Trust-Guided Behavior Adaptation Using Case-Based Reasoning*

Michael W. Floyd and **Michael Drinkwater**
Knexus Research Corporation
Springfield, Virginia, USA
michael.floyd@knexusresearch.com
michael.drinkwater@knexusresearch.com

David W. Aha
Navy Center for Applied Research in AI
Naval Research Laboratory (Code 5514)
Washington, DC, USA
david.aha@nrl.navy.mil

Abstract

The addition of a robot to a team can be difficult if the human teammates do not trust the robot. This can result in underutilization or disuse of the robot, even if the robot has skills or abilities that are necessary to achieve team goals or reduce risk. To help a robot integrate itself with a human team, we present an agent algorithm that allows a robot to estimate its trustworthiness and adapt its behavior accordingly. As behavior adaptation is performed, using case-based reasoning (CBR), information about the adaptation process is stored and used to improve the efficiency of future adaptations.

1 Introduction

A robot can be a beneficial addition to a human team if it improves the team's capabilities, improves productivity, or reduces the risk to the human teammates. This is especially true of semi-autonomous robots that can complete tasks independently or reduce other teammates' workload. However, in order for the team to get the full benefit of the robot they need to trust it and be willing to delegate tasks to it. A lack of trust in the robot could result in teammates 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].

One possibility would be to design a robot that is guaranteed to operate in a trustworthy manner. However, it may be impossible to elicit a complete set of rules for trustworthy behavior if the robot is expected to handle changes in teammates, environments, or mission contexts. The way in which a teammate measures trust in the robot may be user-dependent, task-dependent, or time-varying [Desai *et al.*, 2013]. For example, a teammate's measurement of trust may change in an emergency situation. Similarly, the time-critical nature of the team's mission may make it difficult to get explicit feedback from teammates about the robot's trustworthiness.

We propose an approach that allows a robot to evaluate its trustworthiness and adapt its behavior accordingly. The trust estimate, which we refer to as an *inverse trust estimate*, differs from traditional computational trust metrics in that it measures how much trust other agents have in the robot rather than how much trust the robot has in other agents. Since the robot can only use observable information and not information that is internal to the teammates' reasoning, the inverse trust estimate relies on evaluating the standard interactions between the robot and its teammates (i.e., being assigned tasks and performing those tasks). The inverse trust estimate is not a direct measurement that is able to precisely quantify trust but instead measures trends in trust (e.g., increasing, decreasing, remaining constant) based on observable factors that are known to influence human-robot trust. Using these trends, the robot is able to adapt its behavior in an attempt to find more trustworthy behaviors. Our adaptation approach uses case-based reasoning (CBR) to allow the robot to leverage information from previous behavior adaption to more efficiently adapt to trustworthy behaviors.

In the remainder of this paper we describe how inverse trust and behavior adaptation can be used to allow a robot to adopt trustworthy behaviors regardless of teammates, environment, or context. In Section 2, we describe the robot and how it can modify aspects of its behavior. The inverse trust estimate and how it can be used to classify and evaluate behaviors is discussed in Section 3, followed by how that information can be used to adapt the robot's behavior in Section 4. Our approach is evaluated in a simulated robotics domain in Section 5. We report evidence that case-based behavior adaptation can efficiently adapt the robot's behavior to align with a teammate's preferences. Areas of related work are discussed in Section 6 and concluding remarks are presented in Section 7.

2 Robot Behavior

We make two assumptions about the robot: it behaves semi-autonomously and it has the ability to modify aspects of its behavior. A human operator interacts with the robot by issuing commands or delegating tasks, and the robot acts independently to complete its assignment. The robot has direct control over certain aspects of its behavior that we refer to as the *modifiable components*. These could include changing algorithms (e.g., switching the path planning algorithm it uses), modifying parameter values, or selecting among comparable

*This paper was invited for submission to the Best Papers From Sister Conferences Track, based on a paper that appeared in the 22nd International Conference on Case-Based Reasoning (ICCBR 2014) [Floyd *et al.*, 2014].

data sources to use (e.g., using an alternate map of the environment).

