# Robot Self-Initiative and Personalization by Learning through Repeated Interactions

Martin Mason, Manuel Lopes

# Robot Self-Initiative and Personalization by Learning through Repeated Interactions

Martin Mason
Mt. San Antonio College,USA
profmason@gmail.com

Manuel Lopes[*]
Flowers Team, INRIA, France
manuel.lopes@inria.fr

## ABSTRACT

We have developed a robotic system that interacts with the user, and through repeated interactions, adapts to the user so that the system becomes semi-autonomous and acts proactively. In this work we show how to design a system to meet a user's preferences, show how robot pro-activity can be learned and provide an integrated system using verbal instructions. All these behaviors are implemented in a real platform that achieves all these behaviors and is evaluated in terms of user acceptability and efficiency of interaction.

## Categories and Subject Descriptors

I.2.9 [**Computing Methodologies**]: ARTIFICIAL INTELLIGENCE—*Robotics*

## General Terms

Algorithms

## Keywords

Human-Robot Interaction, Learning by Demonstration

## 1. INTRODUCTION

The aim of this project is to create an efficient personal, or service, robot. This robot should be able to: accept commands from a user to achieve general tasks; and to accomplish tasks autonomously. In particular, the robot can anticipate the user's needs by selecting appropriate tasks according to a user profile and context, and then planning the execution of these tasks without an explicit user request. We envision a robot that comes out-of-the-box with the following capabilities: accept user commands, solve complex tasks, learn and *create profiles to anticipate user needs* and *pro-actively* fulfill them using its own reasoning methods.

This project takes a learning approach to the problem of autonomous task selection by modeling the goals of the user during learning and then achieving autonomous execution of the task by selecting relevant goals and planning sequences of actions to achieve them. Learning can be used to allow the robot to autonomously adapt known tasks to new situations but it can also be implemented in the case where new tasks are desired. The problem of solving new tasks is complex and can even be an ill-posed one and past researchers have considered social guidance [21] as a tool to address this complexity. In the social guidance approach, a human reward signal is combined with environmental rewards in order to program a robot to solve new tasks and so adapt it to its own desires and action preferences. The fields of robot imitation, learning from demonstration [14, 2] and most generally social guided learning [19, 5] have addressed the problem of learning new and ill-posed tasks.

In personal and service robots the level of autonomy required, and expected, by the user must take into account the capabilities of the user, the difficulty of the task, the amount of training time, and other user factors. The concept of semi-autonomy has been already analyzed in terms of the tasks that require different levels of autonomy [12] and it has been proposed that higher levels of autonomy can be achieved through user adaptation. Search and rescue scenarios have considered changing degrees of autonomy, a *mixed initiative* of decision making dependent on the context[8], to allow a reduced number of operators to control a large number of robots. Other aspects of adaptation have considered how to learn the preferences for scheduling or shopping behavior [7, 4].

### 1.1 Approach

Figure 1 shows an overview of our system. We consider two main behaviors of the system: a) user instructed behavior (teleoperation) and b) self-initiated behavior (autonomy). Initially the user instructs the robot verbally to accomplish a task. Every time an action is executed the robot memorizes a description of that state and the classification (good or bad) made by the user. If the user does not classify the state, it is assumed to be a bad state. After several interactions, the robot creates a user profile that it can use autonomously. In the second situation, the robot already has a user profile and so can select its own goals without need for user command. After the robot has moved the world to what it considers a desirable state, the user has the option of correcting the robot and classifying that resulting state as a bad state. This new feedback is used to update the user profile and allows the robot to re-select its goal and try to achieve a state that correctly corresponds to a desired

[*]Research partially funded by INRIA.

state for the user. Having acquired this background knowledge, the robot will be able to start acting as required by the user without an explicit user command, e.g. a vacuum cleaner would automatically clean a dirty floor at the time of the day the user typically prefers.

Our system is thus composed of three main components: a) interaction system that allows for an intuitive interaction; b) a user adaptation module that models the user behaviors and task requirements and c) a goal selection system and task planning and execution algorithm to achieve the goals as defined from the user profile. We validate our system

