Assured Human-Autonomy Interaction through Machine Self-Confidence

by

Matthew Aitken

B.S.E. in Mechanical Engineering, Walla Walla University, 2014

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Aerospace Engineering Science

2016

This thesis entitled:
Assured Human-Autonomy Interaction through Machine Self-Confidence
written by Matthew Aitken
has been approved for the Department of Aerospace Engineering Science

Nisar Ahmed

Eric Frew

Brian Argrow

Dale Lawrence

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Aitken, Matthew (M.S. Aerospace Engineering Sciences)

Assured Human-Autonomy Interaction through Machine Self-Confidence

Thesis directed by Prof. Nisar Ahmed

Autonomous systems employ many layers of approximations in order to operate in increasingly uncertain and unstructured environments. The complexity of these systems makes it hard for a user to understand the systems capabilities, especially if the user is not an expert. However, if autonomous systems are to be used efficiently, their users must trust them appropriately. This purpose of this work is to implement and assess an 'assurance' that an autonomous system can provide to the user to elicit appropriate trust. Specifically, the autonomous system's perception of its own capabilities is reported to the user as the self-confidence assurance. The self-confidence assurance should allow the user to more quickly and accurately assess the autonomous system's capabilities, generating appropriate trust in the autonomous system.

First, this research defines self-confidence and discusses what the self-confidence assurance is attempting to communicate to the user. Then it provides a framework for computing the autonomous system's self-confidence as a function of self-confidence factors which correspond to individual elements in the autonomous system's process. In order to explore this idea, self-confidence is implemented on an autonomous system that uses a mixed observability Markov decision process model to solve a pursuit-evasion problem on a road network. The implementation of a factor assessing the goodness of the autonomy's expected performance is focused on in particular. This work highlights some of the issues and considerations in the design of appropriate metrics for the self-confidence factors, and provides the basis for future research for computing self-confidence in autonomous systems.

Dedication

To my parents, for all of the love and support you have given me over the years.

Acknowledgements

First and foremost I'd like to thank my advisor, Dr. Nisar Ahmed. It has been a pleasure working with you, and I greatly appreciate all of your guidance and support, both in this and in other endeavors. I'd also like to thank the rest of my committee, Dr. Eric Frew, Dr. Brian Argrow, and Dr. Dale Lawrence. Thank you for all of the time and feedback you have put into this project.

I'd also like to thank all of the members of the COHRINT lab for their support. Thank you especially to Nick Sweet, Brett Israelson, and Luke Burks, for always being willing to discuss problems and give advice; and to Jeremy Muesing, for helping keep me sane.

Finally, I'd like to thank my parents, Bob and Kathy Aitken, for encouraging and supporting me through out my education. Without you, I would not be were I am today.

Contents

Figures

Nomenclature

γ	reward discount factor
$\mathcal{A}$	set of possible actions
$\mathcal{B}$	space of all beliefs b
$\mathcal{B}_y$	space of all beliefs b_y
$\mathcal{E}$	set of edges
$\mathcal{G}_R$	a graph model
$\mathcal{N}$	set of nodes
$\mathcal{O}$	set of possible observations
$\mathcal{R}$	the reachable belief space from b_0
$\mathcal{R}^*$	the optimally reachable belief space from b_0 under policy π^*
$\mathcal{S}$	set of possible states
$\mathcal{X}$	space of all observable state components
$\mathcal{Y}$	space of all partially observable state components
$\mathcal{Z}$	the set of all outcomes
π	a probability distribution over $\mathcal{S}$

$\tilde{f}$	a perceived probability distribution over outcomes
$\tilde{r}$	a perceived outcome
a	action
b	a probability distribution over $\mathcal{S}$
b_y	a probability distribution over $\mathcal{Y}$
e	an edge between two nodes
f	a probability distribution over outcomes
f_{sc}	a function that generates self-confidence from the self-confidence factors
n	a node
o	observation
R	Reward function for receiving reward r from state s and action a
r	reward
r^*	the reference value, where above is consider a gain and below a loss
s	state
SC	the autonomous system's self-confidence
T	Probability function between state s and s' for a specific action a
T_x	Probability function between partially observed state x and x' for a specific action a and partially observed state y
T_y	Probability function between partially observed state y and y' for a specific action a and fully observed transition between x and x'

V	the sum of all future expected rewards from belief b
v	a value function mapping actual outcomes to perceived outcomes
w	a weighting function mapping the cumulative probability function to a subjective cumulative probability function
w_e	number of states along an edge e
x	observable state
X_{sc}	the factors of self-confidence
y	partially observable state
Z	Probability function for getting observation o given state s and action a
z	an outcome, in this case a cumulative reward

Chapter 1

Introduction

Autonomous systems are performing an increasing amount of work in military, industrial, and civilian environments. Technical advances in autonomous systems reduce human workload, increase safety and performance, and allow them to operate in increasingly unstructured and uncertain environments. While these systems represent significant potential to benefit their users, actual implementation faces many challenges, especially in the aerospace domain. Two sets of challenges of particular interest are barriers due to certification and regulation, and issues with human-autonomy teaming. At their core, both of these challenges are dealing with trust- the user's willingness to depend on a autonomous system.

Autonomous systems face several barriers related to certification and regulation. Regulatory requirements often require direct human supervision and the ability for manual control, which can severely limit operational capability. Additionally, traditional certification standards requiring process-based type certification are difficult or impossible to apply to sophisticated adaptive/non-deterministic systems [1]. Both of these barriers reflect the assumption that the human is responsible for the 'intelligence' of the system, while the autonomy is there to perform low-level assisting functions that are generally deterministic and predictable. As autonomy takes the responsibility for the intelligence of the system from the human, the certification process will need to be extended to be consistent with autonomy's complexity, non-determinism, and adaptability. One solution to this issue is to switch from process-based certification to performance-based licensing, similarly to how people are licensed. However, while certification relies on certainty, licensing is inherently uncertain and requires a significant amount of trust [2].

Human-autonomy teaming also requires trust. In order for humans and autonomy to work together effectively, they must be able to develop appropriate levels of trust in each other. Autonomous systems, however, are generally opaque to non-expert users, and users may over-trust or mistrust the system. If the user over-trusts the system, the user will misuse the system, which can lead to disastrous consequences. On the other hand, if the user mistrusts the system, then many of the benefits provided by the system may go unrealized [3, 4].

Both of these challenges require trust in autonomy. Not only that, they require that trust be calibrated appropriately. Related questions have been studied by human factors, artificial intelligence, robotics, and control engineering communities. [5, 6, 3, 4, 7, 8, 9, 10]. This work builds off of the trust model shown in Figure 1.1 proposed in Reference [2], which models trust as a closed-loop system between the autonomy and the user. In this model, the users trust is modeled as a multidimensional time-varying process that generates trust actions: actions that are chosen because of the user's trust in the autonomous system.

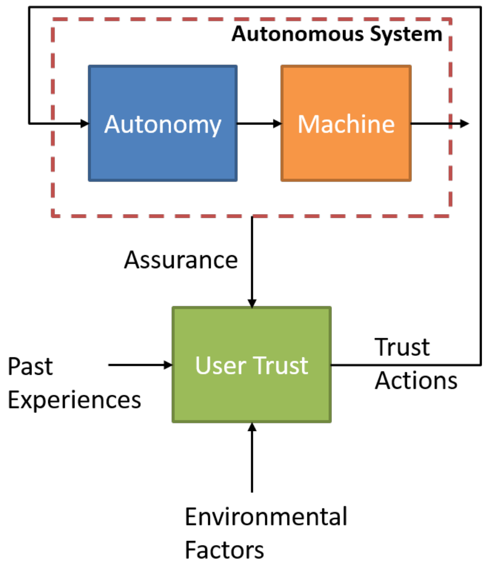


Figure 1.1: Closed loop model of the autonomous system and user trust

The autonomous system can provide assurances to influence the user's trust, which will in turn impact the user's trust actions. If the assurances are designed well, they can calibrate the user's trust, raising or lowering it as necessary. The ideas of trust and assurance will be discussed in detail later on in the thesis.

The purpose of this work is to implement and assess an assurance called 'self-confidence'. Self-confidence is an autonomous system's perceived ability to execute assigned tasks, despite uncertainties in (1) its knowledge of the world, (2) its knowledge of its own state, (3) its reasoning and execution capabilities [11]. Self-confidence is a holistic assessment of the autonomy's capabilities; in other words, it is an autonomous system's self-trust. The goal of self-confidence is to calibrate the user's trust by providing a metric that easily allows the user to understand the autonomy's assessment of the task and its projected ability to achieve a satisfactory result.

In order to ground this study, this work uses an example problem of pursuit-evasion on a road network with an autonomous unmanned ground vehicle. In chapter 2 we describe this scenario in depth and discuss the ideas of trust and assurance in autonomous systems. In chapter 3, we introduce the self-confidence assurance and discuss the reasoning behind it, as well as describing its properties as an assurance. In chapter 4, we consider the autonomy's underlying algorithms and develop a metric to express the autonomy's self-confidence based on those underlying algorithms, and in 5 we design and implement a particular factor of self-confidence called Outcome Assessment. We then analyze the proposed factors and self-confidence metric in several different example scenarios in chapter 6 and present the conclusions of this work in chapter 7.

Chapter 2

Background Information

This section contains technical background information used throughout this document. Section 2.1 describes the grounding scenario that we have chosen to explore self-confidence. Section 2.2 provides the mathematical formulation of Mixed Observability Markov Decision Processes (MOMDPs), which is what we use to model this scenario. Scenario 2.3 details how the grounding scenario is formulated as an MOMDP problem, and section 2.4 describes the SARSOP algorithm that the autonomy uses to solve the MOMDP problem. Finally, section 2.5 goes into more detail about trust in autonomous systems by providing models of how the user trust evolves and interacts with the autonomous system.

2.1 Pursuit Evasion Problem Description

We use a modified version of the UAV-UGS road reconnaissance (U2R2) scenario [12] in order to ground the discussion on self-confidence. In this scenario, one unmanned ground vehicle (UGV) attempts to escape a road network while avoiding a chaser. The chaser must be localized through the use of unattended ground sensors (UGS), which act as motion detectors. The UGS's are scattered throughout the road network, and have sensor characteristics defining rates of false positives and false negatives. Each UGS sends a signal to the ground vehicle anytime a detection is triggered. The scenario ends if either the UGV manages to reach exit states, or if the chaser catches the UGV.

The autonomy in this system resides on board the UGV, allowing it to make navigational decisions based on its belief of the chaser's location. A remote human supervisor is able to monitor and interact with the UGV, and their decisions are influenced by the trust he/she has in the autonomy. The supervisor is assumed to be comfortable with this scenario, but also not expected to have low-level or expert knowledge of the autonomous system. The supervisor is allowed to modify the autonomy's decision making by selecting different planning stances (e.g. 'aggressive', defensive', etc). The supervisor could also provide information and advice to the autonomy by identifying intruder behavior, updating the map with new obstacles or routes, and providing soft sensor data.

