Case-Based Behavior Adaptation Using an Inverse Trust Metric

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

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

Abstract

Robots are added to human teams to increase the team's skills or capabilities but in order to get the full benefit the teams must trust the robots. We present an approach that allows a robot to estimate its trustworthiness and adapt its behavior accordingly. Additionally, the robot uses case-based reasoning to store previous behavior adaptations and uses this information to perform future adaptations. In a simulated robotics domain, we compare case-based behavior adaption to behavior adaptation that does not learn and show it significantly reduces the number of behaviors that need to be evaluated before a trustworthy behavior is found.

1 Introduction

Robots can be important members of human teams if they provide capabilities that humans do not have. These could include improved sensory capabilities, communication capabilities, or an ability to operate in environments humans can not (e.g., rough terrain or dangerous situations). Adding these robots might be necessary for the team to meet its objectives and reduce human risk. However, to make full use of the robots the teammates will need to trust them.

This is especially important for robots that operate autonomously or semi-autonomously. In these situations, the human teammates would likely issue commands or delegate tasks to the robot to reduce their workload or more efficiently achieve team goals. A lack of trust in the robot could result in the humans under-utilizing the it, unnecessarily monitoring the robot's actions, or possibly not using it at all.

A robot could be designed so that it operates in a sufficiently trustworthy manner. However, this may be impractical because the measure of trust might be task-dependent, user-dependent, or change over time (Desai et al. 2013). For example, if a robot receives a command from an operator to navigate between two locations in a city, one operator might prefer the task be performed as quickly as possible whereas another might prefer the task be performed as safely as possible (e.g., avoiding bumping into any obstacles). Each operator has distinct preferences that influence how they will trust the robot's behavior, and these preferences may conflict. Even if these user preferences were known in advance, a change in context could also influence what behaviors are trustworthy. An operator who generally prefers a task to be performed quickly would likely change that preference if the robot was transporting hazardous material, whereas an operator who prefers safety would likely change their preferences in an emergency situation.

The ability of a robot to behave in a trustworthy manner regardless of the operator, task, or context requires that it can evaluate its trustworthiness and adapt its behavior accordingly. The robot may not get explicit feedback about its trustworthiness but will instead need to estimate its trustworthiness based on its interactions with its operator. Such an estimate, which we refer to as an *inverse trust estimate*, differs from traditional computational trust metrics in that it measures how much trust another agent has in the robot rather than how much trust the robot has in another agent. In this paper we examine how a robot can estimate the trust an operator has in it, adapt its behavior to become more trustworthy, and learn from previous adaptations so it can perform trustworthy behaviors more quickly.

In the remainder of this paper we will describe our behavior adaptation approach and evaluate it in a simulated robotics domain. Section 2 presents the inverse trust metric and Section 3 describes how it can be used to guide the robot's behavior. In Section 4, we evaluate our case-based behavior adaptation strategy in a simulated robotics domain and report evidence that it can efficiently adapt the robot's behavior to the operator's preferences. Related work is examined in Section 5 followed by a discussion of future work and concluding remarks in Section 6.

2 Inverse Trust Estimation

Traditional trust metrics are used to estimate the trust an agent should have in other agents (Sabater and Sierra 2005). The agent can use past interactions with those agents or feedback from others to determine their trustworthiness. The information this agent uses is likely internal to it and not directly observable by a third party. In a robotics con-

text, the robot will not be able to observe the information a human operator uses to assess their trust in it. Instead, the robot will need to obtain this internal information to estimate operator trust.

One option would be to directly ask the operator, either as it is interacting with the robot (Kaniarasu et al. 2013) or after the task has been completed (Jian, Bisantz, and Drury 2000; Muir 1987), about how trustworthy the robot was behaving. However, this might not be practical in situations that are time-sensitive or where there would be a significant delay between when the robot wishes to evaluate its trustworthiness and the next opportunity to ask the operator (e.g., during a multi-day search and rescue mission). An alternative that does not require direct operator feedback is for the robot to *infer* the trust the operator has in it.

