Incorporating Transparency During Trust-Guided Behavior Adaptation

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

[1]Knexus Research Corporation; Springfield, VA, USA
[2]Navy Center for Applied Research in AI;
Naval Research Laboratory (Code 5514); Washington, DC, USA
michael.floyd@knexusresearch.com
david.aha@nrl.navy.mil

Abstract. An important consideration in human-robot teams is ensuring that the robot is trusted by its teammates. Without adequate trust, the robot may be underutilized or disused, potentially exposing human teammates to dangerous situations. We have previously investigated an agent that can assess its own trustworthiness and adapt its behavior accordingly. In this paper we extend our work by adding a transparency layer that allows the agent to explain why it adapted its behavior. The agent uses explanations based on explicit feedback received from an operator. This allows it to provide simple, concise, and understandable explanations. We evaluate our system on scenarios from a simulated robotics domain by demonstrating that the agent can provide explanations that closely align with an operator's feedback.

Keywords: Inverse trust, behavior adaptation, explanation, transparency

1 Introduction

Robots can be valuable additions to human teams if they provide additional skills to the team, lessen the humans' workload, or can replace humans in dangerous situations. However, even if a robot provides such benefits, the humans may not utilize it to its full potential if they do not trust it. If the robot is underutilized, it may actually increase the humans' workload (e.g., spending extra time observing the robot's behavior) or exposure to risks (e.g., performing dangerous tasks instead of the robot).

One option would be to hard-code the robot's behavior to ensure trustworthiness. However, this may not be feasible as the type of behavior that is considered trustworthy can depend on the teammate (e.g., their amount of experience working with robots), time (e.g., how long the robot has been with the team), or context (e.g., a routine versus a dangerous situation). Alternatively, the humans could explicitly tell the robot whether it is trustworthy. Yet providing such feedback may not be feasible during run-time if the team is in a time-critical situation. Similarly, if the feedback is given at the end of a mission (e.g., an after action trust survey) the robot may have performed the entire mission while acting in an untrustworthy manner.

Our previous work focused on an inverse trust estimate that allows a simulated robot[1] to estimate its own trustworthiness and adapt its behavior in situations where it believes it is untrustworthy. We investigated case-based methods that allow the agent to adapt its behavior in response to implicit [1] and explicit feedback [2]. In this paper, we extend our work by adding the ability for the agent to explain why it is adapting its behavior. Adding a level of transparency, wherein an automated system can explain the reasons for its actions, can increase the trustworthiness and reliance on automation [3]. By providing such explanations, even in situations where errors occur, it is possible to maintain trust at a higher level than if no explanations are provided.

In the remainder of this paper we will discuss how our case-based approach for behavior adaptation can also be used for explanation. Section 2 reviews how the agent estimates its trustworthiness using an inverse trust estimate and uses that estimate to guide behavior adaptation. In Section 3, we review the feedback model the agent uses to learn how to adapt its behavior in response to explicit feedback. Our approach for allowing the agent to use the same model for explanation is presented in Section 4. In Section 5, we use a military simulation to evaluate the ability of the agent to provide correct explanations to the user. Related work, with a specific focus on human-robot transparency and explanation, is discussed in Section 6, followed by conclusions and areas of future work in Section 7.

2 Trust-guided Behavior Adaptation

We assume that the agent receives commands from a single teammate called the *operator*. The operator provides the agent with high-level commands (e.g., "*move to the flag*", "*patrol for threats*") and it performs the assigned tasks autonomously. The agent also has direct control over the *modifiable components* of its behavior. These could be parameter values (e.g., minimum and maximum speeds), algorithms (e.g., choosing among alternative path planning algorithms), or data sources (e.g., using alternative maps of the environment). For each modifiable component i, the agent is responsible for selecting a value m_i from the partially ordered set $\mathcal{M}_i$ of possible values ($m_i \in \mathcal{M}_i$).

The agent's current behavior B is represented by the tuple containing the currently selected value for each of the n modifiable components: $B = \langle m_1, m_2, \ldots, m_n \rangle$. At any time, the agent can change the values of one or more modifiable components from its current behavior B to a new behavior B' (e.g., changing from $\langle m_1, m_2, \ldots, m_n \rangle$ to $\langle m'_1, m'_2, \ldots, m'_n \rangle$, where at least one $m_i \neq m'_i$). Although the agent can change its behavior for any reason, we will focus on trust-guided behavior adaptation (i.e., changing the robot's behavior in an attempt to make it more trustworthy).

Traditional trust metrics [4] allow an agent to measure its trust in other agents (e.g., teammates). However, for an agent to modify its behavior to be more trustworthy it must estimate another agent's trust in it. To perform such an estimate, we use an *inverse trust metric* [1]. Inverse trust is measured from the agent's perspective, so only

[1] For the remainder of this paper, we use the term *robot* to refer to a physical (or simulated) robot and *agent* to refer to the intelligent agent controlling the robot.

observable indicators of human-robot trust can be used (i.e., none of the human's internal reasoning information can be used).

