Modeling Decision Making on the Use of Automation

Junya Morita (j-morita@jaist.ac.jp)
School of Knowledge Science, Japan Advanced Institute of Science and Technology, Japan

Kazuhisa Miwa (miwa@is.nagoya-u.ac.jp)
Akihiro Maehigashi (mhigashi@cog.human.nagoya-u.ac.jp), Hitoshi Terai (terai@is.nagoya-u.ac.jp)
Graduate School of Information Science, Nagoya University, Japan

Kazuaki Kojima (koj@aoni.waseda.jp)
Faculty of Human Sciences, Waseda University, Japan

Frank E. Ritter (frank.ritter@psu.edu)
College of Information Sciences and Technology, Penn State, USA

Abstract

This paper presents a cognitive model that simulates reliance on automation using a line-tracing task similar to driving where an operator has to track a moving line with a circle by pressing keys on a keyboard (manual control) or rely on automation (auto control). An operator can switch between auto and manual control during the task. The success probabilities of each control mode were systematically varied. An ACT-R model to perform this task was constructed by representing reliance on the automation as production. The model performs this task through productions that manage the perceptual/motor modules. The utility values of these productions are updated based on the rewards in every screen update. We also introduce a meta-level monitoring the internal state of the model. A preliminary run of this model simulated the overall trends of the behavioral data, suggesting some validity of the assumptions made in our model.

Keywords: Automation; ACT-R; Trust.

Introduction

We unconsciously use automation systems like e-mail spam filters, spell checkers, and electronic toll collection (ETC) systems. These systems save time and help us lead more efficient lives. We, however, sometimes face difficult choices about whether to use these systems (Bainbridge, 1983). It has also been pointed out that human decision making using such systems is not always optimal. For example, Parasuraman and Riley (1997) stated that there are two types of maladaptive choices of using automation: misuse, the over-reliance of automation, and disuse, the underutilization of automation. Some studies indicated that human users have an automation bias towards misuse of automation systems (Bahner, Huper, & Manzey, 2008; Singh, Molloy, & Parasuraman, 1997; Skita, Moiser, & Burdick, 2000). On the other hand, other research indicated that human users have a manual bias, a bias towards disuse of automation systems consistent with a need for control (Beck, McKinney, Dzindolet, & Pierce, 2009; Dzindolet, Peterson, Pomranky, Pierce, & Beck, 2003; Dzindolet, Pierce, Beck, & Dawe, 2002).

Vries, Midden, and Bouwhuis (2003) experimentally revealed that the reliance of automation is influenced by both the Capability of Manual control (Cm) and the Capability of Auto controls (Ca). To explain these effects, Gao and Lee (2006) proposed the Extended Decision Field Theory (EDFT model; Figure 1). The model constructs belief of Ca and Cm (Bca, Bcm) based on partially displayed these values. From the belief values, trust (T) and self-confidence (SC) are constructed. Preference of automation (P) is determined by subtracting T from SC. If P exceeds an upper threshold (θ), then the model turns the current control mode to auto. If P falls below a lower threshold ($-\theta$), then the model turns the current control mode to manual. In every cycle, values of Bca, Bcm, T, and SC are updated by differential equations. Although this model clearly explains the reliance on automation in dynamic situations, the model does not have any knowledge about tasks. It cannot interact with a task environment, and it provides no human performance predictions.

This report describes a cognitive process model that interacts with a specific task environment where sequential decision-making is made. Especially, we extend our previous model (Morita et al., in-press) to improve its motor control. To do this, we use ACT-R, a unified theory of cognition (Anderson, 2007). The following subsection briefly shows features of this architecture relating to our model.

ACT-R

One of the most important assumptions of ACT-R is modularity of cognition. ACT-R is composed of several independent modules: goal, production, declarative, perceptual, and motor (Anderson, 2007). A goal module holds the current task goal and other task related information. A production module and a declarative module hold procedural and declarative knowledge respectively. Perceptual modules include a vision and an audio module, which take information from an external environment. A motor module manipulates devices like a keyboard or a mouse in an external environment. Modules other than the production module have buffers to hold temporarily information called a chunk. A production module integrates the other modules by production rules, which consists of a condition/action pair that is used in sequence with other productions to perform a task. Conditions and actions in production rules are specified with buffer contents of each module.