Each modifiable component i has a set of possible values $\mathcal{C}_i$ from which the robot can choose. 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 ($c_i \in \mathcal{C}_i$):

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

The robot can immediately influence how it behaves by switching from its current behavior B to a new behavior B_{new}. In the context of our work, these behavior changes primarily occur when the robot is attempting to behave in a more trustworthy manner. Over the course of operation, the robot can make numerous changes resulting in a sequence of behaviors $\langle B_1, B_2, \ldots, B_n \rangle$.

3 Inverse Trust Estimate

Traditional trust metrics measure how much trust an agent has in other agents [Sabater and Sierra, 2005]. Previous interactions with those agents, observations of the agents, or feedback from others are used to calculate trustworthiness [Esfandiari and Chandrasekharan, 2001]. These metrics are designed to be used in a single direction (e.g., $agent_A$ measuring its trust in $agent_B$) and are not applicable in the inverse (e.g., $agent_B$ measuring how trustworthy $agent_A$ thinks it is). This occurs because the information needed to measure trustworthiness may be internal to the agent using the metric (e.g., how $agent_A$ judged previous interactions with $agent_B$, or private feedback received about $agent_B$).

A robot would need access to the operator's personal experiences and beliefs in order to use a traditional trust metric, but the operator might be unwilling or unable to provide this information. Even if the robot does not have the necessary information to calculate its own trustworthiness it might be able to elicit explicit feedback from the operator (i.e., the results of the operator using a trust metric). This feedback could be provided at run-time (e.g., periodically telling the robot how trustworthy it is [Kaniarasu *et al.*, 2013]) or offline after all tasks have been completed (e.g., filing out a trust survey [Jian *et al.*, 2000; Muir, 1987]). However, this might not be practical if the operator does not have time to provide regular feedback or feedback is needed before all tasks have been completed.

In situations where the robot does not have access to the information needed to use a traditional trust metric and no explicit operator feedback is available, the robot will need to *infer* how trustworthy it is using observable *evidence* of trust. This requires the robot to detect the presence of factors that influence human-robot trust. Numerous factors have been found to influence trust [Oleson *et al.*, 2011] but the strongest indicator is the robot's performance [Hancock *et al.*, 2011; Carlson *et al.*, 2014]. In addition to being the strongest indicator of trust, the robot's performance is also directly observable and has a clear model for its impact on trust [Kaniarasu *et al.*, 2012].

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 by the operator while performing a task. The robot assumes that the operator will view completed tasks as good performance, failed tasks as poor performance, and will interrupt if the robot is performing poorly. Interruptions could also be a result of a change in the operator's goals or a realization that the assigned task was unachievable, but the robot works under the assumption that most interruptions will be related to performance. Task completion and interruptions provide a reasonable basis for estimating inverse trust because they have been found to align closely with changes in operator trust, based on both real-time user feedback [Kaniarasu *et al.*, 2013] and post-run trust surveys [Kaniarasu *et al.*, 2012].

Rather than quantifying precisely how trustworthy the robot is, our inverse trust estimate looks for general trends in the robot's trustworthiness and determines if trust is increasing, decreasing, or remaining constant. We estimate the trustworthiness as follows:

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

where there were n commands issued to the robot while using the current behavior B'. The estimate will increase if the ith command ($1 \leq i \leq n$) was completed successfully and decrease if the command was failed or interrupted ($cmd_i \in \{-1, 1\}$). The ith command also receives a weight w_i which denotes varying levels of success or failure when performing a command. For example, a command that the robot performed poorly would likely be weighted less than a command where the robot damaged itself.

The trust estimate produces a simple step function that the robot can compute online as new information becomes available (i.e., new commands are issued). A more complex or cognitively plausible function could also be used that more closely aligns with the operator's actual trust. However, the additional complexity of such a function might not provide additional benefits if, like with our robot, we seek general trends in trustworthiness rather than a precise value.

