Learning Trustworthy Behaviors Using an Inverse Trust Metric

Michael W. Floyd, Michael Drinkwater, and David W. Aha

Abstract The addition of a robot to a human team can be beneficial if the robot can perform important tasks, provide additional skills, or otherwise help the team achieve its goals. However, if the human team members do not trust the robot they may underutilize it or excessively monitor its behavior. We present an algorithm that allows a robot to estimate its trustworthiness based on interactions with a team member and adapt its behavior in an attempt to increase its trustworthiness. The robot is able to learn as it performs behavior adaptation and increase the efficiency of future adaptation. We compare our approach for inverse trust estimation and behavior adaptation to a variant that does not learn. Our results, in a simulated robotics environment, show that both approaches can identify trustworthy behaviors but the learning approach does so significantly faster.

Key words: inverse trust, behavior adaptation, case-based reasoning

1 Introduction

The addition of a robot to a human team can be beneficial if the robot improves the team's sensory capabilities, performs new tasks, or allows for operation in harsh en-

Michael W. Floyd
Knexus Research Corporation, Springfield, Virginia, USA
e-mail: michael.floyd@knexusresearch.com

Michael Drinkwater
Knexus Research Corporation, Springfield, Virginia, USA
e-mail: michael.drinkwater@knexusresearch.com

David W. Aha
Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory (Code 5514), Washington, DC, USA
e-mail: david.aha@nrl.navy.mil

vironments (e.g., rough terrain or dangerous situations). This may allow the team to better achieve their goals, improve team productivity, or reduce the risk to humans. In some situations, it may not be possible to achieve team goals or guarantee human safety without the robot. However, in order to adequately use the robot the human teammates will need to trust it.

The need to trust a robot teammate is especially true when the robot operates in an autonomous or semi-autonomous manner. In these situations, a human operator would issue a command or delegate a task to the robot and the robot would act on its own to complete its assignment. A lack of trust in the robot could result in the operator underutilizing the robot (i.e., not assigning it tasks it is capable of completing), excessively monitoring the robot's actions, or not using the robot at all (Oleson et al, 2011). Any of these issues could result in an increased workload for human teammates or the possibility of the team being unable to achieve their goal.

One possibility would be to design a robot that is guaranteed to operate in a trustworthy manner. However, this may be impractical if the robot is expected to handle changes in operators, environments, or mission contexts. These changes would make it impractical to elicit a complete set of rules for trustworthy behavior. Additionally, the way in which an operator measures its trust in the robot may be user-dependent, task-dependent, or time-varying (Desai et al, 2013). For example, if the robot received a command to navigate between two locations in an urban environment, one operators might prefer the task be performed as quickly as possible whereas another might prefer the task be performed as safely as possible (e.g., not driving down a road with heavy automobile traffic or potholes). Each of these operators has distinct preferences for how the robot should perform the task, which may conflict, and these preferences will influence how trustworthy they find the robot's behavior. Even if these preferences were known in advance, a change in context could influence the operator's preferences and what is considered trustworthy behavior. The operator who preferred the task to be performed quickly would likely change their preference if the robot was transporting hazardous material, whereas the operator who preferred safety would likely change their preference in an emergency situation.

For a robot to behave in a trustworthy manner regardless of the operator, environment, or context, it must be able to evaluate its trustworthiness and adapt its behavior accordingly. The workload of the human teammates or time-critical nature of the team's mission may make in difficult to get explicit feedback from the operator about the robot's trustworthiness. Instead, the robot will use information from the standard interactions it has with the operator (i.e., being assigned tasks and performing those tasks). Such an estimate, which we refer to as an *inverse trust estimate*, differs from traditional computation trust metrics in that it measures how much trust another agent has in the robot rather than how much trust the robot has in another agent. Additionally, the inverse trust metric does not directly measure trust, since the information necessary to compute such a metric is internal to the operator, but instead estimates trust based on observable factors that are known to influence trust. In this chapter we examine how a robot can estimate the trust an operator has in it, adapt its behavior to become more trustworthy, and learn from previous adaptations so it can find trustworthy behaviors more quickly in the future.

In the remainder of this chapter we describe our inverse trust estimate and how the robot uses it to adapt its behavior. In Section 2 we examine related work on human-robot trust and adapting to user preferences. We define the robot's behavior and the aspects that it can modify in Section 3. Section 4 presents the inverse trust metric and Section 5 describes how that metric is used by the robot to guide behavior adaptation. An evaluation of trust-guided behavior adaption, in a simulated robotics domain, is provided in Section 6 and reports evidence that it can efficiently adapt the robot's behavior to align with the operator's preferences. Concluding remarks and potential areas for future work are presented in Section 7.

2 Related Work

Traditional computational trust metrics are used to measure the trust an agent should have in other agents (Sabater and Sierra, 2005). The agent determines another agent's trustworthiness based on prior interactions or using feedback from peers (Esfandiari and Chandrasekharan, 2001). However, these metrics are not applicable when attempting to determine how trustworthy an agent is in the eyes of another agent. The primary reason for this is because the agent will not have all of the other agent's internal reasoning information available to it (e.g., outcome of past interactions, peer feedback, past experiences, internal model of trust). Instead, the agent will need to acquire a subset of this information and use that to infer trust. In the remainder of this section we will examine factors influencing trust in human-robot interaction and how agents can adapt their behavior to humans.

2.1 Human-Robot Trust

Factors that influence human-robot trust can be grouped into three main categories (Oleson et al, 2011): robot-related factors (e.g., performance, physical attributes), human-related factors (e.g., engagement, workload, self-confidence), and environmental factors (e.g., group composition, culture, task type). While numerous factors have been found to influence human-robot trust (e.g., Li et al (2010), Kiesler et al (2008), Biros et al (2004)), a meta-analysis of numerous studies found the strongest indicator of trust is the robot's performance (Hancock et al, 2011). Similarly, user's identified performance as being among the most important factors they considered in relation to automated cars and medical diagnosis (Carlson et al, 2014).

Kaniarasu et al (2012) have examined the topic of inverse trust and use an online, performance-based measure to identify decreases in trust. Their measurement is based on the number of times a human takes control of the robot or warns the robot it is behaving poorly. They have extended this work to also identify increases in trust, but it requires direct feedback from the operator at regular intervals (Kaniarasu et al, 2013). Saleh et al (2012) have also proposed a measure of inverse trust using a set of expert-authored rules. However, if the robot does not have access to direct feedback or predefined rules, these metrics would not be appropriate to use.