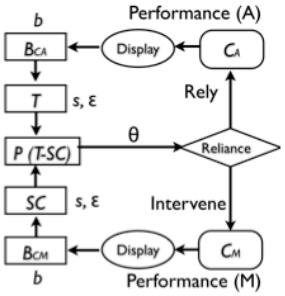

Figure 1: The process of Extended Decision Field Theory (Gao & Lee, 2006), modified and reproduced from the original figure.

ACT-R also includes sub-symbolic cognitive processes. If several rules match to buffer conditions, conflict resolution of production rules is made based on utility values assigned to production rules. The learning of utility is controlled by equation 1:

$$U_i(n) = U_i(n-1) + \alpha[R_i(n) - U_i(n-1)] \quad (1)$$

α is the learning rate; $R_i(n)$ is the reward value given to production i at time n. The learning occurs when a reward is triggered, and all productions that have fired since the last reward are updated. This learning is essentially same as the process presented in the EDFT model using a basic reinforcement learning method.

So far, ACT-R has been used to model many fields of human-machine interaction including driving (Salvucci, 2006), teleoperation (Ritter, Kukreja, & Amant, 2007), water flow control (Lebiere, Gonzalez, & Warwick, 2009), and air-traffic control tasks (Taatgen, 2005). The utility update learning of this architecture also has been applied to strategy selection (Lovett & Anderson, 1996). Therefore, we consider that this architecture is suitable to construct a model of decision making on the use of automation. Through this modeling, we try to find behavioral constraints in the model of decision making on the use of automation.

The Task

To manipulate auto and manual performance in a dynamic situation, and to understand how users change to use automation, we developed a simple tracking task, similar to driving. We call this task the *line-tracing task*. Figure 2 shows the screenshot of the task environment. This environment was developed in Java.

This task requires participants to control the horizontal position of the vehicle (red circle) to follow the black line that scrolls down at 24 pixels per second. The screen is updated every 40 ms. If the vehicle is not on the line, a warning is presented outside of the window. The line is drawn by randomly combining 48 pixels height line patterns of varied angles (30, 45, 90, 135, and 150 degrees).

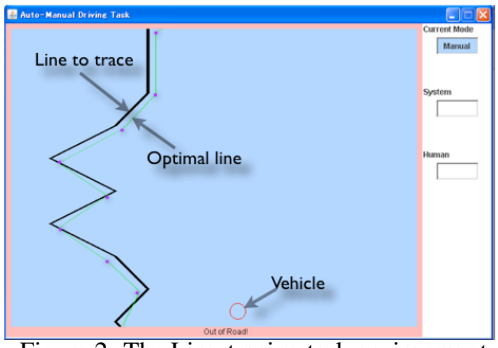

Figure 2: The Line-tracing task environment

The vehicle is controlled by commands of "left", "straight", or "right". If the vehicle receives a left command, the vehicle moves 1 pixel left from the original position. The command is sampled at 48 Hz[1]. Therefore, maximally, the vehicle can move 2 pixels per one pixel scroll of the line.

A participant can chose manual or auto controls to send commands. In the manual control, participants use left and right arrow keys to send commands. If a participant's finger is put on a right arrow key, the vehicle keeps receiving a right command at every 20 ms until this key is released. In the auto control, participants monitor that the auto control moves the vehicle. The auto control tries to follow an optimal line presented as the green line in Figure 2. An optimal line is the shortest line to pass "goals" located on each corner. Figure 2 shows goals as blue dots. If the center of the vehicle is off the optimal line, the auto control system sends a command to correct the vehicle position. In our experiment presented in the next section, the optimal line and goals are not visible to participants.

In both control modes, commands are not always successfully sent to the vehicle. Failures occur at specified rates. In this study, Ca and Cm specify these rates. If Ca or Cm is low, the vehicle controlled by the corresponding mode is lagged, and it becomes hard to follow the line. To conduct the task successfully, participants need to select a suitable mode in each situation. The participants freely change between modes by pressing the spacebar.