3.1 Behavior Classification

The trust estimate is updated by the robot after each successfully completed task, failure, or interruption. The robot continuously monitors the estimate and classifies its current behavior as trustworthy, untrustworthy, or unknown. To perform this classification, two thresholds are used: the trustworthy threshold (τ_T) and the untrustworthy threshold (τ_U).

If the estimate is between the two thresholds ($\tau_U < Trust_{B'} < \tau_T$), the robot cannot confidently classify its behavior as being trustworthy or untrustworthy. In this situation, it continues to monitor the estimate and can only observe general trends in its trustworthiness (increasing, decreasing, or remaining constant). The robot will conclude its behavior is sufficiently trustworthy if the trustworthy threshold is reached ($Trust_{B'} \geq \tau_T$). When a trustworthy behavior is found, the robot will continue to use the behavior but may continue to measure its trustworthiness in case any changes occur the would cause the behavior to no longer be trustworthy (e.g., a new operator or a goal change). However, if

the untrustworthy threshold is reached ($Trust_{B'} \leq \tau_U$), the robot will conclude that its current behavior is untrustworthy and should be changed. The robot will infer that its current behavior has been decreasing the operator's trust and a new, more trustworthy behavior is needed to help regain that trust.

3.2 Evaluated Behaviors

The goal of the robot is to find a behavior that it thinks is trustworthy (i.e., the trustworthy threshold is reached) but as it performs this search it may find certain behaviors to be untrustworthy (i.e., the untrustworthy threshold is reached). When a behavior B is found to be untrustworthy, it is stored as an *evaluated pair* E that also contains the time t it took to be labeled as untrustworthy:

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

The time t is measured from when the robot starts using the behavior to when the untrustworthy threshold is reached. The motivation for storing the time is that it allows for a comparison between untrustworthy behaviors and assigning relative levels of untrustworthiness. 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 completing more tasks, not failing as quickly, or appearing to behave trustworthy for longer periods of time.

The robot maintains a set $\mathcal{E}_{past}$ of previously evaluated behaviors. This set is initially empty but is extended as the robot evaluates more behaviors. After the robot has found n behaviors to be untrustworthy, $\mathcal{E}_{past}$ will contain n evaluated pairs ($\mathcal{E}_{past} = \{E_1, E_2, \ldots, E_n\}$). The set can be thought of as the search path that the robot takes until it finds a behavior B_{final} that reaches the trustworthy threshold.

4 Case-based Behavior Adaptation

Behavior adaptation is used to select a new behavior to evaluate after the current behavior has reached the untrustworthy threshold. We employ case-based reasoning [Richter and Weber, 2013] to perform behavior adaptation in our system. CBR embodies the idea that similar problems tend to have similar solutions. 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}. Using the CBR methodology, the robot attempts to find a trustworthy behavior using information from previous behavior searches. For example, if the robot found a trustworthy behavior for an initial operator, it could use that to help find a trustworthy behavior for a new operator.

Case-based reasoning systems store problem-solution pairs, called *cases*, that represent concrete problem-solving instances. A case C in our system is defined as:

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

The collection of cases that the robot uses is called a *case base*. The case base CB contains all cases that have been stored:

$$CB = \{C_1, C_2, \ldots\}$$

The case base is initially empty but grows each time a new case is created (i.e., a trustworthy behavior is found). It represents all of the problem-solving experience that the robot has collected.

The robot selects a new behavior to perform using the *selectBehavior* function (Algorithm 1). This algorithm performs the case-based reasoning process by comparing the problem the robot is currently attempting to solve (i.e., the set of previously evaluated behaviors $\mathcal{E}_{past}$) to the problems it has previously solved (i.e., the cases in the case base CB). This is motivated by the idea that if two problem are similar then their solutions may also be similar, so the robot can adapt its behavior by switching to the final behavior of the most similar case.