2.2 Behavior Adaptation

Shapiro and Shachter (2002) discuss why an agent with a reward function that is similar to the utility of the user is desirable; it ensures the agent acts in the interests of the user. The agent may need to perform behavior that appears to be sub-optimal if it better aligns with the preferences of the user. Their work involves identifying the underlying influences of the user's utility and modifying the agent's reward function accordingly. This is similar to our own work in that the agent is willing to behave sub-optimally in order to align with the user's preferences, but our robot is not given an explicit model of the user's reasoning process.

Conversational recommender systems (McGinty and Smyth, 2003) use interactions with a user to tailor recommendations to the user's preferences. These systems make initial recommendations and then iteratively improve the recommendations through a dialog with the user. As the system learns the user's preferences through feedback, a model of the user is continuously refined. In addition to learn a user preference model, conversational recommender systems can also learn preferences for how the dialog and interactions should occur (Mahmood and Ricci, 2009). Similarly, search systems have been developed that update their behavior based on a user's preferred search results (Chen et al, 2008). These systems use information from the links a user clicks to infer their preferences and update search rankings accordingly.

Learning interface agents assist users when performing a task (e.g., e-mail sorting (Maes and Kozierok, 1993), schedule management (Horvitz, 1999), note taking (Schlimmer and Hermens, 1993)). These systems observe how users perform certain tasks and learn their preferences. Since these agents are meant to be assistive, rather than autonomous, they are able to interact with the user to get additional information or verify if an action should be performed. Similar to conversational recommender systems, learning interface agents are designed to be assistive with one specific task. In contrast, our robot does not know in advance the specific task it will be performing so it can not bias itself toward learning preferences for that task.

Preference-based planning (Baier and McIlraith, 2008) involves incorporating user preferences into automated planning tasks. These preferences are usually defined a priori but there has also been work to learn the planning preferences (Li et al, 2009). This approach learns the probability of a user performing actions based on the previous plans that user has generated. In our work, that would be equivalent to the operator controlling the robot and providing numerous demonstrations of the task. Such an approach would not be practical if there were time constraints or the operator did not have a fully constructed plan for how to perform the task. For example, the operator might have general preferences for how the robot should move between two locations without knowing the exact route it should take.

Our work also has similarities to other areas of learning during human-robot interaction. When a robot learns from a human, it is often beneficial for the robot to understand the environment from the perspective of the human (Berlin et al, 2006). Breazeal et al (2009) have examined how a robot can learn from a cooperative human teacher by mapping its sensory inputs to how it estimates the human is viewing the environment. This allows the robot to learn from the viewpoint of the teacher and possibly discover information it would not have noticed from its own viewpoint. This is similar to our own work since it involves inferring information about the reasoning of a human. However, like preference-based planning, it involves observing a teacher demonstrate a specific task.

3 Agent Behavior

We assume that the robot, in addition to being autonomous or semi-autonomous, has the ability to control and modify some aspects of its behavior. This could include selecting among comparable algorithms (e.g., switching the path planning algorithm it uses), modifying parameter values, or changing the data it uses (e.g., using an environment map from an alternate source). We call these the *modifiable components* of the robot's behavior.

We define each modifiable component i to have a set of selectable values $\mathcal{C}_i$. If the robot has m modifiable components, its current behavior B is a tuple containing the currently selected value c_i for each modifiable component ($c_i \in \mathcal{C}_i$):

$$B = \langle c_1, c_2, \ldots, c_m \rangle$$

By changing one or more of these components, the robot can immediately influence its behavior by switching from its current behavior B to a new behavior B_{new}. These changes can occur multiple times over the course of operation, resulting in a sequence of behaviors $\langle B_1, B_2, \ldots, B_n \rangle$ that have been used. Since the robot is motivated to perform trustworthy behavior, behavior changes will occur because the current behavior B was found to be untrustworthy (or it anticipates the behavior will be untrustworthy given a change in the team's goals or mission context), at which time we want the robot to perform what it believes to be a more trustworthy behavior.

4 Inverse Trust Estimate

Traditional trust metrics, which measure how much trust an agent has in other agents, use information related to previous interactions with those agents or feedback from others to compute trustworthiness (Sabater and Sierra, 2005). This information is likely internal to the agent and would not be accessible to other agents (both the agent whose trustworthiness is being measured and external observers). In a robotics context, the robot would not have access to the information the operator uses to measure the robot's trustworthiness. If the robot wanted to estimate its own trustworthiness, it would need a method to access the operator's beliefs.

One option would be to elicit explicit feedback from the operator about the trustworthiness of the robot, either at run-time (Kaniarasu et al, 2013) or after the task has been completed (Jian et al, 2000; Muir, 1987). However, this might not be practical if the operator does not have time to provide feedback (e.g., a time-critical mission with a heavy operator workload) or there would be a significant delay in providing feedback (e.g., at the end of a multi-day search and rescue mission). In these situations, even though the operator can not explicitly tell the robot how trustworthy it is, it would still be beneficial for the robot to *infer* its trustworthiness.

Without full knowledge about how the operator measures trust or the necessary internal information to actually compute the trust value, the robot needs to rely on observable *evidence* of trust. As we discussed previously, there are numerous factors that have been found to influence a human's trust in a robot (Oleson et al, 2011). If the robot can directly observe some of these factors, it can attempt to estimate its own trustworthiness. However, some factors may not be easily observable or have clear models of how they influence trust (e.g., the physical appearance of the robot).

One factor that is observable to the robot and has been found to be the strongest indicator of human-robot trust is the robot's performance (Hancock et al, 2011; Carlson et al, 2014). The inverse trust estimate we present is based on the robot's performance and uses the number of times the robot completes an assigned task, fails to complete a task, or is interrupted while performing a task. The robot assumes that the operator will be satisfied with any completed tasks (i.e., the robot is performing well) and unsatisfied when tasks are failed or must be interrupted (i.e., the robot is performing poorly[1]). Task completion and interruption have been found to align with with changes in operator trust (based on user feedback (Kaniarasu et al, 2013) and post-run surveys (Kaniarasu et al, 2012)), so it serves as a viable option for the robot to estimate its own trustworthiness.