Factors that influence human-robot trust can be grouped into three main categories (Oleson et al. 2011): robot-related factors (e.g., performance, physical attributes), human-related factors (e.g., engagement, workload, self-confidence), and environmental factors (e.g., group composition, culture, task type). Although these factors have all been shown to influence human-robot trust, the strongest indicator of trust is robot performance (Hancock et al. 2011; Carlson et al. 2014). Kaniarasu et al. (2012) have used an inverse trust metric that estimates robot performance based on the number of times the operator warns the robot about its behavior and the number of times the operator takes manual control of the robot. They found this metric aligns closely with the results of trust surveys performed by the operators. However, this metric does not take into account factors of the robot's behavior that increase trust.

The inverse trust metric we use is based on the number of times the robot completes an assigned task, fails to complete a task, or is interrupted while performing a task. An interruption occurs when the operator tells the robot to stop its current autonomous behavior. Our robot infers that any interruptions are a result of the operator being unsatisfied with the robot's performance. Similarly, our robot assumes the operator will be unsatisfied with any failures and satisfied with any completed tasks. Interrupts could also be a result of a change in the operator's goals, or failures could be a result of unachievable tasks, but the robot works under the assumption that those situations occur rarely.

Our control strategy estimates whether trust is increasing, decreasing, or remaining constant over periods of time related to how long the robot has been performing its current behavior. For example, if the robot modifies its behavior at time t_A in an attempt to perform more trustworthy behavior, the trust value will be estimated using information from t_A onward. We evaluate the trust value between times t_A and t_B as follows:

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

where there were n commands issued to the robot between t_A and t_B. If the ith command ($1 \leq i \leq n$) was interrupted or failed it will decrease the trust value and if it was completed successfully it will increase the trust value ($cmd_i \in \{-1, 1\}$). The ith command will also receive a weight ($w_i = [0, 1]$) related to the robot's behavior (e.g., a command that was interrupted because the robot performed a behavior slowly would likely be weighted less than an interruption because the robot injured a human).

3 Trust-Guided Behavior Adaptation Using Case-Based Reasoning

The robot uses the inverse trust estimate to infer if its current behavior is trustworthy, is not trustworthy, or it does not yet know. Two threshold values are used to identify trustworthy and untrustworthy behavior: the trustworthy threshold (τ_T) and the untrustworthy threshold (τ_{UT}). Our robot uses the following tests:

- If the trust value reaches the trustworthy threshold ($Trust_{A-B} \geq \tau_T$), the robot will conclude it has found a sufficiently trustworthy behavior.
- If the trust value falls below the untrustworthy threshold ($Trust_{A-B} \leq \tau_{UT}$), the robot will modify its behavior in an attempt to use a more trustworthy behavior.
- If the trust value is between the two thresholds ($\tau_{UT} < Trust_{A-B} < \tau_T$), the robot will continue to evaluate the operator's trust.

In the situations where the trustworthy threshold has been reached or neither threshold has been reached, the robot will continue to use its current behavior. However, when the untrustworthy threshold has been reached the robot will modify its behavior in an attempt to behave in a more trustworthy manner. The ability of the robot to modify its own behavior is guided by the number of behavioral components that it can modify. These modifiable components could include changing an algorithm used (e.g., switching between two path planning algorithms), changing parameter values it uses, or changing data that is being used (e.g., using a different map of the environment). Each modifiable component i will have a set $\mathcal{C}_i$ of possible values that the component can be selected from.

If the robot has m components of its behavior that can be modified, its current behavior B will be a tuple containing the currently selected value c_i for each modifiable component ($c_i \in \mathcal{C}_i$):

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

When a behavior B was found by the robot to be untrustworthy it is stored as an evaluated pair E that also contains the time t it took the behavior to be labeled as untrustworthy:

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

The time it took for a behavior to reach the untrustworthy threshold is used to compare behaviors that have been found to be untrustworthy. A behavior $B^{'}$ that reaches the untrustworthy threshold sooner than another behavior $B^{''}$ ($t^{'} < t^{''}$) is assumed 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 likely performing some trustworthy behaviors or was not performing untrustworthy behaviors as quickly.

As the robot evaluates behaviors, it stores a set $\mathcal{E}_{past}$ of previously evaluated behaviors ($\mathcal{E}_{past} = \{E_1, E_2, \ldots, E_n\}$). It continues to add to this set until it locates a trustworthy behavior B_{final} (when the trustworthy threshold is reached). The set of evaluated behaviors can be thought of as the search path that resulted in the final solution (the trustworthy behavior). The search path information is potentially useful because if the robot can determine it is on a similar search path that it has previously encountered (similar behaviors being labeled untrustworthy in a similar amount of time) then the robot can identify what final behavior it should attempt. To allow for the reuse of past behavior adaptation information we use case-based reasoning (Richter and Weber 2013).