Our inverse trust metric is based on the strong correlation between the agent's performance and human-robot trust [5]. The operator's perception of the agent's performance is not limited to a mission-level evaluation and can be influenced in real-time by both suitable and poor performance [6]. Without any guarantees of explicit feedback from the operator (i.e., the operator may not always have time to provide feedback), our agent uses implicit feedback to estimate its trustworthiness. In particular, it uses three types of implicit feedback related to its performance: successful completion of an assigned task, failure to complete an assigned task, and interruption by the operator. This assumes that completing a task will be viewed as satisfactory performance, whereas failure or interruption will be viewed as poor performance.

The agent estimates the trustworthiness $Trust_B$ of its current behavior B using the influence inf_i of each of the c commands it has completed. Successfully completed commands increase the trust estimate (i.e., $inf_i = 1$) whereas failed or interrupted commands decrease the trust estimate (i.e., $inf_i = -1$). Each command is also given a weight w_i related to its relative importance (e.g., giving a higher weight to more recent commands, giving higher weight to commands that involve human safety).

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

The trust estimate is recomputed after each command and compared to two threshold values: the trustworthy threshold (τ_T) and the untrustworthy threshold (τ_U). If the trust estimate reaches the trustworthy threshold ($Trust_B \geq \tau_T$), the agent concludes it is behaving in a trustworthy manner and continues to monitor its trustworthiness in case of any changes (e.g., a change in operator or mission context). If the trust estimate reaches the untrustworthy threshold ($Trust_B \leq \tau_U$), the agent concludes that its current behavior is untrustworthy and should be changed. Otherwise ($\tau_U < Trust_B < \tau_T$), the agent continues to monitor the trust estimate until it is more confident about its trustworthiness.

In the event that the untrustworthy threshold is reached, the agent changes from its current behavior B to a new behavior B' and begins measuring the trustworthiness of that behavior (i.e., $Trust_{B'}$). The behavior B 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 agent evaluates behaviors, it maintains a set $\mathcal{E}_{past}$ that contains all behaviors that have been found to be untrustworthy ($\mathcal{E}_{past} = \{E_1, E_2, \ldots\}$). This set represents behaviors encountered on the search path taken by the agent before it eventually finds a trustworthy behavior B_{final}. In a case-based reasoning context, the set of previously evaluated behaviors is the *problem* and the trustworthy behavior is the *solution*. A case is created and added to the case base each time the agent finds a trustworthy behavior. We use the following case representation:

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

This representation is motivated by the assumption that operators who find similar behaviors to be untrustworthy (i.e., similar problems) will also find similar behaviors to be trustworthy (i.e., similar solutions). A more detailed description of case acquisition, similarity calculation, case retrieval, and case-based behavior adaptation can be found in [1].

3 Feedback Model

Our inverse trust metric uses *implicit* operator feedback to estimate trust, but it is also possible for the agent to use *explicit* feedback. Although explicit feedback is provided at the operator's discretion (i.e., the agent does not know when or how often it will occur), it provides direct feedback on the agent's performance (e.g., "*go faster*", "*slow down*", "*watch out for obstacles*"). Initially, the agent has no knowledge about the type of feedback it will receive or what each piece of feedback means. As feedback is received, the agent learns a feedback model that contains information about how it should respond to operator feedback. For example, if the operator tells the agent "*go faster*", the agent should learn that this means it should increase its speed.

The agent acquires its feedback model by learning the relationships between its behavior when feedback is received and a trustworthy behavior (i.e., how it was behaving when feedback was received and how it should behave). These relationships are stored in a *feedback base*, where each feedback relationship case FR is defined as:

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

Each feedback relationship case contains a piece of feedback f, a relationship R, and a frequency cnt. For any pair of behaviors B_i and B_j, the relationship R_{ij} encodes how the behaviors differ ($\mathcal{B} \times \mathcal{B} \rightarrow \mathcal{R}$, where $\mathcal{B}$ is the set of all behaviors and $\mathcal{R}$ is the set of all relationships). More specifically, a relationship encodes how each pair of modifiable component values differ ($rel: \mathcal{M}_i \times \mathcal{M}_i \rightarrow \mathcal{O}, \mathcal{O} = \{\prec, \succ, =\}$). The overall relationship R_{ij} is a tuple containing each of the modifiable component relationships ($|B_i| = |B_j| = |R_{ij}| = n, R_{ij} = \langle rel(B_i.m_1, B_j.m_1), \ldots, rel(B_i.m_n, B_j.m_n) \rangle$).

The frequency cnt measures how many times the relationship R was found for feedback f. Since the cases in the feedback base are learned by the agent, it is possible for unnecessary or erroneous relationships to be learned (e.g., the operator gave incorrect feedback). The agent works under the assumption that unnecessary and erroneous relationships occur less frequently than correct ones, so preference is given to relationships with higher frequency values.

