APPENDIX A
SURVEYS

First, we provide Cronbach's alpha scores for the surveys we employ to test for internal reliability.

Scale	α
Agreeableness	.787
Likeability	.928
Negative Situation	.749
Negative Social	.654
Negative Emotions	.766

TABLE I: Cronbach's alpha.

We next provide details regarding the interview questions and hand-crafted questionnaires we administer.

A. Post-Interview Questions

We ask participants five post-interview questions.

1) "Describe your experience using the interface."
2) "Did you feel any information was missing from the explanations provided to you in this study?"
3) "What did you think about the videos you watched in this study?"
4) "Do you think you were able to teach the robot better as the experiment progressed?"
5) "Do you have any other comments or suggestions?"

B. Additional Subjective Metrics

We collect the following subjective metrics in the post-survey questionnaire, but find no significance, and therefore did not describe them in the paper.

- **System Usability:** We use the System Usability Scale (SUS), 10 questions rated on a seven-point scale (Strongly Disagree=1 to Strongly Agree=7) to measure robot and system usability [2].
- **Anthropomorphism:** We use the Anthropomorphism sub-scale of the Godspeed Questionnaire Series, consisting of five questions rated on a five-point scale [1].
- **Methods of Teaching the Robot:** We employ a hand-crafted questionnaire consisting of 17 questions rated on a seven-point scale (Strongly Disagree=1 to Strongly Agree=7) that measured the perceived usefulness and effectiveness of the method of teaching (see Appendix).
- **Trust:** We employ the Multi-Dimensional Measure of Trust (MDMT) questionnaire [6] consisting of 16 questions rated on an eight-point scale (Not at all=0 to Very=7, with an option for Does Not Fit). We measure the overall scale score, as well as participants' score on the Capacity Trust (with Reliable and Capable sub-scales) and Moral Trust (with Ethical and Sincere sub-scales).

C. Hand-Crafted Questionnaires

1) Robotics Prior Experience Survey: Please provide your years of experience with robots (0 to 10+). You must make a selection, even if it is to keep the slider at 0.

Fig. 1: Depicted is the single item robotics prior experience question.

2) Teacher Prior Experience Survey: This section of the survey is a 7-point Likert scale ranging from strongly disagree to strongly agree.

- I have experience teaching.
- I have experience mentoring.
- I have experience tutoring.
- I have experience coaching.
- I have experience training others.

3) Methods of Teaching the Robot: This section of the survey is a 7-point Likert scale (Strongly Disagree to Strongly Agree).

1) Using this method of teaching robots is useful for me
2) Using this method of teaching robots will improve my effectiveness
3) Using this method of teaching robots will improve my performance
4) This method of teaching robots would make it be easy to teach robots behaviors I need
5) Learning this method of teaching robots would be easy
6) This method of teaching robots would be easy to use
7) Using this method to train agents is an idea I like
8) Using this method to train agents would be a pleasant experience
9) Using this method to train agents is a good idea
10) Using this method to train agents is a wise idea
11) I trust this method to teach robots effectively
12) This method to teach robots is reliable
13) This method to teach robots is trustworthy
14) I am concerned about how I use my time to train agents
15) When I will need it I would like to use this method to train robots
16) When I will need it I will intend to use this method to train robots
17) When I will need it I predict I would use this method to train robots