Algorithm 1: Selecting a new behavior

Function: *selectBehavior*($\mathcal{E}_{past}$, CB) ***returns*** B_{new};

```
1  bestSim ← 0; B_best ← ∅;
2  foreach C_i ∈ CB do
3      if C_i.B_final ∉ E_past then
4          sim_i ← sim(E_past, C_i.E_past);
5          if sim_i > bestSim then
6              bestSim ← sim_i;
7              B_best ← C_i.B_final;
8  if B_best = ∅ then
9      B_best ← modifyBehavior(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 performed to ensure that behaviors that have already been evaluated and found to be untrustworthy are not evaluated again (since only untrustworthy behaviors are stored in $\mathcal{E}_{past}$). The robot compares its current set of evaluated behaviors to the set of evaluated behaviors in the remaining cases (line 4). This allows the robot to find the most similar case, store that case's final behavior (lines 5-7), and select that behavior to be used (line 10). The robot immediately switches to this behavior.

If the case base is empty or the final behaviors of all cases have already been evaluated, the *selectBehavior* algorithm will not find any potential behaviors to use (line 8). In this situation, the case-based reasoning system has insufficient problem-solving experience to solve the current problem so an alternate adaptation approach is used. The *modifyBehavior* function performs random walk behavior adaptation. Although other adaptation techniques could also be used, random walk adaptation is used because it does not require any prior knowledge about the operator, task, or domain.

The *modifyBehavior* function selects the 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 behavior B_{new} is found that requires the minimum number of changes to the

modifiable components of $E_{max}.B$ and has not already been evaluated by the robot ($\forall E_i \in \mathcal{E}_{past}, B_{new} \neq E_i.B$). This is based on the assumption that E_{max} is the untrustworthy behavior that is closest to being trustworthy. By making a slight change, the aim is that B_{new} will be closer to being trustworthy. If all possible behaviors have already been evaluated, the robot will stop adapting its behavior and use $E_{max}.B$. This is done so that even if there are no trustworthy behaviors the robot can use it will still attempt to behave in the least untrustworthy way possible.

The *selectBehavior* function relies on computing the similarity between two sets of evaluated behaviors (line 4). The ability to measure similarity is central to case-based reasoning since it allows a system to identify if two problems are similar to each other (i.e., they might have similar solution). This similarity function (Algorithm 2) needs to take into account that these sets may vary in size. This occurs because the number of evaluated behaviors in each set is dependent on how long the trustworthy behavior search took in that instance. For example, a search that quickly found a trustworthy behavior would contain fewer evaluated behaviors than a longer search. 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 of the sets. The algorithm goes through each evaluated behavior E_i in the first set (line 2) and finds the most similar evaluated behavior E_{max} in the second set (line 3). The similarity between behaviors is a function of the similarity of each behavior component:

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

The similarity function for each behavior component will depend on its specific type. For example, a behavior component that represents a binary parameter requires a different similarity function than a component that represents which path planning algorithm is being used. The various similarity functions return values between 0.0 (most dissimilar) and 1.0 (most similar). For example, consider a robot with two modifiable components: speed and padding (how far it attempts to stay away from obstacles when planning its movement). A behavior B_a with a speed of 1 meter/second and a padding of 0.5 meters ($B_a = \langle 1, 0.5 \rangle$) could be compared to a behavior B_b with a speed of 6 meters/second and a padding of 0.5 meters ($B_b = \langle 6, 0.5 \rangle$). The similarity between the behaviors is a function of the similarity of each modifiable component (using a similarity metric for numerical modifiable components[1], where $sim(1, 6) = 0.5$ and $sim(0.5, 0.5) = 1.0$), so they would have a similarity of 0.75 ($sim(B_a, B_b) = \frac{1}{2}(0.5 + 1)$).

If the behaviors stored in E_i and E_{max} are sufficiently similar, based on a threshold λ (line 4), the similarity of their time components are included in the similarity calculation (line 5).

[1]Using the similarity function $sim(v_1, v_2) = (1 - \frac{|v_1 - v_2|}{max - min})$, where max is the maximum the values can take (10 meters/second for speed and 2 meters for padding) and min is the minimum (0 meters/second and 0 meters).

Algorithm 2: Similarity between sets of evaluated behaviors

Function: *sim*($\mathcal{E}_1$, $\mathcal{E}_2$) ***returns*** sim;

