Improving Trust-Guided Behavior Adaptation Using Operator Feedback

Michael W. Floyd[1], Michael Drinkwater[1], and David W. Aha[2]

[1] Knexus Research Corporation; Springfield, VA; USA
[2] Navy Center for Applied Research in Artificial Intelligence; Naval Research Laboratory (Code 5514); Washington, DC; USA
{*first.last*}@knexusresearch.com | david.aha@nrl.navy.mil

Abstract. It is important for robots to be trusted by their human teammates so that they are used to their full potential. This paper focuses on robots that can estimate their own trustworthiness based on their performance and adapt their behavior to engender trust. Ideally, a robot can receive feedback about its performance from teammates. However, that feedback can be sporadic or non-existent (e.g., if teammates are busy with their own duties), or come in a variety of forms (e.g., different teammates using different vocabularies). We describe a case-based algorithm that allows a robot to learn a model of feedback and use that model to adapt its behavior. We evaluate our system in a simulated robotics domain by showing that a robot can learn a model of operator feedback and use that model to improve behavior adaptation.

Keywords: inverse trust, behavior adaptation, adaptable autonomy

1 Introduction

Robots can be valuable members of teams if they provide the team with additional skills, reduce task load, or minimize potential risks to humans. In some scenarios, the robots' contributions could be vital to achieving team goals or successfully completing missions. We focus on semi-autonomous robots that can be issued high-level commands by an operator (e.g., *"move to the river", "patrol for threats"*). However, for the operator to use the robot to its full potential it needs to trust the robot. A lack of trust could result in unnecessarily monitoring the robot's actions, underutilizing the robot, or not using it at all [1].

We have previously examined how a robot can evaluate its trustworthiness using an *inverse trust metric* and adapt its behavior in an effort to engender trust [2]. Unlike traditional trust metrics that can be used to measure a robot's trust in its operator, an inverse trust metric estimates how much trust the operator has in the robot. Our inverse trust metric uses the robot's performance to estimate trust and measures general trends in trustworthiness (i.e., increasing, decreasing, remaining constant). Behavior adaptation is performed using case-based reasoning (CBR) and allows the robot to adapt to changes in operators, missions, or contexts. However, our earlier approach assumes that no explicit

feedback is provided by the operator so only observable indicators of the robot's performance are used. Although such an assumption is beneficial in scenarios where the operator does not have time to provide explicit feedback, it limits the robot's ability to use this information when it is available.

In this paper we describe an extension of our previous work, which only used implied feedback (i.e., the operator allowed the robot to complete a task or interrupted it), to allow the robot to utilize *explicit* operator feedback. No assumptions are made about the format of feedback so it is applicable when feedback comes in different modes (e.g., text, gestures, interface commands) or expressions (e.g., synonymous phrases, different languages). Similarly, no assumptions are made about the frequency of feedback. The robot uses case-based reasoning to learn a model for what the various pieces of feedback mean and uses that model to assist in behavior adaptation (e.g., if the operator says *"go faster"* the robot should increase its speed).

The remainder of this paper describes how a robot can learn a model of operator feedback and use that feedback to adapt its behavior. In Section 2, we provide a summary of our previous work on how a robot can adapt its behavior in an attempt to be a more trustworthy member of a team. Section 3 provides a discussion of related work. Section 4 describes the type of feedback that an operator can give to the robot, and Section 5 presents an approach for learning the meaning of feedback and using feedback to improve behavior adaptation. In Section 6, we evaluate our approach using scenarios from a simulated robotics domain. They indicate that the robot can use a learned feedback model to improve its behavior adaptation performance. Concluding remarks are discussed in Section 7.

2 Trust-Based Behavior Adaptation

The robot receives high-level commands from the operator (e.g., *"move to the building"*, *"patrol for threats"*, *"transport supplies to the hospital"*) and autonomously performs the tasks it is assigned. As the robot performs these tasks, it can control and modify its behavior by changing *modifiable components*. Each modifiable component i represents a single aspect of the robot's behavior and the robot is responsible for selecting a value m_i for that component from the set $\mathcal{M}_i$ of possible values ($m_i \in \mathcal{M}_i$). For example, the robot could select a parameter value from a set of possible values, an algorithm to use from a set of path planning algorithms, or data to use from a set of alternative data sources. Without loss of generality, we assume that the possible values are totally ordered, with an ordering relation between each pair of values ($m_i^j \prec m_i^k$, where $m_i^j, m_i^k \in \mathcal{M}_i$).