APPENDIX B
METHODS

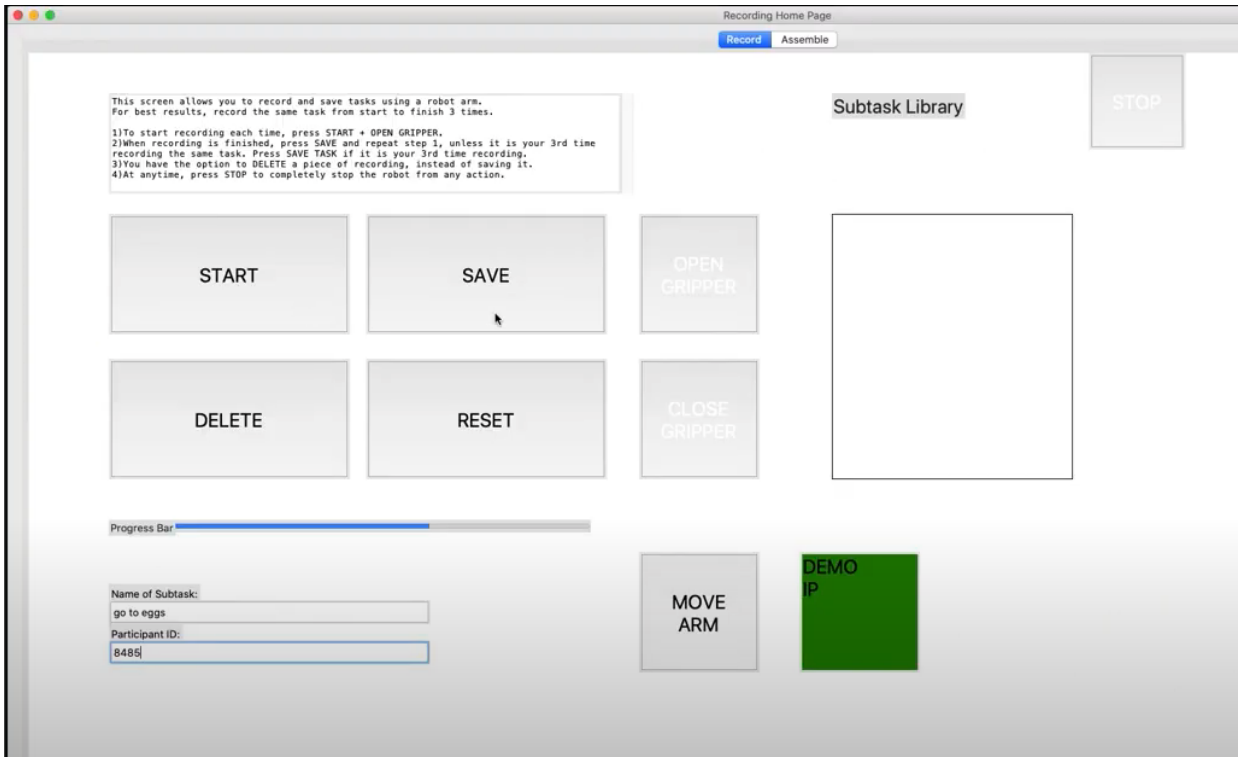
A. Videos

Introduction Video and Optimal Teaching Strategy Videos can be found at the following website.

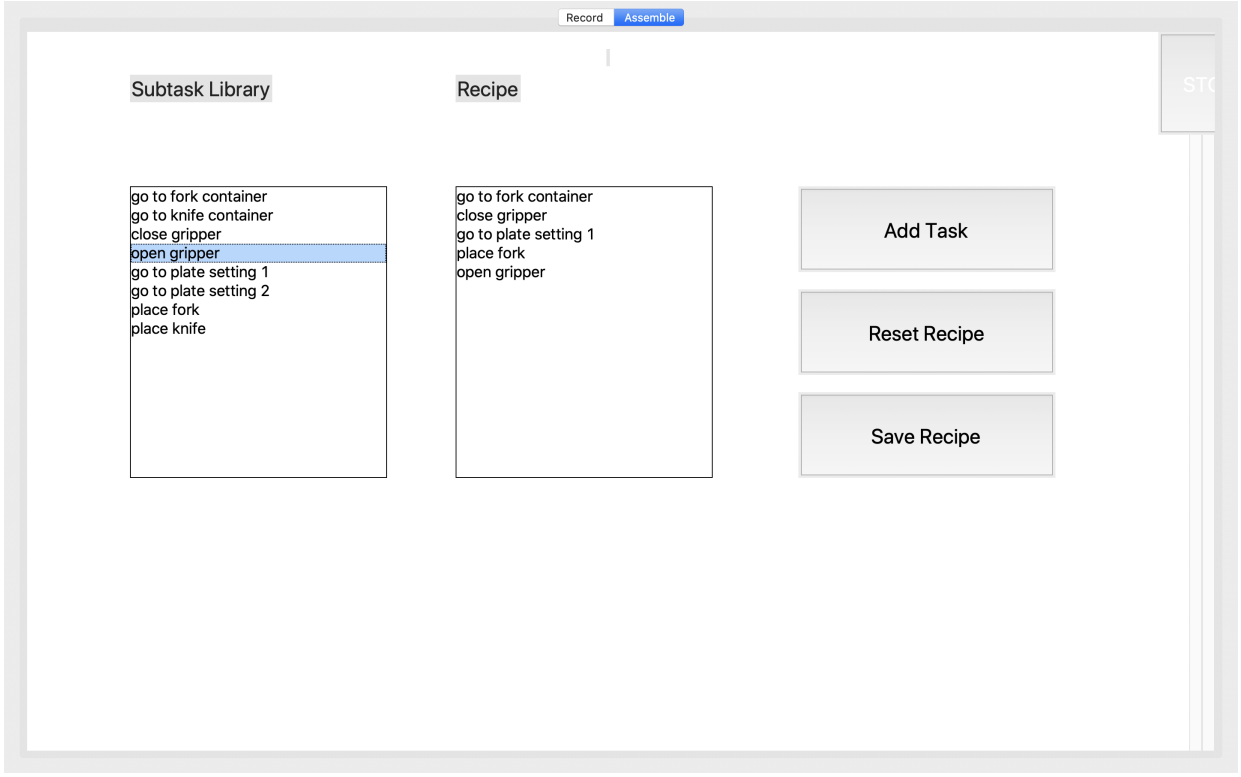
https://sites.google.com/view/moormanetal-rss2023

B. Interface Design

The interface for this study relied on usability heuristics and user experience principles. In order to understand what improvements are needed, we analyzed an existing robot manipulation interface [4] against Nielsen Norman's 10



(a) Depicted is an image of the interface's recording screen.



(b) Depicted is an image of the interface's assemble screen.

Fig. 2: This figure illustrates the screens of the interface.