```
1 totalSim ← 0; num ← 0;
2 foreach E_i ∈ E_1 do
3     E_max ← arg max_{E_j ∈ E_2} (sim(E_i.B, E_j.B));
4     if sim(E_i.B, E_max.B) > λ then
5         totalSim ← totalSim + sim(E_i.t, E_max.t);
6         num ← num + 1;
7 if num = 0 then
8     return 0;
9 return totalSim / num;
```

This ensures that the final similarity value only includes information from behaviors that have a highly similar counterpart in the other set. The similarity function identifies behaviors that have been evaluated in both sets and evaluates if they were found to be untrustworthy in a similar amount of time.

5 Evaluation

We evaluate our behavior adaptation technique in a simulated robotics environment [Knexus Research Corporation, 2015]. The robot is a wheeled unmanned ground vehicle. It receives natural language commands from the operator in an urban environment composed of landmarks (e.g., roads, various types of terrain) and other objects (e.g., houses, humans, vehicles, road barriers).

Our evaluation compares two variations of trust-based behavior adaptation: case-based behavior adaptation and random walk behavior adaptation. While we expect both to allow the robot to adapt to trustworthy behaviors, we evaluate our claim that the case-based approach does so more efficiently.

5.1 Experimental Conditions

Our study uses simulated operators that were selected to represent a subset of the control strategies used by human operators. Each simulated operator has preferences for how the robot should behave and those preferences influence how the robot's behavior is evaluated (i.e., when the robot is allowed to complete a task and when it is interrupted).

Each experiment involves 500 *trials* and in each trial the robot interacts with a single operator. At the start of each trial, the robot randomly selects (with uniform distribution) initial values for each of its modifiable component. Throughout a trial, a series of experimental *runs* occur. A run involves the operator issuing a command to the robot and monitoring the robot as it completes the assigned 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 and may adapt its behavior. At the end of a run, the environment is reset and a new run begins. A trial concludes when the robot finds a trustworthy behavior or evaluates all possible behaviors.

The case-based approach starts each experiment with an empty case base (i.e., no previous problem-solving experience). 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. This case retention strategy is used to prevent adding redundant cases since cases are not added if the existing case base can find a solution. Once a case is added to the case base it can be used during subsequent trials.

The robot used the following thresholds during the experiments: $\tau_T = 5.0$, $\tau_U = -5.0$, $\lambda = 0.95$.

5.2 Evaluation Scenarios

Two evaluation scenarios were used: *Movement* and *Patrol*. In the Movement scenario, the operator issues commands to the robot telling it where to move in the environment (e.g., "move to the flag"). The robot is responsible for interpreting the commands and navigating to the appropriate locations. The operators evaluate the robot based on successful completion of a task, how long the robot has been attempting to complete the task, and how safely the robot is behaving (more details about when the operators interrupt the robot are provided in [Floyd *et al.*, 2014]). Three simulated operators were used: *speed-focused*, *safety-focused*, and *balanced*. The speed-focused operator prefers that the robot completes the task quickly (within 15 seconds) regardless of whether it hits any obstacles. The safety-focused operator prefers that the robot avoids obstacles regardless of how long it takes to complete the task. The balanced operator prefers that the task be completed quickly (within 15 seconds) without hitting any obstacles.

The robot has two modifiable components: *speed* (meters per second) and *obstacle padding* (meters). The speed relates to how fast the robot can move and the padding 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 minimum and maximum acceptable values the robot can use:

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

In the Patrol scenario, six *suspicious objects* are randomly placed in the environment at the start of each run. These objects represent potential threats to the robot or its team. Between 0 and 3 (inclusive) of these objects are selected randomly to be hazardous explosive devices while the remaining objects pose no threat.