Consider an example where the agent has two modifiable components: its speed and its object padding (how far it attempts to stay away from obstacles when planning its movement). The agent receives the feedback "*go faster*" when using a behavior B_1 with a speed of 1.0 meter/second and a padding of 0.5 meters ($B_1 = \langle 1.0, 0.5 \rangle$). Eventually, the agent finds a trustworthy behavior B_2 with a speed of 5.0 meters/second and a padding of 0.5 meters ($B_2 = \langle 5.0, 0.5 \rangle$). The relationship R_{12} between them would show the speed increased while the padding remained constant ($R_{12} = \langle \prec, = \rangle$). If this was the first time this relationship was learned for the feedback "*go faster*", the cnt

value would be 1 ($FR_{example} = \langle$"$go\ faster$", $\langle \prec, = \rangle$,1$\rangle$). If the agent receives the feedback "*go faster*" again, it can retrieve this feedback relationship case and know to increase its speed. A full description of how the agent can learn a feedback model is described in [2]. For the remainder of this paper, we will assume that the agent already has a feedback model available to use.

4 Behavior Adaptation Explanation

As we explained in the previous sections, the agent has two methods for modifying its behavior: adapting in response to implicit feedback and adapting in response to explicit feedback. Adaptation in response to explicit feedback occurs directly after the feedback is received. This provides the operator with a direct connection between their feedback and the behavior change. However, adaptation in response to implicit feedback occurs over a longer period of time (i.e., the entire time the agent is measuring the trustworthiness of a behavior). Since there may not be any single event that caused the agent to change its behavior, it may not be clear to the operator why the behavior change occurred.

To obtain transparency between the agent and the operator, it can provide an explanation when it adapts its behavior in response to implicit feedback. The information contained in the agent's explanation could be in different forms and at varying levels of abstraction (e.g., a visual representation of the agent's trust estimate over time, a list of the commands that were failed or interrupted, an acknowledgement that a behavior change occurred). However, an explanation may not be useful to the operator if it is verbose or difficult to interpret. For example, if the agent provided an explanation that included a complete list of all assigned commands and their results (i.e., all the information it uses to compute the inverse trust estimate), the operator may ignore it.

We designed our agent to provide explanations using a method of communication that we believe will be appropriate for the operator. To achieve this, the agent uses the model of explicit feedback that it has learned from the operator, since the feedback is both understandable (i.e., the operator has used it to communicate with the agent) and succinct (i.e., the operator was able to provide the feedback under real-time constraints). By using the operator's own feedback in explanations, the agent relates that its behavior adaptation is motivated by the predicted actions of the operator. For example, if the agent adapts its behavior by increasing its speed, it can provide the explanation "*I adapted my behavior because I think you were going to tell me to* ***speed up***".

The agent generates an explanation (Algorithm 1) when it adapts its current behavior B_{curr} to a new behavior B_{adapt} (i.e., it performs case-based behavior adaptation when the inverse trust metric reaches the untrustworthy threshold). The relationship R between the two behaviors is calculated (line 2) and compared to the relationship stored in each feedback relationship (i.e., $FR.R$) in the feedback base $FeedbackBase$ (lines 3-4). Any feedback relationship that contains the relationship R is added to the set $\mathcal{F}$ (line 5). The $selectFeedback(...)$ function selects a single piece of feedback from among the feedback items stored in $\mathcal{F}$ (line 6).

Algorithm 1: Adaptation Explanation

Function: *findExplanation*(B_{curr}, B_{adapt}) ***returns*** *expectedFeedback*

```
1 F ← ∅;
2 R ← relationship(B_curr, B_adapt);
3 foreach FR ∈ FeedbackBase do
4     if FR.R = R then
5         F ← F ∪ FR;
6 return selectFeedback(F);
```

We propose four alternative implementations for the $selectFeedback(\dots)$ function:

- **Highest Count**: The feedback relationship FR_i with the largest frequency is selected ($FR_i \in \mathcal{F}, \forall FR_j \in \mathcal{F}, FR_i.cnt \geq FR_j.cnt$) and its feedback $FR_i.f$ is returned. If multiple feedback relationships are tied for the largest frequency, one is selected at random according to a uniform distribution.
- **Highest Group Count**: The feedback relationships are partitioned into k subsets such that all feedback relationships in a subset have the same associated feedback and there is only one subset per feedback type ($\mathcal{P}_1 \cup \mathcal{P}_2 \cup \dots \cup \mathcal{P}_k = \mathcal{F}, \mathcal{P}_1 \cap \mathcal{P}_2 \cap \dots \cap \mathcal{P}_k = \emptyset, \forall FR_i, FR_j \in \mathcal{P}_l, FR_i.f = FR_j.f$). For each subset, the frequency of all feedback relationships in the subset are summed ($sum_l = \sum_{FR_i \in \mathcal{P}_l} FR_i.cnt$) and the subset with the largest summed frequency has its feedback returned. If multiple subsets tie for the largest summed frequency, one is selected at random according to a uniform distribution.
- **Mean Group Count**: The feedback relationships are partitioned and the mean frequency count for each subset is calculated ($\mu_l = \frac{sum_l}{|\mathcal{P}_l|}$). The subset with the largest mean frequency count has its feedback returned. If multiple subsets tie for the largest mean frequency count, then one is selected at random according to a uniform distribution.
- **Random**: One feedback relationship is randomly selected from $\mathcal{F}$ according to a uniform distribution. The feedback stored in the feedback relationship is returned.