If the robot has n modifiable components, its current behavior B is a tuple containing the currently selected value for each modifiable component ($B = \langle m_1, m_2, \ldots, m_n \rangle$). In our work, the robot modifies its behavior in an attempt to increase its trustworthiness. Unlike traditional trust metrics [3] where the robot measures its trust in another agent, an *inverse trust metric* [2] is used to estimate

how much trust an agent has in the robot. Since inverse trust is measured from the robot's perspective, only observable indicators of human-robot trust can be used (i.e., none of the human's internal reasoning information). The robot estimates its trustworthiness based on when it successfully completes assigned tasks, when it fails to complete assigned tasks, and when it is interrupted while performing a task.

The inverse trust metric evaluates the trustworthiness of the current behavior B and tracks trends in trust over time. If the robot has received c commands, the metric will include information from each of those commands. Successfully completed commands will increase the trust estimate, and failed commands or interruptions will decrease the estimate ($cmd_i \in \{-1, 1\}$, with a weight w_i based on the command's relative importance):

$$Trust_B = \sum_{i=1}^{c} w_i \times cmd_i$$

The robot updates this value as more commands are issued and compares it to two thresholds: the trustworthy threshold (τ_T) and the untrustworthy threshold (τ_U). If the estimate reaches the trustworthy threshold ($Trust_B \geq \tau_T$), the robot concludes it has found a sufficiently trustworthy behavior but continues to estimate trust in case there is a change in operator, mission, or goals. If the estimate reaches the untrustworthy threshold ($Trust_B \leq \tau_U$), the robot concludes its behavior is untrustworthy and should be changed. Otherwise ($\tau_U < Trust_B < \tau_T$), the robot continues to monitor the trust estimate until it is more confident about its trustworthiness.

When the untrustworthy threshold is reached, the robot changes its behavior from B to a new behavior B' and begins measuring the trustworthiness of that behavior (i.e., $Trust_{B'}$). The behavior that was found to be untrustworthy, along with the time t it took to reach the untrustworthy threshold, are stored as an *evaluated pair* E ($E = \langle B, t \rangle$).

As the robot evaluates more behaviors, it maintains a set of previously evaluated behaviors $\mathcal{E}_{past}$ that contains all of the behaviors it has found to be untrustworthy ($\mathcal{E}_{past} = \{E_1, E_2, \dots\}$). These previously evaluated behaviors represent the search path the robot has taken while searching for a trustworthy behavior B_{final}. In a CBR context, the previously evaluated behaviors are the *problem* and the final behavior is the *solution*. We use the following case representation:

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

When the robot successfully finds a trustworthy behavior, it creates a new case and stores it in its case base. The motivation for using this case representation is that two operators who find similar behaviors untrustworthy in a similar amount of time may find similar behaviors to be trustworthy. When the robot needs to adapt its behavior (i.e., the trust metric reaches the untrustworthy threshold), it compares the behaviors it has previously evaluated ($\mathcal{E}_{past}$) to each of the cases. If a case is sufficiently similar and its final behavior has not already been evaluated as untrustworthy, the robot switches to use that case's final behavior.

If no cases are sufficiently similar, the robot performs a random walk. This form of behavior adaptation finds the evaluated behavior that took the longest to be labeled as untrustworthy and modifies that behavior. This works under the assumption that the behavior that took the longest to reach the untrustworthy threshold was the closest to being trustworthy, so a slight modification could make it more trustworthy. A more detailed description of case creation, case retrieval, case-based behavior adaptation, and random walk behavior adaptation can be found in [2].

Our previous work only accounted for *implied* feedback (i.e., successes, failures, and interruptions). The primary contribution of this paper is extending our previous work to allow for *explicit* feedback from the operator.

3 Related Work

Kaniarasu et al. [4] have also examined an online, performance-based estimate of operator trust. Their measure tracks only negative factors (e.g., how often the robot is warned of poor performance, or the operator manually controls the robot), so it can identify only decreases in operator trust. To also track increases in trust, they extended their measure to incorporate performance information from the operator at regular intervals [5] (e.g., the operator provides the robot feedback about its performance every 30 seconds). However, this approach is unable to track trust for any periods where explicit feedback in unavailable. Saleh et al. [6] estimate operator trust using a set of expert-authored rules. Since the rules could be different for each operator, mission, or context, this measure requires an expert redefine the rules whenever a change occurs.

In case-based reasoning, research has focused on traditional trust rather than inverse trust, and has generally been examined in the context of agent collaboration [7] or recommendation systems [8]. Case provenance [9] also deals with trust but focuses on the trustworthiness of a case's source rather than the trustworthiness of an agent or system.