Our inverse trust estimate monitors whether trust is increasing, decreasing, or remaining constant while the current behavior B' is being used by the robot. We estimate this value as follows:

$$Trust_{B'} = \sum_{i=1}^{n} w_i \times cmd_i,$$

where there were n commands issued to the robot while it was using the behavior B'. If the ith command ($1 \leq i \leq n$) was interrupted or failed it will decrease the

[1] An interruption could also be a result of the operator identifying a more important task for the robot to perform or failures could be the result of unachievable tasks. The robot works under the assumption that those situations occur rarely and most failures/interruptions are a result of poor performance.

trust estimate and if it was completed successfully it will increase the trust estimate ($cmd_i \in \{-1, 1\}$). The ith command also receives a weight w_i which denotes the relative importance of that command (e.g., a command that resulted in poor performance would likely be given less weight than a command that resulted in the robot injuring a human). While our inverse trust estimate uses a simple step function to represent the current estimate of trust, a more complex (or cognitively plausible) function could be used that more closely aligns with the operator's actual trust. However, the additional computation complexity of such a function might not provide additional benefits if, like with our robot, we seek general trends in trustworthiness rather than an exact trust value.

5 Trust-guided Behavior Adaptation

The robot uses the inverse trust estimate to infer if its current behavior is trustworthy, untrustworthy, or it does not yet know. Since the trust estimate is being updated over time (after each success, failure, or interruption) the robot continuously monitors the estimate and compares it to two threshold values: the trustworthy threshold (τ_T) and the untrustworthy threshold (τ_{UT}).

If the trust estimate is between the two threshold ($\tau_{UT} < Trust_{B'} < \tau_T$), the robot will not make any conclusions and will continue to monitor its trustworthiness. However, if the trust estimate reaches the trustworthy threshold ($Trust_{B'} \geq \tau_T$), the robot will conclude it has found a sufficiently trustworthy behavior. The robot will continue to use its current behavior, since it is believed to be trustworthy, but may continue to measure trustworthiness in case any changes occur (e.g., a new operator or mission goals). Finally, if the trust estimate falls to or below the untrustworthy threshold ($Trust_{B'} \leq \tau_{UT}$), the robot will conclude that it's current behavior is untrustworthy and should be changed. In this situation, the robot will perform behavior adaptation to switch to a new behavior.

Figure 1 shows an example of the robot's estimate of the trustworthiness of its current behavior. The robot initially starts from a baseline value, since it does not know if the behavior is trustworthy or untrustworthy, and updates the estimate as new evidence becomes available. In this example, the robot was issued five tasks to perform. The robot completed the first two tasks successfully (as indicated by the increases in the trust estimate), failed or was interrupted during the third task (as indicated by the decrease in the trust estimate), and then successfully completed the fourth and fifth tasks. The robot's trust estimate is trending upwards, but since it is still between the two thresholds it can not conclude if its behavior is trustworthy or untrustworthy.

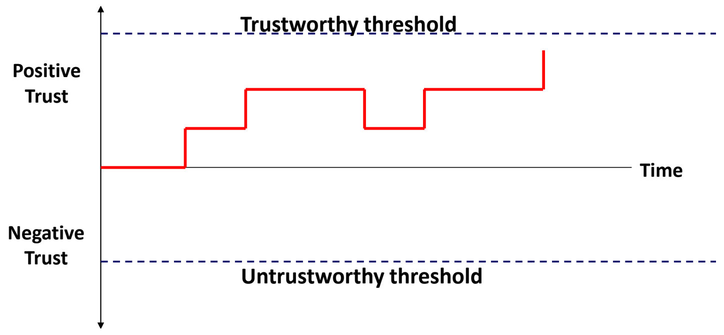

Fig. 1 An example of the robot's trust estimate after being issued five tasks.

5.1 Evaluated Behaviors

When a behavior B is found to be untrustworthy (i.e., the trust estimate reached the untrustworthy threshold), it is stored as an *evaluated pair* E that also contains the time t it took the behavior to be labeled as untrustworthy:

$$E = \langle B, t \rangle$$

The motivation for storing the time it took to label a behavior as untrustworthy, instead of only storing the behavior itself, is that it allows for a comparison between untrustworthy behaviors. This permits a relative level of untrustworthiness so that we can say one behavior is closer to being trustworthy than another. A behavior B' that reaches the untrustworthy threshold more quickly than another behavior B'' ($t' < t''$) is defined to be less trustworthy than the other. This is based on the assumption that if a behavior took longer to reach the untrustworthy threshold then it was either performing some trustworthy actions, was not failing as quickly, or was appearing to behave trustworthy for longer periods of time.

The robot maintains a set $\mathcal{E}_{past}$ of previously evaluated behaviors. This set, which is initially empty, is extended as the robot evaluates more behaviors. If the robot has found n behaviors to be untrustworthy then $\mathcal{E}_{past}$ will contain n evaluated behaviors ($\mathcal{E}_{past} = \{E_1, E_2, \ldots, E_n\}$). However, if the robot determines that a behavior B_{final} is trustworthy (i.e., the trustworthy threshold was reached), that behavior will not be added to the set and the robot will not change its behavior.

The set $\mathcal{E}_{past}$ can be thought of as the search path that was taken to find the trustworthy behavior B_{final}. This can potentially be useful if the robot is performing a new search for a trustworthy behavior (i.e., because of a new operator, mission, or context) and is able to reuse information from the previous search. For example, if two operators find similar behaviors untrustworthy in a similar amount of time, they might also find similar behaviors to be trustworthy.

To make use of information from previous behavior adaptation, we employ case-based reasoning (CBR) (Richter and Weber, 2013). CBR embodies the idea that similar problems tend to have similar solutions. Problem-solution pairs, called *cases*, represent examples of concrete problem solving instances and are stored in a *case base*. Each case C is a pair containing a problem and its solution. In our context, the *problem* is the set of previously evaluated behaviors $\mathcal{E}_{past}$ and the *solution* is the final trustworthy behavior B_{final}:

$$C = \langle \mathcal{E}_{past}, B_{final} \rangle$$

The case base, which is initially empty, grows each time a new case is created. Since each case represents a single problem-solving episode (i.e., finding a trustworthy behavior for an operator in a given context), the case base represents all of the problem solving experience that the robot has collected.