The piece of feedback $expectedFeedback$ that is returned from Algorithm 1 is used to produce a human-readable explanation for the operator. The explanation takes the following form:

"I adapted my behavior because I think you were going to tell me to ***<expectedFeedback>****"*

For example, if Algorithm 1 returned "$drive\ safely$", the generated explanation would be "*I adapted my behavior because I think you were going to tell me to* ***drive safely***".

5 Evaluation

In this section, we evaluate our claim that *the agent produces explanations for its behavior that align with the operator's evaluation of the agent*. The evaluation uses simulated operators so the agent's trustworthiness from a human's perspective cannot be directly measured. However, we can measure the agent's ability to perform actions that have been shown to positively influence trust (i.e., providing explanations). Our evaluation tests the following hypotheses:

H1: The explanations provided by the agent are consistent with the explicit feedback the operator would have provided had the opportunity arisen.
H2: The explanations provided by the agent outperform a random baseline.
H3: The agent provides better explanations using a manually authored feedback base compared to a learned feedback base.

5.1 Domain: eBotworks

We use the eBotworks simulator [7] for our evaluation. eBotworks allows autonomous agents to control simulated robots in a simulated urban environment. In our evaluation, the agent controls an unmanned ground vehicle (UGV) in an environment composed of other agents (e.g., humans, other simulated UGVs), obstacles (e.g., buildings, vehicles, traffic cones, boxes), and ground features (e.g., roads, grass). We chose to use eBotworks because it provides a built-in agent design framework, autonomy modules (e.g., natural language command interpretation, path planning), and allows for evaluation in a non-deterministic and noisy environment.

The scenario we use involves the agent-controlled robot receiving natural language commands from an operator. The commands instruct the agent to patrol between its current location and a goal location. While patrolling, it continuously scans for suspicious objects. If a suspicious object is found, the robot pauses its patrol, moves toward the object, and uses explosive-detection sensors to determine if it is *dangerous* or *harmless*. After classifying a suspicious object, the robot continues patrolling.

The robot has the following modifiable components (and possible values they can take):

- **Speed (meters per second)**: The maximum speed the simulated robot uses when moving through the environment. $\mathcal{M}_{speed} = \{0.5, 1.0, \dots, 10.0\}$
- **Padding (meters):** How far the robot attempts to stay away from obstacles when planning its path. Higher paddings decrease the likelihood that it will collide with obstacles. $\mathcal{M}_{padding} = \{0.1, 0.2, \dots, 2.0\}$
- **Scan Time (seconds)**: How much time the robot spends scanning each suspicious object. Higher scan times increase the probability that the robot will successfully classify the suspicious object. $\mathcal{M}_{scantime} = \{0.5, 1.0, \dots, 5.0\}$
- **Scan Distance (meters)**: How close the robot gets to the suspicious object while scanning it. Lower scan distances increase the probability that the robot will successfully classify the suspicious object. $\mathcal{M}_{scandistance} = \{0.25, 0.5, \dots, 1.0\}$

5.2 Experimental Setup

Our study uses simulated operators to issue commands to the agent and monitor its behavior. The simulated operators were selected to represent a subset of control strategies used by human operators (i.e., when to allow the agent to complete a task and when to interrupt). Two simulated operators are used: *speed-focused* and *detection-focused*. The speed-focused operator prefers the task to be performed quickly (i.e., 95% probability of interrupting if the robot does not complete the task within 120 seconds) and correctly (i.e., 100% probability of interrupting if the robot misses a suspicious object or incorrectly classifies it). The detection-focused operator prefers the task to be performed correctly but is less concerned about speed (i.e., 5% probability of interrupting if the robot exceeds 120 seconds). Both operators place a relatively low emphasis on the robot's safety (i.e., 5% probability of interrupting if the robot comes in contact with an obstacle).

At the start of each experimental *trial*, the robot is assigned a random initial behavior (i.e., a random value for each modifiable component from the set of possible values according to a uniform distribution). Each trial involves multiple *runs*. The robot is placed in an initial position before each run, and six suspicious objects are placed in the environment at random locations (uniformly distributed in predefined regions) and with random appearance (uniformly distributed from a set of small objects that the robot can detect). Each run starts when the operator issues a command to the robot and terminates when it successfully completes the task (i.e., reaches the destination and successfully classifies all suspicious objects) or is interrupted by the operator. At the end of each run, the agent updates its inverse trust estimate, compares the current trust estimate to the thresholds, and may adapt its behavior. The environment is then reset to its initial conditions before the next run.