Experiments

Before describing the model, we summarize here two experiments that examined the use of automation in this task (more details are reported in Maehigashi et. al., in press) .

In experiment 1, the baseline performance of the manual and the auto controls were examined. The participants were required to control the vehicle using the manual control mode. There were five conditions where Cm values were manipulated from 30% to 70%. Every participant experienced all the five conditions in random order. Each condition lasted 40 s. The performance of the experiment was compared to

[1]If a key-press event is detected, a flag of sending commands is on. This flag is off when a key-release event is detected. Therefore, the manipulation of the command rate is not influenced by a key-repeat rate setting in an operating system.

the auto-mode performance measured in the corresponding Ca conditions. The results confirmed that the manual performance is lower than the auto performance in each condition.

Experiment 2 specified the ratio of automation use during the task. The participants conducted the task with the auto and manual control modes. They could freely change the mode during the task. Combining five levels of Ca and Cm values, 25 experimental conditions were prepared. Every participant experienced all 25 conditions. As in experiment 1, each condition lasted 40 seconds. The results of experiment 2 indicate that participants could adaptively select a suitable mode in a given situation.

Model

Simulated task environment

The model presented here extends our previous model (Morita et al., in-press) to become closer to the actual interaction with the task environment. The model interacts with a simulated task environment developed in the ACT-R graphical user interface that is part of ACT-R 6 (Anderson, 2007). The simulated environment is same as the original environment in the keyboard layout, the screen update rates, the line scrolling speed, the vehicle size, the line width, and the screen size. The auto control mode is also implemented with Common Lisp in the simulated task environment. However, unlike the original environment, visible goal positions are set at each corner to allow the model to perceive the path.

Process of the model

The model uses the production, goal, vision and motor modules of ACT-R 6. Eleven production rules are included in the model. These rules consist of a basic perception-action cycle. Figure 3 indicates this cycle, presenting the rules in boxes. The cycle consists of a perceptual (the top part of the figure) and motor process (the bottom part of the figure) similar to previous driving models in ACT-R (Salvucci, 2006; Ritter et al., 2007).

In the perceptual process, the model picks visual information from a visual location buffer that holds location information of objects in the environment. The *FindVehicle* rule finds the horizontal position of the vehicle, and places it into the goal buffer. The *FindGoal* rule finds the horizontal position of the nearest goal position, and placed it into the goal buffer. The position information in the goal buffer is used in the subsequent motor process. After the motor process, information in the goal buffer is cleared to begin the next cycle.

The subsequent motor process depends on the current mode. In each mode, there is a rule to switch the current mode (*ToAuto* / *ToManual*). These mode-switching rules send a command to release currently pressed keys to the motor module. After finishing the key-release, the *PressSpace* rule sends a motor command pressing the spacebar.

The mode swiching rules compete with other rules in each condition. In the auto mode, the *ToManual* rule conflicts with the *KeepA* rule that just clears the goal buffer. In the manual mode, the *ToAuto* rule competes with the *KeepM*, *ToLeft*, *ToRight*, *LtoS*, and *RtoS* rules. These five rules have different conditions specifying the vehicle and the goal positions, and current move-commands (left, right, straight). The action clauses of the *ToLeft*, *ToRight*, *LtoS*, *RtoS* rules send a command to hold or release a key to the motor module[2]. The *KeepM* rule does not have any action clauses relating the motor module. This rule just clears the goal buffer.

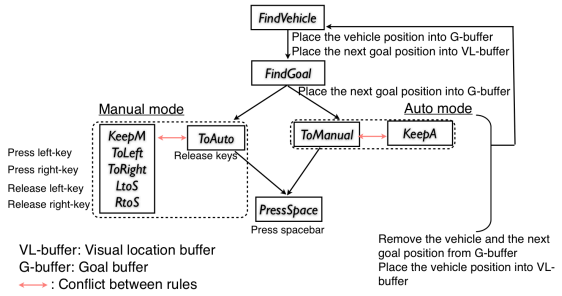

Figure 3: The basic cycle of the model.