The robot receives commands from its operator telling it to patrol between its current location and a destination location. While navigating to the destination, the robot is responsible for locating suspicious objects nearby. If a suspicious object is detected, the robot pauses its patrol, moves toward the object, scans the object with its explosive detector, labels the object as *explosive* or *harmless*, and resumes its patrol behavior. The accuracy of the robot's explosives detector is a function of how long the robot spends scanning the objects (longer scan times result in improved accuracy) and its proximity to the object (smaller scan distances result in improved accuracy). In addition to speed and padding, *scan time* (seconds) and *scan distance* (meters) are modifiable components of the robot's behavior:

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

The simulated operators in this scenario will also consider the robot's ability to identify and label suspicious objects when evaluating the robot. The robot will be interrupted if it does not scan one or more suspicious objects or incorrectly labels an object. Two simulated operators are used in this scenario: *speed-focused* and *detection-focused*. The speed-focused operator prefers for the robot to complete the patrol correctly and within a fixed time limit (120 seconds). The detection-focused operator prefers that the task be performed correctly regardless of time.

5.3 Results

Both case-based behavior adaptation and random walk behavior adaptation resulted in similar trustworthy behaviors for each operator. This includes finding trustworthy behaviors in similar ranges (e.g., that the speed-focused operator prefers higher speeds) or similar relations between values (e.g., the interdependence between scan time and scan distance). Furthermore, the trustworthy behaviors aligned with what an outside observer would intuitively consider trustworthy for each operator.

The primary difference between the two adaptation approaches is how many behaviors need to be evaluated before a trustworthy behavior is found. Table 1 shows the mean number of evaluated behaviors (and 95% confidence interval) when interacting with each operator over 500 trials. Additionally, this table shows the results when the operator is selected at random 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 operator it is currently receiving commands from.

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. At the beginning of an experiment, when the case base is small or empty, the case-based approach relies on random walk to generate cases so the initial results are similar to those of random walk. However, as more cases are added the number of random walk adaptations decreases until the robot generally only performs a single case-based adaptation before finding a trustworthy behavior. Our results indicate that most cases are stored during trials that occur near the start of an experiment. Even in the random operator experiments, the case-based approach is able to store cases related to several different operators (three in the Movement scenario and two in Patrol), and quickly differentiate between them.

These results indicate that the efficiency of the case-based approach could be further improved if the system was given an initial case base to use. Having an existing case base that was generated during training sessions would reduce the

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

number of expensive random walk adaptations required during time-sensitive missions. Random walk adaptation is used because it requires no explicit knowledge about the domain, task, or operator. However, a more intelligent search that is able to use direct feedback from the operator or learn the root causes of interruptions would reduce the cost of case generation.

6 Related Work

Existing approaches for measuring inverse trust differ from our own in that they require regular operator feedback or predefined rules. Robot performance, measured based on the number of times a human takes control of the robot or warns the robot, has been used to measure decreases in a robot's trustworthiness [Kaniarasu *et al.*, 2012]. In order to also detect increases in trust, direct feedback from the operator at regular intervals is required [Kaniarasu *et al.*, 2013]. A measure of inverse trust using a set of expert-authored rules has also been proposed [Saleh *et al.*, 2012]. However, without existing knowledge of these rules the robot would be unable to measure its trustworthiness.