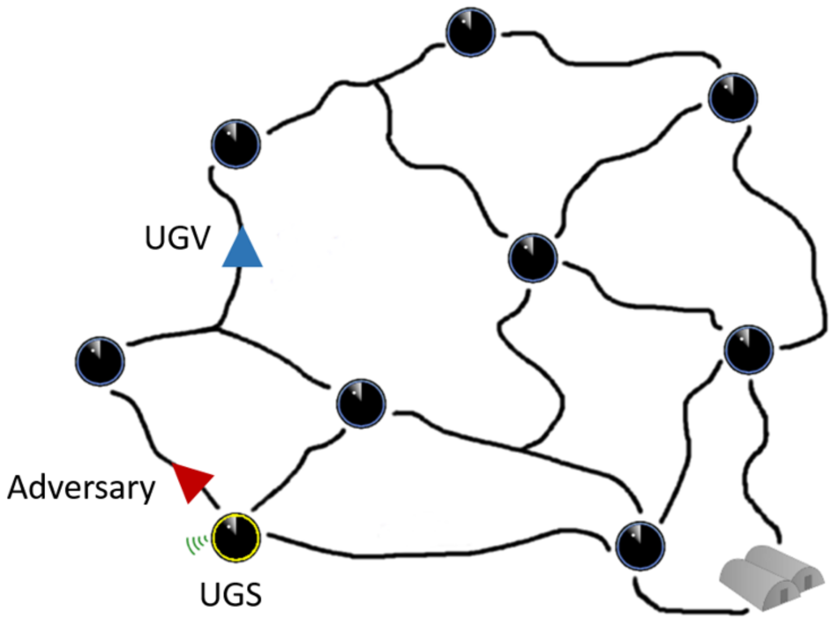


Figure 2.1: An example road network

The reasoning behind this scenario is two-fold. One, the scenario requires the autonomy to operate in a complex and uncertain environment, allowing the autonomy to generate meaningful self-confidences based on the approximations and uncertainty it necessarily needs to use to make decisions. Two, the supervisor's interactions with the autonomy are expected to be dependent on their trust in the autonomy. For example, the supervisor may refuse to let the autonomy execute its plan for a new scenario if the supervisor does not trust it, which could be because the autonomy had previously failed in other scenarios.

2.2 The Mixed Observability Markov Decision Process

In order for the autonomy to make decisions, it needs a method of understanding the world. For this scenario, we formulate the road network escape problem as a mixed observability Markov decision process (MOMDP) [13], which is a factored partially observable Markov decision process (POMDP) [14]. For any Markov Decision Process (MDP), the goal is find a policy for the decision maker that specifies what actions they should take at any state to maximize a reward function. To understand a MOMDP, we will first look at a POMDP.

2.2.1 The Partially Observable Markov Decision Process

A POMDP is a discrete-time stochastic control process. The goal is to determine the actions that the agent should take in order to maximize a reward function. Formally, a POMDP is a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{O}, T, Z, R, \gamma)$, where $\mathcal{S}$ is a set of possible states, $\mathcal{A}$ is a set of actions, and $\mathcal{O}$ is a set of observations. At each step, the process is in a state s. The agent chooses an action a, and at the next step the process moves to state s' with probability $T(s'|s,a) = p(s'|s,a)$. The agent then receives an observation about the state $o \in \mathcal{O}$ with probability $Z(o|s',a) = p(o|s',a)$, and a reward $R(s,a)$. This is shown in Figure 2.2. An agent's goal is to take the action at each state that will maximize its expected discounted cumulative reward (utility), given by $\sum_{t=0}^{\infty} \gamma^t r_t$, where the discount rate $\gamma \in [0,1]$ determines how immediate rewards are balanced against the expected sum of future rewards. Different reward functions $R(s,a)$ will generate different behaviors, as the actions the agent takes depend on the reward for taking those actions.

Since the state is unknown, the agent must maintain a belief b which is a probability distribution over the state space $\mathcal{S}$. Because the state has the Markov property, all that is need to determine b' is the previous belief b, the action taken a, and and the received observation o. This is given by equation 2.1, where $b(s)$ is the probability that the process is in state s and η is a normalizing constant given by equation 2.2.

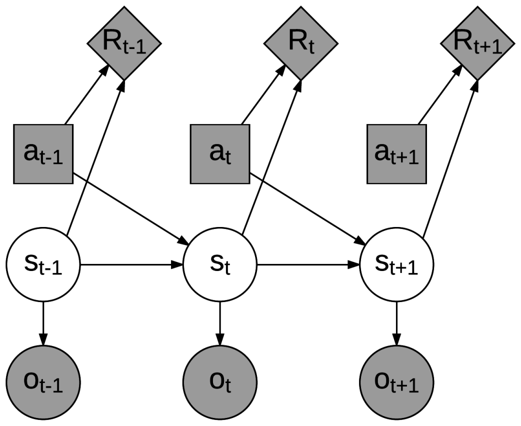


Figure 2.2: The POMDP model

$$b'(s') = \frac{1}{\eta} Z(o|s', a) \sum_{s \in \mathcal{S}} T(s'|s, a) b(s) \tag{2.1}$$

$$\eta = \sum_{s' \in \mathcal{S}} Z(o|s', a) \sum_{s \in \mathcal{S}} T(s'|s, a) b(s) \tag{2.2}$$

After the agent determines its belief, it then takes an action according to its policy π. A policy π specifies an action $a = \pi(b)$ for any belief $b \in \mathcal{B}$. A policy may be any mapping between actions and beliefs, but the optimal policy π^* is the policy that maximizes the expected reward from a starting belief b_0, given by equations 2.3 and 2.4, where V^π is the value function and r_t is the expected reward for any belief and action pair

$$V^\pi(b_0) = \sum_{t=0}^{\infty} \gamma^t r_t(b_t, a_t) = \sum_{t=0}^{\infty} \gamma^t \sum_{s \in \mathcal{S}} R(s, a_t) b_t(s) \tag{2.3}$$

$$\pi^* = \underset{\pi}{\operatorname{argmax}} \, V^\pi(b_0) \tag{2.4}$$

For simple problems, a dynamic programming technique called value iteration can be used to determine the optimal policy [15]. In practice, however, finding the exact solutions to POMDPs is computationally intractable. Current methods often use sampling techniques and extensions of value iteration to approximate the optimal policy, and other methods exploit the structure of the problem to reduce the amount of computation necessary to achieve a near optimal policy [16, 17]. Additionally, if only part of the state is observed, and the other part is always known, the POMDP model can be simplified to a mixed observability Markov decision process (MOMDP), which can also reduce computation time [13].

2.2.2 POMDP to MOMDP

A MOMDP model is specified as a tuple $(\mathcal{X}, \mathcal{Y}, \mathcal{A}, \mathcal{O}, T_x, T_y, Z, R, \gamma)$, where the state space is factored as $\mathcal{S} = \mathcal{X} \times \mathcal{Y}$. $\mathcal{X}$ is the space of all possible values of fully observable state components x and $\mathcal{Y}$ is the space of all possible values of the partially observable state components y. Instead of a single transition probability T, the MOMDP considers the transition probabilities for each component separately. The conditional probability function $T_x(x'|x, y, a) = P(x'|x, y, a)$ gives the probability that the fully observable state variable has value x' if the agent takes action a in state (x, y), and $T_y(y'|x, y, a, x') = P(y'|x, y, a, x')$ gives the probability that the partially observable state variable has value y' if the agent takes action a in state (x, y) and the fully observable state variable has value x'. Everything else remains as it is in the POMDP.

The main benefit of the MOMDP over the POMDP is that the belief space $\mathcal{B}$ becomes significantly smaller. Since x is known, any belief is simply represented as $b = (x, b_y)$. This reduces the problem dimension from $|\mathcal{X}||\mathcal{Y}|$ to simply $|\mathcal{Y}|$, leading to a vast improvement in computational efficiency for POMDP solvers.

2.3 MOMDP Formulation of the Pursuit-Evasion Scenario

In this pursuit-evasion scenario, we model the road network as a finite set of states. Specifically, the map of possible states for the UGV and the pursuer is a discrete space realization of a road network specified by the graph model $\mathcal{G}_R = (\mathcal{N}, \mathcal{E})$ consisting of a finite set of nodes $\mathcal{N}$ and edges $\mathcal{E}$. The nodes represent intersections in the road network, while the edges represent the streets between the intersections. The edges are divided into multiple states $[1, 2, ...w_e]$, where w_e

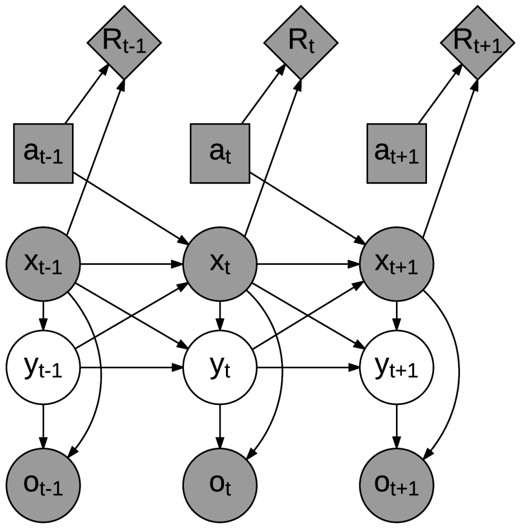


Figure 2.3: The MOMDP model: The state is factored into two components: $s = (x, y)$, where x is fully observable, and y is partially observable

is the number of states along an edge e between two nodes. This gives a final list of discrete states $\zeta \in [1, 2...d]$ where $d = \sum_{e \in \mathcal{E}} w_e + |\mathcal{N}|$, where each state is either a node n or a tuple of edge and position along the edge (e, ζ). With both the pursuer and the UGV, the full state is described as $s = (x, y)$ where x is the UGV's state and y is the pursuer's state, giving a total state space size of d^2.

The actions available to the autonomy are UGV movements between adjacent states. Along an edge, the UGV may move to the next state, the previous state, or remain at the current state. At a node, the UGV may enter any of the connecting edges, or it may remain at the intersection. For simplicity, we assume the UGV can only move one state at a time, and when an action is taken to move to an adjacent state, the UGV moves with probability 1, as shown in equation 2.5:

$$p(x'|x, a) = \begin{cases} 1 & \text{if } (x, a) \to x' \\ 0 & \text{else} \end{cases} \tag{2.5}$$

At the same time, the pursuer moves according to a model specified by the transition probability $T_y(y'|x, y, a, x')$, which encapsulates the autonomy's assumptions of the pursuer's behavior. We implement several different motion models for the pursuer, including random walk, where the pursuer moves to an adjacent state to its current position with uniform probability, and a targeted mode, where the pursuer always moves along the shortest path to the UGV. The human can identify the pursuer's behavior by selecting a specific predefined motion model, which changes T_y accordingly.