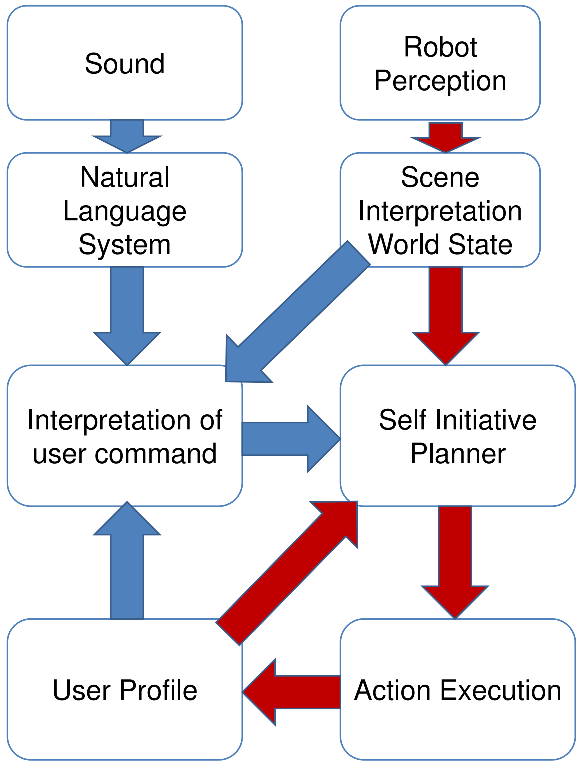

**Figure 1: System overview**

using different user studies. A first study evaluates if our representation is able to describe the user's preferences for our chosen task. A second study studies how the interaction is able to efficiently acquire the required data. The final study evaluates the amount of data required to acquire a good profile and the amount of corrective feedback needed to improve it.

### 1.2 Related Work

Several studies have considered how a robot can learn from humans and achieve increasing levels of autonomy. The most common setting is that where an explicit training phase exists with a large demonstration and an offline learning method that is followed by full autonomy of the robot without further adaptation ([14, 2]).

Recent approaches have consider a more efficient way to provide such training information to either increase the sample efficiency, e.g. reduce the amount and time required for training, and the quality of the solution, e.g. warranty that the number of errors is low. In the case of known tasks, several works have considered how a reinforcement learning agent can be given extra "shaping" information to increase the speed of learning [11, 21]. In these works an extra reward signal is given, asynchronously, to the robot and results showed several interesting aspects of the differences in users' feedback. Other approaches considered the case of unknown tasks where either the policy is directly estimated using either regression or classification methods [3, 6] or using inverse reinforcement learning techniques [15, 9]. In these approaches, the user provides an initial demonstration to the learner and at runtime the user can directly correct the learner either after a learner's request [3, 6] or on their own initiative [11, 21]. The request, or the user's guidance, always refers to the current state. In the work of [15] the learner itself guides the learning process and requests information on states that might differ from the current one. This last approach has the potential benefit of directly obtaining the most useful information but, in particular for the case of physical agents, it can lead to requests for information about unreachable situations.

Our work follows this line of research and shares several features in relation to previous works in terms of acquiring data from a user and accepting correction during acquisition and runtime. Our training system is inspired by a mixed initiative system [8] where at first the robot is tele-operated using verbal commands and with experience it becomes able to select its goals and execute its actions. At any time the user can stop the robot and verbally operate it. We consider the case where the robot is able to solve complex tasks but that it does not know what particular task the user is interested in. Our robot will be initially "micro-managed" through the steps of a given process, but in the end it only considers the high-level results as important for the user profile.

Our system can thus be seen as an inverse reinforcement learning problem with a simplified reward structure, where the final state of a task is achieved but the robot is left free to achieve that final state by its own means using built-in task solving methods [18, 13]. This contrasts with other explicit inverse reinforcement learning approaches, e.g. [1, 20], where a reward is found to explain not only the final state but also the trajectory to reach such state. Such approaches require full trajectories to learn the task and cannot infer a task from simple state classifications. Thus our approach requires less information to be acquired from the user. Also, those projects were developed in an offline setting where a demonstration is acquired and latter processed, not in an interactive scenario.

## 2. DIALOG SYSTEM

A **dialog system** is used to receive the users' commands and clarification responses and to provide cooperative supervision of the task execution. The system is not limited to a fixed sentence structure and allows users to speak naturally to the system. The grammar is defined from the concepts that can be grounded in the robot's own perceptual and motor skills. The system implements an optional confirmation and clarification request system. Although such confirmations must be ensured in critical systems, for personal robots it is a cause of fatigue and makes the interaction very unnatural.

Upon hearing a command the robot executes the most probable inference from the information available, not only the most probable spoken sentence but the one that is more probable taking into account the current world state. This can be seen as a primitive form of self-initiative where the robot executes the most likely command. Initially the robot will behave within the constraints imposed by the average profile we created for the task at hand. After some time decisions by the robot will be aligned with the particular user.

The decision was made to have the system always provide a motor response to any user input without relying on verbal clarification requests. As a result the system will require techniques for dealing with situations where either the user provides incomplete information or the natural language system fails to recognize part of the user's speech. In the domain of the cleaning problem, three techniques were employed to fill in incomplete information. The first technique was to maintain an interaction history and fill in unknown information with values from the previous interaction. The second was to have the robot search for the closest object to the robot and to have the robot interact with that object. Finally, a planner can be used to determine the best possible action (see Section 4 for details of this planner). The current world state was analyzed and an optimal world state was generated. The next step in transitioning from the current world state to the optimal state was used to fill in the incomplete information. All three of these approaches resulted in the robot taking some form of smart action.

## 3. WORLD REPRESENTATION AND USER PROFILING

This Section presents the method for acquiring user profiles from interaction data. We start by describing the world representation and its consequences in terms of generalizing knowledge and transfer between different environments. Then we define what a user profile is and how can it be learned. The following section illustrates how this profile gives robots self-initiative that takes into account user preferences.