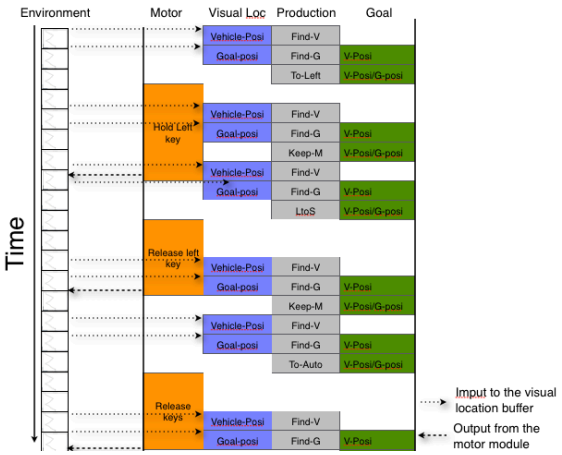

Figure 4: Time flow diagram.

Figure 4 presents a time flow diagram showing the relations between the screen updates of the environment and the model cycles. The environment regularly updates the screen every 40 ms. Individual rule firings take 50 ms, but the cycle of the model is not regulated. There are delays in the visual and motor processes. The process of the visual location module itself has no delay. However, encoding the location into the goal buffer lags 10 ms. from the actual environment. The delay of the motor control is larger than that of the perceptual module. The ACT-R motor module needs preparation and execution time, which depends on the status of the motor module. These delays disadvantage manual control compared to automatic control.

[2]The default ACT-R implementation does not include key-press and key-release functions. We used a customised module in which the time parameter of key-punch is used.

Learning and mode switching

The model adaptively learns to use a suitable mode in a given situation. We used the default reinforcement-learning algorithm in ACT-R 6 presented in section 1. The model receives rewards in every screen update. When the vehicle is off the line, rules used in the previous screen receive no reward ($Ri(n) = 0$). Otherwise, rules used in the previous screen receive positive rewards ($Ri(n) = 10$). This trigger corresponds to the warning in the actual task (Figure 2).

This study uses a small learning rate ($\alpha = .01$) compared to the default setting ($\alpha = .2$) because the rewards are frequently triggered during the task. The noise parameter added to the utility values (egs) is also set to 1. The initial utility values of the mode switching rules (U_{ToAuto} and $U_{ToManual}$) are set to 5, and the initial utility values of other rules are set to 10. This setting corresponds to the cost of mode switching. The initial utility values will not change unless the vehicle moves off the line because positive rewards and the initial values of utility are the same.

In ACT-R, strategy selection is often modeled by conflict resolutions based on utility values of rules (Lovett & Anderson, 1996). According to this paradigm, the mode-switching rules fire when its utility values exceed a utility value of a competing rule. However, our task has differences from tasks used in the previous studies. As Figure 4 indicates the motor actions have delays. There are time-gaps between rule firing and manual control execution. These gaps make rewarding difficult because the next cycle begins before the motor action completes. In addition, the structures of conflict are not the same between the manual and the auto control modes. The *ToAuto* rule conflicts with five rules in the manual mode. On the other hand, the *ToManual* rule conflicts with only the *KeepM* rule in the auto mode.

To solve this problem, we assumed meta-level decision making in addition to the standard conflict resolution. The following code indicates the conditions of the *ToAuto* rule.

```
=goal>
    isa move-vehicle
    - vehicle-loc nil
    - goal-loc nil
    - previous-rule to.auto
    current-mode manual
    !eval! (<= *self-conf* *trust*)
```

The last line is an !eval! condition that has two global variables, *trust* and *self-conf*[3]. The *ToManual* rule also has a similar condition, "!eval! (>= *self-conf* *trust*)." The Lisp function outside of the ACT-R model sets the utility values of the *KeepA* and *KeepM* rules into *trust* and *self-conf* respectively. By putting these conditions, *ToManual* fires only when *self-conf* exceeds *trust*, and the utility values of the *ToManual* rule exceeds that of the *KeepA* rule. Similarly, the *ToAuto* rule fires only when *trust* exceeds *self-conf*, and the utility values of the *ToAutol* rule exceeds that of the competing production rules.