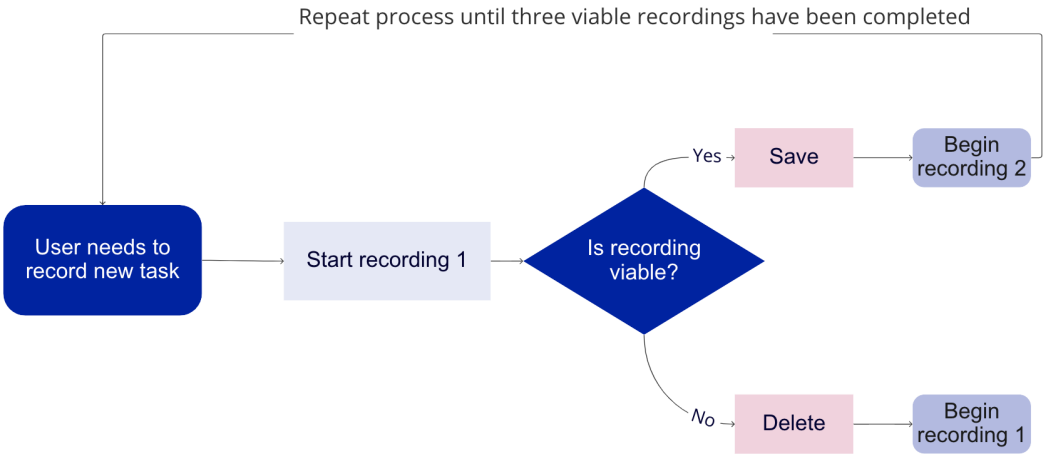


Fig. 3: Depicted is a flow chart of the record interface tab.

heuristics principles to determine which design elements could be improved [5].

The design entails a recording screen (Figure 2a) and an "assemble" screen (Figure 2b). The recording screen allows users to facilitate robot teaching and learning on demand, record their actions three times (to optimize robot learning), and save learned tasks into a library. System status is visible via the recording progress bar, which helps notify users of a robot's overall learning progress. In the assemble screen, the *library* concept allows users to save, categorize, and visualize all of the robot's learned behavior; it is created to match the user's mental model of a knowledge repository and allows users to quickly navigate to learned robot commands and be able to call to action quickly. The *queue* concept allows users to pick existing learned tasks from the library, arrange them in a logical sequence, and form coherent, continuous, longer tasks; it is created to maximize personalization for users while accounting for a robot's technical capabilities.

The design also incorporated a stop button that will halt all robot movement immediately. This feature aligns with the Nielsen Norman Group's heuristics principles [5] in ensuring enough user freedom and control in an interactive environment. Though the experiment is designed to be in a controlled setting, this design added another user exit option to increase both confidence in robot teaching as well as user safety.

APPENDIX C
STUDY DESIGN

A. Domains and Tasks

- **Block Touching:** A blue, green, and red block are placed in front of the robot. Participants are asked to teach the robot to touch the blocks in a particular order using the gripper of the robot. The tasks for the block touching domain are as follows.
  1) Touch the red block, then touch the blue block.
  2) Touch the blue block, then touch the green block.
  3) Touch the green block, then touch the blue block, then touch the red block.
- **Box Packing:** Two plastic bananas, two Jell-O boxes, and two SPAM cans, along with a cardboard box are laid out in front of the robot (these food items are chosen to be in compliance with items commonly used in robotics benchmarking, such as the YCB dataset [7]). Participants are asked to teach the robot to pick up and place a combination of these food items into the cardboard box. In this domain, bananas served as distractor items, and are not relevant to the tasks the participant must teach the robot to accomplish. The tasks for the box packing domain were as follows.
  1) Pack 2 SPAM cans and 1 jello box in the box.
  2) Pack 1 SPAM can in the box.
  3) Pack 2 jello boxes in the box.
- **Table Setting:** Two forks, two knives, and two plates are placed in front of the robot. Participants are asked to teach the robot to pick up the utensils and place them in designated locations around the two plate settings. Duct tape has been wrapped around the utensils in order to facilitate manipulation of the utensils by the robot's gripper. The tasks for the table setting domain were as follows.
  1) Place a fork and knife for plate setting 2 and a knife for plate setting 1.
  2) Place a fork for plate setting 1.
  3) Place a fork and knife for plate setting 1.
- **Soil Mixing:** A bucket of manure, a bucket of sand, and a bucket of lime are placed in front of the robot, along with a mixing bowl into which scoops of each of these materials are to be poured. Participants are asked to teach the robot to create different soil mixtures for different plants using the scoop, which has been placed in the robot's gripper. The tasks for the soil mixing domain were as follows.
  1) Assemble 2 scoops of sand and 1 scoop of lime.

2) Assemble 1 scoop of manure and 1 scoop of lime.
3) Assemble 1 scoop of sand and 1 scoop of manure.

- **Medicine Dispensing:** Four kinds of medicine (red pill cup, green pill cup, yellow pill cup, and TUMS pill cup) along with three trays labeled persons 1, 2, and 3 are placed in front of the robot. Participants are asked to pick and place these medicine cups into the appropriate person's tray to dispense the proper medication to each person. Note that here the TUMS pill cup and person 3 tray both serve as distractor items, and are not relevant to the tasks the participant must teach the robot in this domain. The tasks for the medicine dispensing domain were as follows.
  1) Serve the red pills to person 1, and the yellow pills to person 2.
  2) Serve the green pills to person 1.
  3) Serve the yellow pills to person 1 and the red pills to person 2.

B. Orderings

We ordered the conditions based on a Latin square design to ensure that the ordering is balanced. Each participant experienced one of the conditions.

TABLE II: This table shows the domain orders for each of the four conditions.