The observations received by the autonomy at each time step are simply a binary detection/no detection from every UGS in the road network. Each sensor has a false positive rate f_p and a false negative rate f_n that define the probability $Z(o_i|y)$ of detecting the pursuer, given in equation 2.6. The observation o that the autonomy receives is the tuple of all binary observations from every sensor in the road network, $o = [o_1, o_2, ...]$.

$$Z(o_i = \text{detect}|y) = \begin{cases} 1 - f_n & \text{if } y = \zeta_i \\ f_p & \text{else} \end{cases} \tag{2.6}$$

Finally, the human can choose from several predefined reward functions, whose form is shown in 2.7. Generally, there is a small cost or reward for taking an action, a large negative reward for being caught, and large positive reward for reaching the exit state. Changing the reward function changes how the autonomy behaves. Increasing the reward for escaping can cause the autonomy to be more aggressive in attempting to get to the exit, while increasing the cost of being caught can make the autonomy be more cautious, waiting to gather a more accurate position estimate of the pursuer's location.

$$R(s, a) = \begin{cases} R_{escape} & \text{if } x = \zeta_{exit} \\ R_{caught} & \text{if } x = y \text{ or } x' = y \\ R_{action\ cost} & \text{else} \end{cases} \tag{2.7}$$

2.4 Policy Generation and SARSOP Algorithm

In order to solve this MOMDP problem, the autonomy implements a solution technique called Successive Approximations of the Reachable Space under Optimal Policies, or SARSOP [18]. Like other sampling-based algorithms, SARSOP attempts to approximate π^* by using value iteration on a sample of the belief space, rather than the entire belief space $\mathcal{B}$. The idea behind SARSOP is that while the belief space $\mathcal{B}$ is large, the reachable belief space $\mathcal{R}(b_0)$ is smaller, and even smaller still is the optimally reachable space $\mathcal{R}^*(b_0)$, as shown in Figure 2.4. The reachable belief space is a subset of beliefs reachable from an initial belief b_o under arbitrary sequences of actions, and the optimally reachable belief space is a subset of beliefs reachable from an initial belief b_0 under the optimal policy π^*. By focusing on $\mathcal{R}^*(b_0)$, the algorithm can increase efficiency by ignoring areas of the belief space that are unreachable. Since $\mathcal{R}^*(b_0)$ is unknown, SARSOP uses heuristic exploration and on-line learning techniques to iteratively converge on $\mathcal{R}^*(b_0)$.

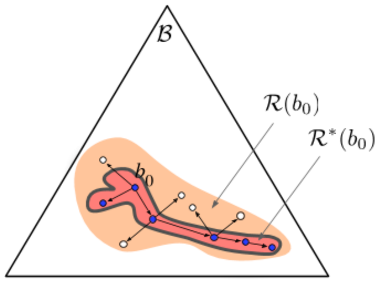


Figure 2.4: Optimally reachable space, reachable space, and belief space: $\mathcal{R}^*(b_0) \subseteq \mathcal{R}(b_0) \subseteq \mathcal{B}$ [18]

The algorithm uses the fact that the value function of equation 2.3 can be approximated by a piecewise-linear convex function given in equation 2.8 and shown in Figure 2.5:

$$V(b) = \max_{\alpha \in \Gamma}(\alpha \cdot b) \tag{2.8}$$

Γ is a finite set of vectors called α-vectors, and each one is associated with an action. The policy generated by the algorithm is simply choosing the action a associated with the α-vector that results in the largest $V(b)$.

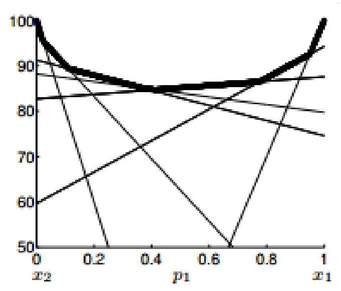


Figure 2.5: An example α-vector value function for a one-dimensional belief [15]

2.5 Trust and Assurance

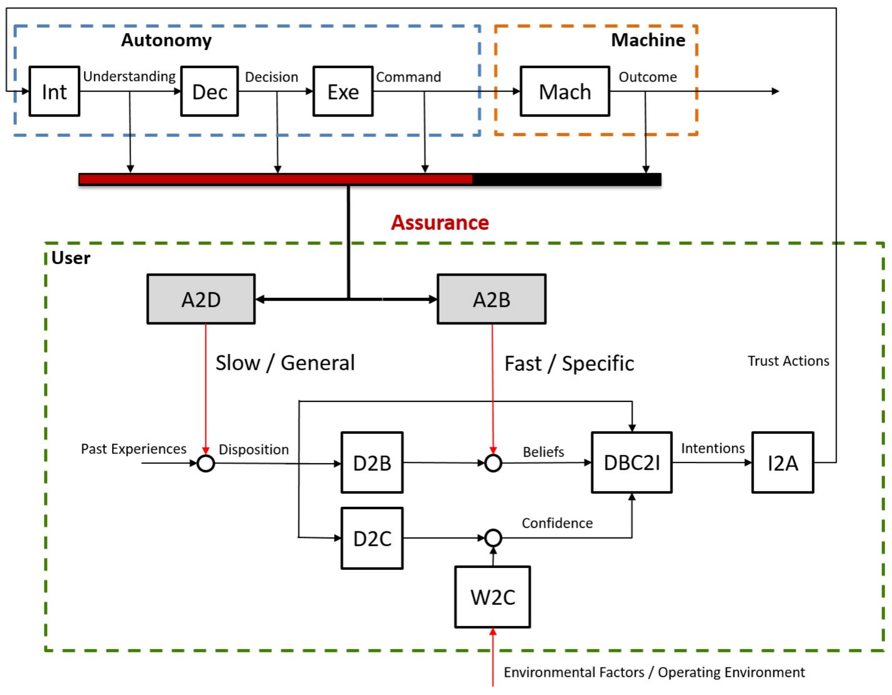


Figure 2.6: Closed loop model of the autonomous system and user trust

Trust has been studied across many different disciplines, including psychology, business, and engineering. While many different methods of exploring trust have been proposed, this work is an extension of the model proposed in Reference [2] and shown in Figure 2.6. This model combines ideas from many different disciplines, but ultimately takes a control engineering perspective on trust. The fundamental assertion of this model is that the user's trust in an autonomous system can be affected by past experiences, environmental factors, and by the autonomy itself. In turn, the user's trust affects how it interacts with the autonomy, creating a closed loop system between the user and the autonomous system. This model defines the following key terms:

- **Autonomous System**: an agent or system comprised of a machine being driven or controlled by some form of autonomy. An autonomous system always interacts with a human user.
- **Autonomy**: the ability to perform complex mission-related tasks with substantially less human intervention for more extended periods of time, sometimes at remote distances, than current systems.
- **Machine**: the physical system and supporting low-level electronics and/or software in an autonomous system commanded by the autonomy.
- **Trust**: a user's willingness to depend on the autonomous system. Trust depends on a particular system or situation, as well as a user's background dispositions and beliefs. Trust always leads to an action.
- **Assurance**: the autonomous system's ability to affect the user's trust. The term is not intended to have a positive or negative connotation - assurances can decrease trust.

2.5.1 The User Trust Model and Trust Actions

The user trust model, shown in Figure 2.7, describes how a user's trust changes due to various inputs. It models trust as a time-varying, multivariate process that ultimately affects how the user interacts with the autonomy through trust actions. The user trust model is broadly broken up into three main sections: disposition, belief, and system confidence. Each one tends to change at different rates and respond to different inputs.

The user's disposition is a reflection of their propensity to trust in general (Trusting Stance) and their general perception of autonomy (Faith in Autonomy). Their propensity to trust is weighing of potential benefits against the risk trusting the unknown, and is a reflection of their natural willingness to trust others. Their general perception of autonomy indicates whether they think that a type of autonomy is good willed and will perform in the expected manner. This is similar to optimism and pessimism.

The user's system confidence is influenced by the current environment (Normality) and background knowledge (Structural Assurance), as well as the user's disposition. Structural Assurance encapsulates the effect of things like regulations, testing, and expert opinions on a user's trust. Normality describes the effects that the environment has on trust, for instance, a rough storm reducing the trust in an airline autopilot.

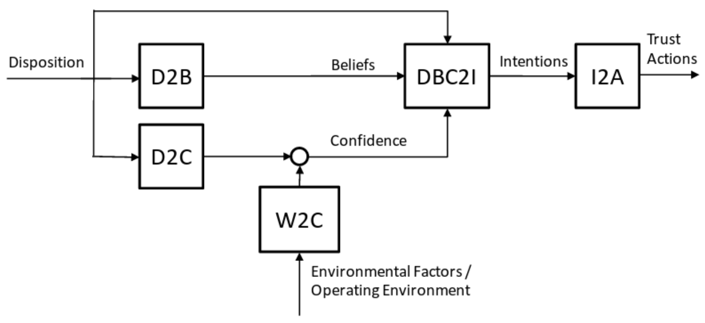


Figure 2.7: The User Trust Model

Finally, the user's belief is specific to the current autonomous system. These beliefs influenced by their system confidence and their disposition, but they are also affected by the assurances provided by the autonomous system. There are four main sections of the user's beliefs, and they are Benevolence, Integrity, Competence, and Predictability. Benevolence is how well the user believes the autonomous system is interpreting the user's actual intent. Integrity is how well the user believes the autonomous system is communicating necessary information back to the user. Competence is the user's belief in the autonomous system's capabilities, and predictability is the user's belief that past performances are indicative of future performance.

All of these dispositions, system confidences, and beliefs come together to affect how the user interacts with the autonomous system. These interactions that are affected by the user's trust are called Trust Actions. These actions change how the autonomous system is used, and they give the autonomy insight into the user's current trust.

2.5.2 The Autonomous System and Assurances

The Autonomous System Model in Figure 2.8 divides the autonomous system into two parts: the autonomy, and the machine. As defined before, the autonomy is responsible for the 'intelligent' behavior of the system, while the machine is the physical system and supporting low-level software. The autonomous system is also broadly divided into four elements: Interpreter, Decision, Execution, and Machine.

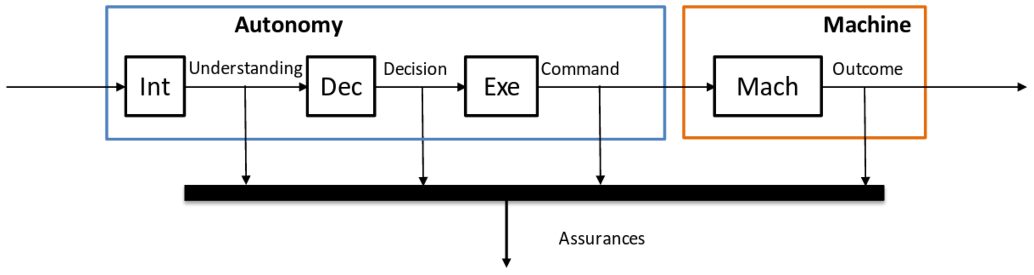


Figure 2.8: The Autonomous System Model

The interpreter element is responsible for decoding the user's trust action into an 'understanding', related to the user's intent. The decision element then takes the understanding and uses it to generate decisions based on the autonomy's understanding of the user's intention. The decision is then used by the execution element to generate commands to send to the machine. The machine then takes those commands and executes them, generating an observable outcome. At each step in the process, the autonomous system can offer assurances to the user, giving them a greater understanding of the autonomous system. By giving the user these assurances, the autonomy is able to influence the user's trust to match the system's capabilities.