[3]ACT-R architecture provide the !eval! condition to allow the modeler to add arbitrary conditions to a production rule.

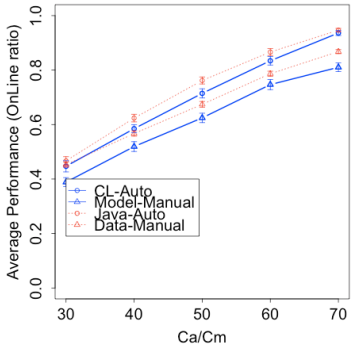

Figure 5: Performance of the model and the data in the baseline simulation. Error bars represent the standard error of means (SEM).

Simulations

In this paper, we present two simulation experiments that simulate the experiments presented in section 3.

Baseline simulation

First, we conducted a simulation of experiment 1 to confirm the correspondence of base performance of the auto and manual modes.

Method In experiment 1, the participants could not use the auto control mode (Data-Manual: $n = 65$). Similarly, we ran the model with the initial control mode as the manual, and removed the *ToAuto* rule from the model (Model-Manual: $n = 100$). We also compared baseline auto performance between the original environment (Java-Auto: $n = 65$) and the simulated environment (CL-Auto: $n = 100$).

Results Figure 5 indicates the performances of the four conditions in each Ca/Cm level, showing the ratio of time that the vehicle is on the line. From this figure, it can be observed that the performance of the all four lines increases with higher Ca/Cm levels, consisting with the manipulations of capability. In addition, we can confirm that the auto controls are better than the manual controls in both the experiment data and the simulation. This result indicates the manual disadvantages in this task. Although the performance of the model is relatively lower than that of the data, the correlations between the experiment and the simulation are high [Auto: $r^2 = .994$, $p < .01$. Manual: $r^2 = .997$, $p < .01$].

Simulation with two modes

This simulation is conducted to simulate experiment 2 that specifies the automation use ratio.

Method In experiment 2, the participant ($n = 35$) could use the auto control mode. They conducted the task in 25 conditions where Ca and Cm levels were manipulated (5 levels of Ca ranging from 30% to 70% v.s. 5 levels of Cm ranging from 30% to 70%). Similarly, the model conducted the task choosing two modes of control in the 25 conditions ($n = 50$).

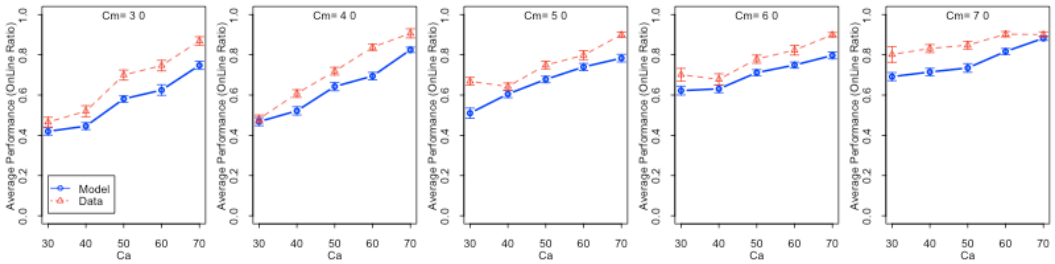

Figure 6: Performance of the simulation 2. Error bars represent the standard error of means (SEM).

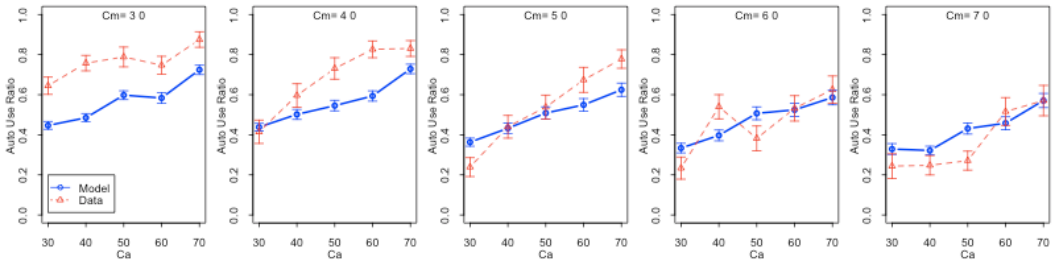

Figure 7: Auto use ratio of the simulation 2. Error bars represent the standard error of means (SEM).