Table Ordering	Soil Ordering	Box Ordering	Medicine Ordering
1	2	3	4
3	1	4	2
4	3	2	1
2	4	1	3

Condition 1: Table Setting, Soil Mixture, Box Packing, Medicine Dispensing

Condition 2: Soil Mixture, Medicine Dispensing, Table Setting, Box Packing

Condition 3: Medicine Dispensing, Box Packing, Soil Mixture, Table Setting

Condition 4: Box Packing, Table Setting, Medicine Dispensing, Soil Mixture

APPENDIX D
TAMP SUB-TASK FORMULATION

Task and Motion Planning

Task and motion planning (TAMP) problems integrate multi-modal motion planning with representational strategies originating in long horizon task planning [3, 8]. TAMP is a framework that allows for an agent to break down and complete long-horizon tasks while taking into account multi-faceted constraints in a similar manner to Multi-Modal Motion Planning. As such, each sub-task is described by its preconditions, `pre`, that need to be set in order for a sub-task to take place, constraints `con` that is the set of constraint functions that must hold for all continuous actions during the duration of the sub-task, and effects (or transition), `eff`, the changes made to the robot, objects and environments after a sub-task has been executed.

```
goToRedPills(s_i, τ, s_r, p_t, ..., p_p3):
    con: Move(s_i, τ, s_r), CFreeE(τ),
         CFreeTUMS(p_t, τ), ...,
         CFreePerson3(p_p3, τ)
    pre: holdingCup=False, LocRobot=s_i,
         LocTUMS=p_r, ..., LocPerson3=p_p3
    eff: LocRobot=s_r, holdingCup=False
goToGreenPills(s_i, τ, s_g, p_t, ..., p_p3):
    con: Move(s_i, τ, s_g), CFreeE(τ),
         CFreeTUMS(p_t, τ), ...,
         CFreePerson3(p_p3, τ)
    pre: holdingCup=False, LocRobot=s_i,
         LocTUMS=p_r, ..., LocPerson3=p_p3
    eff: LocRobot=s_g, holdingCup=False
goToYellowPills(s_i, τ, s_y, p_t, ..., p_p3):
    con: Move(s_i, τ, s_y), CFreeE(τ),
         CFreeTUMS(p_t, τ), ...,
         CFreePerson3(p_p3, τ)
    pre: holdingCup=False, LocRobot=s_i,
         LocTUMS=p_r, ..., LocPerson3=p_p3
    eff: LocRobot=s_y, holdingCup=False
grasp[cup](s_i, τ, s_j, p_c):
    con: Grasp(s_i, τ, s_j)
    pre: holdingCup=False, LocRobot=s_i,
         Loc[cup]=p_c
    eff: holdingCup=True, LocRobot=s_j
goToPerson1(s_i, τ, s_p1, p_t, ..., p_p3):
    con: Move(s_i, τ, s_p1), Upright(τ),
         CFreeE(τ), CFreeTUMS(p_t, τ), ...,
         CFreePerson3(p_p3, τ)
    pre: holdingCup=True, LocRobot=s_i,
         LocTUMS=p_r, ..., LocPerson3=p_p3
    eff: LocRobot=s_p1, holdingCup=True
goToPerson2(s_i, τ, s_p2, p_t, ..., p_p3):
    con: Move(s_i, τ, s_p2), Upright(τ),
         CFreeE(τ), CFreeTUMS(p_t, τ), ...,
         CFreePerson3(p_p3, τ)
    pre: holdingCup=True, LocRobot=s_i,
         LocTUMS=p_r, ..., LocPerson3=p_p3
    eff: LocRobot=s_p2, holdingCup=True
release[person](s_i, τ, s_j, p_p):
    con: Release(s_i, τ, s_j)
    pre: holdingCup=True, LocRobot=s_i,
         Loc[person]=p_b
    eff: holdingCup=False, LocRobot=s_j
```

Fig. 4: Sample formulation in the medicine domain.

Sample Formulation

Next, we provide a sample formulation for the Medicine Dispensing domain, shown in Figure 4. Each task takes in an initial state s_i, a trajectory provided by a demonstration τ, the final state the robot should reach at the end of the task s_x (x varies depending on the task), and a collection of poses of all objects in the environment.

Variable τ represents a single trajectory. There are few different types of constraints included in this formulation. $\texttt{Move}(s_i, \tau, s_j)$ $\forall s_i, s_j \in \mathcal{S}$ is a constraint over trajectories where the robot moves from s_i to s_j. The constraints for $\texttt{CloseGripper}(s_i, \tau, s_j)$ and $\texttt{OpenGripper}(s_i, \tau, s_j)$ are

similar, they represent trajectories that allow the robot to grasp and release the cup respectively. There are also collision-free constraints, such as `CFreeE` and `CFreeTUMS` that tell the robot not to collide with the environment or any of the objects in the environment. The go to person sub-tasks also include an `Upright` constraint telling the robot to keep the cup upright when holding it.

In the go to pills sub-tasks the robot is not holding anything in its gripper, i.e., `holdingCup=False`. As explained above, this means that its range of motion is not nearly as limited as the robot does not have to worry about any object that it may be manipulating. The go to person sub-tasks are transfer mode subtasks [3], as `holdingCup=True`, and so the robot's range of motion is limited as it has to worry about not spilling the pills while moving the cup; hence `Upright` is included in `con`. `grasp` performs a kinematic switch when grasping a cup; this switch can be seen in the sub-task formulation as `holdingCup=False` in `pre` being switched to `holdingCup=True` in `eff`. The reverse can be seen in `release`.

APPENDIX E
QUOTES

Getting better over time

We first wanted to study how domain experience impacts the quality of demonstrations. In our post-interview, we found that the majority of participants agreed that they were able to teach the robot better as the experiment progressed. Two sample quotes supporting this qualitative finding are as follows.

> "I learned from both my experience teaching the robot, and watching the videos and seeing you teach the robot how to do things."

> "After watching the video of how you accomplished the overall work, then I understood what I should do."