Chapter 3

Trust and Self-Confidence

Before designing an assurance for a particular system, it is necessary to first understand what the assurance is, what it is attempting to accomplish, and how it should affect the user's trust. This chapter explores the nature of self-confidence and what it means to an autonomous system. We also discuss the projected usefulness of self-confidence to the certification of autonomous systems and to human-autonomy teaming, especially in cases where the user is in a supervisory 'on-the-loop' role. Finally, we discuss how self-confidence as an assurance will affect the user's trust.

3.1 Self-Confidence Definition

Several definitions of self-confidence have been proposed in recent works. Reference [11] provides a brief summary of the relevant literature and identifies four different views of self-confidence before proposing a (mostly) unifying definition of self-confidence used in this work. The four views on self-confidence are the *anthropomorphic view*, the *uncertainty view*, the *experiential view*, and the *stability view*. The anthropomorphic view defines self-confidence to be similar to how humans express self-confidence, while the experiential view expresses self-confidence based on past experience. The uncertainty view simply defines self-confidence to be the probability of success or failure, and the stability view defines self-confidence to be the sensitivity of the probability of success to uncertainty. All of these views seem to reflect different parts of a more general concept: understanding an autonomy's ability to do a specific task. This leads to the following definition of self-confidence.

We define self-confidence to be an agent's perceived ability to achieve assigned goals (within a defined region of autonomous behavior) after accounting for (1) uncertainties in its knowledge of the world, (2) uncertainties of its own state, and (3) uncertainties about its reasoning process and execution abilities. In other words, self-confidence is how the autonomy trusts itself to accomplish a specific task. This includes, but is not limited to, taking into account how well it has done in similar circumstances in the past (experiential), how well it thinks it will do now (uncertainty), and how changing circumstances will affect it (stability). It is a holistic self-assessment of all information, processes, and decisions that go into the execution of a particular task.

3.1.1 Certainty and Goodness

Self-confidence asks two main questions: "how certain is the analysis?", and "how good is the anticipated performance?" To answer the first question, the autonomy must consider the three uncertainties in the definition. Even if it is highly confident in the goodness of its anticipated performance (e.g. a very high probability of success, or a generally good outcome), it may be very unsure in its analysis (e.g. there are a lot of 'unknown unknowns', the solver is known to return overly-optimistic results on this type of problem, etc). Alternatively, the autonomy may be highly confident in its analysis as well as in its anticipated performance. In the first case, the autonomy would have a low self-confidence, but in the second it would have a high self-confidence.

To answer the second question, the autonomy must consider how it thinks the task will go. This is not the expected utility, but rather a consideration of how likely it is to get a good outcome verses a bad one. However, the goodness of an outcome distribution is not just the probability of success; most tasks have some cost or reward function associated with them. For instance, the outcome of an inspection task is generally not success/fail, but a function of what areas were surveyed. Depending on the function, spending time inspecting one area may be more beneficial than briefly surveying the whole area, but it may also be that the opposite is true. Because outcomes are rarely black and white, the second part of self-confidence must capture the goodness of the autonomous system's anticipated performance during the task. Section 5 will go into more detail about the difference between goodness and utility.

Without the answer to both of these questions, the autonomy can not report self-confidence. It must know both in order to not overestimate its own capabilities. Without certainty, the user has no way of distinguishing between correct and incorrect analysis. Without goodness, the user has no assurance as to how well the autonomy will do. Without both, the user is more likely to incorrectly trust the autonomous system.

3.1.2 Insight into Capabilities

Self-confidence is closely tied to the idea competency boundaries [8]. As an autonomous system approaches its competency boundaries, its self-confidence should generally decrease. If the problem is near or outside of the limits of the autonomy's solving ability, any solution generated by the autonomy should be suspect and the autonomy should not be very self-confident. Similarly, if the information that the autonomy is given is not enough for it to make good decisions, it should also not be self-confident. Because of this relationship between self-confidence and competency boundaries, self-confidence should give the user insight into what the system is actually capable of.

Since self-confidence is a function of each part of the autonomy, it can be a useful tool to guide the interrogation of an autonomous system. For instance, the user can compare the autonomous system's self-confidence in different scenarios to draw conclusions about the autonomous system's capabilities. Though we define self-confidence to be a singular metric in 3.3, the process that goes into generating self-confidence can be used in other ways, like highlighting low-confidence areas of the autonomous system.

3.1.3 Imperfect Metric

An autonomy's self-confidence and the autonomy's actual capabilities will most likely never match because the autonomy can generally never perfectly predict the exact real world outcomes of its actions, nor can it usually completely assess the autonomy's internal processes. Nonetheless, it is still a useful metric in the same way that an noisy measurement is useful for estimation. While the autonomy's self-confidence may be erroneous, it will give an approximate indication of the system's capabilities, allowing an easier estimate of the system's abilities. This should result in a much quicker calibration of user trust to the autonomy's capabilities, as discussed in the following section.

3.2 The Goal of Self-Confidence

Autonomous systems require appropriate trust from the user in order to be used effectively, because trust guides the user's reliance on the system. When the user's trust in the autonomous system is matched to its actual capabilities, their trust is *calibrated*, that is, they are not overtrusting or distrusting the system. However, if their trust is not calibrated, overtrust may lead to misuse, and distrust may lead to disuse [3].

In the case of overtrust, the user overestimates the autonomous system's capabilities. This can lead to a misuse of the system and possibly fatal consequences. As one example, a study on human-robot interaction found that human participants would follow an emergency robot past emergency exits and even into dark, barricaded rooms, even when the participants where shown the robot not working previous to the emergency [5]. In spite of the evidence of the faulty robot, the participants continued to trust the robot even when it was not appropriate. In other cases, inadequate monitoring of autopilots in airplanes has led to several accidents [3].

In the case of distrust, the user underestimates the autonomous system's capabilities and thus disuses the system, removing any benefit that the system could bring. This is especially true with systems that generate warnings, as too many false alarms can lead to 'creative disablement' of the systems [3]. Since the cost of false alarms is low relative to the cost of missing a true alarm, systems tend to minimize the chance of a missed true alarm. This generally leads to increasing false alarms, which can then cause a user to mistrust and then disuse the system. In other cases, where the autonomy is not trusted to run without human supervision, like search and rescue operations and UAV flights, the benefit of having extra autonomous agents is lost.

The goal of self-confidence is to calibrate a user's trust in an autonomous system in order to avoid misuse and disuse. Several papers indicate that trust can be enhanced by providing the user with richer, more insightful assurances from the autonomy about its internal processes [3, 6, 4]. The more the user understands about the system, the easier it is for them to correctly calibrate their trust. Self-confidence should easily allow the user to understand the autonomous system's capabilities and limitations. This can be particularly useful for both certification and human-autonomy teaming.

3.2.1 Self-Confidence for Certification

Self-confidence could be particularly helpful during certification procedures for autonomous systems. Current regulations based on exhaustive testing or formal verification and validation methods can be impractical or impossible to apply to sophisticated, non-deterministic systems. While outcome-based testing over a few representative samples may indicate performance of the autonomy, they do not give the necessary insight into how the autonomy might react to unanticipated circumstances. However, self-confidence coupled with some explanatory assurances could allow the autonomy to be licensed in a way that is similar to how humans are licensed. The autonomous system could be tested out in various scenarios, and at the same time queried about its decisions and its confidence in its ability to perform the given tasks. A licensor could then make a more informed decision than would be possible with just a sample-based testing approach, and at the same time allow the autonomy to become licensed without having to undertake a costly or impossible certification procedure [2].

3.2.2 Self-Confidence for Human-Autonomy Teaming

Self-confidence is also expected to enhance human-autonomy interaction and teaming. Many of the challenges of human-autonomy teaming arise from the opaqueness of autonomy's decision-making and inappropriate reliance on autonomous systems [4]. Additional insights like self-confidence can help foster trust and understanding between the human and autonomous team members, improving the overall performance of the team [19]. Self-confidence can also help the user identify the autonomy's weak areas, allowing the human and autonomy to better compliment each other by allowing each to use the strengths of the other. This is especially important as human-autonomy interaction begins to shift towards integrated systems that require dynamic cooperation and task management [11].

3.3 Self-Confidence as an Assurance

Implementing self-confidence as an assurance requires understanding how it should affect the user's trust model (Figure 2.7). Self-confidence is specifically targeted at affecting the user's belief trust state, although it will also effect the user's disposition over time. Specifically, self-confidence as an assurance should mainly affect the competence belief state, and may have a small effect on the Integrity, Benevolence, and Predictability belief states. This section will discuss the expected effects on each of these user trust states, as well as the temporal and functional properties of the self-confidence assurance.

3.3.1 Effect on Trust States

Self-confidence as an assurance is meant to affect the user's belief in the competence of the system. As discussed in section 3.2, it is designed to help the user assess the system's capabilities. As the user's belief in the competence of the autonomy changes, so does their intentions to rely on the autonomy. This results in changing of trust actions, which change how the autonomy operates and thus changes the autonomy's self-confidence. This closing of the loop between the user and the autonomy through the feedback of assurances and trust actions should allow the team to more quickly gain an efficient equilibrium, where the combined performance of the autonomy and the user is at its best.

The self-confidence assurance will also affect have smaller side effects on the Benevolence, Integrity, and Predictability states, as well as the user's disposition towards similar autonomy. These generally are because of the additional understanding of the system afforded by self-confidence. For example, the user's belief of the systems predictability may increase when the user sees that the autonomy generated very different self-confidences for cases that looked similar to the user. Without self-confidence, the user might view the autonomy as unpredictable, but with the self-confidence they may see that what they thought were to similar circumstances were actually very different to the autonomy. Similarly, the user may change their view on the system's benevolence because they are more able to decide whether an 'incorrect' action was due to a misalignment between the user's and the autonomy's intentions or if it was just simply a mistake. In the case of Integrity, just the presence of the self-confidence assurance may increase the user's trust in the system because of the additional information the system is providing. Finally, the user's disposition towards autonomy may change based on whether or not the autonomy provides self-confidence to the user. All of these secondary affects are not necessarily unique to self-confidence, but it is important to consider the side affects of particular assurance before implementing it.

3.3.2 Time Frames

The term 'reliance' can often tend to be interpreted as simply a choice to use or not use the system as a whole, which somewhat limits trust to only be useful at that decision point. However, many autonomous systems are not simply 'go/no go' systems. This especially true in the areas we are looking at, with humans in 'on-the-loop' roles or in human-autonomy teams. Trust, and therefore reliance, is a dynamic state that changes over time. When self-confidence should change, though, is problem specific. In general, we will talk about three time frames when it comes to self-confidence: long term, in-task, and pre-task self-confidence.

