Maximizing Agent Utility in Human-Robot Collaborative Assembly Tasks

Kenneth Shaw, Jaskaran Grover, Changliu Liu

***Abstract*—Human-robot collaboration (HRC) is increasingly important in assembly line tasks as humans and robots have varied strengths that can cooperate safely together to increase efficiency. However, due to the time-varying nature of human collaborators, it is challenging for the robot to efficiently identify the ideal response to the human plan. Currently, most solutions use the trajectory of the human arm as the primary detection method for the human's underlying plan and respond with one appropriate robot action to maximize the utility given the situation. Neither the human nor the robot is expected to provide further help if the task is proving difficult for the other agent, therefore not maximizing the utility of each agent. In this paper, we will improve upon this by grouping robot tasks with the same underlying goal together, allowing the robot to switch between them to maximize the utility of each agent.**

I. Introduction

In factories and many other industrial settings, robots have proven to be efficient at many repetitive tasks. That being said there are also many cases such as small assembly where robots aren't capable of completing the task as effectively. Human-robot collaboration has shown much promise in areas such as manufacturing and has caught the continued interest of academia and industry alike to fill this gap. In a collaborative human-robot environment, physical safety is combined with software checks to ensure comprehensive, guaranteed safety. Software also facilitates the tasks allocated to each agent, robot, and human. This combination can allow robots and humans to work in close proximity together to complete the task, using their complementary strengths most efficiently. This paper examines the allocation strategy of these tasks and proposes one such method.

This allocation strategy must find the proper robot action to match that of the human's that maximizes the utility of the human and the robot. For example, if one were placing a window seal on a car with an assistive robot, there are many different actions the robot and the human could take with various success rates, durations and strain on the human. The robot could attempt to do it fully autonomously but possibly run into issues of alignment and placement. The human could assist the robot through the path, helping it place the door seal. Additionally, the human could complete the whole action by itself. This places a large strain on the human but has a higher guaranteed success rate. There are existing various strategies for this task allocation problem that will be explained below.

II. Related Work

Much work has been previously done on the task allocation problem with a variety of applications. In human-robot collaboration, one common way is by using advanced planners to decide which task to match to the human based on the status of the objects and obstacles in the environment. One such implementation is SHARY, which uses a hierarchy of shared agent tasks and communication mechanisms between the two agents. It uses a human aware task planner that incorporates the feasibility of the task and human load to decide on the correct task to assign. It also maintains a list of suspended and currently active tasks to more efficiently manage the next upcoming tasks for the human-robot team [4].

Additionally, the Berkeley MSC lab has specified human-robot tasks as part of a hierarchy of sub-tasks that can be completed sequentially together to fulfill one larger task. This hierarchy enables the robot to better predict the next action and trajectory of the human which therefore enables the robot to predict and respond earlier to the status of the task [3].

Theory of Mind paradigms use cues from the human, environment, and goals to better estimate the mental state of the human [11]. For instance, eye gaze has already shown promise in finding a target object that a person is looking at [2]. Additionally, inverse planning looks to understand the human's goals based on their abilities in perception, in this case using a supervisor to keep track of the human's understanding and facilitates communication between the agents [5].

In these previously explained human-robot collaborative environments, the robot is responsive to the action of the human and meets it with a singular action that is programmed offline to help the human achieve the underlying intention, where the intention is the underlying goal that would be achieved by conducting such a task. However, when examining human-human teams, this is largely not true. For instance, when human-human teams work together on intentions such as lifting heavy objects, each human monitors the success of the other one and adapts to help and improve the success of the whole team. When playing a game such as soccer, players are constantly monitoring the success of others to improve their own actions and the success of the team. The focus of this paper will be to model this in human-robot collaboration.