Each *case* C is composed of a problem and a solution. In our context, the *problem* is the previously evaluated behaviors and the *solution* is the final trustworthy behavior:

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

These cases are stored in a *case base* and represent the robot's knowledge about previous behavior adaptation.

When the robot modifies its behavior it selects new values for one or more of the modifiable components. The new behavior B_{new} is selected as a function of all behaviors that have been previously evaluated for this operator and its case base CB:

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

The $selectBehavior$ function (Algorithm 1) attempts to use previous adaptation experience to guide the current adaptation. The algorithm iterates through each case in the case base and checks to see if that case's final behavior has already been evaluated by the robot. If the behavior has been evaluated, that means the robot has already found the behavior to be untrustworthy so the robot does not try to use it again. The remaining cases have their set of evaluated behaviors ($C_i.\mathcal{E}_{past}$) compared to the robot's current set of evaluated behaviors ($\mathcal{E}_{past}$). The most similar case's final behavior is returned and will be used by the robot. If no such behaviors are found (the final behaviors of all cases have been examined or the case base is empty), the $modifyBehavior$ function is used to select the next behavior to perform. It selects an evaluated behavior E_{max} that took the longest to reach the untrustworthy threshold ($\forall E_i \in \mathcal{E}_{past}(E_{max}.t \geq E_i.t)$) and performs a random walk (without repetition) to find a behavior B_{new} that required the minimum number of changes from $E_{max}.B$ and has not already been evaluated ($\forall E_i \in \mathcal{E}_{past}(B_{new} \neq E_i.B)$). If all possible behaviors have been evaluated and found to be untrustworthy the robot will stop adapting its behavior and use the behavior from E_{max}.

The similarity between two sets of evaluated behaviors (Algorithm 2) is complicated by the fact that the sets may vary in size. The size of the sets depend on the number of previous behaviors that were evaluated by the robot in each set and there is no guarantee that the sets contain identical behaviors. To account for this, the similarity function looks at the overlap between the two sets and ignores behaviors that have been examined in only one of the sets. Each evaluated behavior in the first set has its behavior matched to an evaluated behavior E_{max} in the second set that contains the most similar behavior ($sim(B_A, B_B) = \frac{1}{m}\sum_{i=1}^{m} sim(B_A.c_i, B_B.c_i)$, where the similarity function will depend on the specific type of behavior component). If those behaviors are similar enough, based on a threshold λ, then the similarity of the time components of these evaluated behaviors are included in the similarity calculation. This ensures that only matches between evaluated behaviors that are highly similar (i.e., similar behaviors exist in both sets) are included in the similarity calculation.

Algorithm 1: Selecting a New Behavior

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

```
bestSim ← 0; B_best ← ∅;
foreach C_i ∈ CB do
    if C_i.B_final ∉ E_past then
        sim_i ← sim(E_past, C_i.E_past);
        if sim_i > bestSim then
            bestSim ← sim_i;
            B_best ← C_i.B_final;
if B_best = ∅ then
    B_best ← modifyBehavior(E_past);
return B_best;
```

Algorithm 2: Similarity between sets of evaluated behaviors

Function: *sim*($\mathcal{E}_A$, $\mathcal{E}_B$) ***returns*** sim;

```
totalSim ← 0; num ← 0;
foreach E_i ∈ E_A do
    E_max ← arg max_{E_j ∈ E_B} (sim(E_i.B, E_j.B));
    if sim(E_i.B, E_max.B) > λ then
        totalSim ← totalSim + sim(E_i.t, E_max.t);
        num ← num + 1;
if num = 0 then
    return 0;
return totalSim / num;
```

4 Evaluation

In this section, we describe an evaluation for our claim that the case-based reasoning approach is able to adapt to and perform trustworthy behaviors more quickly that a random walk approach. We conducted this study in a simulated environment with a simulated robot and operator.

4.1 eBotWorks Simulator

Our evaluation uses the eBotworks simulation environment (Knexus Research Corporation 2013). eBotworks is a multi-agent simulation engine and testbed that allows for multi-modal command and control of unmanned systems. It allows for autonomous agents to control simulated robotic vehicles while interacting with human operators, and for the autonomous behavior to be observed and evaluated.