Long term self-confidence refers to the autonomy's self-confidence over many tasks. In the same way that the human's disposition towards a type of autonomy changes gradually over time, the long term self-confidence refers to the autonomy's general ability to execute a category of tasks. Pre-task self-confidence refers to the self-confidence an autonomy has in executing a specific task before it receives any feedback about the task. Finally, in-task self-confidence reflects the autonomy's moment-by-moment confidence in its ability to finish a task. The main difference between pre-task self-confidence and in-task self-confidence is that in-task self-confidence takes into account information it learned during the execution of a particular task. While these definitions depend on the definition of task (e.g. is the task a set of goals, or just one goal), they are meant to parallel the important time frames in the user's trust. As a task goes on, the autonomy's self-confidence may increase or decrease because of feedback it receives during the task, and the self-confidence assurance should communicate that to the user. However, in cases where the user only checks on the autonomy at the beginning of tasks, this temporal specificity may not be necessary. Regardless, it is important to make sure that the assurance is temporally specific enough for the chosen time frame. For this thesis, we will restrict the self-confidence assurance to be only the autonomy's pre-task self-confidence.

3.3.3 Properties

Sections 3.1 and 3.2 discussed what self-confidence is and what its purpose is. However, it is not exactly clear what should be reported to the user[1] , or how the certainty and goodness parts of self-confidence interact. This section will assert the properties that the self-confidence assurance will have.

- **Self-confidence is a single metric** Since self-confidence should be easy to interpret, we define the self-confidence assurance to be a single metric to be reported to the user. We define this metric to be on a scale of -1 to 1, where -1 is 'complete certainty in an absolutely bad outcome', 1 is 'complete certainty in an absolutely good outcome', and 0 is 'complete uncertainty in the result being good or bad'. Whether this comes from uncertainty in the analysis process or from known uncertainty in the outcome is not distinguished.

- **As certainty decreases, goodness matters less to self-confidence** In the limit of complete uncertainty, it does not matter what the anticipated performance is, self-confidence will always go to zero.

- **As certainty increases, goodness matters more to self-confidence** If the autonomy is completely certain, the goodness of the anticipated performance is all that matters.

[1] See discussion in section 7.1.3 on *how* it should be reported

Chapter 4

Self-Confidence for Pursuit-Evasion Scenario Autonomous System

4.1 Autonomous System Overview

Before we begin to design the self-confidence for the system, we must understand the autonomous system. As discussed in 2.5.2, the autonomous system is divided into two parts, the autonomy and the machine. The autonomy is responsible for the high-level decision making and the interaction with the user, and the machine is low-level software and physical hardware that allows the autonomy to interact with the world. In this scenario, the autonomy is the software handling the implementation of the MOMDP for decision-making, and the machine is the physical UGV, as well as its low-level software that provides basic navigation. The self-confidences that we will explore in this chapter are just those associated with the autonomy; we do not explore self-confidences associated with the machine. This is in part due to the nature of POMDPs, and will discussed more in 4.4.

4.1.1 The Autonomy

The autonomy is the part of the system responsible for the high-level decision making. As described in sections 2.1-2.4, the autonomy creates a MOMDP model of the world from given information: a graph representation of the road network, a specific model of chaser behavior and machine behavior, sensor locations and characteristics, an initial belief of the location of both the chaser and the UGV, and a specific reward function. It then uses the SARSOP algorithm to generate a policy that it can use to look up the actions that it should take for any given UGV location and belief on the chaser's location. The autonomy then uses this policy to send commands to the machine, telling it to move to one state in the road network to another.

4.1.2 The Machine

The machine takes the commands generated by the autonomy and turns them into actions. In this case, the machine is a Clearpath Robotics Husky^{TM}, shown in Figure 4.1. The Husky takes the actions generated by the policy and translates them into goal positions along the road edges, and then uses a path planner to generate velocity commands. It is also responsible for handling the communication to the UGSs and the user interface.

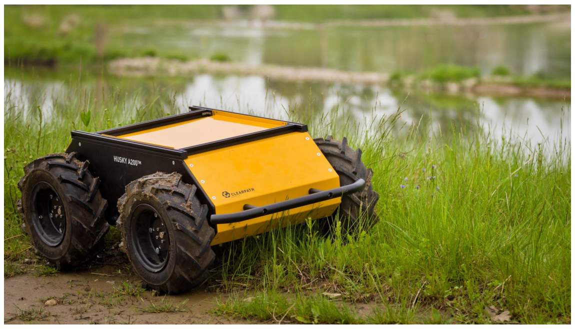


Figure 4.1: The Clearpath Robotics Husky [20]

4.2 Self-Confidence Implementation

In Figure 4.2, we model the process that the autonomous system uses to execute an assigned task. The first block is the interpretation block, where the autonomy takes the user's commands and information to generate a model of the world. In this case, the interpreter allows the user to select from several stances ('aggressive','defensive','moderate',) that correspond to MOMDP reward functions. It also allows the user to identify the chaser's behavior in the same way ('chasing','searching','random walk', etc). The rest of the model (the road network, the observations, and the UGV state transition probabilities) are taken from pre-existing knowledge. The solver (in this case, the APPL software running the SARSOP algorithm) takes the MOMDP model and generates a policy, which is then executed by the autonomy and the machine. Along with the policy, the autonomy also generates an outcome distribution describing the probability of getting a specific outcome, or cumulative reward. This outcome distribution is the autonomy's anticipation of what will happen, and does not necessarily match reality.

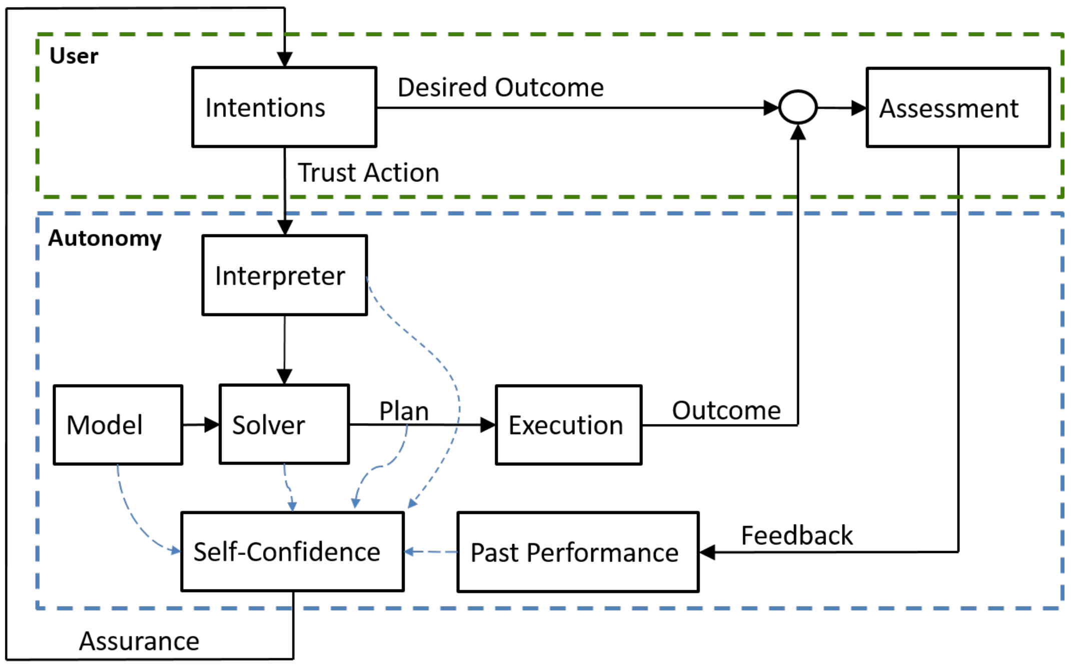


Figure 4.2: The Self-Confidence Implementation Model

Then, taking the principles from chapter 3 and the user-autonomy system model in 2.5, we define a set of factors that will effect the autonomous system's self-confidence. These factors are:

- x_1 Command Interpretation
- x_2 Model Validity
- x_3 Solver Quality
- x_4 Outcome Assessment
- x_5 Past Performance

Command Interpretation, Model Validity, and Solver Quality are the autonomy's confidence in the interpreter, model, and solver elements. Past Performance is the autonomous system's confidence in its ability to complete the current task given how well it has done in previous experiences. Finally, Outcome Assessment is the autonomous system's confidence in achieving a good outcome. In this case, the autonomy does include its confidence in its ability to execute the commands, as this is already accounted for in the Model Validity factor. This is discussed in more detail in section 4.4.

The autonomous system's self-confidence is a function of these factors, as shown in Equations 4.1 and 4.2.

$$X_{sc} = [x_1, x_2, x_3, x_4, x_5, ...] \tag{4.1}$$

$$SC = f_{sc}(X_{sc}) \tag{4.2}$$

The next several sections will discuss each factor individually. Then Chapter 5 will go into detail about the implementation of the outcome assessment factor for this scenario.

4.3 Command Interpretation

Command Interpretation asks the question: "Are the user and the autonomy 'on the same page?'". Since the autonomy's ability to accomplish an assigned task depends on its ability to understand the user's intent, this factor is meant to convey the autonomy's confidence in its understanding of the user's intentions. This factor relates mainly to the certainty aspect of self-confidence. Since the autonomy measures its performance against its understanding of the user's intentions, it must also be certain that its understanding is correct. This means that even if the autonomy is completely confident in the goodness of its anticipated performance, its self-confidence will not be high unless it also believes it is using the correct performance measure.

While self-confidence as a whole mainly effects the user's Competence belief state, this factor may also heavily effect the Benevolence belief state, as that is the state that describes the user's perceived difference between the user's and the autonomy's intent. This factor will not change unless the autonomy can observe something about the user's intent. This will happen whenever the user gives the autonomy new instructions, but may also change if the user gives the autonomy an indication of the how the user thinks the autonomy is doing.

If the user is attempting to maximize a total expected reward (like the cumulative reward of the POMDP model), one way the autonomy may learn the user's intentions is through inverse reinforcement learning. Inverse reinforcement learning attempts to recover a reward function through observations of an agent's actions [21]. An autonomous system using this method to learn the user's intentions may report the statistics of this inference problem as the command interpretation factor. In this scenario, however, the autonomy is limited to a set of predefined reward functions that are meant to represent the user's intent. In this case, the autonomy may simply compare its outcome assessment to feedback that the user gives. For instance, if the autonomy thinks that it is consistently doing very well at a repeating task, but the user consistently rates the autonomy's performance as 'poor', it may lower its command interpretation.

4.4 Model Validity

Model Validity is meant to assess the goodness of the underlying models that it uses to make decisions. Since a decision is only as good as the information that led to it, model validity is another factor in self-confidence. Model validity focuses heavily on the uncertainties in the autonomy's model of the world, and thus the certainty aspect of self-confidence.

Model validity mainly affects the user's Competence belief state. Model validity changes any time the autonomy has opportunity to evaluate the models that it is using. This may come at the initialization, like acknowledging the affect of discretization on motion models. It may also change as the autonomy makes measurements about its environment. If measurements do not match the autonomy's model, then model validity may decrease. For instance, in the very simple case of a Kalman Filter, a measure of model validity might be the results of chi-square test for consistency of measurement innovations [22].

In this scenario, model validity is assessing the accuracy of the road network as a representation of the environment, the accuracy of the sensor models and observation probabilities, the accuracy of the chaser's motion model and corresponding transition probabilities, and the accuracy of the UGV's motion model and corresponding transition probabilities. In particular, all of the transition models require both a time and space discretization of the the continuous of the road network, which can lead to inaccuracy in the transition probabilities.