In this new human-robot collaborative environment, when a human is struggling on a task the robot will perform further actions to help assist the human. This can better efficiently utilize the resources of the robot as well as the human. In this case, there is an intention or underlying mission that the human must complete. One intention can then have many different tasks that will complete this same end result. The human will provide the initial intention with its first task, but then the robot will decide continued shared tasks to minimize the expected completion time of the intention and maximize the resources available to it.

To ensure safety and allow for trajectory and task functionality, SERoCS (Safe and Efficient Robot Collaboration System) will be used as the underlying basis for the experiments. The robust cognition algorithms monitor the environment and track the human using a Kinect camera by using joint positions and then infer a plan of the human. Once that is understood, an efficient task planning algorithm based on the CFS algorithm dictates the robot plan the long term trajectory. Finally, a safe motion planning and control algorithm for safe human-robot interaction ensures that the long term goal and safety constraints are met. This framework addresses the broad challenges in a human-robot collaborative environment [7].

The structure of the rest of the paper is as follows. In Section III, the mathematical framework of robot task allocation is proposed. In Section IV, the SERoCS framework modifications are discussed. Section V concludes the paper.

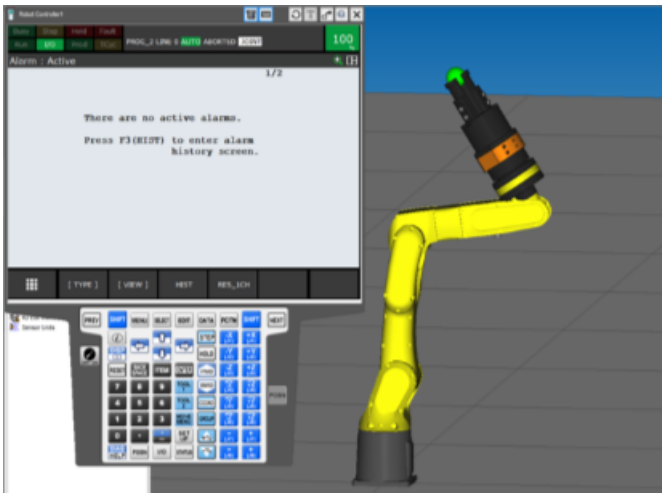


Fig. 1: RoboGuide FANUC Robot Simulator

III. Problem Framework

In a typical human-robot collaborative environment, the robot's task allocation strategy must decide which task to conduct to help the human. This task should attempt to maximize the utility, or usefulness of each agent at completing the task. In this paper, the task that the human is trying to complete is tied closely with the trajectory or action that the human is conducting. Each task, however, will now have an underlying intention. This intention is the desired goal state of the target objects at the end of the successful completion of the tasks. The robot will try to switch tasks to complete the intention faster.

Each intention will have a group of synonymous tasks that can be conducted to achieve the same end intention. As an example, placing a car seal onto a door can be done in three ways, (1) by placing the seal completely automatically with the robot, (2) by having the human help align the robot and door together to complete the task or (3) completely by human action alone. Each of these n tasks within the intention will be defined as a tuple $< P_n, T_n, K_n >$ where P is the probability of success, T is the task time completion distribution, and K is an empirical understanding of the strain and unpleasantness on the human of the action and the process of switching actions. In the same example as above, the completely autonomous procedure might have a fast completion time and little strain on the human, but the success rate could be low. The completely human operated procedure would have a greater amount of human strain but also a higher completion rate and consistent completion time.

Focusing on the task completion time aspect, independent trials can be conducted to gather data and understand this random variable. From prior industrial engineering research, task completion time of tasks such as factory assembly tasks can typically be modelled and fitted to a Weibull distribution [9]. The Weibull distribution is a right skewed distribution, where most tasks will likely be completed earlier rather than later and is defined as follows where $k > 0$ is the shape parameter and $\lambda > 0$ is the scale parameter of the probability density function:

$$f(x;\lambda,k) = \begin{cases} \frac{k}{\lambda}\left(\frac{x}{\lambda}\right)^{k-1} e^{-(x/\lambda)^k} & x \geq 0 \\ 0 & x < 0 \end{cases}$$