5.2 Behavior Adaptation

Behavior adaptation, which we have only described abstractly to this point, is performed when the currently evaluated behavior reaches the untrustworthy threshold and the robot needs to select a new behavior to perform. The new behavior B_{new} is selected as a function of the set of previously evaluated behaviors $\mathcal{E}_{past}$ and the robot's case base CB:

$$B_{new} = selectBehavior(\mathcal{E}_{past}, CB)$$

The *selectBehavior* function (Algorithm 1) searches for a case C_i in CB with a set of evaluated behaviors that is most similar to $\mathcal{E}_{past}$. The motivation for this is that if they have similar problems then they might have similar solutions, so the robot can adapt its behavior by switching to the final behavior stored in C_i.

Algorithm 1: Selecting a new behavior

```
Function: selectBehavior($\mathcal{E}_{past}$, CB) returns $B_{new}$;
1  $bestSim \leftarrow 0$; $B_{best} \leftarrow \varnothing$;
2  foreach $C_i \in CB$ do
3      if $C_i.B_{final} \notin \mathcal{E}_{past}$ then
4          $sim_i \leftarrow sim(\mathcal{E}_{past}, C_i.\mathcal{E}_{past})$;
5          if $sim_i > bestSim$ then
6              $bestSim \leftarrow sim_i$;
7              $B_{best} \leftarrow C_i.B_{final}$;
8  if $B_{best} = \varnothing$ then
9      $B_{best} \leftarrow modifyBehavior(\mathcal{E}_{past})$;
10 return $B_{best}$;
```

The algorithm iterates through each case in the case base (line 2) and checks to see if the case's final behavior has already been evaluated (line 3). This check is done to ensure that behaviors that have already been found to be untrustworthy are not evaluated again. The sets of evaluated behaviors of the remaining cases are compared to the robot's current set of evaluated behaviors using a similarity metric (line 4). The most similar case's final behavior is stored (lines 5-7) and returned to the robot (line 10). This behavior is immediately used by the robot and the robot begins measuring the trustworthiness of that behavior. If no similar cases were found (i.e., the case base was empty or the final behaviors of all cases have already been evaluated), the *modifyBehavior* function is used to select the next behavior to perform (line 9).

The *modifyBehavior* function selects an evaluated behavior E_{max} that took the longest to reach the untrustworthy threshold ($\forall E_i \in \mathcal{E}_{past}, E_{max}.t \geq E_i.t$). A random walk (without repetition) is performed to find a behavior B_{new} that requires the minimum number of changes to $E_{max}.B$ and has not already been evaluated ($\forall E_i \in \mathcal{E}_{past}, B_{new} \neq E_i.B$). This is based on the assumption that E_{max} is the least untrustworthy of the evaluated behaviors and that a slight change might lead to a more trustworthy behavior. If all possible behaviors have been evaluated and found to be untrustworthy, the robot will stop adapting its behavior and use $E_{max}.B$.

Algorithm 1 relies on calculating the similarity between two sets of evaluated behaviors (line 4). This similarity (Algorithm 2) is complicated by the fact that the sets may vary in size. This occurs because the number of evaluated behaviors in each case is dependent on how long the search took in that instance. Similarly, there is no guarantee that the same behaviors were evaluated in each set. To account for this, the similarity function looks at the overlap between the two sets and ignores behaviors that have only been evaluated in one set. The algorithm goes through each evaluated behavior in the first set (line 2) and finds the most similar evaluated behavior E_{max} in the second set (line 3). The similarity between two behaviors is a function of the similarity of each behavior component:

$$sim(B_1, B_2) = \frac{1}{m} \sum_{i=1}^{m} sim(B_1.c_i, B_2.c_i),$$

where the similarity function for each behavior component will depend on its specific type. For example, a behavior component that represents a binary parameter value would require a different similarity function than a component that represents which path planning algorithm to use.

If the two evaluated behaviors, E_i and E_{max}, are sufficiently similar, based on a threshold λ (line 4), then the similarity of their time components are included in the similarity calculation (line 5). This ensures that the final similarity value includes information from only behaviors that have a highly similar counterpart in the other set. This function will return a high similarity (up to a maximum of 1.0) when similar behaviors took nearly the same time to reach the untrustworthy threshold and a low similarity (to a minimum of 0.0) when similar behaviors had a noticeable difference in the time they took to reach the untrustworthy threshold.

Algorithm 2: Similarity between sets of evaluated behaviors

```
Function: sim($\mathcal{E}_1$, $\mathcal{E}_2$) returns sim;
1 $totalSim \leftarrow 0$; $num \leftarrow 0$;
2 foreach $E_i \in \mathcal{E}_1$ do
3     $E_{max} \leftarrow \arg\max_{E_j \in \mathcal{E}_2} (sim(E_i.B, E_j.B))$;
4     if $sim(E_i.B, E_{max}.B) > \lambda$ then
5         $totalSim \leftarrow totalSim + sim(E_i.t, E_{max}.t)$;
6         $num \leftarrow num + 1$;
7 if $num = 0$ then
8     return 0;
9 return $\frac{totalSim}{num}$;
```

6 Evaluation

In this section, we evaluate our behavior adaptation technique in a simulated robotic environment. Two variations of trust-based behavior adaptation are used: case-based behavior adaptation and random walk behavior adaptation. While we expect both approaches to allow the robot to adapt to trustworthy behaviors, we will evaluate our claim that the case-based approach can find trustworthy behaviors more efficiently.

6.1 eBotWorks Simulator

Our evaluation uses the eBotWorks simulation environment Knexus Research Corporation (2013), a multi-agent simulation engine and testbed for unmanned systems. In eBotWorks, autonomous agents control simulated robotics vehicles and can receive multimodal commands from human operators. We chose to use eBotWorks based on its flexibility in autonomous behavior modeling, ability to interact with agents using natural language commands, and built-in experimentation and data collection capabilities.

In our experiments, we use a single robot that is a wheeled unmanned ground vehicle (UGV). The robot uses eBotWorks' built-in natural language processing (for interpreting user commands), sensing, and path-planning modules. The environment is composed of landmarks (e.g., roads, various types of terrain) and objects (e.g., houses, humans, vehicles, road barriers). The actions performed by the robot are non-deterministic and the robot also suffers from limited observability and potential sensor errors.

6.2 Experimental Conditions

Our initial study uses simulated operators to facilitate a larger-scale evaluation than if real human operators were used[2]. The simulated operators were selected to represent a subset of the control strategies used by human operators. Each simulated operator has unique preferences for how the robot should behave and these preferences will influence how the robot's performance is evaluated (i.e., when the operator allows the robot to complete a task and when it interrupts).