Zero-shot transfer

We found that participants can perform zero-shot transfer of knowledge regarding sub-task abstraction, teaching efficiency, and sub-task redundancy to novel domains. One participant in particular commented on this in the transition between the box packing and table setting domains. They describe their gained understanding about abstracting sub-tasks to account for item multiplicity.

> "I followed again the example of the [table setting domain] video so I knew that the two cans of SPAM and two cans of Jell-O were basically equivalent so I only trained it to go to the space and then separately trained it to close the gripper to pick up an item separately..."

Hierarchical Task Planning

In our interviews, we found that participants spoke about low-level motion control in the first couple domains. One participant, when asked whether they thought they taught the robot better as the experiment progressed, stated the following.

> "Earlier, there was a lot of joints and the ways the robot could move that I didn't know of. Or many things that I felt it couldn't do but it could have that I got to learn while watching the videos. So yeah, as the experiment progressed, I felt pretty comfortable and easier to control the robot."

In the latter domains, however, participants shifted to discussing higher-level strategies and task planning. One participant, when initially asked about their experience after block touching (the first domain), describes low-level strategies for manipulating the robot.

> "I just noticed that it had some joints that could be moved to make it to teach it to do what I wanted so I just moved those and like pointed to the different blocks and then saved as I went."

The same participant, after their final domain in the Box packing domain, describes their high-level task planning.

> "So for this tasks I simply act out the robot how to grab either Jell-O, bananas, or SPAM and then I taught it how to go to the box and then I taught it how to grab and drop so that you would grab either banana, Jell-O, and SPAM and then go to the box and then drop it there."

The contrast between thinking about the low-level movement in the first demo domain and the higher-level abstractions in the last domain encountered is especially apparent with these answers.

Workload

We also investigate the effect of demonstration abstraction on participants' perceived workload. Our findings indicated an inverse correlation between sub-task count and perceived workload. One participant noted fatigue and a high perceived workload at the end of the study.

> "Overall, it was very time-consuming, repeating the motions over and over again. I found myself getting really tired near the end, like the last [domain]."

APPENDIX F
CODER INSTRUCTIONS

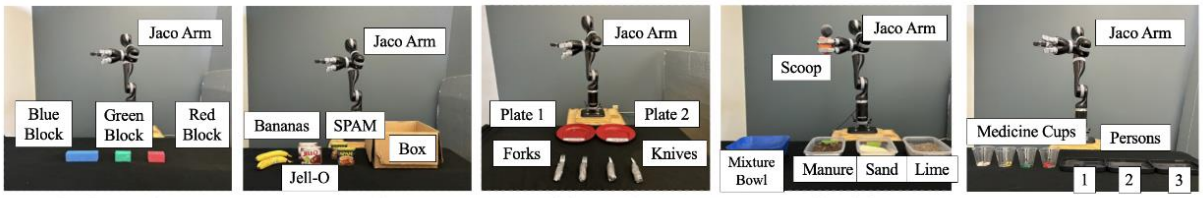
Following are the coder instructions from which we obtain our abstraction and redundancy scores, cited as [4] in our work.

Coder Instructions

Context

Learning from Demonstration enables end-users, who are not robotics experts, to shape robot behavior. In our work, we are interested in how the quality of demonstrations changes over time, after trial and error teaching the robot in different domains.

Domains



Block Touching: A blue, green, and red block are placed in front of the robot. Participants are asked to teach the robot to touch the blocks in a particular order using the robot gripper.

Box Packing: Two plastic bananas, two Jell-O boxes, and two Spam cans, along with a cardboard box are laid out in front of the robot. Participants are asked to teach the robot to pack (pick up and place) a combination of these food items into the cardboard box.

Table Setting: Two forks, two knives, and two plates are placed in front of the robot. Participants are asked to teach the robot to set the table by picking up the utensils and placing them in designated locations around the two plate settings.

Soil Mixing: A bucket of manure, a bucket of sand, and a bucket of lime are placed in front of the robot, along with a mixing bowl into which scoops of each of these materials are to be poured. Participants are asked to teach the robot to create different soil mixtures for different plants. In this domain, the scoop is placed in the robot's gripper by the experimenter.

Medicine Dispensing: Four kinds of medicine (red pill cup, green pill cup, yellow pill cup, and TUMS pill cup) along with three trays labeled persons 1, 2, and 3 are placed in front of the robot. Participants are asked to dispense the proper medication to each person by picking and placing medicine cups into the appropriate person's tray.

Definitions

Domain: The participant will work with the robot in 5 domains, as listed above.

Task: In each domain, the participant must teach the robot 3 tasks (like assemble exactly two scoops of sand and one scoop of lime).

Subtask: The participant teaches the robot sub-tasks (like go to sand) that can be used to accomplish tasks.

Demonstration: The participant physically moved the robot, *demonstrating* to record how to accomplish a subtask.

What data do we have?

In each domain, participants would provide a series of demonstrations to teach subtasks that can be reused to accomplish tasks. They then used these subtasks to accomplish 3 tasks in each domain. We have notes that describe the subtasks provided, and we also have the list of subtasks names assigned for each task. For instance, in the soil mixing domain, a participant could record the following subtasks:

- Go to sand bin
- Go to manure bin
- Go to lime bin
- Scoop
- Go to mixing bucket
- Drop

Then, they would assign these subtasks to the following tasks:

Soil Mixing Domain	Task 1	Task 2	Task 3
Task description	Assemble exactly two scoops of sand and one scoop of lime.	Assemble exactly one scoop of manure and one scoop of lime.	Assemble exactly one scoop of sand and one scoop of manure
Subtasks assigned to task	∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to lime bin ∗ scoop ∗ Go to mixture bin ∗ Dump	∗ Go to manure bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to lime bin ∗ Scoop ∗ Go to mixture bin ∗ Dump	∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to manure bin ∗ Scoop ∗ Go to mixture bin ∗ Dump

Types of scoring

Given the notes of each subtask recorded by participants, how do we evaluate the quality of the demonstrations used for each task? We look at two metrics: abstraction score and redundancy score.