Each time the robot is interrupted, the operator generates a piece of natural language feedback. The feedback comes in five categories: speed feedback (e.g., "*go faster*"), safety feedback (e.g., "*be careful*"), false positive feedback (e.g., "*that wasn't a threat*"), false negative feedback (e.g., "*that was a threat!*"), and missed object feedback (e.g., "*you missed one!*"). Although the operators provide multiple synonymous pieces of feedback for each category (e.g., "*speed up*", "*go faster*"), for this evaluation we treat all synonymous feedback as equivalent. The feedback is never actually provided to the robot (i.e., the robot cannot use it for adaptation). Instead, the feedback is logged and used to evaluate any explanations provided by the robot. For each trial, all feedback generated during that trial is recorded.

Each trial ends in one of two possible outcomes: the robot labels its behavior as trustworthy or as untrustworthy. If the behavior is found to be trustworthy, the data from the trial is discarded. This occurs because no behavior adaptation occurred and therefore no explanations were provided by the robot. However, if the behavior is found to be untrustworthy, case-based behavior adaptation is performed (i.e., adaptation in response to implicit feedback). The robot uses a case base that was learned during a previous study[2]. The robot's current behavior (i.e., the one randomly generated at the

[2] The case base described in [1] labelled *Patrol Random*. It contains cases learned from both the speed-focused and detection-focused operators (25 total cases).

start of the trial) and the behavior returned by case-based behavior adaptation are used to generate an explanation. Eight variant methods for generating explanations are evaluated using Algorithm 1. The variants differ based on the feedback base that is used (i.e., *authored* by an expert or *learned*[3] by the robot) and the method for selecting feedback (i.e., *Highest Count*, *Highest Group Count*, *Mean Group Count*, *Random*). We also use a baseline approach that randomly selects an explanation according to a uniform distribution (labelled as *Baseline* to avoid confusion with the *Random* feedback selection method).

At the end of each trial, the robot's explanation is compared to the explicit feedback generated by the operator during the trial. The following metrics are computed:

- **Most Common**: The percentage of trials where the robot's explanation matched with the most common piece of feedback provided by the operator.
- **Matched One**: The percentage of trials where the robot's explanation matched at least one piece of feedback given by the operator.
- **Mean Rank**: The mean rank of the robot's explanation relative to a list that is ranked by the number of times each piece of feedback occurs during a trial. If the robot's explanation does not appear in the ranked list, it is given a value of the size of the ranked list plus one.

The average from 1000 trials was collected and the process was repeated 25 times (i.e., 25,000 total trials). The robot used a trustworthy threshold of $\tau_T = 5.0$, an untrustworthy threshold of $\tau_U = -5.0$, and the case-based adaptation approach described in [1].

5.3 Results

Figure 1 shows results for the *Most Common* and *Matched One* metrics, and Figure 2 shows the *Mean Rank* results (error bars show 95% confidence intervals). The results using the expert-authored feedback base are combined into a single entry, labelled as *Expert*. This was done for simplicity because the results were identical regardless of the explanation selection method used. The expert did not include redundancy so each relationship only appears once in the expert-authored feedback base. This causes the same explanation to be returned regardless of which explanation selection method is used.

All results that use our explanation approach (i.e., *Highest Count*, *Highest Group Count*, *Average Group Count*, *Random*, and *Expert*) were statistically significant improvements over *Baseline* (using a paired *t*-test with $p < 0.001$). These results provide support for **H2**. *Expert* outperformed the other approaches across all three metrics. The primary benefits of using the expert-authored feedback base are that there are no erroneous or redundant feedback relationships. Compared to the best results when using a learned feedback base (i.e., *Highest Count*, *Highest Group Count*, and *Average Group Count*), *Expert* can better provide an explanation that matches at least

[3] The learned feedback base is identical to the feedback base described in [2] where feedback is given by the operator 100% of the time. It contains feedback from both operators.

one piece of feedback given by the operator (91% of the time vs. 81% of the time), and regularly provides the most common piece of feedback (75% of the time vs. 61% of the time). These results support **H3**. However, the results are promising as they indicate that, while our approach for explanation works best with an expert-authored feedback base, reasonable performance can be achieved using a learned feedback base. Given that both the expert-authored and learned feedback bases resulted in explanations that closely matched feedback from the operator, the results provide evidence that **H1** is supported.

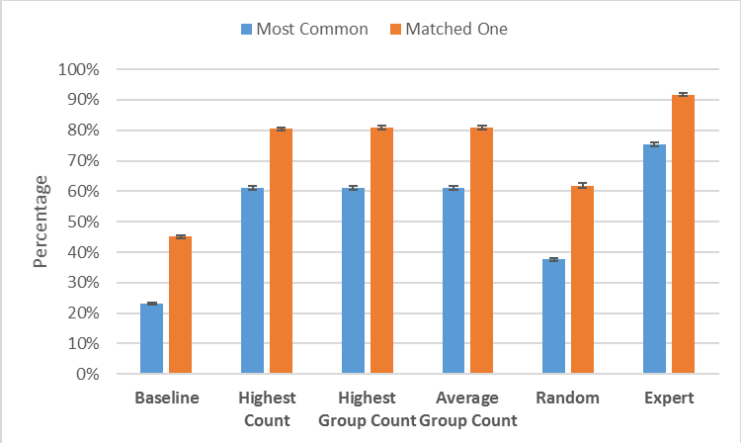

Fig. 1. The percentage of explanations that matched the most common feedback provided by the operator (Most Common) or any feedback provided by the operator (Matched One).