### 3.1 World Representation

A very important aspect of the way a robot can adapt to a user is the model the robot uses to represent the world. Most research in *human-robot interaction* consider a propositional representation of the world with a fixed number of objects, each with a known number of properties. This representation is the one typically used in robotics due to the computational efficiency of the algorithms used for learning and planning. Yet such representations are very limiting when the environment changes either in its geometry or in the number and types of objects it contains. A representation that instead considers *predicates* or *relations* between entities can more easily generalize between different domains.

In this work we consider a middle ground and represent the state of the environment as a vector in a feature space. For this we created a large bank of features and used the results of an online survey to select the relevant ones for a particular task. We do not restrict ourselves to having the robot copy the final state of the environment by memorizing the desired locations of objects, but rather have it learn what makes a desired state and what does not. This approach is as rich and as general as the feature set used and so a great deal of expert knowledge goes into the creation of such features. Another option is to create a large bank of features and allow the learning system to automatically select the relevant ones.

In our work we consider several levels of representation. At a low level we consider the locations of objects in world coordinates $x$. This representation is acquired from the robot's sensors and is mainly used for the control and navigation system. At a medium level we consider discrete regions of the world $s$ to plan sequences of actions in a more efficient way. At the higher level we consider the description of the configuration of the objects as a feature vector $\mathbf{f}()$. We consider thus that the environment geometry, the objects' locations and properties and all the relations between the entities is encoded in that set of features. A particular configuration is represent as a point in this feature space $f = \mathbf{f}(x) = \left[\ f_1(x) \dots f_n(x)\ \right]^T$

Within this representation the specific coordinates of objects are lost and so the system will not be able to replicate, in a metric way, an observed configuration. We want a system that is able to generalize among different environments and number of objects. For example, if we have a knife, fork, plate and spoon, we would not store the geometric coordinates of any of the items, but rather that the knife and fork are close together and that the plate is always between the knife and spoon. This would allow for generalization in cases where additional knives, forks or spoons were added to the system and the robot could then organize these items according to the same relational rules.

### 3.2 User Profiling

Each user's request is a demand for the robot to move the world from state $x$ to a new state $y$, implicitly from a less to a more desired state. Reaching the desired state may require several intermediate steps, where the intermediate states may be less desired than the initial state and where the final state is preferred to any of the other. From such sequences we acquire a database of what each user considers to be a good ($l = 1$) or a bad state ($l = 0$). Computationally speaking we have a dataset consisting feature-label pairs, $D = \{(f(x_1), l_1), \dots, (f(x_i), l_i), \dots (f(x_n), l_n)\}$. We assume that the user's preferences can be described as a function that defines whether a given world state is good or bad.

With carefully designed features it is possible to adapt to a great variety of different user preferences. An advantage of this approach is that it is easy to map between different environments as long as the features themselves are able to cope with such variety. For instance if the user explains how to set a table for four people, the robot should be able to set a table for any number of people. In this case the features would describe the relations between each of the objects, cutlery, glasses and plates. We note that this user profile considers only the final results of the tasks that the user instructs the robot to do and ignores the particular sequence of actions the user use to achieve it.

### 3.3 Learning and Generalization

To be able to generalize the information from the user profile to different environments and object configurations we need to learn a classifier. We are interested in computing a classifier that varies along the feature-space previously described. Possible methods for this include kernel logistic regression [23], beta regression [16] and others. We adopt beta regression due to its computational efficiency and its ability to include prior information such as the proportion of good versus bad states.

At each feature vector extracted from a particular world state, the desired classifier is simply a Beta distribution. Using a standard Bayesian approach, we compute a posterior distribution over the probability of that feature being a good state using the corresponding conjugate prior, the Binomial distribution. For notational convenience, let us for the moment denote the parameters of the binomial at a state x as a vector $p(f)$ as a short notation for $p(f(x))$. Using this notation, we want to estimate $P(p(f))$. This posterior distribution describes not only the probability of being a good state but also the confidence on that prediction. Let $fi$ be some state observed in the demonstration $\mathcal{D}$, and let $\alpha(f)$ and $\beta(f)$ denote, respectively the number of times that, in the demonstration, the feature state was classified as a good state, or a bad state, respectively. We have:

$$\begin{aligned} P(p(f)) &\propto \mathsf{Bin}(\alpha(f); \alpha(f) + \beta(f))\mathsf{Beta}(\alpha_0, \beta_0) \\ &= \mathsf{Beta}(p(f); \alpha(f) + \alpha_0, \beta(f) + \beta_0) \end{aligned}$$