Our work on inverse trust is related to learning a user's preferences. The ability to incorporate a user's preferences has been examined in areas such as learning interface agents and preference-based planning [10]. Learning interface agents build a model of a user by watching the user perform a task (e.g., e-mail sorting [11] or schedule management [12]) and later assisting the user in performing that task. Similarly, preference-based planners can learn a user's planning preferences by observing the user perform a planning task [13]. In our work, these demonstration-based approaches would be equivalent to the operator manually controlling the robot and performing the task. This would not be practical in time-sensitive situations or when the operator does not have a fully constructed plan for how a task should be performed.

Both user preferences and feedback play an important role in human-in-the-loop CBR systems, such as conversational case-based reasoning systems [14]. These interactions tend to be in the form of dialogs between the user and the system, whereas in our work interactions are one-sided (i.e., information is passed

only from the operator to the robot) and sporadic. Conversational recommender systems [15] use feedback to refine a model of the user and iteratively improve the recommendations that are provided. Similarly, feedback can also be used to tailor what questions to ask a user [16], thereby influencing what feedback will be provided in the future. Whereas our system is designed to work in a variety of tasks and missions, these approaches focus on a single task (i.e., recommendation).

The meaning of explicit operator feedback is learned by the robot by determining a relationship between its behavior when the feedback was received and its final trustworthy behavior. Relationships are similar to compound critiques [17] in recommender systems in that they represent the changes that should be made to a set of features. More generally, learning behavior relationships is similar to rule-induction [18]. The primary difference between these approaches and our own is that behavior relationships are generated using two data points (e.g., the behavior when feedback was received and the trustworthy behavior), rather than a full or partial subset of observations.

4 Operator Feedback

The primary methods used by the operator to interact with the robot are issuing commands and interrupting (i.e., implied feedback). However, the operator can also provide additional information to the robot in the form of *explicit feedback*. For the remainder of this paper, when referring to feedback we mean explicit feedback rather than implied feedback.

Feedback is provided at the operator's discretion, so no assumptions are made about when it will occur or how often it will be provided. The frequency of feedback is operator-specific (i.e., some operators prefer to provide more feedback) but is also influenced by the operator's workload. For example, an operator would likely have less time to provide feedback to the robot during an emergency situation. In the extreme case, the robot would not receive any feedback from the operator.

For a robot that supports multimodal interaction, feedback can be provided by any of the available modes (e.g., written text, speech, gestures, user interface commands). This allows the operator to provide the same feedback in a variety of ways. For example, the operator can tell the robot to stop by saying the word *"stop"*, making a hand gesture, or pressing a keyboard button. Similarly, the operator can provide the same feedback in a variety of ways using a single mode of interaction (e.g., *"go faster"*, *"speed up"*, *"get going"*, saying it in other languages). The robot could use computational semantics or machine translation to group similar utterances, but this might not be feasible due to the robot's computational constraints. Similarly, the robot would need a method for grouping similar pieces of feedback from different modalities.

In some domains, the format of feedback can be formally defined such that the robot has a prior model of what feedback it can receive and what each piece of feedback means. However, this requires the operator to be aware of the

format and structure its feedback accordingly. It would be difficult to enforce this requirement if the robot is expected to interact with a variety of operators with minimal training (e.g., a robot that is part of an ad-hoc team or a mass-produced consumer robot). Even if the operator is fully aware of how to correctly provide feedback, the format might limit how expressive the feedback can be. This would make it difficult to provide feedback if the team encounters new environments, new tasks, or unforeseen events. Instead, we will examine how the robot can learn a model of operator feedback without any prior knowledge about the frequency, format, or modality of feedback.

Each time feedback is provided by the operator, the robot stores a pair F containing the feedback f and the behavior B that was being used by the robot when the feedback was received ($F = \langle f, B \rangle$). This representation encodes the circumstances under which the operator decided to provide feedback (i.e., how the robot was behaving) as well as the information the operator was trying to convey to the robot (i.e., the feedback). This makes the assumption that the operator's feedback is a direct response to how the robot is currently behaving. If the operator provides feedback about a previous behavior (e.g., *"You were driving too slowly five minutes ago."*), this encoding will erroneously attribute that feedback to the current behavior. However, we anticipate that such delayed feedback will be relatively rare compared to feedback about the current behavior or delayed feedback that is still valid for the current behavior (e.g., the robot was driving slowly five minutes ago and is still driving slowly).

Over the course of operation, the robot will maintain a set $\mathcal{F}_{received}$ of received feedback ($\mathcal{F}_{received} \subseteq \mathcal{F}$, where $\mathcal{F}$ is the set of all possible feedback items). This set, which will be empty initially, will be extended when the robot receives a new feedback item ($\mathcal{F}_{received} = \bigcup_{i=1}^{n} F_i$, where n is the number of feedback items received).