Normally certainty in execution might be a factor in self-confidence, but in this case it is not included in the self-confidence factors. This is because model validity already accounts for certainty in execution by considering the UGV's motion model. In other words, the MOMDP model accounts for uncertainty in the UGV's actions and allows the solver to incorporate that information in its decision making. Since the model and solver are already accounting for the fact that the execution maybe uncertain, the affect of uncertainty in execution is included in the model validity factor.

There are many different ways to approach generating model validity. Some, like the surprise index [23], measure model validity based on how 'surprised' the autonomy is when it gets unexpected observations (i.e. observations that are statistically unlikely to occur according to the autonomy's model). This is similar to the previously mentioned chi-square test for Kalman filters. However, these types of assessments require in-task calculations based on sensor measurements or other observations received online, and can not give an assessment before the task begins. Other metrics focus on the generation of the model, attempting to capture the fundamental flaws of models (e.g. linearized dynamics for nonlinear system, discretization of time/space, etc) [24, 25].

4.5 Solver Quality

Solver Quality assesses the autonomy's ability to make appropriate decisions based on the information it is given. In cases where the autonomy cannot solve the problem exactly, the autonomy must use approximations to find a solution. Solver quality asks if those approximations are appropriate and if the solver can be relied upon to generate good solutions. This is the 'uncertainty in its reasoning processes' part of self-confidence. In some cases, the autonomy may have too much information to handle effectively during time-constrained operations. In other cases, the autonomy's approximations may be wrong and cause the autonomy to make incorrect decisions.

Solver quality greatly effects the user's Competence belief state. If the autonomy doesn't think that is competent enough to solve the problem it has been given, then that should be reflected in the user's Competence belief. Generally, the solver quality assessment changes whenever the autonomous system reanalyses a situation. In some cases, like this scenario, the analysis all happens before the task begins, and so the solver quality factor does not change throughout the task. In other situations where the autonomy is constantly taking in new information and replanning, the solver quality factor will change much more quickly.

In this scenario, the autonomy's solver is the SARSOP algorithm. There are several ways the solver quality factor could possibly be computed. For example, SARSOP does not do well with very large or very complex state spaces. In fact, the state space size and connectivity between states is a good indicator of whether or not the solver will be able to converge to a good solution within a bounded time. For a problem with state space size n, SARSOP has to reason over a n-dimensional belief state (the *curse of dimensionality*), as well as an exponentially-growing set of reachable beliefs (the *curse of history*) [16]. While SARSOP reduces the curse of history by attempting to only consider optimally reachable beliefs in its backup and pruning operations, it is still an exponential problem.

Other metrics can be reported by the algorithm itself, for instance the SARSOP algorithm reports lower and upper bounds on the expected cumulative reward. These converge to an expected value as the solver converges towards the optimal policy, so a measure of solver quality might be based on the distance between the two bounds. In chapter 6, examples of the solver succeeding and failing are discussed in detail, as well as how convergence rate or state space size based confidence measures might react.

4.6 Outcome Assessment

Outcome Assessment evaluates how good the autonomy thinks the outcome of the task will be. Even if the autonomy perfectly understands and analyzes the user's problem, it may not be within the autonomous system's capabilities to achieve a good result. While very simple cases may simply be success/failure, most cases have some sort of cost or reward function that allows the autonomy to assess how well it expects to do. This is not just a measure of expected cumulative reward, rather it is an assessment of the goodness of the expected outcome, and the goodness of the distribution of possible outcomes in general.

This factor may be especially helpful to the user, as it allows the autonomy to tell the user what is and what is not within its capabilities. Without this factor, the autonomy is generally judged based on the task outcomes, and if it is a bad outcome the user's trust in the autonomy goes down. This is especially true during the user's and autonomy's first interactions; bad outcomes at the beginning negatively effect the user's trust significantly more than they do after the user has been using the autonomy for a longer period of time [3]. With this factor, however, the user may be able to see that the task was outside of the system's capabilities. This allows the user to maintain trust in the system, even though the system is not competent to perform the specific task.

In the very simple case of success or failure, the autonomy's outcome assessment may simply be the probability of success. However, most cases are not that simple, so other methods must be used to assess the goodness of the distribution of possible outcomes. For this scenario, the autonomy is able to generate a distribution $P(outcome)$ over the possible outcomes for the task. How the autonomy generates an outcome assessment from this distribution is discussed in more detail in chapter 5.

This factor is especially interesting when we consider what would happen to self-confidence if the autonomy is completely certain in its assessment. If the model is perfect, the solver is perfect, and the reward function is an exact representation of the user's intent, then the outcome distribution generated by the autonomy is the *true* outcome distribution- an exact representation of what will happen in reality. In this case, since the model is perfectly valid, the solver is exact, and the interpretation is perfect, the only self-confidence factors left are outcome assessment and past performance. Since past performance is meant to be a correction factor aligning the generated self-confidence with the autonomy's true capabilities and the generated outcome function *is* the autonomy's true capabilities, past performance won't factor in either. This is what happens to the autonomy's self-confidence in the limit of complete certainty as discussed in 3.3.3: all that matters is the goodness of anticipated performance, which is what outcome assessment is.

Since the outcome distribution is the true outcome distribution, the outcome assessment factor x_4 is the true outcome assessment factor x_4^{true}. As stated above, the other self-confidence factors do not matter to self-confidence, so the autonomy's true self-confidence SC^{true} is just a function f_{sc} of the true outcome assessment x_4^{true}, as shown in the left side of Equation 4.3.

$$f_{sc}(x_4^{true}) = SC^{true} \approx SC = f_{sc}(X_{sc}) \tag{4.3}$$

However, the assumption of complete certainty in assessment is expected to never be true. When it is not true, self-confidence SC will just approximate SC^{true}. This means that the self-confidence function f_{sc} of all of the factors X_{sc} should approximate SC^{true}. This gives a nice test to see how well different formulations of self-confidence and its factors actually work. This will be illustrated in more detail in chapter 5.

4.7 Past Performance

Past Performance is unique in this set of factors; it does not come from any of the autonomy's assessment of the current task. Rather, it asks how well the autonomy has done in similar circumstances. This allows the autonomy's past experience to lower its self-confidence if it hasn't done well in similar circumstances, and raise its self-confidence if tends to underestimate its performance. Alternatively, if the autonomy has never encountered a certain circumstance, past performance may act as a prior by assessing the similarity between the current circumstance and past circumstances. If the circumstance is completely dissimilar to previous ones, it may reduce the autonomy's self-confidence, simply because it has no experience.

Since the other self-confidence factors are not exact, past performance could act as a correction factor to more closely align the autonomous system's self-confidence with the true self-confidence, as in Equation 4.3. This requires the past performance factor to be able to assess the similarity between the current task and the previous tasks and to generate a correction factor based on the similarity as well as the actual outcome of previous tasks. One possible way similarity could be be assessed is to use traces collected during task execution to infer task invariants [26]. The system could then use the inferred invariants to assess the similarity of a task based on the presence or absence of violations of those invariants in the current task. It also could be possible to apply techniques from the reinforcement learning domain, where task similarity measures are used for task transfer problems [27].

Chapter 5

Outcome Assessment for Road Network Scenario

Now that the factors of self-confidence have been identified for this particular scenario, we can go into more detail into the actual creation and implementation of a self-confidence factor. In section 4.6, we defined outcome assessment to be the assessment of goodness of a distribution $P(outcome)$ over the anticipated possible outcomes that the autonomy could achieve. This outcome distribution is generated by the autonomy before the task begins and an example distribution is shown in Figure 5.1. This section uses the research presented in the previous chapters to formulate an outcome assessment factor on this distribution that can be reported to the user.

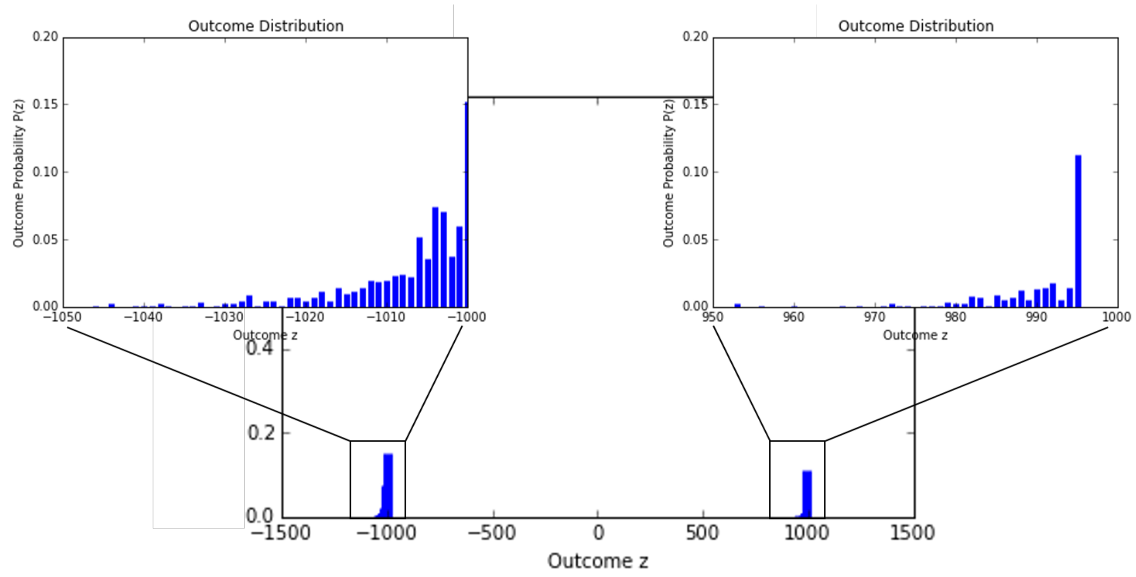


Figure 5.1: An Example Outcome Distribution

5.1 Design Insights

There are several key insights that go into the formulation of the outcome assessment factor. The first is that in the limit of complete certainty in assessment, the autonomy's self-confidence is just a function of the outcome assessment factor, as shown in Equation 4.3 and discussed in section 4.6. This means it should also follow all of the properties that we've assigned to self-confidence as a whole, ignoring the certainty aspect. For simplicity, we can let $SC = f_{sc}(x_4) = x_4$ by defining outcome assessment to on the same scale as the self-confidence assurance: $[-1, 1]$, where $x_4 = -1$ means 'absolutely bad outcomes' and $x_4 = 1$ means 'absolutely good outcomes'.

The next key insight is that this confidence factor is not just the expected utility of the distribution. While the expected utility certainly does give some information about the goodness of the distribution, is not all that is required. The goodness of a outcome distribution should be a reflection of how likely the autonomy is to achieve a good outcome vs a bad one. This will be discussed in depth after the outcome assessment equation has been introduced, and comparisons will be made between it and other metrics, like expected utility, variance, and entropy.

The final key insight is that the autonomy and the user may interpret the goodness of a outcome differently. The autonomy is a rational agent who attempts to maximize the expected utility. Behavioral economic theory shows that humans often evaluate probabilistic outcomes differently than a rational agent would, and so the autonomy's rational evaluation of the outcome distribution may not match the humans [28].