In other words, the posterior distribution of $p(f_q)$ is also a Beta distribution with parameters $\alpha(f) + \alpha_0$ and $\beta(f) + \beta_0$. In order to generalize $P(p(f))$ to unvisited states, and following the approach in [16], we assume that the parameters of the Beta distribution depend smoothly on $f$. For this we define a kernel, $k(\cdot, \cdot)$, and use standard kernel regression to extrapolate the parameters of the Beta from the training dataset to unvisited states. Specifically, for any query point $f(x_q)$ and a given training set, we have

$$\begin{aligned} \hat{\alpha}(f_q) &= \sum_{g \in \mathcal{D}} k(f_q, g)\alpha(g) + \alpha_0 \\ \hat{\beta}(f_q) &= \sum_{g \in \mathcal{D}} k(f_q, g)\beta(g) + \beta_0 \end{aligned}$$

Finally, the posterior mean of the distribution over the parameters $p(f)$ – that we will henceforth use as our classifier at $f_q$, is given by $\hat{p} = \frac{\hat{\alpha}_{f_q}}{\hat{\alpha}_{f_q} + \hat{\beta}_{f_q}}$. The algorithm requires setting the parameters $\theta$ that define the kernel $k(\cdot, \cdot)$. We estimate these parameters from the training data using a cross-validation approach. For each point, we compute the posterior distribution. Then, we use a minimum least square criterion between the predicted mean $\hat{p}$ at each point and the observed empirical one $\bar{p} = \frac{\bar{\alpha}}{\bar{\alpha} + \bar{\beta}}$.

$$L_\theta(\mathcal{D}) = \sum_{i=1}^{n} (\hat{p}_i - \bar{p}_i)^2.$$

By minimizing this criteria we can optimize the kernel parameters. In our experiments, we used a Gaussian kernel with a diagonal covariance matrix, that is, with independent bandwidths for each dimension of the features. The function was minimized using the L-BFGS-B algorithm [22] method and positive constraints for the kernel bandwidths. This step is very important to do *feature selection*. As described before, our system will rely on a large bank of features to allow it to adapt to a wider variety of user's preferences. By optimizing the kernels the system is automatically selecting which dimensions provide relevant information and which dimensions are redundant. To improve efficiency, dimensions with very narrow bandwidth can be removed.

## 4. TASK EXECUTION

This Section discusses how the robot plans and executes its actions. We note that even when the robot is following directly the instructions of the user, the robot still requires a high-level of autonomy. Since the user is only able to provide high-level verbal commands, the robot requires complex perceptual and motor actions to understand the commands (as was discussed in 2). When encountering a new situation the robot should proactively act in the world to change it to a more desirable state according to a given user profile.

A user's preferences are described as a function $V$ that assigns to each feature vector $f(x)$, a probability $V(f(x))$ that this feature vector represents a good state for that user. Our problem can then be defined as follows: Taking into account an initial state $f(x)$, find a plan that reaches a more desirable state $f(y)$ such that $V(f(y) > V(f(x))$. In other words, the goal state is any state where the probability of it being a desirable state is high. Within this formulation we can use different optimization methods to find such maxima, either global or local. Different methods will entail different compromises in terms of efficiency and requirements on the motor primitives of the system. The two main methods we consider are to either to just find good (optimal) desired states or to search for a realizable path to a desired state.

### 4.1 Unrestricted Planner

We can consider first a case where we have to find a (global) maximum of the function $V(f(x))$. Note that this is a problem of dimension $N^M$ where $N$ is the number of objects in the space and $M$ is the number of possible locations. A brute force approach will not work because the number of possible solutions grows exponentially. We thus use simulated annealing [10], a global optimization method. This method is able to search the solution space and find a good solution for the problem.

Local methods such as the L-BFGS-B algorithm [22] or other gradient method approaches can also be used but are subject to more local minima as solutions. Choosing between the two approaches (global vs local) involves a compromise between the efficiency (speed) of finding a solution and the quality of the solution found.

### 4.2 Restricted Planner

Unrestricted planners are more efficient because they ignore the restrictions caused by the world dynamics and robot capabilities, and so assume that the robot is able to plan how to reach any goal. After a goal state is selected, a planner must determine a path for the robot to move from its current state to this goal state. If the planner can't find a valid path, then a slightly less optimal goal state must be chosen and the process repeated. In order to avoid this process of having to repeatedly evaluate optimal states to see if they are reachable, a planner that is restricted to only reachable states can be implemented. This second class of methods are much slower but when a desired goal is found, the path to reach it is also found. Another distinction between the two methods is in whether they yield a local maximum or a global maximum.

We consider that we want to find a state that is considered good by the user. Several search methods can be considered [18]. We considered a depth-first search problem where each node represents the locations of objects in discrete cells of the world. If the objects are located in cells $c_i$, then their coordinates are given by $x(c_i)$. The value of each node is the probability that a given node represents a desired world state, $V(f(x(c_i)))$. Figure 2 shows the search method be-