Abstraction Score

The abstraction score describes how well the subtasks were abstracted, to maximize the usability of the subtasks. Recall the goal in the soil mixing domain of making soil mixtures for plants. If, for instance, the goal was 2 scoops of lime and 1 scoop of sand.

Insufficiently Abstracted	Optimal Abstraction	Overly Abstracted
– Go to lime, scoop, go to bucket, drop, go to lime, scoop, go to bucket, drop, go to sand, scoop, go to bucket, drop	– Go to sand – Go to lime – Scoop – Go to bucket – Drop scoop	– Move left – Move right – Move up – Move down
The problem here is that we can only reuse this subtask in the exact same scenario (this subtask cannot be used to scoop 1 scoop of lime and 2 scoops of sand).	This optimally abstracted set of tasks will generalize to any number of scoops of lime and sand required.	The problem here is that it is unclear how to combine these tasks to accomplish the goal, since they are too abstracted.

To calculate abstraction score, one point is awarded for each subtask that can be used towards the task when using the sub-tasks to accomplish the task. For example, the insufficiently abstracted example above would receive 1 point as their abstraction score as they only recorded one subtask that could be applied to the task. If they had instead recorded two subtasks:

- Go to lime, scoop, go to bucket, drop, go to lime, scoop, go to bucket, drop
- Go to sand, scoop, go to bucket, drop

Then they would receive an abstraction score of 2.

The optimal abstraction score example above would be used as follows to accomplish the task:

- Go to lime
- Scoop
- Go to bucket
- Drop scoop
- Go to lime
- Scoop
- Go to bucket
- Drop scoop
- Go to sand
- Scoop
- Go to bucket
- Drop scoop

Resulting in an abstraction score of 12.

Redundancy Score

The redundancy score describes how (un)necessary tasks are. A sub-task is deemed redundant if its goal can be met by another sub-task or a combination of sub-tasks previously taught. Sub-task redundancy is defined with respect to a given set of subtasks being taught. For instance, for the same context, where the goal is 2 scoops of lime and 1 scoop of sand, the following table describes two levels of redundancy.

Redundant	Optimal
Go to sand. Scoop sand. Go to lime. Scoop lime. Go to bucket. Drop lime. Drop Sand.	Go to sand. Go to lime. Scoop. Go to bucket. Drop scoop.
The problem here is that we can only reuse drop sand for drop lime (simply drop), so drop lime is unnecessary, and redundant. Similarly, we can use scoop sand for scoop lime (simply scoop), so scoop lime is unnecessary and redundant.	This is the proper subtask breakdown, where the demonstrator recognizes that drop can work for sand, lime, and manure, and scoop similarly can work for sand, lime and manure.

To calculate redundancy score, one point is given for each subtask that can be done using other sub-tasks recorded in the domain. For instance, the redundant example above receives a redundancy score of 2 (one for the redundant drop subtask and one for the redundant scoop subtask). However, the optimal example above receives a redundant score of 0 as no redundant subtasks were taught.

Note that the redundancy score is computed based upon the subtasks taught by that participant, not the optimal subtask list.

Key for each domain

Soil Mixing Domain	Task 1	Task 2	Task 3
Task description	Assemble exactly two scoops of sand and one scoop of lime.	Assemble exactly one scoop of manure and one scoop of lime.	Assemble exactly one scoop of sand and one scoop of manure
Optimal Subtask List	∗ Go to sand bin ∗ Go to manure bin ∗ Go to lime bin ∗ Scoop ∗ Go to mixture bin ∗ Dump		
Optimal Subtask Assignment	∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to lime bin ∗ scoop ∗ Go to mixture bin ∗ Dump	∗ Go to manure bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to lime bin ∗ Scoop ∗ Go to mixture bin ∗ Dump	∗ Go to sand bin ∗ Scoop ∗ Go to mixture bin ∗ Dump ∗ Go to manure bin ∗ Scoop ∗ Go to mixture bin ∗ Dump
Max Abstraction Score	12	8	8

Block Touching (Demo) Domain	Task 1	Task 2	Task 3
Task description	Touch the red block then touch the blue block	Touch the blue block then touch the green block	Touch the green block then touch the blue block then touch the red block
Optimal Subtask List	∗ Go to and touch the green block ∗ Go to and touch the blue block ∗ Go to and touch the red block		
Optimal Subtask Assignment	∗ Go to and touch the red block ∗ Go to and touch the blue block	∗ Go to and touch the blue block ∗ Go to and touch the green block	∗ Go to and touch the green block ∗ Go to and touch the blue block ∗ Go to and touch the red block
Max Abstraction Score	2	2	3