5.2 Outcome Assessment Formulation

Reference [29] proposes a model called Cumulative Prospect Theory (CPT) for describing how humans assess uncertain outcomes. There are three main differences that separate CPT from Expected Utility Theory, which is what the autonomy uses in its decision making (MOMDP's attempt to maximize the expected utility received). The first difference is that humans evaluate outcomes relative to a reference point, rather than zero. The second difference is that humans weight losses (values below the reference point) differently than they do gains (values above the reference point). A simple example of this is loss aversion, or the tendency for humans to weight losses higher than they weight gains. The final difference is that human's tend to overweight the probability of extreme outcomes, and underweight the probability of average outcomes. CPT combines these differences into a modified version of Expected Utility Theory given by Equation 5.1:

$$U(f) := \int_{-\infty}^{0} v(r)\frac{d}{dr}(w(F(r)))dr + \int_{0}^{+\infty} v(r)\frac{d}{dr}(-w(1-F(r)))dr \tag{5.1}$$

where f is the ordered distribution of all outcomes, z is an outcome (in this scenario a cumulative reward), $F(z)$ is the cumulative probability of all outcomes up to z, v is a value function, and w is a weighting function. The value function v maps outcomes to values, and the weighting function w maps cumulative probabilities to subjective cumulative probabilities. Example v and w are shown in Figures 5.2 and 5.3 If $v(z) = z$ for all z, and if $w(F(z)) = F(z)$ for all $F(z)$, then this simply reduces to E[$\mathcal{Z}$]

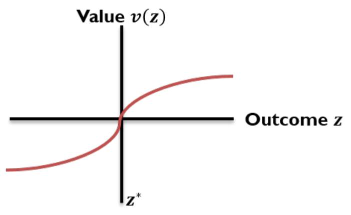


Figure 5.2: Typical Value Function

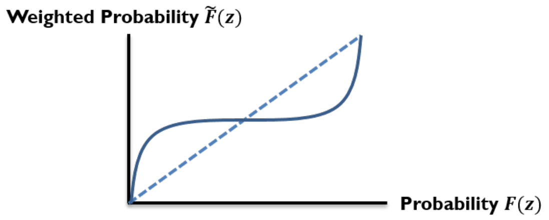


Figure 5.3: A Typical Weighting Function

This allows the autonomy to take into consideration the user's irrationality when it comes to interpreting uncertain outcomes. However, we already stated that we do not want to use the expected value. Instead, we will use the ratio between two parts of the function. The two parts we will compare are the upper and lower partial moments of the transformed outcome assessment function around z^* ($v(z^*) = 0$), which is given in Equation 5.2, where $\tilde{z} = v(z)$ and $\tilde{(}f)$ is the perceived probability distribution over $\tilde{z}$:

$$UPM/LPM = \frac{\int_0^{+\infty} \tilde{z}\tilde{f}(\tilde{z})d\tilde{z}}{\int_{-\infty}^{0} \tilde{z}\tilde{f}(\tilde{z})d\tilde{z}} \tag{5.2}$$

In short, this is the ratio of the probabilities of getting gains weighted by their magnitudes to the probability of getting losses weighted by their magnitudes. This ratio is inspired by Reference [30], which uses a similar measure to measure the goodness of an asset return distribution. At its core, this is capturing the ratio between good perceived outcomes and bad perceived outcomes relative to a reference value, which is the user's lowest good outcome. When the moments are balanced, this ratio is 1. It approaches 0 as the lower moment dominates the upper moment, and ∞ as the upper moment dominates the lower moment. To transform this into an easily interpretable scale, we use the logistic-log function given in Equation 5.3:

$$x_4 = 2/(1 + e^{(-k(log(UPM/LPM)))}) - 1 \tag{5.3}$$

This gives a final outcome assessment that maps outcome distributions to a number between -1 and 1, representing the goodness of the distribution. Figure 5.4 shows the curve when $k = 1$, where k is a parameter that controls the steepness of the curve around 0.

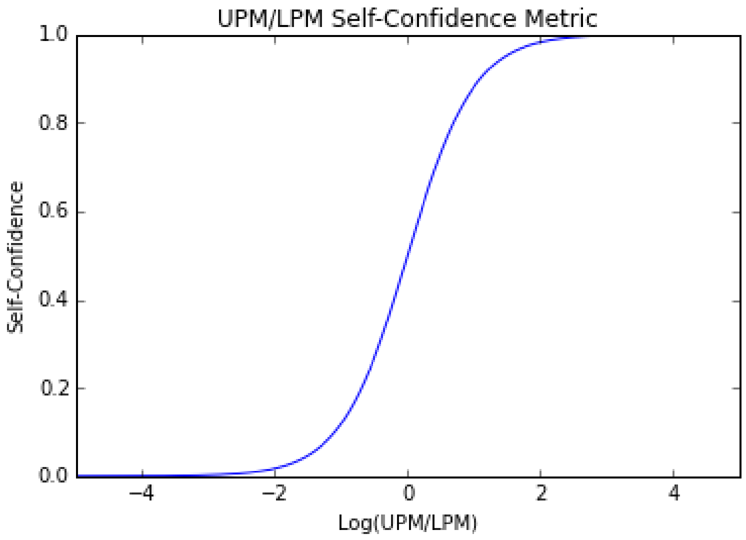


Figure 5.4: The Outcome Assessment Curve

The reasoning behind this metric is that it is important to know how likely it is to get an outcome above or below the reference value, as well as the relative difference between the outcome and the reference value. Alternative metrics such as the mean, variance, entropy, and skewness were also considered, but lacked many desirable properties. While the difference between the mean and the reference value does give indication of whether the expected outcome is good or bad, it does not give enough information about the distribution of the outcome, i.e. how much probability was above or below the reference value. Variance and entropy both give a good idea about how spread out the outcome distribution is, but they do not communicate the relative value of the outcomes compared to the reference value. Skewness and kurtosis both have similar problems because they are moments about the mean. The difference between the outcome assessment metric and other metrics are further shown in the examples in the next section.

5.3 Properties and Evaluation of Outcome Assessment

This metric has most of the properties of self-confidence that we previously asserted in sections 3.3.1 (ignoring the certainty aspects), and we will argue that is a good candidate to effect the user's Competence belief state. It is a single metric, reported just like self-confidence on a scale of -1 to 1, where -1 is 'completely bad', and 1 is 'completely good'. As the outcome distribution gets worse (losses become more probable), the metric approaches -1, and as gains become more probable the metric moves toward 1.

The following outcome distributions have been chosen to explore the properties of the outcome assessment metric. Each outcome distribution has all of the mass on one or two specific outcomes. This is because each partial moment in the metric can be equivocated to a specific outcome and probability which will give the same result. For simplicity, in each of these cases $z^* = 0$, $v(z) = z$, and $w(F(z)) = F(z)$ unless specified.

In these first two cases, both of the outcome distributions have a mean at $z = 10$. However, in Figure 5.5 $P(z > z^*) = 1$, which means all outcomes are gains. In Figure 5.6, $P(z > z^*) = \frac{1}{2}$, which means half of the outcomes are losses. In both cases, any metric based on the mean would

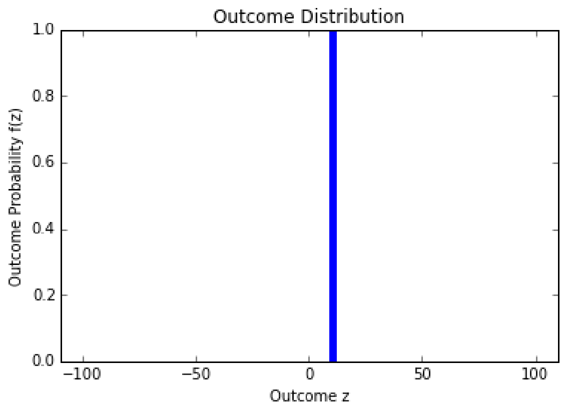


Figure 5.5: $P(z = 10) = 1$

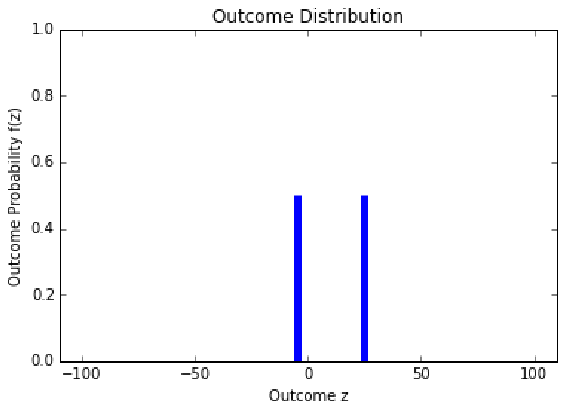


Figure 5.6: $P(z = -5, 25) = \frac{1}{2}$

have the same outcome assessment, but the UPM/LPM metric differentiates between the two. In the first case, the ratio is 10/0, and $x_4 = 1$, because there is no way for the autonomy to achieve a loss. In the second case, the ratio is $\frac{12.5}{2.5}$, which gives an outcome assessment of $x_4 = \frac{2}{3}$. Note that it is a still a pretty good outcome, because on average the outcome is a good outcome, but it is not as high because of the possibility of getting losses.

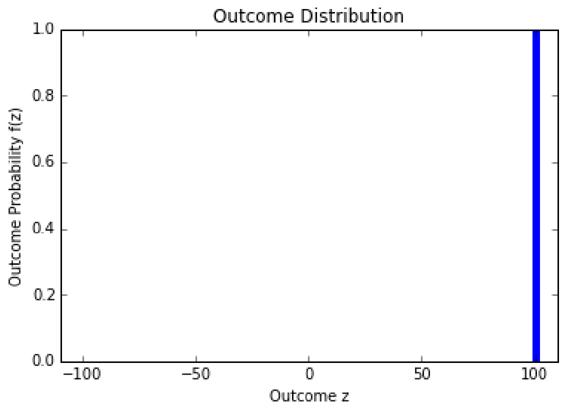


Figure 5.7: $P(z = 100) = 1$

In the next example, shown in Figure 5.7, the expected outcome is much higher than in Figure 5.5. In this case, reporting the mean of the function does show that the outcome distribution in Figure 5.7 is better than the outcome distribution in Figure 5.5. However, both of these have the same outcome assessment: $x_4 = 1$. While at first this might seem strange, it fits the definition of outcome assessment. In both cases the autonomy is perfectly confident in its ability to achieve a gain. Remember, self-confidence is simply meant to affect the user's trust in the system's capabilities, and in these cases the autonomous system is always going to achieve above the user's minimal good outcome (reference value).