Once the completion time is fit to this distribution, the distribution's cumulative distribution function is useful to find the area under the curve or the probability of completing the task within a certain amount of time. The CDF is as follows[6]:

$$1 - e^{-(x/\lambda)^k} \quad x \geq 0$$

However, the completion time is only one part of the characteristics that define the cost of the task. Each task has a total success rate over a long period of time as well as a cost of the human strain of conducting and switching to the task. Therefore the total cost can be defined as follows, where there are l task pairs, p_n is the probability of reaching this point, x_n is the duration, and c_n is the cost function of switching tasks.

$$\min : \sum_{n=0}^{l} p_n * x_n * c_n \tag{1}$$

The probability of reaching this point or p_n is the sum of the probability of not completing the previous trials.

$$p_n = 1 - \sum_{m=0}^{n-1} (1 - e^{-(x_m/\lambda_m)^{k_m}}) \tag{2}$$

Combining these we obtain:

$$\min : \sum_{n=0}^{l} (1 - \sum_{m=0}^{n-1} (1 - e^{-(x_m/\lambda_m)^{k_m}}) * x_n * c_n) \tag{3}$$

Therefore, the goal of the robot will be to minimize intention cost or the above function which consequently minimizes the strain on the human as well as task completion time but also guaranteeing task completion, assuming the final $p_l = 0$. It will do this by finding the correct time to switch tasks, or finding the proper vector of x_n to pursue the tasks for. Intuitively, once the success rate of the current task becomes low over time (e.g., the robot is trying to put a window seal on a car door and is failing) then the actions of the robot will switch to a task that will be more likely to be completed.

To organize the relationship between these task tuples, a sequence structure can be proposed. Continuing on the previous example, it is likely that the robot should try to completely autonomously add the door seal to the car first, but then when time dictates it's failing and getting stuck, it should then increasingly allow the human to intervene and complete the action together. Like in this example, the general structure of the net can be understood solely by domain knowledge of the task at hand. Many different permutations can be confidently ignored, and a general sequential structure can be easily presented. A convenient way to represent this is as a Petri Net as below.

Each tasks' forward progress within them are not considered online. For example, if the robot was halfway across the door, that information won't be considered. Only time elapsed on the task will be used as information to minimize the expected cost based on the offline model.

In summary, once the switching times x_n are calculated offline, these will be stored by the robot and can be used at each intention to minimize the estimated cost to complete it. The robot will create an offline compiled list that will dictate which task the robot should complete and when to minimize the total cost of completing the goal or intention, therefore maximizing the utility of the agent team.

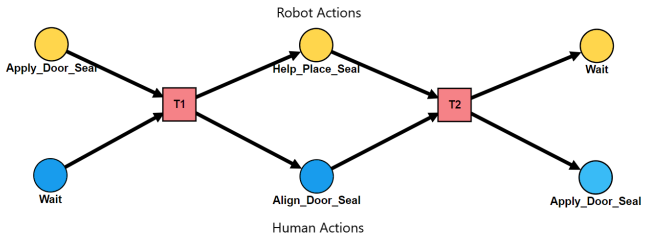


Fig. 2: Petri Net showing an example intention and task sequence, where the robot actions are in yellow, the transitions are in red, and the human actions are in blue.