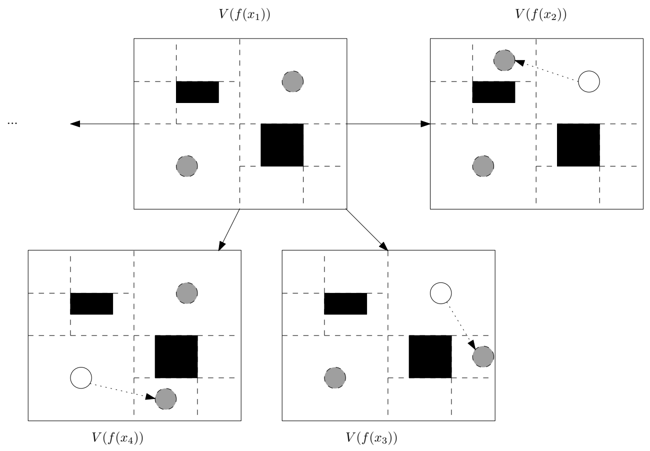

**Figure 2: Search tree. Each square represents a world state with obstacles (black squares) and objects (dark circles), the search system proceeds by generating valid states from previous ones until a state is found that has a high probability of being a target state.**

havior. When a solution is found, the plan to move the world state from the initial state to the final one is just the path of the search. Informed search methods could speed up this process but if the problem has many local minima then it is difficult to create heuristics to guide the search.

## 5. EXPERIMENTAL SETUP

A deliberate decision was made to demonstrate self initiation and autonomous goal selection using a physical system and not a simulation. In a physical system, the robot is able to grasp, move and release objects and the user receives visual and kinesthetic confirmation of the system's performance which increases the engagement of the user with the system. Relying on simulated environments does not allow effective measurement of the quality of interactions due to the different user modalities and expectations. This section presents the experimental setup, the environment, the perceptual and motor capabilities of the robot and the human-robot verbal interface.

### 5.1 Robot and Environment

We consider a robot with a holonomic base and a humanoid torso that is able to grasp and transport release objects, navigate and avoid obstacles, see Figure 3. We rely on an overhead camera and on-board sensors to perform the navigation and object interactions tasks. We considered a simplified context consisting on a simple “Cleaning” task which consists of taking randomly distributed baskets or blocks and redistributing them into a clean or “Tidy” state.

In order to create a simplified relational representation of the world state, we selected a set of thirty six features which were sufficiently rich to represent a wide variety of possible states, yet constrained enough that they could be searched using the power of a modern desktop computer. With three classes of objects the total number of features is 30, and they include: Intra Class position (x and y); Intra

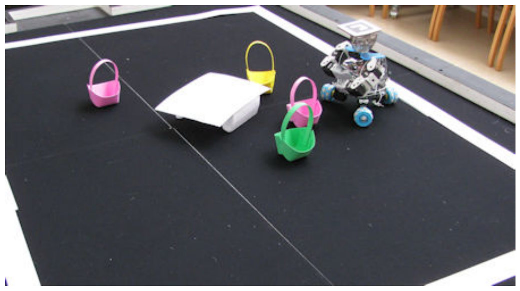

**Figure 3: Environment and robotic setup**

Class dispersion(x and y); Class Distance to four landmarks (x and y); Inter Class distance(x and y).

### 5.2 Dialog System

The Dialog system is implemented using the Nuance SDK [17]. The task domain greatly restricts the dialog. From each utterance the Natural Language Interface (NLI) only needs to extract the following information: Action, Object, Object Description, Target, and Target Description. This information is divided up into the slots by the Natural Language parser of the Nuance system. Due to the self initiation aspect of this project and the Task Based dialog system, each communication does not require all information to be available.

## 6. EXPERIMENTS AND RESULTS

This section shows a series of experiments that validate our approach. We consider four validations: a) the richness of the representation used, b) the effectiveness of the interaction, c) the efficiency of acquisition of a user profile and d) the user's evaluation of the autonomously selected goals.

### 6.1 Establishing User Preferences and Profiles

Since each user could have a different definition of successful goal completion, a survey was designed to establish common features among tidy scenes. This will allow us to know if there is a common concept of ”tidiness” and also if our representation is rich enough to separate the different classes. For the survey ten images were prepared using a mixture of structured scenes and randomly generated scenes. The ten images were presented in random order. Twenty six re-

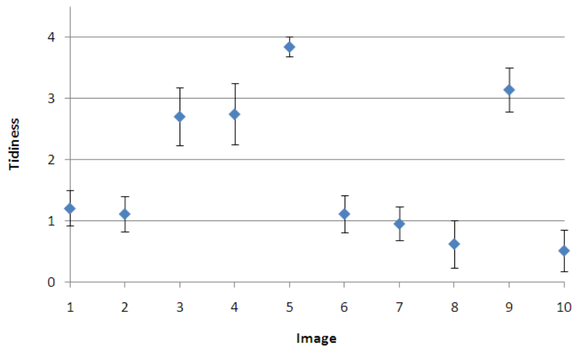

**Figure 4: The tidiness score for each image is calculated. The error bars show the inter-rater reliability of that image.**