There are no significant differences between the results when using the *Highest Count*, *Highest Group Count*, or *Average Group Count* explanation selection methods. However, all three were significant improvements over *Random* explanation selection. This improvement occurred because some feedback relationships in the feedback base are erroneous. A higher count value in a feedback relationship indicates that the relationship has been observed more often (i.e., less likely to be the result of a single error), so the three methods that use the count value are better able to reduce the influence of erroneous relationships.

The primary reason why none of the approaches were able to achieve ideal performance (i.e., always providing an explanation that matched the most common piece of feedback given by the operator) is because the behavior adaptation process also introduced error. The case base used to perform behavior adaptation was learned (i.e., it may contain erroneous cases) and similarity assessment was performed using only a single evaluated behavior (i.e., $\mathcal{E}_{past}$ contains only a single evaluated pair so limited information was available during case retrieval). However, even with these sources of error our approach was still able to provide reasonable explanations.

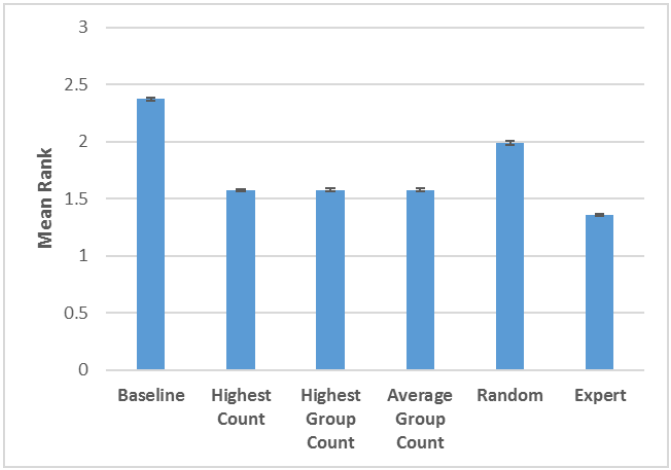

Fig. 2. The mean rank of the robot's explanation relative to a list that is ranked by the number of times each piece of feedback occurs during a trial.

6 Related Work

The Situation awareness-based Agent Transparency (SAT) model aims to improve human-robot teaming by providing situational awareness, reducing user overhead, and allowing the user to appropriately calibrate their trust in the robot [8]. The SAT model is implemented as a user interface that provides three levels of transparency: the robot's status (e.g., current state, goals, plans), the robot's reasoning process, and the robot's projections (e.g., future environment states). The transparency offered by the SAT model is significantly more complex than our approach and is designed for a user that is continuously monitoring the robot (i.e., constantly observing the robot through the interface). Instead, we focus on providing transparency for an operator who is performing their own tasks and may only monitor the robot sporadically.

Explanation in AI systems can be divided into *internal explanations* and *external explanations* [9]. Internal explanations are used by the system as part of its reasoning process. For example, DiscoverHistory [10] identifies discrepancies between observed environment states and expected environment states, and generates explanations for why those discrepancies occurred (e.g., the actions of other agents). These explanations provide the system with information about unobservable parts of the environment and allow it to respond more intelligently to unexpected events. The other category of explanation, external explanations, differs in that they aim to explain the system's reasoning process to a user. While many internal explanations can be used as external explanations, they are not intended to be understandable by a user (e.g., formatting, presentation, amount of content).

Storing concrete problem solving instances gives case-based reasoning an inherent advantage in providing explanations when compared to other learning approaches [11]. In some domains, providing the user with the cases themselves may be a sufficient

explanation (e.g., a help desk system). Cunningham et al. [12] have shown that providing the retrieved case as an explanation improves user satisfaction compared to displaying a rule or giving no explanation. However, using the case as an explanation requires that the user can clearly understand why it is similar to the input problem and why the solution is appropriate [13]. In our system, explanations are a result of two separate case-based reasoning processes: one using the behavior adaptation case base and the other using the feedback base. Although the cases stored in the feedback base are relatively simple triples (i.e., feedback, relationship, and frequency), the behavior adaptation cases require a more complex similarity calculation. The complexity of retrieval, use of multiple interconnected case bases (i.e., the results of behavior adaptation are used as input when generating an explanation), and time constraints of the operator make it unsuitable to directly use the cases as explanations.