In each condition, the model conducted the task for 40 seconds. The initial mode was randomly set.

Results Figure 6 presents the performance of the model and the experimental data. Each of the five graphs indicates the performance in each Cm level, and the horizontal axis of the each graph indicates Ca levels. The figure indicates an increase of the performance of the model and the experiment data with higher Ca and Cm levels. The correlations of the model and the data are high [$r^2 = .875$, $p < .01$] although some differences between the model and the data can be observed. For example, the model is lower than the data in combination of the low Cm and the high Ca levels (e.g., Cm = 30/Ca = 70). Similarly, the performance of the model fell below that of the data in combination of the high Cm and the low Ca level (e.g., Cm = 70/Ca = 30). These differences suggest some difference in automation use between the model and the experiment data.

Figure 7 indicates the auto use ratio in each Ca and Cm level, which represents how long the auto mode is used during the task. Comparison of the five graphs reveals decreases of auto use ratio with increases of the Cm level. We can also see an increase of the auto use ratio with increases of the Cm level from each graph. The model shares these tendencies with the data, and we obtained a significant correlation between the data and the model [$r^2 = .784$, $p < .01$]. These results suggest that the adaptive learning on the mode selection was made in both the experiment and the simulation. The model is, however, less adaptive compared to the data. The auto use ratio of the model is lower in the low Cm levels such as $Cm = 30$, and the model's lines is flatter than the data's line in $Cm = 50$.

Conclusion

This paper described an ACT-R model that simulates decision making about the use of automation. The results of the simulations show overall correspondence with the experimental data, suggesting some validity of the assumptions made in our model.

We consider that this study is characterized as the connection of the cognitive process model (ACT-R) to an abstract model (EDFT). Figure 8 summarizes the process of our model in the framework of the EDFT model. Like the EDFT model, the utility module of ACT-R represents Belief and Trust in automation systems. However, unlike the previous model, our model has knowledge to execute a task and simulates performing the task. Our model also does not receive Ca/Cm directly. Randomized course conditions influence the performance (success/failure) of the task. Moreover, complex perceptual/motor factors are involved in the manual mode performance. Therefore, the reliance of the automation interacts with the performance of the task in our model. As Bainbridge (1983) implied, to understanding decision making about the use of automation, one needs to consider monitoring the performance of auto and manual performance. This study is a first step to include performance factors into modeling use of automation.

Our model is also different from previous models of strategy selection in ACT-R. Unlike the previous studies, our task requires and uses a complex perceptual/motor process. In such a situation, rewarding behavior is not easy problem. We introduced a function to monitor self-confidence and trust to solve this problem. We consider that the conditions comparing utility values represents a type of meta-cognitive decision making, monitoring the internal states of the model. This monitoring process is not straightforward, but it is influenced by noise (egs). We need to be careful about assuming this

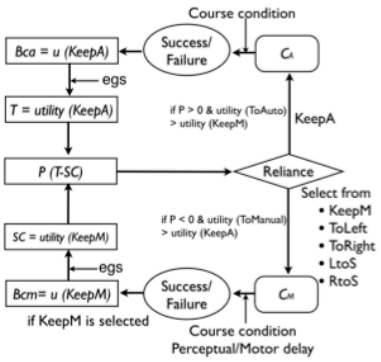

Figure 8: Correspondence with the EDFT model.

process, but we could not simulate overall trends of the experiment without this assumption. This suggests the default sub-symbolic computation is not enough for explaining the use of this automation in this task.