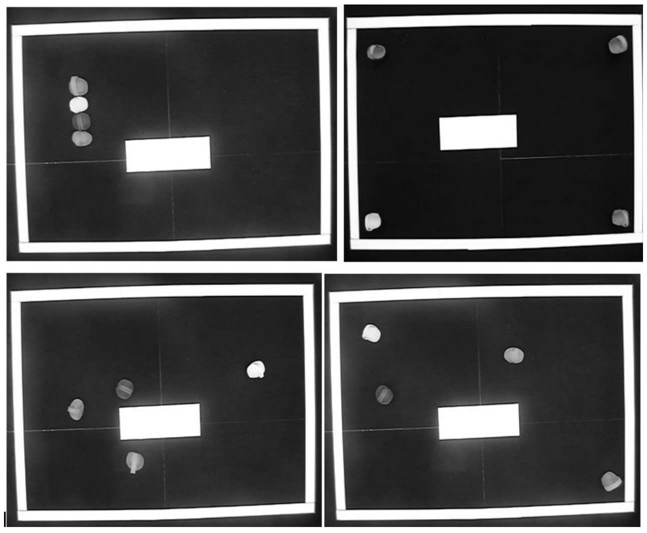

**Figure 5: The two tidiest (top) and least tidy (bottom) images in the survey sample.**

spondents completed the survey selected from past students and members of the Robotics Society of Southern California. The survey respondents ranged in age from six to seventy two with an average age of 41 years and standard deviation of 12 years. The respondents were 82% male. Each participant was asked to rank the 10 images based on the neatness of the scene from very tidy (4) to very untidy (0). After the data was collected, a tidiness score was calculated for each image. Figure 4 shows the results of the survey.

The two tidiest and least tidy images are shown in Figure 5 below. The first image showed an extremely small dispersion between objects. The second image showed all the images located in the corners which meant that the dispersion was maximized while the distance between the objects and the perimeter was minimized. The untidiest images have the objects randomly distributed. A set of feature vectors using the features previously described was calculated for each of the images. It was determined that these features were sufficient to distinguish all the images in the sample using the learning algorithm.

## 6.2 Interaction Validation

This experiment investigated how well a user can instruct the robot. Table 1 summarizes the results. In two separate trials, users were asked to interact with the robot to get it to transition a scene from the same initial state of scattered objects to a tidy state defined as having all the objects in the corner of the scene. In third and fourth trials, users were asked to interact with the robot to transition a simplified scene from an untidy to a tidy state. Survey participants varied in age between 21 and 43, evenly split between male and female, with three of the participants being university students and one a homemaker.

Each participant was told that their goal was to instruct the robot to tidy up the room and that the system could understand a variety of voice prompts. In each of the trials, participants were given the sample prompt "Please pickup the yellow block" as an example of something they could instruct the robot to do. After this prompt, users were allowed to instruct the robot on their own. Below is a portion of one user's dialog:

**Table 1: Validation of the Human Machine Interaction system**

| User Gender | Male | Male | Female | Female |
|---|---|---|---|---|
| Language Native | N | Y | Y | Y |
| # Interactions | 17 | 17 | 10 | 9 |
| Robot Failures | 4 | 2 | 1 | 1 |
| Language Failures | 4 | 1 | 1 | 0 |
| Efficiency | 0.53 | 0.82 | 0.8 | 0.89 |
| Inter. Time (sec) | 301 | 276 | 309 | 289 |

- User: "Get me a block"
- Robot: Proceeds to nearest block and picks it up.
- User: "Put it in the corner"
- Robot: Proceeds to top left corner and drops the block
- User: "Get me the yellow block"
- Robot: Moves to the yellow block and picks it up.
- User: "Put it in the corner, put it in the corner"
- Robot: Proceeds again to the top left corner and drops the block.
- User: "This is tidy"'

Another user chose to issue more detailed commands.

- User: "Get the Pink block and put it in the upper left hand corner.'
- Robot: Moves to the pink block and picks it up.
- User: "Put it in the upper left hand corner"
- Robot: Proceeds to the top left hand corner and drops the block.

The efficiency of the interaction was measured by the ratio of the successful interactions to the total number of interactions required to complete the task. The robot was deemed to have failed the interaction if it either failed to complete the task, the user asked the robot to do something that was outside of its domain, or the robot did something that did not match the users' intention. Each participant rated each interaction as a success or failure. Language failures are defined as failures of the natural language system to parse the user input into usable commands. The exit survey asked users to rate their satisfaction with the system's performance on a scale from zero to four with four being *Very Satisfied*, and resulted in an average user rating of 3.25. We saw that the users were able to command the robot and were satisfied with the behavior of the robot.