Models of trust in case-based reasoning systems have focused on traditional trust rather than inverse trust (e.g., in the context of recommender systems [Tavakolifard *et al.*, 2009] or collaborative search [Briggs and Smyth, 2008]). Case provenance [Leake and Whitehead, 2007], where a case-based reasoning system considers the reliability of a case's source, also takes trust into account. Our work also has similarities to conversational case-based recommender systems [McGinty and Smyth, 2003] that tailor recommendations to a user's preferences. Recommendations are iteratively improved by learning a user model through an interactive dialog. This is similar to learning interface agents [Maes and Kozierok, 1993; Schlimmer and Hermens, 1993] that observe a user performing a task and assist with that task in the future. However, both conversational recommender systems and learning interface agents are designed to assist with only a single task. In contrast, our robot does not know in advance the specific task it will be performing so it cannot bias itself toward learning preferences for that task.

In preference-based planning [Baier and McIlraith, 2008], a user's predefined preferences are incorporated into automated planning tasks. Instead of being defined in advance, the user's planning preferences can also be learned from previous plans the user has generated [Li *et al.*, 2009]. In our work, this would be equivalent to an operator manually controlling the robot in order to provide demonstrations to the robot. This would not be practical in time-sensitive situations or when the operator did not have a fully constructed plan for how the robot should perform the task (e.g., the operator might not know or care about the exact route the robot takes).

In human-robot interaction, it is often beneficial for the robot to attempt to interpret the environment from the perspective of a human teacher [Berlin *et al.*, 2006]. This can allow the robot to discover information it would not have seen from its own viewpoint during a demonstration of a task [Breazeal *et al.*, 2009]. This is similar to our work in that it attempts to interpret information from a secondary perspective but, like with preference-based planning, requires the teacher to provide demonstrations.

7 Conclusions

In this paper, we described our approach to inverse trust estimation and how it can be used by a semi-autonomous robot that is part of a human team. The inverse trust estimation differs from a traditional trust metric in that it allows the robot to infer how much trust an operator has in it rather than measure how trusting it is of an operator. The robot uses this trust estimation to determine when it should adapt its behavior in order to be a more trustworthy member of the team.

The robot learns as it adapts its behavior by storing information about previously evaluated behaviors. Case-based reasoning is used to leverage this information and find trustworthy behaviors more efficiently. We demonstrated the efficiency of case-based adaptation in a simulated robotics domain and found it significantly outperformed an adaptation approach that does not learn.

The primary benefit of this approach is that it does not require any background knowledge about the operator, tasks, environment, or context. However, this also limits the approach by restricting it to relying on an expensive random walk adaptation when acquiring cases. Future work will look at how supplemental information, like occasional operator feedback, can be used to improve the efficiency of adaptation. We also plan to allow the robot to reason about its goals and the goals of the team. This would allow the robot to verify it is trying to achieve team goals and identify if any sudden goal changes occur.

Acknowledgments

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

References

[Baier and McIlraith, 2008] Jorge A. Baier and Sheila A. McIlraith. Planning with preferences. *AI Magazine*, 29(4):25–36, 2008.

[Berlin *et al.*, 2006] Matt Berlin, Jesse Gray, Andrea Lockerd Thomaz, and Cynthia Breazeal. Perspective taking: An organizing principle for learning in human-robot interaction. In *21st National Conference on Artificial Intelligence*, pages 1444–1450, 2006.