Each experiment is composed of 500 *trials* and in each trial the robot interacts with a single simulated operator. At the start of a trial, the robot randomly selects (with a uniform distribution) initial values for each of its modifiable components. Throughout the trial, a series of experimental *runs* occur. Each run involves the operator issuing a single command to the robot and monitoring the robot as it performs the task. During a run the robot will complete the task, fail to complete the task, or be interrupted by the operator; it will update its trust estimate accordingly. At the end of each run the environment is reset and a new run begins. A trial concludes when the robot has either found a trustworthy behavior or evaluated all possible behaviors.

The case-based behavior adaptation approach starts each experiment with an empty case base. A case is stored at the end of a trial if the robot found a trustworthy behavior and performed at least one random walk adaptation (i.e., the robot could not find a solution in its case base so it used the *modifyBehavior* function). This case retention strategy is used to prevent adding redundant cases. An added case can be used during any of the subsequent trials in the experiment.

The robot's trustworthy threshold was set to $\tau_T = 5.0$ and its untrustworthy threshold set to $\tau_{UT} = -5.0$. These thresholds were set to allow some fluctuation between increasing and decreasing trust while still identifying trustworthy and untrustworthy behaviors quickly. When calculating the similarity between sets of evaluated behaviors, a similarity threshold of $\lambda = 0.95$ was used (i.e., behaviors must be at least 95% similar to be matched).

6.3 Evaluation Scenarios

We selected two scenarios of increasing complexity: *movement* and *patrolling for threats*. While the Movement scenario is a relatively simple task, the Patrol scenario requires a more complex behavior with a larger set of modifiable components.

[2] We plan to validate these findings in a series of user studies.

6.3.1 Movement Scenario

The initial task the robot is required to perform involves moving between two locations in the environment (Figure 2). The simulated operator issues natural language commands to tell the robot where to move (e.g., “move to the flag”) and the robot is responsible for navigating to that location. Three metrics are used by the operators to assess the robot’s performance:

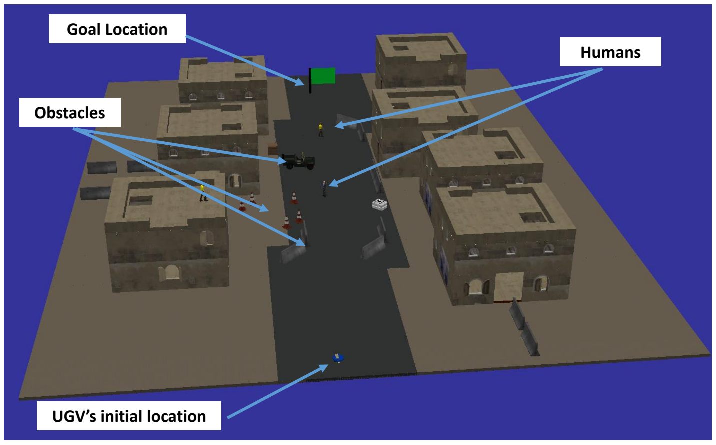

Fig. 2 The environment configuration for the Movement scenario.

- **Task Duration**: The operator has an expectation about the amount of time the task should take to complete ($t_{complete}$). If the robot does not complete the task within that time, the operator may, with probability p_α, interrupt the robot.
- **Task Completion**: If the operator determines that the robot has failed to complete the task (e.g., the robot is stuck or moved to the wrong location), the robot will be interrupted.
- **Safety**: The operator may interrupt the robot, with probability p_γ, if the robot collides with any obstacles.

We use three simulated operators in this scenario:

- **Speed-focused operator**: This operator prefers the robot to move to the destination quickly regardless of whether it hits any obstacles ($t_{complete} = 15$ seconds, $p_\alpha = 95\%$, $p_\gamma = 5\%$).
- **Safety-focused operator**: This operator prefers the robot to avoid obstacles regardless of how long it takes to reach the destination ($t_{complete} = 15$ seconds, $p_\alpha = 5\%$, $p_\gamma = 95\%$).
- **Balanced operator**: This operator prefers a balanced mixture of speed and safety ($t_{complete} = 15$ seconds, $p_\alpha = 95\%$, $p_\gamma = 95\%$).

In this scenario, the robot has two modifiable behavior components: *speed* and *obstacle padding*. Speed, measured in meters per second, relates to how fast the robot can move. Padding, measured in meters, relates to the distance the robot will attempt to maintain from obstacles during movement. The set of possible values for each modifiable component ($\mathcal{C}_{speed}$ and $\mathcal{C}_{padding}$) are based on the robot's capabilities (i.e., minimum and maximum accepted values with fixed increments):

$$\mathcal{C}_{speed} = \{0.5, 1.0, \ldots, 10.0\}$$
$$\mathcal{C}_{padding} = \{0.1, 0.2, 0.3, \ldots, 2.0\}$$

6.3.2 Patrolling Scenario

In the second scenario, the robot patrols for threats as it moves between two locations in the environment (Figure 3). At the start of each run, six *suspicious objects* are randomly placed in the environment. These suspicious objects represent potential threats, and between 0 and 3 (inclusive) of them are designated as hazardous explosive devices (selected randomly with a uniform distribution). The remaining suspicious objects are not hazardous to the robot or the team.

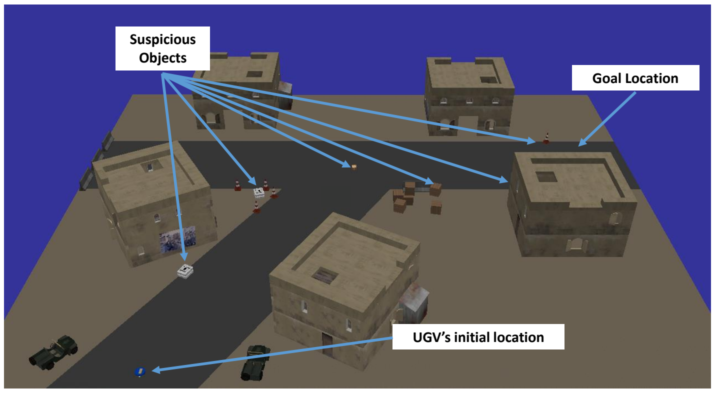

Fig. 3 The environment configuration for the Patrol scenario.