## 6.3 Acquiring User Profiles

This experiment evaluated the amount of data required to create a good user profile. We defined two profiles: a) All the objects in the top right corner; b)Pink objects go to the top right corner and all the other objects go to the bottom left corner. To train the profiles, the objects were randomly distributed about the scene and then the robot was instructed to transition the random state, step by step until the scene was in a final state that the user judged matched one of the profile definitions. After each user instruction, the profile was updated and when the user judged that the state was tidy according to the profile definition, the interaction was complete. These interactions were repeated several times to determine how much training was required to generate a good profile. A profile is good if, for a give user and a large set of world states, the system is able to use the profile to

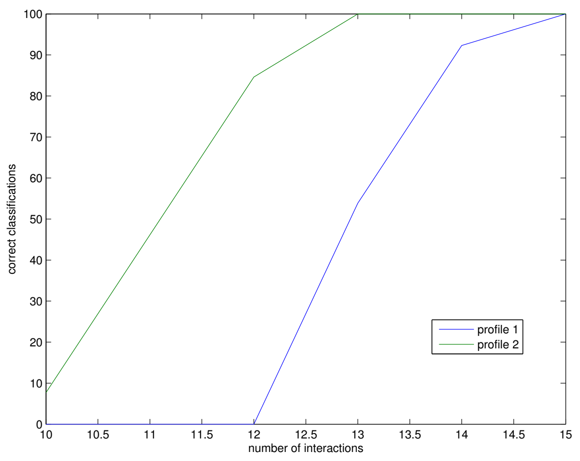

**Figure 6: Profile acquisition**

automatically find a good world state that matches the user classification. After each interaction, the self initiative system was used with the generated profile to find good world states for a set of thirteen test images. The users rated each world state so generated according to how well they matched the original profile. Figure 6 shows the evolution for both profiles. We see that, for our domain, around $10-13$ interactions are sufficient to acquire a good profile where each interaction consists of between $8-10$ user's instructions.

## 6.4 Using User profiles

This experiment validated how well the robot could utilize user profiles to *perform the task uninstructed* and achieve a final state that corresponds to each particular user preferences. Based on the profiles acquired before, we presented the robot with an initial world configuration and the robot determined a plan to reach a suitable final state. The different profiles produced substantially different results that corresponded to the users' preferences, see Figure 7. To achieve the desired state, the robot needed to move all the four objects.

Taking into account our current `Python` implementation, the unrestricted planner was much more effective. With this planner a good plan is found in less than 8 *seconds*. Using the restricted planner with a breath-first search method (see [18]) we were able to plan tasks with 4 objects in an environment where each object could be at 45 different locations, corresponding to more than 3 million states. Typical plans consisted of a sequence of nine actions. However, the restricted planner often took several minutes to find a solution which rendered it inappropriate for use in a human interaction setting. Informed search methods should lead to substantial speed gains for this restricted planner.

## 7. CONCLUSIONS AND DISCUSSION

In this work we presented a robotic system that interacts naturally with a user. The robot is able to learn user profiles (defined as a database of good and bad world states) by acquiring knowledge from the different interactions episodes it has with the user. Using such knowledge the robot is able to pro-actively act in different environments and world configurations taking into account the user's preferences. We have demonstrated results using a real robot platform performing room tidying tasks. Such autonomy goes beyond previous work where autonomy was only considered at the execution level and not at the goal selection level.

We note that this scenario is very simplistic in terms of robotic complexity but no assumptions is made about the particular scenario. The description of the world in terms of features is very general, particularly when considering that we automatically perform feature selection, and current planning systems and optimization methods can easily deal with problems much larger than the one showed. Our physical implementation tested the learning system with user interactions in the presence of motor and perceptual errors that, if large, might reduce the user acceptability and the quality of the learning profiles. Results showed that the system worked even with such sources of noise. In a simulated world such evaluation would be difficult to make.

There exits systems that learn using information from interactions with a user. The choice among the different approaches will have to take into account the capabilities of the user, the type of task and the a-priori knowledge and skills of the robot. If the system needs to learn not only the task but also how to solve it then our approach can not be used and approaches such as [1, 20] or [3, 6] will better if, respectively, the reward or the policy have simpler representations. In those approaches it is also assumed that the user is able to provide either trajectories and/or the correct action. Our approach assumes that the robot is equipped with planning skills and that the user is only able to provide information about the required goal states. For instance learning how to grasp an object will be much better addressed with other methods because they include a more detailed trajectory matching. On the other side, learning the preferences about travel destinations, shopping bags contents, or the comfort levels of light and temperature in a room will be better addressed with our system because the means to achieve such goals are irrelevant to the preferences of the users.

As future work we intend to improve the efficiency of the restricted planner using other hybrid methods, preferably allowing the robot to learn the world model. Adding more modalities, such as pointing, to the interface will increase the efficiency of the system. Another interesting new prospect for this approach is to consider a joint-work problem where the robot and the human must accomplish the task cooperatively. Finally, this work can be adapted to a wide variety of human interactions where the task can be defined in terms of relationships between objects or primitives and their environment.

## 8. REFERENCES

[1] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the 21st International Conference on Machine Learning (ICML'04)*, pages 1–8, 2004.

[2] B. Argall, S. Chernova, and M. Veloso. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469–483, 2009.

[3] S. Chernova and M. Veloso. Learning equivalent action choices from demonstration. In *Intelligent Robots and Systems, 2008. IROS 2008. IEEE/RSJ International Conference on*, pages 1216–1221. IEEE, 2008.