We use a simulated urban environment (Figure 1) containing landmarks (e.g., roads) and objects (e.g., houses, humans, traffic cones, vehicles, road barriers). The robot is a wheeled unmanned ground vehicle (UGV) and uses eBotwork's built-in natural language processing (for interpreting user commands), locomotion, and path-planning modules. The actions performed by a robot in eBotworks are non-deterministic (e.g., the robot cannot anticipate its exact position after moving).

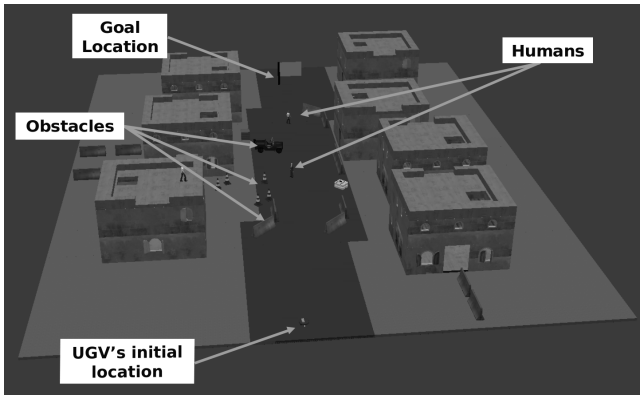

Figure 1: Simulated urban environment in eBotworks

4.2 Simulated Operator

In this study we will use a simulated operator to issue commands to the robot. The simulated operator assesses its trust in the robot using three factors of the robot's performance:

- **Task duration**: The simulated operator has an expectation about the amount of time that the task will take to complete ($t_{complete}$). If the robot does not complete the task within that time, the operator may, with probability p_α, interrupt the robot and issue another command.
- **Task completion**: If the operator determines that the robot has failed to complete the task (e.g., the UGV is stuck), it will interrupt.
- **Safety**: The operator may interrupt the robot, with probability p_γ, if the robot collides with any obstacles along the route.

4.3 Movement Scenario

The simple task the robot is required to perform involves moving between two locations in the environment. At the start of each run, the robot will be placed in the environment and the simulated operator will issue a command for the robot to move to a goal location. Based on the robot's performance (task duration, task completion, and safety), the operator will allow the robot to complete the task or interrupt it. When the robot completes a task, fails to complete it, or is interrupted, the scenario will be reset by placing the robot back at the start location and the operator will issue another command.

We us three simulated operators:

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

Each of the three simulated operators will control the robot for 500 experimental trials, with each trial terminating when the robot determines it has found a trustworthy behavior. For the case-based approach, a case is added to the case base at the end of any trial where the robot performs at least one random walk adaptation of its behavior. When no random walk adaptations are performed, the robot was able to find a trustworthy behavior using the cases in its case base so there is no need to add another case.

The robot has two modifiable behavior components: *speed* (meters per second) and *obstacle padding* (meters). Speed relates to how fast the robot can move and obstacle padding relates to the distance the robot will attempt to keep from obstacles during movement. The set of possible values for each modifiable component ($\mathcal{C}_{speed}$ and $\mathcal{C}_{padding}$) are determined from minimum and maximum values with fixed increments.

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

We test our robot using a trustworthy threshold of $\tau_T = 5.0$ and an untrustworthy threshold of $\tau_{UT} = -5.0$. When calculating the similarity between sets of evaluated behaviors the robot uses a similarity threshold of $\lambda = 0.95$ (behaviors must be 95% similar to be matched).

4.4 Results

We found that both the case-based behavior adaptation and the random walk behavior adaptation strategies resulted in similar trustworthy behaviors for each simulated operator. For the speed-focused operator, the trustworthy behaviors had higher speeds regardless of padding ($3.5 \leq speed \leq 10.0$, $0.1 \leq padding \leq 1.9$). The safety-focused operator had higher padding regardless of speed ($0.5 \leq speed \leq 10.0$, $0.4 \leq padding \leq 1.9$). Finally, the balanced operator had higher speed and higher padding ($3.5 \leq speed \leq 10.0$, $0.4 \leq padding \leq 1.9$). In addition to having similar value

ranges, there were no statistically significant differences between the distributions of those values for the two strategies.