Sørmo et al. [13] identify five goals of explanation in CBR: *transparency* (i.e., how the answer was reached), *justification* (i.e., why the answer is good), *relevance* (i.e., why a question is relevant), *conceptualization* (i.e., clarify the meaning of a concept), and *learning* (i.e., teach the user). Our work falls under the transparency category since it is focused on how the robot made a decision. To a lesser extent, our explanations also provide justification by presenting the reason the agent thinks a behavior change was necessary (i.e., it is good because it aligns with the operator's preferences). Our work differs from the traditional use of explanations in CBR in that we are not explaining an answer that is given to a user, but instead explaining an agent's reasoning process. Our explanations are *cognitive explanations* since they explain the reasoning of an intelligent system [14]. Cognitive explanations have also been examined in ambient intelligence systems [15]. Similar to our work, they discuss how a system can explain to the user why a behavior was chosen. However, it differs in that it attempts to explain the reasoning that resulted in an incorrect or unexpected behavior being chosen, whereas we focus on explaining why a behavior was changed.

Issue-based prediction [16], like our own work, stores an explanation as the solution portion of a case. Case-based reasoning and a weak domain model are used to explain which side will win in a legal dispute. Our work uses much simpler explanations and does not require any domain knowledge during reasoning, instead learning knowledge about operator feedback. However, in a legal domain the explanations are much more complex and benefit from domain knowledge. FormuCaseViz [17] is similar to our own work in that the explanations are meant to reduce the cognitive burden on the user. The system visually displays the differences between the target problem and similar cases, helping the user to quickly understand the similarities and differences. This approach differs from our own in that it attempts to explain aspects of the CBR process to the user whereas we focus on explaining the agent's reasoning.

Explanation has been identified as an important feature of recommender systems, with numerous systems implementing explanation capabilities [18]. While many recommender systems explain why they gave a particular recommendation, explanations have also been used to explain why follow-up questions were asked in conversational recommender systems [19]. Muhammad et al. [20] have examined how the explainability of a case can be used to guide retrieval in a case-based recommender. Their approach is based on the idea that a useful case should be both similar and allow

for informative explanations. Our work differs from recommender systems in that the agent is not providing alternatives for the operator to choose from, but is instead justifying a decision that has already been made.

It is not surprising that trust models have been examined in the context of case-based recommender systems [21], given the relationship between providing explanations and trust. Additionally, trust is an important factor in case-based agent collaboration [22] and case provenance [23]. Unlike our work, which focuses on inverse trust, these investigations examine traditional trust (e.g., trust in another agent or trust in the source of a case). Even outside of the CBR literature, most existing work focuses on traditional trust [4]. The two exceptions are the work of Kaniarasu et al. [24] and Saleh et al. [25]. Kaniarasu et al. use negative performance factors (e.g., how often the robot is warned of poor performance) and periodic performance feedback from the operator (e.g., whether the robot is currently performing well) to estimate the operator's trust. This differs from our own work in that their approach requires explicit performance feedback to estimate trust. Saleh et al. instead use a set of expert-authored rules to estimate operator trust. Unlike our inverse trust estimate, which can be used regardless of operator or mission context, their approach requires the rules to be redefined by an expert whenever a change in context occurs.

7 Conclusions

In this paper, we examined how an agent, which controls a simulated robot and performs trust-guided behavior adaptation, can provide explanations when it modifies its behavior. Introducing a layer of transparency should increase the agent's trustworthiness by allowing it to present its operator with the motivation for any behavior changes. Our approach uses two case-based reasoning processes, both of which use cases that are learned while interacting with the operator. The first process, which was the focus of our prior work, involves evaluating the robot's trustworthiness and selecting a new behavior if the robot is behaving in an untrustworthy manner. When the robot adapts its behavior, a second case-based reasoning process is used to generate an explanation for why the change occurred. Given that the human-robot team may be in a time-sensitive situation, we designed our explanations to be simple, concise, and understandable.

Our evaluation involved an operator instructing a robot to patrol a simulated environment, identify suspicious objects, and classify them as threats or harmless. As the robot completed its tasks, it evaluated its trustworthiness and adapted its behavior if it determined it was untrustworthy. Our results indicate that the explanations provided by the robot for its behavior adaptations aligned closely with the explicit feedback provided by the operator. The primary area we wish to address in future work is to validate our results in a user study. While our system was based on the findings of existing research on human-robot trust and transparency, we plan to independently validate those findings using human operators in our robotics environment.

Acknowledgements

Thanks to ONR for sponsoring this research. Thanks also to Michael Drinkwater for his assistance in developing the eBotworks scenarios we used to evaluate our agent, and to the reviewers for their comments.