[4] M. desJardins, E. Eaton, and K. Wagstaff. Learning user preferences for sets of objects. In *Proceedings of*

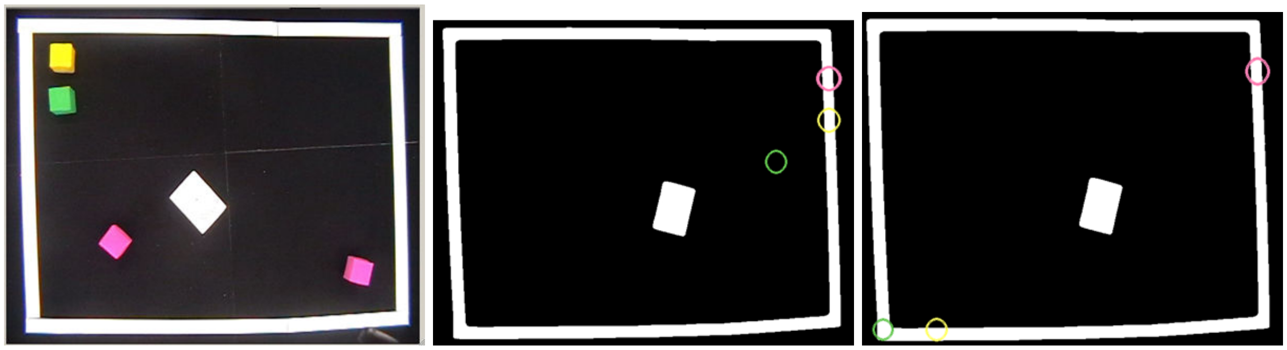

**Figure 7: Final states reached by the robot from the same initial state (top) taken into account two different user profiles: (middle) *all object in a corner*, and (right) *pink objects in bottom left and all other in top corner.***

*the 23rd international conference on Machine learning*, page 280. ACM, 2006.

[5] T. W. Fong, I. Nourbakhsh, and K. Dautenhahn. A survey of socially interactive robots. *Robotics and Autonomous Systems*, 2003.

[6] D. Grollman and O. Jenkins. Dogged learning for robots. In *Robotics and Automation, 2007 IEEE International Conference on*, pages 2483–2488, 2007.

[7] Y. Guo and C. Gomes. Learning optimal subsets with implicit user preferences. In *Proceedings of the 21st international joint conference on Artifical intelligence*, pages 1052–1057, 2009.

[8] B. Hardin and M. Goodrich. On using mixed-initiative control: a perspective for managing large-scale robotic teams. In *Proceedings of the 4th ACM/IEEE international conference on Human robot interaction*, pages 165–172. ACM, 2009.

[9] R. Jaulmes, J. Pineau, and D. Precup. Active learning in partially observable markov decision processes. In *NIPS Workshop on Value of Information in Inference, Learning and Decision-Making*, 2005.

[10] S. Kirkpatrick, C. Gelatt Jr, and M. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671, 1983.

[11] W. B. Knox and P. Stone. Combining manual feedback with subsequent mdp reward signals for reinforcement learning. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems (AAMAS'10)*, pages 5–12, 2010.

[12] C. Lachi, G. Teti, G. Tambumni, E. Datteri, and P. Dario. Adaptable semi-autonomy in personal robots. In *IEEE International Worksop on Robot and Human Interactive Communication*, 2001.

[13] S. LaValle. *Planning algorithms.* Cambridge Univ Pr, 2006.

[14] M. Lopes, F. Melo, L. Montesano, and J. Santos-Victor. Abstraction levels for robotic imitation: Overview and computational approaches. In O. Sigaud and J. Peters, editors, *From Motor to Interaction Learning in Robots.* Springer, 2009.

[15] M. Lopes, F. S. Melo, and L. Montesano. Active learning for reward estimation in inverse reinforcement learning. In *European Conference on Machine Learning (ECML/PKDD)*, Bled, Slovenia, 2009.

[16] L. Montesano and M. Lopes. Learning grasping affordances from local visual descriptors. In *IEEE 8TH International Conference on Development and Learning*, China, 2009.

[17] Nuance. *Nuance Speech Recognition System Version 8.5*, 2004.

[18] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach.* Prentice Hall, second edition, 2003.

[19] K. Severinson-Eklundh, A. Green, and H. Hüttenrauch. Social and collaborative aspects of interaction with a service robot. *Robotics and Autonomous Systems*, 42(3-4):223–234, 2003.

[20] D. Silver, J. A. Bagnell, and A. Stentz. Learning from demonstration for autonomous navigation in complex unstructured terrain. *International Journal of Robotics Research*, 29(1):1565 – 1592, October 2010.

[21] A. L. Thomaz and C. Breazeal. Teachable robots: Understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence Journal*, 172:716–737, 2008.

[22] C. Zhu, R. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-BFGS-B: Fortran subroutines for large-scale bound-constrained optimization. *ACM Transactions on Mathematical Software*, 23(4):550–560, 1997.

[23] J. Zhu and T. Hastie. Kernel logistic regression and the import vector machine. In *Adv. Neural Information Proc. Systems*, pages 1081–1088, 2002.