As the robot moves between the start location and the goal location (given by a natural language command from the operator), it scans for suspicious objects nearby. When a suspicious object is detected, it pauses its patrolling behavior, moves toward the object, scans it with its explosives detector, and labels the object as an *explosive* or *harmless*. The robot then resumes its patrolling behavior. The accuracy of the robot's explosives detector is a function of how long the robot spends scanning the object (long scan times result in improved accuracy) and its proximity to the object (smaller scan distances result in improved accuracy). In addition to the speed and padding, the *scan time*, measured in seconds, and *scan distance*, measured in meters, are also modifiable components of the robot's behavior. The possible values for these are:

$$\mathcal{C}_{scantime} = \{0.5, 1.0, \ldots, 5.0\}$$
$$\mathcal{C}_{scandistance} = \{0.25, 0.5, \ldots, 1.0\}$$

In addition to the task duration, task completion, and safety factors described in the Movement scenario, the simulated operators will also base their decision to interrupt the robot on its ability to successfully identify and label suspicious objects. An operator will interrupt the robot if it does not scan one or more suspicious objects (e.g., it drives by without noticing it) or incorrectly labels a harmless object as an explosive. If the robot incorrectly labels an explosive device as harmless, the object will eventually detonate and the robot will fail its task. The robot assigns higher weight to failures due to missing explosive devices (3 times higher than other failures or interruptions) because of the danger such failures cause to human teammates and bystanders.

We use two simulated operators in this scenario:

- **Speed-focused operator**: The operator prefers that the robot performs the patrol task within a fixed time limit ($t_{complete} = 120$ seconds, $p_{\alpha} = 95\%$, $p_{\gamma} = 5\%$).
- **Detection-focused operator**: The operator prefers the task be performed correctly regardless of time ($t_{complete} = 120$ seconds, $p_{\alpha} = 5\%$, $p_{\gamma} = 5\%$).

6.4 Trustworthy Behaviors

We found that both case-based behavior adaptation and random walk behavior adaptation resulted in similar trustworthy behaviors for each operator. This includes values falling within similar ranges of trustworthy values (e.g., for the safety-focused operator in the Movement scenario the padding never went below 0.4 meters in any trial) or similar relations between values (e.g., in the Patrol scenario there was a relation between scan time and scan distance). Furthermore, the trustworthy behaviors aligned with what an outside observer would intuitively consider trustworthy for each operator (e.g., that the speed-focused operator will prefer higher speeds).

The trustworthy behaviors for each of the operators in the Movement scenario are shown in Figures 4, 5, and 6. Each dot represents the trustworthy behavior found during a single trial using random walk adaptation. Although 500 trials were performed for each operator, fewer than 500 dots appear in each figure because some trials converged to the same parameter values. This is more prevalent when case-based behavior adaptation is performed since the final behaviors stored in cases occur much more frequently than other behaviors (i.e., those solutions are repeatedly reused). However, the trustworthy behaviors found by the case-based approach fall within the same regions as the random walk behaviors.

The speed-focused operator (Figure 4) causes the robot to converge to higher speed values regardless of padding while the safety-focused operator (Figure 5) results in higher padding values regardless of speed. For the balanced operator (Figure 6), both speed and padding must be high.

In the Patrol scenario, there are similar differences in the range of values for certain behavior components. The speed-focused Patrol operator causes the robot to converge to higher speed values (speed ≥ 2.0) whereas the detection-focused operator has no such restriction. However, unlike in the Movement scenario there are also interdependencies among behavior components. For example, none of the trustworthy behaviors for the speed-focused Patrol operator have both a medium speed ($2.0 \leq speed \leq 4.0$) and high scan time. This is because the robot needs to account for longer scan times by driving faster. Similarly, there is an interdependence between scan time and scan distance. The robot only selects a poor value for one of the modifiable components (low scan time or high scan distance) if it selects a good value for the other (high scan time or low scan distance).

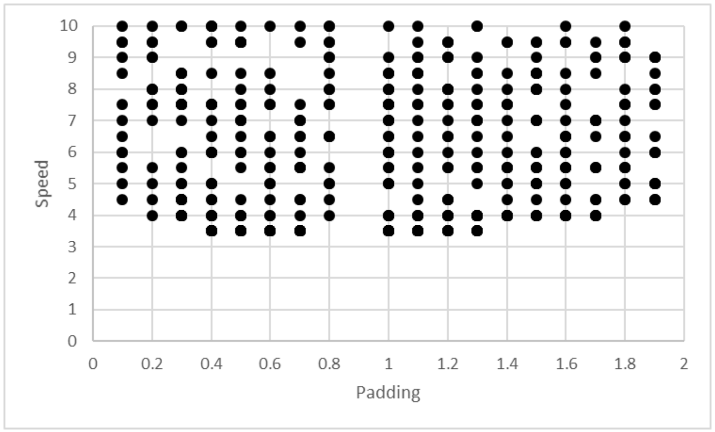

Fig. 4 Trustworthy behaviors for speed-focused operator in the Movement scenario.

These results fit with the definitions of the simulated operators and our intuition on the behaviors they would find trustworthy. However, one noticeable exception occurred in the Movement scenario where behaviors that appear to be trustworthy are actually not. For both the speed-focused and balanced operators (Figures 4 and 6), no trustworthy behaviors were found when padding $= 0.9$. For both of these operators, larger (padding $= 1.0$) and smaller (padding $= 0.8$) values were found to be trustworthy. This occurred in the results for both the case-based and random walk approaches, and in a follow up evaluation where the robot was forced to use behaviors with padding $= 0.9$. The reason this padding value was found to be untrustworthy was because of the environment. The padding value resulted in a direct, but narrow, path to the destination. This required the robot to slow down when navigating through the narrow path and caused it to exceed its time limit (this is why

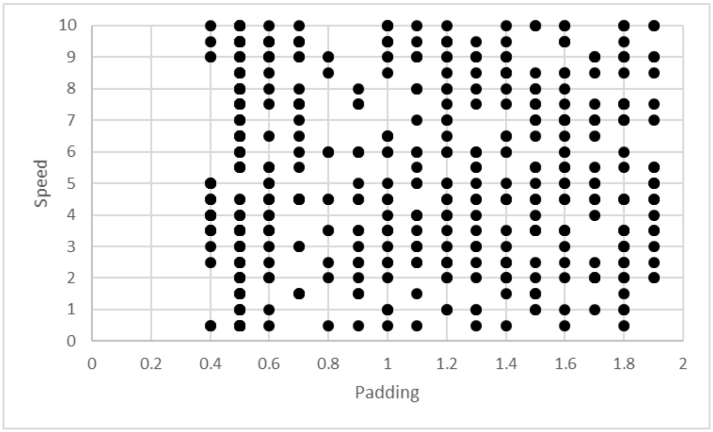

Fig. 5 Trustworthy behaviors for safety-focused operator in the Movement scenario.