[Breazeal *et al.*, 2009] Cynthia Breazeal, Jesse Gray, and Matt Berlin. An embodied cognition approach to mindreading skills for socially intelligent robots. *International Journal of Robotic Research*, 28(5), 2009.

[Briggs and Smyth, 2008] Peter Briggs and Barry Smyth. Provenance, trust, and sharing in peer-to-peer case-based web search. In *9th European Conference on Case-Based Reasoning*, pages 89–103, 2008.

[Carlson *et al.*, 2014] Michelle S. Carlson, Munjal Desai, Jill L. Drury, Hyangshim Kwak, and Holly A. Yanco. 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*, pages 20–27, Palo Alto, USA, 2014.

[Desai *et al.*, 2013] Munjal Desai, Poornima Kaniarasu, Mikhail Medvedev, Aaron Steinfeld, and Holly Yanco. Impact of robot failures and feedback on real-time trust. In *8th International Conference on Human-robot Interaction*, pages 251–258, 2013.

[Esfandiari and Chandrasekharan, 2001] Babak Esfandiari and Sanjay Chandrasekharan. On how agents make friends: Mechanisms for trust acquisition. In *Proceedings of the 4th Workshop on Deception, Fraud and Trust in Agent Societies*, pages 27–34, Montreal, Canada, 2001.

[Floyd *et al.*, 2014] Michael W. Floyd, Michael Drinkwater, and David W. Aha. How much do you trust me? Learning a case-based model of inverse trust. In *Proceedings of the 22nd International Conference on Case-Based Reasoning*, pages 125–139, Cork, Ireland, 2014. Springer.

[Hancock *et al.*, 2011] Peter A. Hancock, Deborah R. Billings, Kristin E. Schaefer, Jessie Y.C. Chen, Ewart J. De Visser, and Raja Parasuraman. 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, 2011.

[Jian *et al.*, 2000] Jiun-Yin Jian, Ann M. Bisantz, and Colin G. Drury. Foundations for an empirically determined scale of trust in automated systems. *International Journal of Cognitive Ergonomics*, 4(1):53–71, 2000.

[Kaniarasu *et al.*, 2012] Poornima Kaniarasu, Aaron Steinfeld, Munjal Desai, and Holly A. Yanco. Potential measures for detecting trust changes. In *7th International Conference on Human-Robot Interaction*, pages 241–242, Boston, USA, 2012.

[Kaniarasu *et al.*, 2013] Poornima Kaniarasu, Aaron Steinfeld, Munjal Desai, and Holly A. Yanco. Robot confidence and trust alignment. In *Proceedings of the 8th International Conference on Human-Robot Interaction*, pages 155–156, Tokyo, Japan, 2013.

[Knexus Research Corporation, 2015] Knexus Research Corporation. eBotworks. http://www.knexusresearch.com/products/ebotworks.php, 2015. [Online; accessed February 27, 2015].

[Leake and Whitehead, 2007] David Leake and Matthew Whitehead. Case provenance: The value of remembering case sources. In *7th International Conference on Case-Based Reasoning*, pages 194–208, 2007.

[Li *et al.*, 2009] Nan Li, Subbarao Kambhampati, and Sung Wook Yoon. Learning probabilistic hierarchical task networks to capture user preferences. In *21st International Joint Conference on Artificial Intelligence*, pages 1754–1759, 2009.

[Maes and Kozierok, 1993] Pattie Maes and Robyn Kozierok. Learning interface agents. In *11th National Conference on Artificial Intelligence*, pages 459–465, 1993.

[McGinty and Smyth, 2003] Lorraine McGinty and Barry Smyth. On the role of diversity in conversational recommender systems. In *5th International Conference on Case-Based Reasoning*, pages 276–290, 2003.

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

[Oleson *et al.*, 2011] Kristin E. Oleson, Deborah R. Billings, Vivien Kocsis, Jessie Y.C. Chen, and Peter A. Hancock. Antecedents of trust in human-robot collaborations. In *Proceedings of the 1st International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, pages 175–178, 2011.

[Richter and Weber, 2013] Michael M. Richter and Rosina O. Weber. *Case-Based Reasoning - A Textbook*. Springer, 2013.

[Sabater and Sierra, 2005] Jordi Sabater and Carles Sierra. Review on computational trust and reputation models. *Artificial Intelligence Review*, 24(1):33–60, 2005.

[Saleh *et al.*, 2012] Jamil Abou Saleh, Fakhreddine Karray, and Michael Morckos. Modelling of robot attention demand in human-robot interaction using finite fuzzy state automata. In *International Conference on Fuzzy Systems*, pages 1–8, 2012.

[Schlimmer and Hermens, 1993] Jeffrey C. Schlimmer and Leonard A. Hermens. Software agents: Completing patterns and constructing user interfaces. *Journal of Artificial Intelligence Research*, 1:61–89, 1993.

[Tavakolifard *et al.*, 2009] Mozhgan Tavakolifard, Peter Herrmann, and Pinar Öztürk. Analogical trust reasoning. In *3rd International Conference on Trust Management*, pages 149–163, 2009.