5 Feedback Model

We have described how the robot can record feedback but not how it can leverage that information. This section will present methods that allow the robot to learn from feedback and use that feedback in an attempt to improve its behavior adaptation performance.

5.1 Learning the Feedback Model

Since the meaning of feedback is initially unknown to the robot, it needs to learn a feedback model. The feedback items themselves do not provide enough information to build the model because they capture only what the robot's behavior was at the time the feedback was received. The robot also needs to know what it should have done in response to the feedback. Since feedback is received while searching for a trustworthy behavior, when the robot finds a trustworthy behavior it can use that behavior to build its feedback model.

The feedback model is structured as a case base that contains guidelines for how the robot should adapt its behavior in response to feedback. We refer to this case base as the *feedback base* to differentiate it from the case base used for case-based behavior adaptation. A case FR is defined as:

$$FR = \langle f, R, cnt \rangle$$

Each case contains a piece of feedback f, a relationship R, and a frequency count cnt. The relationship represents guidelines for how the robot should adapt its behavior in response to the feedback. For any pair of behaviors (e.g., the behavior when feedback was received and a final trustworthy behavior), the relationship encodes how the two behaviors differ ($relation : \mathcal{B} \times \mathcal{B} \rightarrow \mathcal{R}$, where $\mathcal{B}$ is the set of all behaviors and $\mathcal{R}$ is the set of all relationships). More specifically, the relationship encodes how the modifiable components of each behavior differ. A relationship can be determined for each pair of modifiable components ($rel : \mathcal{M}_i \times \mathcal{M}_i \rightarrow \mathcal{O}, \mathcal{O} = \{\prec, \succ, =\}$). The relationship R_{ij} between two behaviors B_i and B_j contains the relationship between each of their modifiable components ($|B_i| = |B_j| = |R_{ij}|, R_{ij} = \langle rel(B_i.m_1, B_j.m_1), rel(B_i.m_2, B_j.m_2), \dots \rangle$).

Consider an example where the robot has two modifiable components: its speed and its object padding (how far it attempts to stay away from obstacles when planning its movement). If the robot receives the feedback *"go faster"* while using a speed of 1 meter/second and a padding of 0.5 meters ($B_1 = \langle 1, 0.5 \rangle$), and eventually finds a behavior with a speed of 5 meters/second and a padding of 0.5 meters ($B_2 = \langle 5, 0.5 \rangle$) to be trustworthy, the relationship will show the speed increased and the padding remained constant ($R_{12} = \langle \prec, = \rangle$).

The *cnt* value stores the number of times that feedback f resulted in the relationship R being observed. The motivation for storing this value is that it is possible to observe unnecessary relationships or erroneous relationships. An unnecessary relationship would occur if the robot changed one or more modifiable components when it did not need to (e.g., in an attempt to go faster the robot changed both its speed and its padding), whereas an erroneous relationship would occur when the operator gives incorrect feedback (e.g., telling the robot to go faster when it is already driving fast enough). We make the assumption that correct relationships, even if they contain unnecessary modifications, will occur more frequently than erroneous relationships. Using this case definition, the feedback can be thought of as the *problem*, the relationship as the *solution*, and the frequency count a measurement of the quality of a relationship.

Algorithm 1 shows the process the robot uses to refine its feedback model. The algorithm is used at the end of a search when the robot has found a trustworthy behavior (this is also when it creates and stores behavior adaptation cases). It receives as input the set of received feedback items $\mathcal{F}_{received}$, the trustworthy behavior B_{final} that was found, and its feedback base $FeedbackBase$. The algorithm iterates through each of the feedback items (line 1) and checks to see if the behavior when feedback was received differs from the final behavior (line 2). This check is performed to ensure the robot stores only cases when feedback required it to adapt its behavior (i.e., it would never store the relationship

$\langle =, =, =, \ldots \rangle$). If the behaviors differ, the relationship between the behaviors is computed (line 3). If the feedback base already contains a case with that feedback and relationship, the frequency count for that case is increased (line 5-8). Otherwise, a new case is created and added to the feedback base (lines 9-11). Once all feedback items have been processed, the set is emptied (line 12) and can again be extended as new feedback is received.