IV. Experiment Setup

As mentioned previously, SERoCS provided the basic framework for the human-robot collaboration paradigm. Importantly, it provides the ability to set and carry out cartesian goals which are converted into joint based long term trajectories using the CFS algorithm. It uses Kinect environment data and a short term safety planner to keep the robot and human safe together [7]. When SERoCS was used at Berkeley, the MSC lab used FANUC robots along with two computers. To communicate with the robot, a FANUC system called DSA was used, where the desired motions were converted into torque commands and provided to the robot using EtherCat. One host computer ran the longer term plan and trajectory optimization, where a second target computer ran the safety controller as well as communicated with the robots. At CMU, even if the one physical robot we have is similar, the control mechanism for the robot is different and only one PC was desired. In this case, a UDP protocol called StreamMotion is used to command angle positions for each of the six

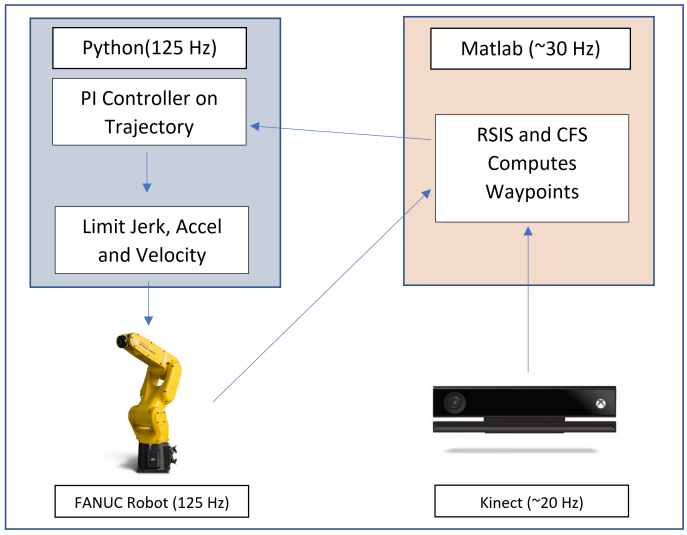


Fig. 3: Final Matlab to StreamMotion Trajectory Framework

joints of the FANUC robot or the accompanying simulator. Within every 8ms, the computer must respond with a target trajectory before the next status packet is received from the robot, which consists of the position and other diagnostic data. This immediately posed significant problems to the implementation because the long term planner can only be solved at a frequency of 30Hz.

To solve this problem, first, an experimental interface and a decoder for StreamMotion packets was created with Matlab and used to communicate with the FANUC robot. The intention was to integrate this with SERoCS since it was also written in Matlab and Simulink. After much experimentation, however, it continued to result in a MOTN-604 error because Matlab was not expedient enough to satisfy the 8ms constraint consistently. To avoid this issue, a Python-based UDP library was trialed. Thankfully, the Python provided bytes object was very useful for decoding and prototyping could be done very quickly. By itself, Python could easily keep up with the StreamMotion packets from the robot simulator. However, Python must now also communicate with the Matlab code running at approximately 30Hz. To architect this, a Matlab wrapper for the Python terminal was used to communicate positions in joint space to Python. To allow for the StreamMotion packets to be sent out possibly simultaneously and allow for Matlab to open the Python terminal when desired, the StreamMotion portion of the Python code was placed in its own high priority thread. Initially, the Matlab data was simply cached and repeated to the robot, effectively a zero hold method.

It was then found that the trajectory plan must be smoothed online to follow manufacturer-defined jerk, acceleration and velocity constraints before sending them to the robot. First, it was thought that a trajectory generation method like polynomials splines or lines could be used to smooth out the trajectory and avoid those constraints [1] [8]. To maintain expedient calculations, a line generation method was tried between the current point and next target point at each time the data was received from Matlab, where the expected amount of time

between each Matlab command was used to determine the number of points to generate. If a point from Matlab was received earlier than expected, the line would be redrawn to the new target position. If it was received later, then the robot would enter a zero order hold until the point was received. However, the constraint on jerk was still exceeded very often. Therefore, to meet this restriction, it was decided to attempt to apply these constraints online at 125 Hz and constrain the trajectory before sending it to the robot. The velocity, acceleration, and jerk was calculated as follows, where δt is the constant step size between two consecutive time intervals i.e. $\delta t = t_k - t_{k-1}$