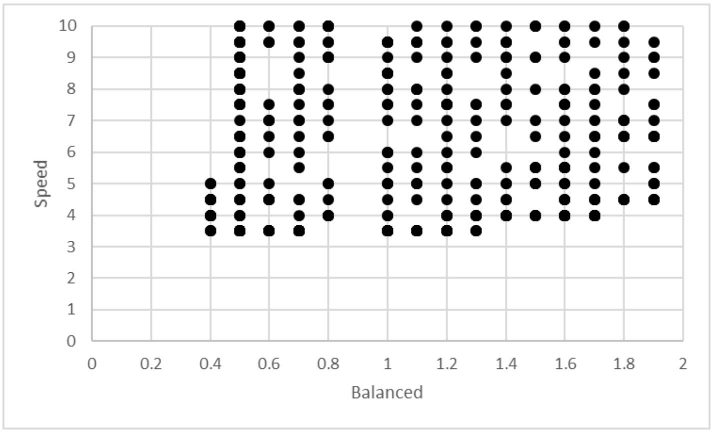

Fig. 6 Trustworthy behaviors for balanced operator in the Movement scenario.

that padding value was not an issue for the safety-focused operator). However, when the padding was lowered the path became large enough that the robot could drive through without slowing down. Similarly, when the padding was increased the path was eliminated so the robot took a slightly longer but much easier path. These results show that even if we had a general ideal about what behaviors would be considered trustworthy there is still the possibility of seemingly trustworthy behaviors being untrustworthy. It is beneficial for the robot to be able to adapt and overcome any issues that are not anticipated, especially if it operates in a dynamic or unknown environment.

6.5 Efficiency

The primary difference between the case-based and random walk approaches was related to how many behaviors needed to be evaluated before a trustworthy behavior was found. Table 1 shows the mean number of evaluated behaviors (and 95% confidence interval) when interacting with each operator over 500 trials. In addition to being controlled by a single operator during each experiment, we also examined a condition where the operator is selected at random (with equal probability) at the start of each trial. This represents a more realistic situation where the robot is required to interact with a variety of operators but does not know which particular operator it is currently interacting with. This variant is labeled as *Random* and was performed in both scenarios. The table also shows the number of cases acquired during the case-based behavior adaptation experiments (each experiment started with an empty case base).

Table 1 Mean number of behaviors evaluated before finding a trustworthy behavior.

Scenario	**Operator**	**Random Walk**	**Case-based**	**Cases Acquired**
Movement	*Speed-focused*	20.3 (±3.4)	1.6 (±0.2)	24
Movement	*Safety-focused*	2.8 (±0.3)	1.3 (±0.1)	18
Movement	*Balanced*	27.0 (±3.8)	1.8 (±0.2)	33
Movement	*Random*	14.6 (±2.9)	1.6 (±0.1)	33
Patrol	*Speed-focused*	344.5 (±31.5)	9.9 (±3.9)	25
Patrol	*Detection-focused*	199.9 (±23.3)	5.5 (±2.2)	22
Patrol	*Random*	269.0 (±27.1)	9.3 (±3.2)	25

The case-based approach required significantly fewer behaviors to be evaluated in all seven conditions (using a paired t-test with $p < 0.01$). This is because the case-based approach learns from previous adaptations and uses that information to quickly find trustworthy behaviors. At the beginning of an experiment, when the robot's case base is empty, the case-based approach relies on performing random walk adaptation. As the case base grows, the number of random walk adaptations decreases until the agent generally performs a single case-based adaptation before finding a trustworthy behavior. Even in the random operator experiments when the case base contains cases from several different operators (three in the Movement scenario and two in Patrol), the case-based approach can quickly differentiate between operators and select a trustworthy behavior. Operators with fewer restrictions on trustworthy behaviors (i.e., a higher percentage of the behavior space is considered trustworthy), like the safety-focused and detection-focused operators, had the lowest mean number of adaptations to find a trustworthy behavior.

6.6 Discussion

The primary limitation of the case-based approach is that it relies on random walk search when it does not have any suitable cases to use. This is especially prevalent early on when the robot has a small or empty case base. For example, if we consider only the final 250 trials for each Patrol scenario operator, the mean number of behaviors evaluated is lower than the overall mean (4.2 for the speed-focused, 2.8 for the detection-focused, and 3.3 for the random). This is because the robot performs the expensive random walk adaptation more often in the early trials, so it performs more efficiently on the later trials. These expensive adaptations occur infrequently (only in trials where a case is stored) but increase the mean number of behaviors that are evaluated.

Two primary solutions exist to reduce the number of behaviors examined during case-based behavior adaptation: improving search and seeding the case base. Random walk search is used because it requires no explicit knowledge about the domain, task, or operator. However, a more intelligent search that could identify relations between interruptions and modifiable components would likely improve adaptation time (e.g., an interruption when the robot is close to objects may require a change to the padding value). This could reduce the cost of each search, whereas seeding the case base would attempt to minimize the number of searches required. A set of initial cases could be provided to the robot so that it would not need to acquire as many on its own. However, these two solutions introduce their own potential limitations. A more informed search requires introducing domain knowledge, which may be difficult or expensive to obtain, and seeding the case base requires an expert to manually author cases (or another method of automatic case acquisition). The specific requirements of the application domain will ultimately influence whether fasted behavior adaptation or lower domain knowledge requirements are more important.

7 Conclusions

In this chapter we have described our approach for inverse trust estimation and how a robot can use it to adapt its behavior. Rather than traditional trust metrics that directly measure how much trust an agent has in another agent, our inverse trust estimate attempts to infer how much trust another agent has in it. As such, it can not be thought of as an explicit measurement of trust but rather a best-guess estimate based on observable indicators of trust (e.g., the operator's response to the robot's performance). Our approach relies more on the general trends in its trustworthiness (increasing, decreasing, or constant) rather than requiring a precise numerical value.