Algorithm 1: Process the feedback items received during a search

Function: *processFeedback*($\mathcal{F}_{received}$, B_{final}, $FeedbackBase$);

```
1  foreach F_i ∈ F_received do
2      if F_i.B ≠ B_final then
3          R_i ← relation(F_i.B, B_final);
4          exists ← false;
5          foreach FR_j ∈ FeedbackBase do
6              if FR_j.f = F_i.f and FR_j.R = R_i then
7                  FR_j.cnt ← FR_j.cnt + 1;
8                  exists ←true;
9          if !exists then
10             FR_new ← ⟨F_i.f, R_i, 1⟩;
11             FeedbackBase ← FeedbackBase ∪ FR_new;
12 F_received ← ∅;
```

5.2 Using the Feedback Model

We have previously described how the robot stores the feedback it receives (Section 4) and will now describe how the robot uses the feedback model it has learned to adapt its behavior. Algorithm 2 is called when the operator provides the robot with feedback. A new feedback item is created from the received feedback and current behavior (line 1), and is stored in the set of feedback items (lines 2). The algorithm iterates through all feedback relationships in the feedback base (line 5) and stores the most frequent feedback relationship for the given feedback (lines 6-8). This is because there can be multiple feedback relationships for each type of feedback, so only the best relationship (i.e., the one with the highest frequency value) is used. If no feedback relationship is found (i.e., the feedback base is empty or no relationship has been found for that feedback yet), the robot does not change its behavior (lines 9-10). However, if a feedback relationship is found, then the robot uses the *applyRelationship(...)* function to modify its behavior.

The *applyRelationship(...)* function does the following:

1. The current behavior B_{curr} is stored along with its current trust estimate $Trust_{B_{curr}}$ and evaluation time t_{curr}. These are stored because behavior

Algorithm 2: Receive feedback from the operator

Function: *receiveFeedback(*f*,* B_{curr}*,* $\mathcal{F}_{received}$*,* $FeedbackBase$*)* ***returns*** B_{new};

```
1  F_new ← ⟨f, B_curr⟩;
2  F_received ← F_received ∪ F_new;
3  bestFrequency ← 0;
4  R_best ← ∅;
5  foreach FR_i ∈ FeedbackBase do
6      if FR_i.f = f and FR_i.cnt > bestFrequency then
7          bestFrequency ← FR_i.cnt;
8          R_best ← FR_i.R;
9  if R_best = ∅ then
10     return B_curr;
11 return applyRelationship(B_curr, R_best);
```

adaptation is triggered by feedback, not by the behavior being labelled as trustworthy or untrustworthy. Since the feedback can result in unnecessary behavior changes (e.g., erroneous feedback or incorrect feedback relationships), this allows the robot to continue evaluating the behavior at a later time.

2. A new behavior B_{new} is selected under the conditions that it has not already been found to be untrustworthy ($\forall E_i \in \mathcal{E}_{past}, E_i.B \neq B_{new}$) and it satisfies the relationship R_{best}. Ideally, the new behavior will satisfy the entire relationship ($relation(B_{curr}, B_{new}) = R_{best}$). However, if no behaviors meet the entire relationship (e.g., the relationship requires decreasing the robot's speed but the speed is already at its minimum value), B_{new} will be a behavior that partially satisfies the relationship.
3. If the new behavior has already been partially evaluated (i.e., its trust estimate and time were previously stored in Step 1), the trust estimate and evaluation time are loaded. This allows the robot to continue its previous evaluation of the behavior and avoids spending longer than necessary evaluating behaviors.

The feedback process works under the assumption that errors, either in the feedback provided by the operator or in the feedback model learning, are unavoidable. However, the relationships' frequency counts are used to reinforce correct relationships while ignoring poor relationships. For example, consider the situation where feedback is received, a relationship is selected, and applying the relationship results in a trustworthy behavior being found. Since the feedback is stored (lines 1-2 of Algorithm 2), the robot will generate the relationship again when it processes the feedback items (using Algorithm 1). This increases the relationship's frequency count and can increase the chance that it is used again in the future (i.e., that it will have the highest frequency count for that feedback). Similarly, if applying a relationship does not result in a trustworthy behavior being found, the robot will continue to adapt its behavior until a trustworthy

behavior is found (e.g., using further feedback, case-based adaptation, or random walk adaptation). When feedback items are eventually processed, a different relationship will likely be generated and have its frequency count increased. Since the unsuccessful relationship does not increase its frequency count it may be less likely to be used in the future (i.e., it may no longer have the highest frequency for that feedback).

6 Evaluation

In this section, we evaluate our claim that *learning a feedback model and using operator feedback can improve the performance of behavior adaptation.* Case-based behavior adaptation has previously been found to allow a robot to efficiently locate a trustworthy behavior [2]. However, it requires using the significantly less efficient random walk behavior adaptation to acquire cases. Since the robot starts with an empty case base and learns cases during during deployment, random walk behavior adaptation serves as a bottleneck. We focus on evaluating the improvements the feedback model provides compared to random walk adaptation. Our evaluation tests the following hypotheses:

H1: Learning and using a feedback model will demonstrate improved performance compared to random walk behavior adaptation.
H2: The performance improvement will increase as the model learns from feedback.