$$v(t_k) = \frac{q(t_k) - q(t_{k-1})}{\delta t},$$
$$a(t_k) = \frac{v(t_k) - v(t_{k-1})}{\delta t} = \frac{q(t_k) - 2q(t_{k-1}) + q(t_{k-2})}{\delta t^2},$$
$$j(t_k) = \frac{a(t_k) - a(t_{k-1})}{\delta t} = \frac{q(t_k) - 3q(t_{k-1}) + 3q(t_{k-2}) - q(t_{k-3})}{\delta t^3}$$

and these constraints were applied:

$$v_{min} \leq v(t) \leq v_{max},$$
$$a_{min} \leq a(t) \leq a_{max},$$
$$j_{min} \leq j(t) \leq j_{max} \ \forall t$$

where v_{min}, v_{max} are the minimum and maximum limits on the velocity allowed by the robot manufacturer and acceleration and jerk are a and j respectively.

For instance, the application of the constraints was done as below for jerk:

$$a\,(t)_{max} = j_{max} + \ a\,(t-1)\,,$$
$$v\,(t)_{max} = \ a\,(t)_{max} + \ v\,(t-1)\,,$$
$$d\,(t+1) = d\,(t) + v\,(t)_{max}$$

for acceleration:

$$v(t)_{\max} = a_{\max} + v(t-1),$$
$$d(t+1) = d(t) + v(t)_{\max}$$

and for the velocity:

$$d(t+1) = d(t) + v_{\max}$$

However, this proved to be unstable. For example, in a ramp input, the actual position would initially be much below the desired position. The robot will maximize the jerk and acceleration constraints to meet the desired positional trajectory. Once meeting the trajectory, the velocity and acceleration would be much larger than that of the ramp input. The robot would then use a maximum negative jerk, attempting to correct itself. However, then the velocity and acceleration would be much lower than the ramp input. This process would repeat throughout the trajectory and become unstable under larger inputs.

To improve the stability, a PI controller was tried to facilitate the tracking of the robot to the desired trajectory. This promised to easily smooth out the trajectory, be stable and minimize error while being fast enough for online use. In this case, let $q_d(t)$ be the desired angle at the current timestep, and $q_d(t-1)$ be the previously desired angle. The error is defined as $e(t) = q_d(t) - q_d(t-1)$. A feedback and feedforward controller where PI is the feedback component was then used as follows:

$$q_d(t) = q_d(t-1) + K_p e(t) + K_i \int_0^t e(t)dt \qquad (4)$$

As demonstrated by the plot attached, this fulfilled the requirements for the robot trajectory.

This work completed on the FANUC robot will be integral to future work on the robot in the lab for human-robot collaboration and other experiments. Further research in task assignment will be conducted on the robot in the future.

V. Conclusion

The work completed on the FANUC robot as explained in the previous section will be very helpful for future projects in the lab. We have also theoretically shown that this task assignment paradigm has the ability to more efficiently optimize a human-robot collaborative assembly task and maximize the utility of each agent. However, in the future, additional work must be done on experimental situations. This will enable a better understanding of various task types and their cost and duration distributions. As for other research directions, this basic assignment structure could use additional data to make decisions on what the robot needs to accomplish, including verbal communication, eye gaze and other Theory of Mind type body cues. One could evaluate the skill and fatigue level of the human and adjust the degree of robot collaboration and automation in this case extending to different task action pairs online [10]. The robot could also provide additional information to help the human orient in the task. Finally, there was an assumption that forward progress to one task will be ignored when deciding whether to move to the next one. This was chosen to simplify the calculations of the issue but either a more advanced model or reinforcement learning could be considered to choose the optimal time to switch between tasks. Again, the framework designed here in the experiment section will be useful with SERoCS and many other experiments in the lab for the future.