The primary benefit of this behavior adaptation approach is that it does not require any background knowledge about the tasks, environment, context, or operators. Each time the robot successfully finds a trustworthy behavior, it stores information about the adaptation process and uses that to improve the efficiency of future adaptations. This allows it to constantly learn behavior adaptation knowledge with each trial.

We evaluated our trust-guided behavior adaptation algorithm in a simulated robotics environment by comparing it to a variation that does not learn. In the two scenarios, Movement and Patrol, both approaches converged to trustworthy behaviors but the case-based algorithm required significantly fewer behaviors to be evaluated. This is advantageous because the operator is more likely to stop using the robot the longer the robot behaves in an untrustworthy manner.

Although we have shown the benefits of trust-guided behavior adaptation, several areas of future work exist. Although much of our work is based on studies in human-robot interaction, our initial evaluation has been limited to simulation studies. An ongoing area of our research is to validate our findings in a series of user studies. Next, our robot is only concerned with undertrust. In longer scenarios, the robot should also evaluate situations of overtrust where the operator trusts the robot too much and allows the robot to behave autonomously even when its performance is poor. We also plan to expand our inverse trust estimate by incorporating other trust factors and adding mechanisms that promote transparency (Kim and Hinds, 2006) between the robot and the operator. Transparency would allow information exchange between the robot and the operator, and allow the robot to verify or refine assumptions it has been using (e.g., which goals the team is currently trying to achieve). Many of these areas for future work revolve around taking our existing approach, which requires minimal domain knowledge, and allowing for extra knowledge to be incorporated if it ever becomes available.

Acknowledgments

Thanks to the United States Naval Research Laboratory and the Office of Naval Research for supporting this research.

References

Baier JA, McIlraith SA (2008) Planning with preferences. AI Magazine 29(4):25–36

Berlin M, Gray J, Thomaz AL, Breazeal C (2006) Perspective taking: An organizing principle for learning in human-robot interaction. In: 21st National Conference on Artificial Intelligence, pp 1444–1450

Biros DP, Daly M, Gunsch G (2004) The influence of task load and automation trust on deception detection. Group Decision and Negotiation 13(2):173–189

Breazeal C, Gray J, Berlin M (2009) An embodied cognition approach to mindreading skills for socially intelligent robots. International Journal of Robotic Research 28(5)

Carlson MS, Desai M, Drury JL, Kwak H, Yanco HA (2014) Identifying factors that influence trust in automated cars and medical diagnosis systems. In: AAAI Symposium on The Intersection of Robust Intelligence and Trust in Autonomous Systems, pp 20–27

Chen K, Zhang Y, Zheng Z, Zha H, Sun G (2008) Adapting ranking functions to user preference. In: 24th International Conference on Data Engineering Workshops, pp 580–587

Desai M, Kaniarasu P, Medvedev M, Steinfeld A, Yanco H (2013) Impact of robot failures and feedback on real-time trust. In: 8th International Conference on Human-robot Interaction, pp 251–258

Esfandiari B, Chandrasekharan S (2001) On how agents make friends: Mechanisms for trust acquisition. In: 4th Workshop on Deception, Fraud and Trust in Agent Societies, pp 27–34

Hancock PA, Billings DR, Schaefer KE, Chen JY, De Visser EJ, Parasuraman R (2011) A meta-analysis of factors affecting trust in human-robot interaction. Human Factors: The Journal of the Human Factors and Ergonomics Society 53(5):517–527

Horvitz E (1999) Principles of mixed-initiative user interfaces. In: 18th Conference on Human Factors in Computing Systems, pp 159–166

Jian JY, Bisantz AM, Drury CG (2000) Foundations for an empirically determined scale of trust in automated systems. International Journal of Cognitive Ergonomics 4(1):53–71

Kaniarasu P, Steinfeld A, Desai M, Yanco HA (2012) Potential measures for detecting trust changes. In: 7th International Conference on Human-Robot Interaction, pp 241–242

Kaniarasu P, Steinfeld A, Desai M, Yanco HA (2013) Robot confidence and trust alignment. In: 8th International Conference on Human-Robot Interaction, pp 155–156

Kiesler S, Powers A, Fussell SR, Torrey C (2008) Anthropomorphic interactions with a robot and robotlike agent. Social Cognition 26(2):169–181

Kim T, Hinds P (2006) Who should i blame? effects of autonomy and transparency on attributions in human-robot interaction. In: 15th IEEE International Symposium on Robot and Human Interactive Communication, pp 80–85

Knexus Research Corporation (2013) eBotworks. http://www.knexusresearch.com/products/ebotworks.php, [Online; accessed December 1, 2014]

Li D, Rau PP, Li Y (2010) A cross-cultural study: Effect of robot appearance and task. International Journal of Social Robotics 2(2):175–186

Li N, Kambhampati S, Yoon SW (2009) Learning probabilistic hierarchical task networks to capture user preferences. In: 21st International Joint Conference on Artificial Intelligence, pp 1754–1759

Maes P, Kozierok R (1993) Learning interface agents. In: 11th National Conference on Artificial Intelligence, pp 459–465

Mahmood T, Ricci F (2009) Improving recommender systems with adaptive conversational strategies. In: 20th ACM Conference on Hypertext and Hypermedia, pp 73–82

McGinty L, Smyth B (2003) On the role of diversity in conversational recommender systems. In: 5th International Conference on Case-Based Reasoning, pp 276–290

Muir BM (1987) Trust between humans and machines, and the design of decision aids. International Journal of Man-Machine Studies 27(56):527–539

Oleson KE, Billings DR, Kocsis V, Chen JY, Hancock PA (2011) Antecedents of trust in human-robot collaborations. In: 1st International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support, pp 175–178

Richter MM, Weber RO (2013) Case-Based Reasoning - A Textbook. Springer

Sabater J, Sierra C (2005) Review on computational trust and reputation models. Artificial Intelligence Review 24(1):33–60

Saleh JA, Karray F, Morckos M (2012) Modelling of robot attention demand in human-robot interaction using finite fuzzy state automata. In: International Conference on Fuzzy Systems, pp 1–8

Schlimmer JC, Hermens LA (1993) Software agents: Completing patterns and constructing user interfaces. Journal of Artificial Intelligence Research 1:61–89

Shapiro D, Shachter R (2002) User-agent value alignment. In: Stanford Spring Symposium - Workshop on Safe Learning Agents