The difference between the two behavior adaption approaches was related to the number of behaviors that needed to be evaluated before a trustworthy behavior was found. Table 1 shows the mean number of evaluated behaviors (and 95% confidence interval) when interacting with each operator type (over 500 trials for each operator). In addition to being controlled by only a single operator, we also examined a condition in which the operator is selected at random with equal probability. This represents a more realistic scenario where the robot will be required to interact with a variety of operators without any knowledge about which operator will control it.

Table 1: The mean number of behaviors evaluated

Operator	Random Walk	Case-based
Speed-focused	20.3 ($\pm$3.4)	1.6 ($\pm$0.2)
Safety-focused	2.8 ($\pm$0.3)	1.3 ($\pm$0.1)
Balanced	27.0 ($\pm$3.8)	1.8 ($\pm$0.2)
Random	14.6 ($\pm$2.9)	1.6 ($\pm$0.1)

The case-based approach required significantly fewer behaviors to be evaluated in all four experiments (using a paired t-test with $p < 0.01$). This is because the case-based approach was able to learn from previous adaptations and use that information to quickly find trustworthy behaviors. At the beginning, when the robot's case base is empty, the case-based approach is required to perform adaptation that is similar to the random walk approach. As the case base size grows, the number of times random walk adaptation is required decreases until the agent generally only performs a single case-based behavior adaptation before finding a trustworthy behavior. Even when the case base contains cases from all three simulated operators, the case-based approach can quickly differentiate between the users and select a trustworthy behavior. The number of adaptations required for the safety-focused operator was lower than for the other operators because a higher percentage of behaviors are considered trustworthy. The robot, which started the experiments for each operator with an empty case base, collected 24 cases when interacting with the speed-focused operator, 18 cases when interacting with the safety-focused operator, 33 cases when interacting with the balanced operator, and 33 cases when interacting with a random operator.

The primary limitation of the case-based approach is that it relies on the random walk search when it does not have any suitable cases to use. Although the mean number of behaviors evaluated by the case-based approach is low, the situations where random walk is used (and a new case is created) require an above-average number of behaviors to be evaluated (closer to the mean number of behaviors evaluated when only random walk is used). The case-based approach uses random walk infrequently, so there is not a large impact on the mean number of behaviors evaluated over 500 trials, but this would be an important concern as the problem scales to use more complex behaviors with more modifiable components. Two primary solutions exist to improve performance in more complex domains: improved search and seeding of the case base. Random walk search was used because it requires no explicit knowledge about the domain or the task. However, a more intelligent search that could identify relations between interruptions and modifiable components (e.g., an interruption when the robot is very close to objects requires a change to the padding value) would likely improve adaptation time. Since a higher number of behaviors need to be evaluated when new cases are created, if a set of initial cases were provided to the robot it would be able to decrease the number of random walk adaptations (or adaptations requiring a different search technique) it would need to perform.

5 Related Work

In addition to Kaniarasu et al. (2012), Saleh et al. (2012) have also proposed a measure of inverse trust and use a set of expert-authored rules to measure trust. Unlike our own work, while these approaches measure trust, they do not use this information to adapt behavior. Shapiro and Shachter (2002) discuss the need for an agent to act in the best interests of a user even if that requires sub-optimal performance. Their work examines identifying factors that influence the user's utility function and updating the agent's reward function accordingly. This is similar to our own work in that behavior is modified to align with a user's preference, but our robot is not given an explicit model of the user's reasoning process.

Conversational recommender systems (McGinty and Smyth 2003) iteratively improve recommendations to a user by tailoring the recommendations to the user's preferences. As more information is obtained through dialogs with a user, these systems refine their model of that user. Similarly, learning interface agents observe a user performing a task (e.g., sorting e-mail (Maes and Kozierok 1993) or schedule management (Horvitz 1999)) and learn the user's preferences. Both conversational recommender systems and learning interface agents are designed to learn preferences for a single task whereas our behavior adaptation requires no prior knowledge about what tasks will be performed.

Our work also has similarities to other areas of learning during human-robot interactions. When a robot learns from a human, it is often beneficial for the robot to understand the environment from the perspective of the human. Breazeal et al. (2009) have examined how a robot can learn from a cooperative human teacher by mapping its sensory inputs to how it estimates the human is viewing the environment. This allows the robot to learn from the viewpoint of the teacher and possibly discover information it would not have noticed from its own viewpoint. This is similar to preference-based planning systems that learn a user's preferences for plan generation (Li, Kambhampati, and Yoon 2009). Like our own work, these systems involve inferring information about the reasoning of a human. However, they differ in that they involve observing a teacher demonstrate a specific task and learning from those demonstrations.