6.1 eBotworks Simulator

We use the eBotworks simulator [19] for our evaluation. eBotworks allows autonomous agents to control simulated robotic vehicles while interacting with human operators using a variety of command modalities (e.g., speech, text, user interface commands). This simulator was selected based on its built-in agent design framework, autonomy modules (e.g., natural language command interpretation and path planning), and experimentation and data collection capabilities. Additionally, eBotworks allows for non-deterministic environments and noisy sensory inputs.

In our evaluation, the robot is a wheeled unmanned ground vehicle (UGV) operating in an urban environment that is composed of ground features (e.g., paved roads, grass), objects (e.g., houses, vehicles, road barriers, traffic cones), and other agents (e.g., humans, other robots). The scenario we use in the evaluation involves the robot receiving commands from an operator to patrol between an initial location and a goal location. While patrolling, the robot continuously scans for suspicious objects. If a suspicious object is found, the robot moves toward it and uses its sensor for detecting explosives to determine if the object is a threat or harmless. After classifying each suspicious object, the robot continues patrolling.

In this scenario, the robot has four modifiable components of its behavior: speed, padding, scan time, and scan distance. Speed, measured in meters

per second, controls how quickly the robot moves through the environment, while padding, measured in meters, controls how far the robot attempts to stay away from obstacles when planning its path (i.e., lower padding makes it more likely to bump into objects). Scan time, measured in seconds, is how much time the robot spends scanning each suspicious object, and scan distance, measured in meters, is how close the robot gets to suspicious objects while scanning. Longer scan times and smaller scan distances increase the probability that the robot will successfully classify objects as threats or harmless. The possible values for each modifiable component are: $\mathcal{M}_{speed} = \{0.5, 1.0, \ldots, 10.0\}$, $\mathcal{M}_{padding} = \{0.1, 0.2, \ldots, 2.0\}$, $\mathcal{M}_{scantime} = \{0.5, 1.0, \ldots, 5.0\}$, $\mathcal{M}_{scandistance} = \{0.25, 0.5, \ldots, 1.0\}$.

6.2 Experimental Conditions

Our study uses simulated operators that issue natural language commands to the robot and monitor its performance. The simulated operators were selected to represent a subset of the control strategies of human operators, and each operator's preferences influence when the robot is able to complete a task and when it is interrupted. The operators evaluate the robot based on how quickly the task is completed, how safely it is completed, and how well it identifies and correctly classifies suspicious objects. Two simulated operators are used: *speed-focused* and *detection-focused*. The speed-focused operator prefers the task to be completed quickly (i.e., 95% probability of interrupting if the robot exceeds 120 seconds) and correctly (i.e., 100% probability of interrupting if the robot misses a suspicious object or incorrectly classifies it), with less focus on safety (i.e., 5% probability of interrupting if the robot hits an obstacle). The detection-focused operator prefers the task be completed correctly, but is less concerned with speed (i.e., 5% probability of interrupting if the robot exceeds 120 seconds) or safety.

The operators can give four types of natural language feedback in the following categories: speed feedback, safety feedback, false positive feedback (i.e., classifying a harmless object as a threat), and false negative feedback (i.e., missing a suspicious object or classifying a threat as harmless). Each category of feedback has three synonymous pieces of feedback that the operators can use interchangeably and with equal probability (e.g., *"go faster"*, *"speed up"*, *"get going"*). Although we use a simulated operator, this is done to represent that human operators may not use a fixed vocabulary for feedback. Every time an operator interrupts the robot it can, with probability p_f, give the robot feedback.

For each feedback probability $p_f \in \{0.00, 0.05, 0.10, \ldots, 1.00\}$, we perform 50 experimental *trials* and start from an initially empty feedback base (i.e., the robot has no feedback model at the start of the first trial with each feedback probability). At the start of each trial the robot is assigned a random initial behavior and a random operator (both with uniform distribution). A trial concludes when the robot successfully finds a trustworthy behavior or has evaluated all possible behaviors. Each trial is composed of numerous experimental *runs*. At the start of each run the environment is reset, the robot is placed at the start position, and *six suspicious* objects are placed in the environment (their

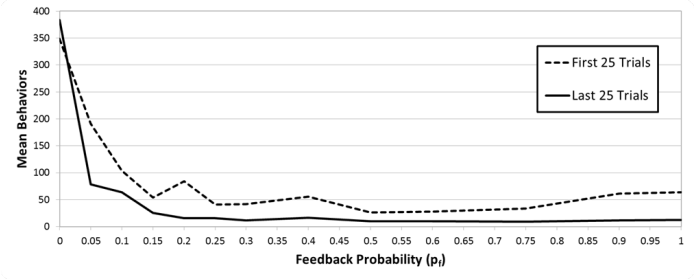

Fig. 1. Mean number of behaviors evaluated before a trustworthy behavior is found using a variety of feedback probabilities.