References

1. Floyd, M.W., Drinkwater, M., & Aha, D.W. (2014). How much do you trust me? Learning a case-based model of inverse trust. *Proceedings of the Twenty-Second International Conference on Case-Based Reasoning* (pp. 125-139). Cork, Ireland: Springer.
2. Floyd, M.W., Drinkwater, M., & Aha, D.W. (2015). Improving trust-guided behavior adaptation using operator feedback. *Proceedings of the Twenty-Third International Conference on Case-Based Reasoning* (pp. 134-148). Frankfurt, Germany: Springer.
3. Dzindolet, M.T., Peterson, S.A., Pomranky, R.A., Pierce, L.G., & Beck, H.P. (2003). The role of trust in automation reliance. *International Journal of Human-Computer Studies*, **58**(6), 697-718.
4. Sabater, J., & Sierra, C. (2005). Review on computational trust and reputation models. *Artificial Intelligence Review*, **24**(1), 33-60.
5. Hancock, P.A., Billings, D.R., Schaefer, K.E., Chen, J.Y., De Visser, E.J., 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.
6. Kaniarasu, P., Steinfeld, A., Desai, M., & Yanco, H.A. (2012). Potential measures for detecting trust changes. *Proceedings of the Seventh International Conference on Human-Robot Interaction* (pp. 241-242). Boston, USA: ACM.
7. Knexus Research Corporation. (2016). *eBotworks*. Retrieved from [http://www.knexusresearch.com/products/ebotworks.php]
8. Chen, J.Y.C., Barnes, M.J., Selkowitz, A.R., Stowers, K., Lakhmani, S.G., & Kasdaglis, N. (2016). Human-autonomy teaming and agent transparency. *Proceedings of the Twenty-First International Conference on Intelligent User Interfaces* (pp. 28-31). Sonoma, USA: ACM.
9. Aamodt, A. (1993). Explanation-driven case-based reasoning. *Proceedings of the First European Workshop on Case-Based Reasoning* (pp. 274-288). Kaiserslautern, Germany: Springer.
10. Molineaux, M., Kuter, U., & Klenk, M. (2012). DiscoverHistory: Understanding the Past in Planning and Execution. *Proceedings of the Eleventh International Conference on Autonomous Agents and Multi-Agent Systems* (pp. 989-996). Valencia, Spain: IFAAMAS.
11. Leake, D.B. (1996). CBR in context: The present and future. In D.B. Leake (Ed.), *Case-Based Reasoning: Experiences, Lessons, and Future Directions*. Menlo Park, USA: AAAI Press/MIT Press.

12. Cunningham, P., Doyle, D., & Loughrey, J. (2003). An evaluation of the usefulness of case-based explanation. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 122-130). Trondheim, Norway: Springer.
13. Sørmo, F., Cassens, J., & Aamodt, A. (2005). Explanation in case-based reasoning—perspectives and goals. *Artificial Intelligence Review,* **24**(2), 109-143.
14. Roth-Berghofer, T. (2004). Explanations and case-based reasoning: Foundational issues. *Proceedings of the Seventh European Conference on Case-Based Reasoning* (pp. 389-403). Madrid, Spain: Springer.
15. Kofod-Petersen, A., & Cassens, J. (2007). Explanations and context in ambient intelligent systems. *Proceedings of the Sixth International and Interdisciplinary Conference on Modeling and Using Context* (pp. 303-316). Roskilde, Denmark: Springer.
16. Brüninghaus, S., & Ashley, K.D. (2003). Combining case-based and model-based reasoning for predicting the outcome of legal cases. *Proceedings of the Fifth International Conference on Case-Based Reasoning* (pp. 65-79). Trondheim, Norway: Springer.
17. Massie, S., Craw, S., & Wiratunga, N. (2004). Visualisation of case-case reasoning for explanation. *Proceedings of the Seventh European Conference on Case-Based Reasoning Workshops* (pp. 135-144). Madrid, Spain.
18. Tintarev, N., & Masthoff, J. (2007). A survey of explanations in recommender systems. *Proceedings of the Twenty-Third International Conference on Data Engineering Workshops* (pp. 801-810). Istanbul, Turkey: IEEE.
19. McSherry, D. (2005). Explanation in recommender systems. *Artificial Intelligence Review*, **24**(2), 179-197.
20. Muhammad, K., Lawlor, A., Rafter, R., & Smyth, B. (2015). Great explanations: Opinionated explanations for recommendations. *Proceedings of the Twenty-Third International Conference on Case-Based Reasoning* (pp. 244-258). Frankfurt am Main, Germany: Springer.
21. Tavakolifard, M., Herrmann, P., & Öztürk, P. (2009). Analogical trust reasoning. *Proceedings of the Third International Conference on Trust Management* (pp. 149-163). West Lafayette, USA: Springer.
22. Briggs, P., & Smyth, B. (2008). Provenance, trust, and sharing in peer-to-peer case-based web search. *Proceedings of the Ninth European Conference on Case-Based Reasoning* (pp. 89-103). Trier, Germany: Springer.
23. Leake, D.B., & Whitehead, M. (2007). Case provenance: The value of remembering case sources. *Proceedings of the Seventh International Conference on Case-Based Reasoning* (pp. 194-208). Belfast, Northern Ireland: Springer.
24. Kaniarasu, P., Steinfeld, A., Desai, M., & Yanco, H.A. (2013). Robot confidence and trust alignment. *Proceedings of the Eighth International Conference on Human-Robot Interaction* (pp. 155-156). Tokyo, Japan: ACM.
25. Saleh, J.A., Karray, F., & Morckos, M. (2012). Modelling of robot attention demand in human-robot interaction using finite fuzzy state automata. *Proceedings of the International Conference on Fuzzy Systems* (pp. 1-8). Brisbane, Australia: IEEE.