having too many queries and complicating the interaction.

The model, while designed to be simple, could be improved significantly to reduce query numbers without reducing the generality of the system. Increasing the number and type of features to label would definitely improve the model, whether using perception to gain more knowledge of the environment, or using previous timesteps' labels of a particular new user to predict future labels on-line. These are definitely aspects of this project I would be interested in seeing completed in the future.

5 Conclusion

In conclusion, this project worked to deliver supportive actions to a human worker during a chair assembly task that adapted to the user's preferences over time. A method of doing so, keeping in mind the desire to minimize disturbance to the human worker and the desire to have a simple and robust model, was outlined using active learning and demonstrated in simulation and using the Baxter robot. A link to the video of the robot simulation can be found at the index page for this project on the DUS's website.

6 Acknowledgements

I'd like to thank Professor Scassellati for allowing me the opportunity to work in the Social Robotics Laboratory towards the completion of this project. I'd also sincerely like to thank Corina Grigore for all her help and guidance through this work, from regular meetings to debugging at the very end. I learnt alot from everyone in the lab and I am so grateful to have had this experience to wrap my final year. Thank you.

References

[1] Limall, Eduardo Jos, et al. POLIBOT POwer Lines Inspection RoBOT. Industrial Robot: An International Journal, vol. 45, no. 1, 2018, pp. 98109., doi:10.1108/ir-08-2016-0217.

[2] Bogue, Robert. What are the prospects for robots in the construction industry? Industrial Robot: An International Journal, vol. 45, no. 1, 2018, pp. 16., doi:10.1108/ir-11-2017-0194.

[3] Pratt, Jerry, et al. Series elastic actuators for high fidelity force control. Industrial Robot: An International Journal, vol. 29, no. 3, 2002, pp. 234241., doi:10.1108/01439910210425522.

[4] Lindner, Lars, et al. Mobile robot vision system using continuous laser scanning for industrial application. Industrial Robot: An International Journal, vol. 43, no. 4, 2016, pp. 360369., doi:10.1108/ir-01-2016-0048.

[5] Tegin, Johan, and Jan Wikander. Tactile sensing in intelligent robotic manipulation a review. Industrial Robot: An International Journal, vol. 32, no. 1, 2005, pp. 6470., doi:10.1108/01439910510573318.

[6] Zeylikman, Sofya, et al. The HRC Model Set for Human-Robot Collaboration Research. CoRR, abs/1710.11211, Nov. 2017.

[7] Bogue, Robert. Exoskeletons and robotic prosthetics: a review of recent developments. Industrial Robot: An International Journal, vol. 36, no. 5, 2009, pp. 421427., doi:10.1108/01439910910980141.

[8] Brigham, Anderson, and Andrew Moore. Active Learning for Hidden Markov Models: Objective Functions and Algorithms. Proceedings of the 22nd International Conference on Machine Learning, 2005.

[9] Ha, Do M, et al. RiSH: A robot-Integrated smart home for elderly care. Robotics and Autonomous Systems, vol. 101, Mar. 2018, pp. 7492., doi:https://doi.org/10.1016/j.robot.2017.12.008.

[10] Pereira-Santos, Davi, et al. Empirical investigation of active learning strategies. Neurocomputing, 2017.

[11] Predicting Supportive Behaviors based on User Preferences for Human-Robot Collaboration. International Conference on Autonomous Agents and Multiagent Systems (AAMAS), 2018.

[12] Settles, Burr. Active Learning. Synthesis Lectures on Artificial Intelligence and Machine Learning, vol. 6, no. 1, 2012, pp. 1114., doi:10.2200/s00429ed1v01y201207aim018.