appearance and location are randomly selected each run). Between 0 and 3 of the objects (inclusive) are selected randomly to be threats while the remaining objects are harmless. A run concludes when the robot successfully completes the assigned tasks, fails, or is interrupted. At the end of a trial the robot updates its trust estimate and may adapt its behavior (either using random walk behavior adaptation or based on feedback). The robot stores and uses feedback (Algorithm 2) at the end of any run where feedback is provided, and updates the feedback base (Algorithm 1) at the end of each trial where a trustworthy behavior is found.

Since we are assessing how using feedback improves random walk behavior adaptation, which is used by case-based behavior adaptation to acquire cases, the robot uses only random walk adaptation. The robot uses a trustworthy threshold of $\tau_T = 5.0$ and an untrustworthy threshold of $\tau_U = -5.0$. These thresholds were selected to allow some fluctuation between increasing and decreasing trust while still identifying trustworthy and untrustworthy behaviors quickly.

6.3 Results

The mean number of behaviors that were evaluated before a trustworthy behavior was found is shown in Figure 1. The results are further divided into the mean for the first 25 trials and last 25 trials. When comparing the results when no feedback model is learned or used (i.e., $p_f = 0.0$) to when feedback is used (i.e., $p_f > 0.0$), using feedback results in a statistically significant improvement (using a paired t-test with $p < 0.001$). This provides evidence that hypothesis **H1** is supported.

Figure 1 also shows evidence that when feedback is used the performance increases in later trials. When $p_f > 0.0$, the performance in the last 25 trials (i.e., when the robot has had time to build a feedback model) is an improvement over the first 25 trials (i.e., when the model is empty or still being refined). Figure 2 examines this further by displaying the running mean (i.e., the value for trial N is the mean of the first N trials) using four feedback probabilities ($p_f \in \{0.00, 0.05, 0.50, 1.00\}$). In early trials, performance is poor because the

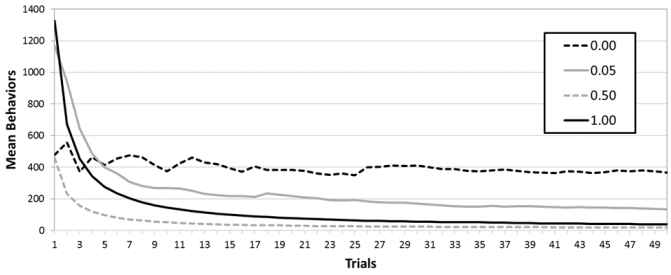

Fig. 2. Running mean number of behaviors evaluated over 50 trials.

feedback model is still being learned. The differences in performance in the first trials is because each of those trials starts at a random behavior, some of which are further from a trustworthy behavior than others. However, regardless of their early performance, all evaluations that used feedback (i.e., all but $p_f = 0.0$) had a mean that decreased as the number of trials increased. The improvement occurs because the robot refines its feedback model over time and improves its ability to adapt in response to feedback. This shows support for hypothesis **H2**.

6.4 Discussion

Even when feedback is relatively rare (e.g., $p_f = 0.05$), the robot can still improve its performance significantly. Additionally, there is no statistically significant difference in performance when p_f values between 0.15 and 1.0 are used. This indicates that this approach does not require near-constant feedback, but can perform well using moderate amounts of feedback. Similarly, since feedback is most important when the robot needs to do random walk behavior adaptation, the robot could request additional feedback when case-based behavior adaptation fails. This would be beneficial because it would not only improve the robot's ability to acquire additional behavior adaptation cases but would also inform the operator that a period of sub-optimal behavior should be expected (i.e., using random walk behavior adaptation to acquire cases rather than the more efficient case-based behavior adaptation).

At the end of the evaluation, the feedback bases contained between 81 and 309 feedback cases (mean of 175.25), with the majority of cases having low frequency counts (i.e., their relations were rarely found for their feedback item). The cases with the highest frequency counts tended to contain the relationships we would expect given the feedback. However, some cases with high frequency counts displayed unexpected relationships. For example, with speed-related feedback the relationships often indicated that speed should be increased and padding decreased. This relationship arises because lower padding allows the robot to navigate through narrow pathways and make tighter turns, ultimately increasing its speed.