Box Packing Domain	Task 1	Task 2	Task 3
Task description	Pack exactly two SPAM cans and exactly one jello box in the box	Pack exactly one SPAM can in the box	Pack exactly two jello boxes in the box
Optimal Subtask List	∗ Go to SPAM ∗ Go to jello box ∗ Close gripper ∗ Go to box ∗ Open gripper		
Optimal Subtask Assignment	∗ Go to SPAM ∗ Close gripper ∗ Go to box ∗ Open gripper ∗ Go to SPAM ∗ Close gripper ∗ Go to box ∗ Open gripper ∗ Go to jello box ∗ Close gripper ∗ Go to box ∗ Open gripper	∗ Go to SPAM ∗ Close gripper ∗ Go to box ∗ Open gripper	∗ Go to jello box ∗ Close gripper ∗ Go to box ∗ Open gripper ∗ Go to jello box ∗ Close gripper ∗ Go to box ∗ Open gripper
Max Abstraction Score	12	4	8

Table Setting Domain	Task 1	Task 2	Task 3
Task description	Place the fork and knife for plate setting 2 and knife for plate setting 1	Place fork for plate setting 1	Place knife and fork for plate setting 1
Optimal Subtask List	∗ Go to fork container ∗ Go to knife container ∗ Close gripper ∗ Go to plate setting 1 ∗ Go to plate setting 2 ∗ Go to plate's fork loc ∗ Go to plate's knife loc ∗ Open gripper		
Optimal Subtask Assignment	∗ Go to fork container ∗ Close gripper ∗ Go to plate setting 2 ∗ Go to plate's fork loc ∗ Open gripper ∗ Go to knife container ∗ Close gripper ∗ Go to plate setting 2 ∗ Go to plate's knife loc ∗ Open gripper ∗ Go to knife container ∗ Close gripper ∗ Go to plate setting 1 ∗ Go to plate's knife loc ∗ Open gripper	∗ Go to fork container ∗ Close gripper ∗ Go to plate setting 1 ∗ Go to plate's fork loc ∗ Open gripper	∗ Go to fork container ∗ Close gripper ∗ Go to plate setting 1 ∗ Go to plate's fork loc ∗ Open gripper ∗ Go to knife container ∗ Close gripper ∗ Go to plate setting 1 ∗ Go to plate's knife loc ∗ Open gripper
Max Abstraction Score	15	5	10

Medicine Dispensing Domain	Task 1	Task 2	Task 3
Task description	Serve red pills to person 1, and yellow pills to person 2	Serve green pills to person 1	Serve yellow pills to person 1 and red pills to person 2
Optimal Subtask List	∗ Go to yellow pill cup ∗ Go to red pill cup ∗ Go to green pill cup ∗ Close gripper ∗ Go to person 1 location ∗ Go to person 2 location ∗ Open gripper		
Optimal Subtask Assignment	∗ Go to red pill cup ∗ Close gripper ∗ Go to person 1 location ∗ Open gripper ∗ Go to yellow pill cup ∗ Close gripper ∗ Go to person 2 location ∗ Open gripper	∗ Go to green pills cup ∗ Close gripper ∗ Go to person 1 location ∗ Open gripper	∗ Go to yellow pill cup ∗ Close gripper ∗ Go to person 1 location ∗ Open gripper ∗ Go to red pill cup ∗ Close gripper ∗ Go to person 2 location ∗ Open gripper
Max Abstraction Score	8	4	8

How is the data organized?

Available is a list describing each of the subtasks recorded for that domain. These are notes taken by the experimenter. Available is also the recipe constructed by the participant for each task (task 1 – 3) in that domain. This uses the names the participant called each of their subtasks (the naming scheme may differ between people).

For each person, the coder must read the tasks recorded, understand how they differ from the optimal key provided, and look at how the participant used these subtasks for each task in the domain. Then, they score the abstraction score and redundancy of that domain, by summing the abstraction score across tasks of the domain, and summing redundancy score across tasks of the domain.

The written notes are to provide context into the observed behavior of the recording of subtasks. However, it is possible that participants recorded subtasks that they never used, i.e. in the recipe for the task (if they for instance change their strategy half way through). Therefore, when calculating score please verify that the subtask was used in the recipe saved.

More Information

This section details background in Task and Motion Planning (TAMP) to enable us to answer the question: *How to tell if a subtask is valid/ can be applied to the task?*

Pre-requisits

1) an understanding of TAMP (https://arxiv.org/abs/2010.01083)
2) familiarity with the domain that you are rating. This familiarity can be gained by actually solving the domain and understanding the possible abstractions within the domain to solve the given tasks.

Concepts needed to score –

There are two concepts that a rater of TAMP abstractions needs to understand: Mode Changes and Bottleneck states.

Mode changes: When we interact with objects, we change the mode of contact with them. Consider picking up a fork. When we have no fork in the gripper the gripper is bound by fewer constraints of movement. When the fork is in the gripper the gripper is bound by additional constraints such as the fork not colliding with anything else in the world. This change in constraints of movement lead to obvious points of sub-task creation. Pick up fork needs to be a different sub-task from move fork to the plate.

Bottleneck States: Bottle neck states are states that an agent needs to reach to solve a larger family of sub-tasks. A classic example of a bottleneck state is a door. To access the gas stove or the kitchen cabinet or to cook, you first need to cross the bottleneck state of the kitchen door. Across these bottlenecks the agent has access to a greater number of sub-tasks. In pick and place domains a classic bottle neck is the "pre-grasp" position. Consider the location just above the fork, this allows the robot to pick up the fork in different orientations. Hence, this location just above the fork is considered a pre-grasp position. The robot does not have to constrain its gripper pose when moving to the pre-grasp position, but when initiating the "grasp" sub-task the robot's gripper needs to be strictly oriented to pick

the object correctly. A similar situation occurs when the robot needs to place knives and forks next to a plate. The move to plate is a good sub-task to teach as all utensils would require the robot to move to the plate. After which the robot can move to the knife's location and place the knife or move to the fork's position to place the fork. Notice that for a different plate we would need to teach a novel reach plate task as the plates are at different locations, however the knives and forks are at the same relative position when compared to the plate's position, making the move to fork or knife position sub-tasks repeatable when taught once. Without teaching this bottleneck abstraction of move to plate, the user would have to teach place knife next to plate 1 and place knife next to plate 2 as different tasks, and repeat these sub-tasks for the forks, the napkins, the spoons, etc, for each plate. Now they just have to teach how to reach the bottleneck state of reaching a plate, and then the robot can just repeat the placement of the knives and forks from previous relative sub-tasks.