However, using an !eval! condition is not supported by ACT-R theory. ACT-R is designed as a unified cognitive theory that combines sub models from various fields (Anderson, 2007). It is difficult to combine sub models using !eval! conditions. Therefore, we need to consider other methods of modeling this mechanism, which can contribute the development of a unified theory of cognition, particulary metacognition.

There are several other limitations of this study. Considering the result in Figure 7, we need to explore mechanisms that lead to more adaptive learning. We also need to conduct analysis on more detailed behavior such as the frequency or timing of mode switching during the task. The task used in this study is also relatively simple and artificial. Connecting to more complex tasks is required. The factors involved in automation use are broad. In future work, we will explore other factors of automation use, such as cognitive load, emotion, mental models, and individual differences.

Acknowledgments

This study is supported through the CREST program from the Japan Science and Technology Agency (JST), and DTRA (HDTRA 09-1-0054). The authors wish to acknowledge the helpful comments and suggestions made by Dan Bothell, William Kennedy, Christian Lebiere, Walt Mankowski, Ling Rothlock, Matthew Walsh, and John Yen.

References

Anderson, J. R. (2007). *How can the human mind occur in the physical universe?* New York: Oxford University Press.

Bahner, J. E., Huper, A. D., & Manzey, D. (2008). Misuse of automated decision aids: Complacency, automation bias and the impact of training experience. *International Journal of Human-Computer Studies*, *66*(9), 688-699.

Bainbridge, L. (1983). Ironies of automation. *Automatica*, *19*(6), 775–779.

Beck, H. P., McKinney, J. B., Dzindolet, M. T., & Pierce, L. G. (2009). Effects of human-machine competition on intent errors in a target detection task. *Human Factors*, *51*(4), 477-486.

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.

Dzindolet, M. T., Pierce, L. G., Beck, H. P., & Dawe, L. A. (2002). The perceived utility of human and automated aids in a visual detection task. *Human Factors*, *44*(1), 79–94.

Gao, J., & Lee, J. D. (2006). Extending the decision field theory to model operators' reliance on automation in supervisory control situations. *IEEE Transactions on Systems Man and Cybernetics*, *36*(5), 943–959.

Lebiere, C., Gonzalez, C., & Warwick, W. (2009). Convergence and constraints revealed in a qualitative model comparison. *Journal of Cognitive Engineering and Decision Making*, *3*(2), 131–135.

Lovett, M. C., & Anderson, J. R. (1996). History of success and current context in problem solving: Combined influences on operator selection. *Cognitive Psychology*, *31*(2), 168–217.

Maehigashi, A., Miwa, K., Terai, H., Kojima, K., Morita, J., & Hayashi, Y. (in-press). Experimental investigation about misuse and disuse in human automation system interaction. In *Hci international 2011*.

Morita, J., Miwa, K., Maehigashi, A., Terai, H., Kojima, K., & Ritter, F. E. (in-press). Modeling human-automation interaction in a unified cognitive architecture. In *The 20th behavior representation in modeling & simulation (brims) conference 2011*.

Parasuraman, R., & Riley, V. (1997). Humans and automation: Use, misuse, disuse, abuse. *Human Factors*, *39*(2), 230–253.

Ritter, F. E., Kukreja, U., & Amant, R. S. (2007). Including a model of visual processing with a cognitive architecture to model a simple teleoperation task. *Journal of Cognitive Engineering and Decision Making*, *1*(2), 121–147.

Salvucci, D. D. (2006). Modeling driver behavior in a cognitive architecture. *Human Factors*, *48*, 362–380.

Singh, I. L., Molloy, R., & Parasuraman, R. (1997). Automation-induced monitoring inefficiency: role of display location. *International Journal of Human-Computer Studies*, *46*(1), 17–30.

Skita, L. J., Moiser, K., & Burdick, M. D. (2000). Accountability and automation bias. *International Journal of Human-Computer Studies*, *52*(4), 701–717.

Vries, P. de, Midden, C., & Bouwhuis, D. (2003). The effects of errors on system trust, self-confidence, and the allocation of control in route planning. *International Journal of Human-Computer Studies*, *58*(6), 719–735.