7 Conclusions

In this paper, we presented an extension of our work on trust-guided behavior adaptation to allow for the incorporation of explicit operator feedback. Since the robot learns the feedback model, it does not require that the operator limits feedback to a fixed vocabulary (e.g., the operator can use synonyms for feedback). Similarly, behavior adaptation is not dependent on feedback so feedback is used only when it is available. Our approach is beneficial because it does not require a predefined feedback model but learns one over time. This model is continuously refined and updated as more information becomes available, improving the robot's response to feedback over time. However, a limitation of our approach is that new feedback is incorporated into the feedback model only after a trustworthy behavior is found. Until that point, the robot can use feedback to adapt but cannot refine the feedback model.

We evaluated our approach in a simulated robotics environment where the robot was responsible for patrolling an urban environment, identifying suspicious objects, and classifying them as threats or harmless. Our results indicate that by learning a feedback model and using it to assist in behavior adaptation the robot can significantly improve its behavior adaptation performance. Although the robot did not initially have a feedback model, it quickly learned one and used it to improve future performance.

One area of future work we plan to address is using the feedback base to allow the robot to explain its reasoning behind behavior adaptation. In this sense, the robot would search for similar solutions to its proposed solution (i.e., the relationship between the current behavior and the new behavior) and retrieve their associated problems (i.e., what feedback the operator might have been considering). This adds transparency between the robot and operator by providing information about the robot's reasoning process and can further increase trust [20]. We also plan to investigate how the robot can reason about its goals and the team's goals to ensure they compliment each other, and to detect any unexpected goal changes. Additionally, we plan to evaluate our trust-guided behavior adaptation approach in a series of user studies.

Acknowledgments

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

References

1. Oleson, K.E., Billings, D.R., Kocsis, V., Chen, J.Y., Hancock, P.A.: 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. (2011) 175–178

2. Floyd, M.W., Drinkwater, M., Aha, D.W.: 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, Springer (2014) 125–139
3. Sabater, J., Sierra, C.: Review on computational trust and reputation models. Artificial Intelligence Review **24**(1) (2005) 33–60
4. Kaniarasu, P., Steinfeld, A., Desai, M., Yanco, H.A.: Potential measures for detecting trust changes. In: 7th International Conference on Human-Robot Interaction. (2012) 241–242
5. Kaniarasu, P., Steinfeld, A., Desai, M., Yanco, H.A.: Robot confidence and trust alignment. In: 8th International Conference on Human-Robot Interaction. (2013) 155–156
6. Saleh, J.A., Karray, F., Morckos, M.: Modelling of robot attention demand in human-robot interaction using finite fuzzy state automata. In: International Conference on Fuzzy Systems. (2012) 1–8
7. Briggs, P., Smyth, B.: Provenance, trust, and sharing in peer-to-peer case-based web search. In: 9th European Conference on Case-Based Reasoning. (2008) 89–103
8. Tavakolifard, M., Herrmann, P., Öztürk, P.: Analogical trust reasoning. In: 3rd International Conference on Trust Management. (2009) 149–163
9. Leake, D., Whitehead, M.: Case provenance: The value of remembering case sources. In: 7th International Conference on Case-Based Reasoning. (2007) 194–208
10. Baier, J.A., McIlraith, S.A.: Planning with preferences. AI Magazine **29**(4) (2008) 25–36
11. Maes, P., Kozierok, R.: Learning interface agents. In: 11th National Conference on Artificial Intelligence. (1993) 459–465
12. Horvitz, E.: Principles of mixed-initiative user interfaces. In: 18th Conference on Human Factors in Computing Systems. (1999) 159–166
13. Li, N., Kambhampati, S., Yoon, S.W.: Learning probabilistic hierarchical task networks to capture user preferences. In: 21st International Joint Conference on Artificial Intelligence. (2009) 1754–1759
14. Aha, D.W., McSherry, D., Yang, Q.: Advances in conversational case-based reasoning. Knowledge Eng. Review **20**(3) (2005) 247–254
15. McGinty, L., Smyth, B.: On the role of diversity in conversational recommender systems. In: 5th International Conference on Case-Based Reasoning. (2003) 276–290
16. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: 20th ACM Conference on Hypertext and Hypermedia. (2009) 73–82
17. McCarthy, K., Reilly, J., McGinty, L., Smyth, B.: On the dynamic generation of compound critiques in conversational recommender systems. In: 3rd International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems. (2004) 176–184
18. Quinlan, J.R.: Generating production rules from decision trees. In: 10th International Joint Conference on Artificial Intelligence. (1987) 304–307
19. Knexus Research Corporation: eBotworks. `http://www.knexusresearch.com/products/ebotworks.php` (2015) [Online; accessed May 6, 2015].
20. Kim, T., Hinds, P.: 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. (2006) 80–85