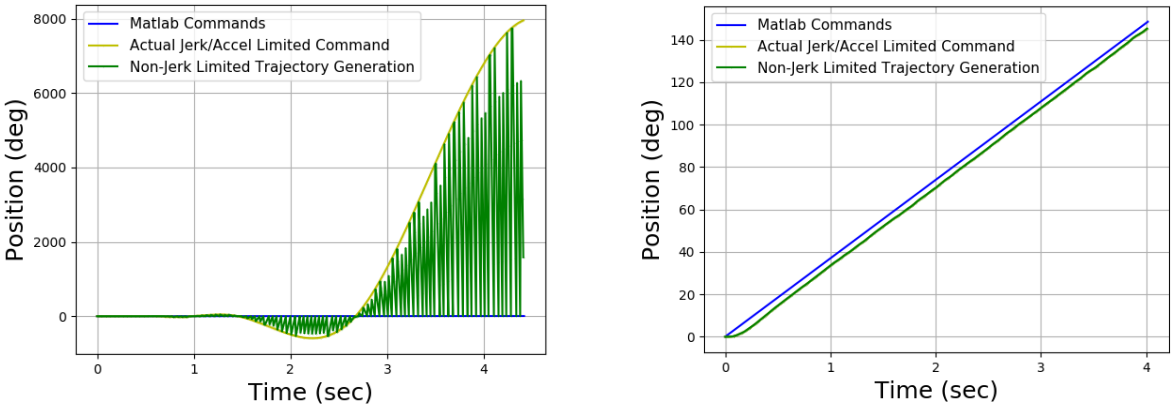
VI. Acknowledgements

This work was supported by the Robotics Institute Summer Scholars Program, the National Science Foundation REU program, and related to National Science Foundation Award #1734109. The authors would additionally like to recognize FANUC for their technical support.

References

[1] P. Allen, "Coms4733: Trajectory planning, class notes," November 2015.
[2] R. M. Aronson and H. Admoni, "Semantic gaze labeling for human-robot shared manipulation," in *ACM Symposium on Eye Tracking Research and Applications (ETRA)*. ACM, June 2019.
[3] Y. Cheng, L. Sun, and M. Tomizuka, "Towards better human robot collaboration with robust plan recognition and trajectory prediction," *arXiv preprint arXiv:1903.02199*, 2019.

[4] A. Clodic, H. Cao, S. Alili, V. Montreuil, R. Alami, and R. Chatila, "Shary: a supervision system adapted to human-robot interaction," in *Experimental robotics*. Springer, 2009, pp. 229–238.
[5] S. Devin and R. Alami, "An implemented theory of mind to improve human-robot shared plans execution," in *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 2016, pp. 319–326.
[6] S. Kotz, N. Balakrishnan, and N. L. Johnson, *Continuous multivariate distributions*. Wiley, 2000.
[7] C. Liu, T. Tang, H.-C. Lin, Y. Cheng, and M. Tomizuka, "Serocs: Safe and efficient robot collaborative systems for next generation intelligent industrial co-robots," *arXiv preprint arXiv:1809.08215*, 2018.
[8] S. Macfarlane and E. A. Croft, "Jerk-bounded manipulator trajectory planning: design for real-time applications," *IEEE Transactions on Robotics and Automation*, vol. 19, no. 1, pp. 42–52, 2003.
[9] B. Rummel, "Beyond average: Weibull analysis of task completion times," *Journal of Usability Studies*, vol. 12, no. 2, pp. 56–72, 2017.
[10] B. Sadrfaridpour, H. Saeidi, J. Burke, K. Madathil, and Y. Wang, "Modeling and control of trust in human-robot collaborative manufacturing," in *Robust Intelligence and Trust in Autonomous Systems*. Springer, 2016, pp. 115–141.
[11] B. Scassellati, "Theory of mind for a humanoid robot," *Autonomous Robots*, vol. 12, no. 1, pp. 13–24, 2002.



(a) Linear Interpolation (Unstable)

(b) PI Tracking Controller (Stable)

Fig. 4: One joint trajectory under ramp input