6 Conclusions

In this paper we have presented an inverse trust measure to estimate an operator's trust in a robot's behavior and to adapt its behavior to increase an operator's trust. The robot also learns from previous behavior adaptations using case-based reasoning. Each time it successfully finds a trustworthy behavior, it records that behavior as well as the untrustworthy behaviors that it evaluated.

We evaluated our trust-guided behavior adaptation algorithm in a simulated robotics environment by comparing it to a behavior adaptation algorithm that does not learn from previous adaptations. Both approaches converge to trustworthy behaviors for each type of operator (speed-focused, safety-focused and balanced) but the case-based algorithm requires significantly fewer behaviors to be evaluated before a trustworthy behavior is found. This is advantageous because the chances that the operator will stop using the robot increases the longer the robot is behaving in an untrustworthy manner.

Although we have shown the benefits of trust-guided behavior adaptation, several areas of future work exist. We have only evaluated the behavior in a simple movement scenario but will soon test it on increasingly complex tasks where the robot has more behavior components that it can modify (e.g., scouting for hazardous devices in an urban environment). In longer scenarios it may be important to not only consider undertrust, as we have done in this work, but also overtrust. In situations of overtrust, the operator may trust the robot too much and allow the robot to behave autonomously even when it is performing poorly. We also plan to include other trust factors in the inverse trust estimate and add mechanisms that promote transparency between the robot and operator. More generally, adding an ability for the robot to reason about its own goals and the goals of the operator would allow the robot to verify it is trying to achieve the same goals as the operator and identify any unexpected goal changes (e.g., such as when a threat occurs).

Acknowledgments

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

References

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

Carlson, M. S.; Desai, M.; Drury, J. L.; Kwak, H.; and Yanco, H. A. 2014. Identifying factors that influence trust in automated cars and medical diagnosis systems. In *AAAI Symposium on The Intersection of Robust Intelligence and Trust in Autonomous Systems*, 20–27.

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

Hancock, P. A.; Billings, D. R.; Schaefer, K. E.; Chen, J. Y.; De Visser, E. J.; and Parasuraman, R. 2011. A meta-analysis of factors affecting trust in human-robot interaction. *Human Factors: The Journal of the Human Factors and Ergonomics Society* 53(5):517–527.

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

Jian, J.-Y.; Bisantz, A. M.; and Drury, C. G. 2000. Foundations for an empirically determined scale of trust in automated systems. *International Journal of Cognitive Ergonomics* 4(1):53–71.

Kaniarasu, P.; Steinfeld, A.; Desai, M.; and Yanco, H. A. 2012. Potential measures for detecting trust changes. In *7th International Conference on Human-Robot Interaction*, 241–242.

Kaniarasu, P.; Steinfeld, A.; Desai, M.; and Yanco, H. A. 2013. Robot confidence and trust alignment. In *8th International Conference on Human-Robot Interaction*, 155–156.

Knexus Research Corporation. 2013. eBotworks. http://www.knexusresearch.com/products/ebotworks.php. [Online; accessed April 9, 2014].

Li, N.; Kambhampati, S.; and Yoon, S. W. 2009. Learning probabilistic hierarchical task networks to capture user preferences. In *21st International Joint Conference on Artificial Intelligence*, 1754–1759.

Maes, P., and Kozierok, R. 1993. Learning interface agents. In *11th National Conference on Artificial Intelligence*, 459–465.

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

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

Oleson, K. E.; Billings, D. R.; Kocsis, V.; Chen, J. Y.; and Hancock, P. A. 2011. Antecedents of trust in human-robot collaborations. In *1st International Multi-Disciplinary Conference on Cognitive Methods in Situation Awareness and Decision Support*, 175–178.

Richter, M. M., and Weber, R. O. 2013. *Case-Based Reasoning - A Textbook*. Springer.

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

Saleh, J. A.; Karray, F.; and Morckos, M. 2012. Modelling of robot attention demand in human-robot interaction using finite fuzzy state automata. In *International Conference on Fuzzy Systems*, 1–8.

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