How to award points –

Abstraction score - We give points for each sub-task the user taught that is necessary in solving the task. The sub-tasks are necessary if without them the overall task could not have been solved. For example, to place a fork from the fork location next to plate 1 we can break down subtasks in many ways. A user might just pick the fork and place it next to plate 1 in one task. This is a necessary sub-task to solve this problem, hence the user gets one point. Or the user might decide to move to the fork location (pre-grasp), pick up the fork (grasp), move to the plate 1 (bottleneck), move to fork location (pre-place), place fork (place). These provide 5 sub-tasks all necessary and repeatable hence the user gets 5 points. The users might not distinguish between a pre-grasp and a grasp or a pre-place and place and they might only get 3 points (for pick, move to bottleneck and place).

Redundancy score - Sometimes users teach sub-tasks but do not use them, or use them but they were unnecessary to begin with. For example, a lot of users teach a sub-task to move the robot back to a home position before the robot does the next sub-task. This is not necessary. The users do it because of a poor understanding of how the robot works and this is penalized when measuring the redundancy score.

What about over-abstracting? Are we providing too many points for users to create too many sub-tasks? Does moving 1 cm to the left a sub-task – no, there is no mode change or reaching of a bottleneck state because of this sub-task, hence it is not necessary in solving the task. However, you can teach two ways of getting to the fork, is that important? What if I make a different sub-task to pick every fork in the fork location, now I have tons of sub-tasks. Well picking up one fork from a fork basket is no different from picking another so just wasted your time, and this is a redundant sub-task!

How the scoring strategy applies to each domain –

Now for each domain we will provide our scoring key and list of bottleneck and mode changes. These should be sufficient to score both the abstraction scores and redundancy scores.

Soil domain -

The mode changes here happen when contacting the objects being picked with the scoop. The bottleneck states are the pre-grasp positions over the positions of the objects and the mixing bowl. The optimal abstraction moves over objects (bottleneck state / pre-grasp), scoops them (grasp), moves to the mixing bowl (bottleneck state /pre-place), dumps contents (place).

Table setting domain -

The mode changes happen when forks or knives are picked up (grasp) or placed (released). The bottle neck states are the pre-pick and pre-place over the knife and fork locations and the move to plate, move to fork pick location (which is a fork dispenser like a drawer partition with forks alone) and a move to knife pick location (again a knife dispenser).

Medicine delivery domain -

The mode changes here are pick and place of the medicine boxes. The bottle neck states are to get to the position of each of the colored medicine pill boxes and person's location for delivery of the medicines as pre-grasp and pre-place positions.

Box packing domain -

The mode changes are in the pick and place of the items and are taught as close gripper and open gripper. The bottle-neck abstractions are to reach the location of the spam containers, and jello-box containers storage locations, and reach the location of the box being packed. Again, the spam and jello boxes are present at the same location (like a shelf) and do not need the robot to be taught different tasks to reach them.

Block touching -

Here the only mode change is when the blocks are touched. There is no need of a bottleneck state as there are no constraints over the pose of the arm when touching the blocks.

REFERENCES

[1] Christoph Bartneck, Dana Kulić, Elizabeth Croft, and Susana Zoghbi. Measurement instruments for the anthropomorphism, animacy, likeability, perceived intelligence, and perceived safety of robots. *International journal of social robotics*, 1(1):71–81, 2009.

[2] John Brooke. System usability scale (sus): a quick-and-dirty method of system evaluation user information. *Reading, UK: Digital equipment co ltd*, 43:1–7, 1986.

[3] Caelan Reed Garrett, Rohan Chitnis, Rachel Holladay, Beomjoon Kim, Tom Silver, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. Integrated task and motion planning. *Annual review of control, robotics, and autonomous systems*, 4:265–293, 2021.

[4] Nakul Gopalan, Nina Moorman, Manisha Natarajan, and Matthew Gombolay. Negative result for learning from demonstration: Challenges for end-users teaching robots with task and motion planning abstractions. In *Proceedings of Robotics: Science and Systems.*, 2022.

[5] Jakob Nielsen. *Usability engineering*. Morgan Kaufmann, 1994.

[6] Daniel Ullman and Bertram F Malle. What does it mean to trust a robot? steps toward a multidimensional measure of trust. In *Companion of the 2018 acm/ieee international conference on human-robot interaction*, pages 263–264, 2018.

[7] Yu Xiang, Tanner Schmidt, Venkatraman Narayanan, and Dieter Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. *arXiv preprint arXiv:1711.00199*, 2017.

[8] Kai Zhang, Eric Lucet, Julien Alexandre Dit Sandretto, Selma Kchir, and David Filliat. Task and motion planning methods: applications and limitations. In *19th International Conference on Informatics in Control, Automation and Robotics ICINCO 2022)*, pages 476–483. SCITEPRESS-Science and Technology Publications, 2022.