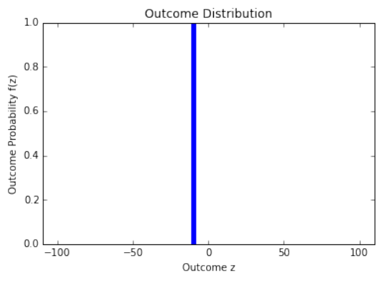


Figure 5.8: $P(z = -10) = 1$

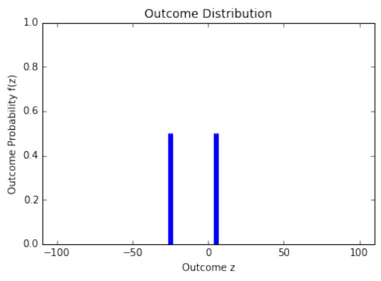


Figure 5.9: $P(z = -25, 5) = \frac{1}{2}$

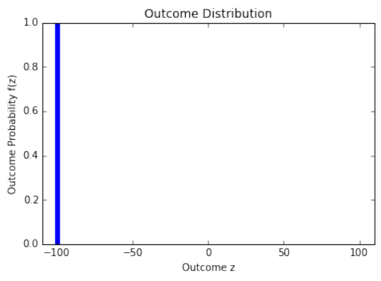


Figure 5.10: $P(z = -100) = 1$

The next three figures (Figures 5.8-5.10) are similar to the first three, except that the expected outcome is now negative. Similarly to above, the means are -10, -10, and -100, and the outcome assessment is $x_4 = -1$, $-\frac{2}{3}$, and -1. In the left and right outcome distributions, it is impossible to receive a gain, but in the middle outcome distribution it is possible to receive a good outcome, which raises the outcome assessment.

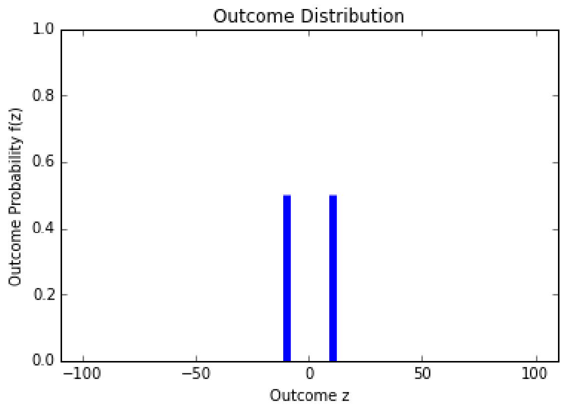


Figure 5.11: $P(z = -10, 10) = \frac{1}{2}$

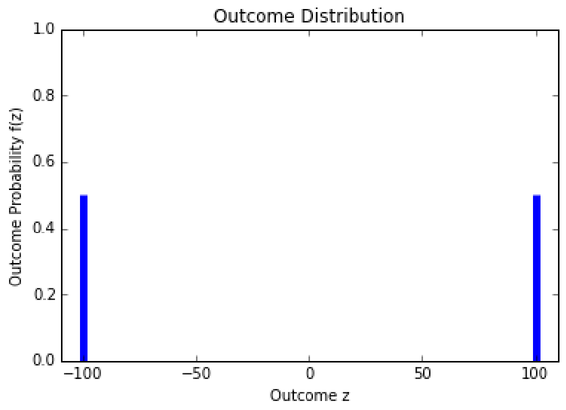


Figure 5.12: $P(z = -100, 100) = \frac{1}{2}$

In Figures 5.11 and 5.12, both outcome distributions have the same expected utility of zero. They also have the same outcome assessment, $x_4 = 0$. This is because the gains are as good as the losses are bad. In practice, these distributions often get interpreted differently than being exactly equal, depending on the value and weighting functions that the users use. For instance, if v is loss averse, both of these will result in negative outcome assessments.

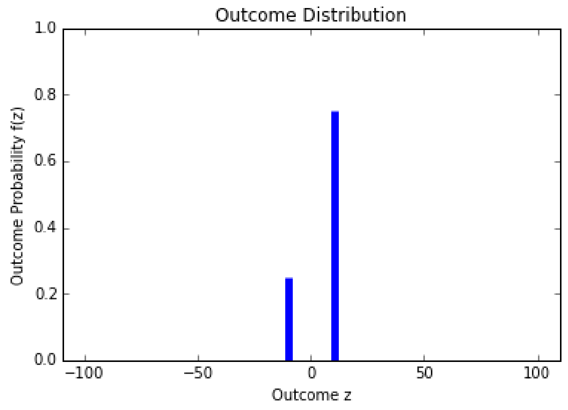


Figure 5.13: $P(z = -10) = \frac{1}{4},\ P(z = 10) = \frac{3}{4}$

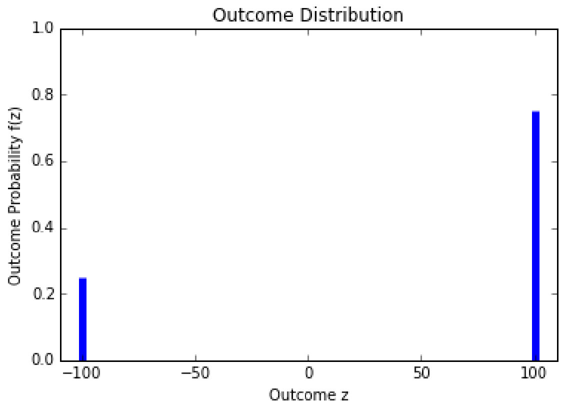


Figure 5.14: $P(z = -100) = \frac{1}{4},\ P(z = 100) = \frac{3}{4}$

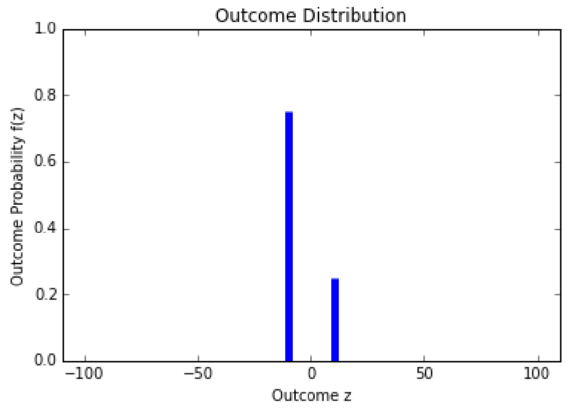


Figure 5.15: $P(z = -10) = \frac{3}{4},\ P(z = 10) = \frac{1}{4}$

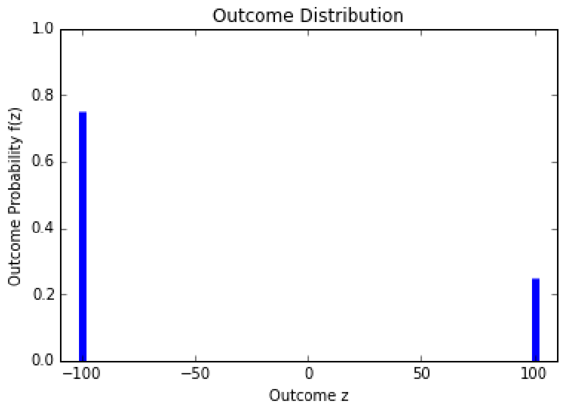


Figure 5.16: $P(z = -100) = \frac{3}{4},\ P(z = 100) = \frac{1}{4}$

In the last four figures, Figures 5.13-5.16, the outcome distributions are asymmetric. In the first two, the means are 5 and 50 and the outcome assessments are $x_4 = 0.5$ and 0.5, respectively. In the second two, the means are -5 and -50 and the outcome assessments still $x_4 = -0.5$ and 0.5, respectively. This is because self-confidence is simply measuring favorability of a distribution. In these cases, the outcomes are equally distant to the reference so the ratio of the probabilities is the only thing that matters. This is similar to the simple case of success/fail, and in fact in this case this formula reduces to $2P(good) - 1$, or more simply $P(good) - P(bad)$. Like above, however, these distributions get interpreted differently when including the user's value and weighting functions.

The last four cases also showcase an interesting property of the outcome assessment metric: it is just a relative metric. It doesn't matter if the value of the outcomes are -1 and 1 or -1000 and 1000, x_4 will still be the same. This is true for the other metrics that have been considered as well; like mean, entropy, variance, etc. If the user truly values two outcomes to be completely opposite (even after accounting for loss aversion and other preferences), outcome assessment will be the same, regardless of the magnitude of the values of the outcome. Since the user's value function has already been accounted for, it would be incorrect for the autonomy to have a lower goodness of outcome assessment in the $[-1000, 1000]$ outcome distribution than the $[-1, 1]$ based on the greater loss, because the greater loss is balanced by the greater gain. (Had the user's value function not been accounted for, loss aversion could say that losing an extra 999 is worse than gaining 999, so the autonomy's outcome assessment should lower. But that has already been accounted for with the value function.)

However, the autonomy could have different outcome assessments for those two distributions for other reasons. One reason could be the stability of the outcome assessment to a change in z^*. A shift of ± 1 of z^* in the first outcome distribution $[-1, 1]$ could change the outcome assessment to be either $x_4 = -1$ or $x_4 = 1$, because all of the outcomes would be losses or gains, respectively. But the same shift in z^* would barely affect x_4 for the outcome distribution $[-1000, 1000]$. Whether or not these two outcome distributions should have different outcome assessments is an open question, but for now we will go with the UPM/LPM metric, which leaves them to be the same.

Overall, this outcome assessment measure is measuring a outcome-adjusted ratio between the gains and the losses (good outcomes and bad outcomes), after accounting for human biases. With an outcome assessment of 1, the autonomy will only get good outcomes, and with outcome assessment of -1, the autonomy will only get bad outcomes. For outcome assessments in between, the autonomy weighs the probability of getting good outcomes vs bad outcomes weighted by their ‘goodness’ or ‘badness’- their relative distance to the reference value. It does not replace reporting the expected utility, in fact this measure is meant to compliment the reporting of the expected utility.

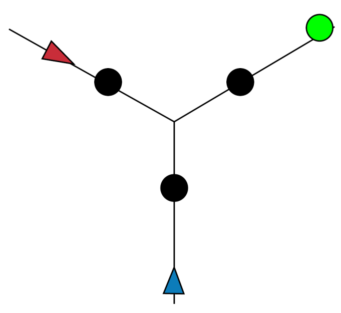
Chapter 6

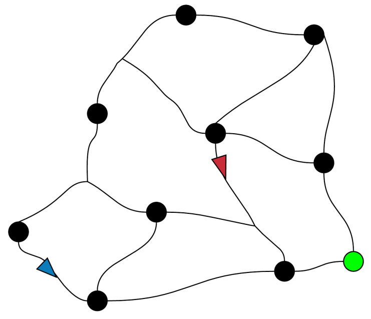
Examples Scenarios and Discussion

In order to see how self-confidence works in an actual scenario, we generate a set of example trial scenarios with different road networks and chaser behaviors. The road network is one of three shapes, shown in Figures 6.1a-6.1c. Each road network has either 0 or 4 states between each node, effectively increasing the lengths of the edges (travel time between adjacent nodes is the *not* same). The chaser is given one of two behaviors: a chasing mode where it always moves one state closer to the autonomy's location at every time step, and a random walk mode where it randomly chooses between adjacent states. These behaviors change the chaser's transition probabilities $P(y'|y)$ in the MOMDP model. For each trial, the solver is allowed 10 minutes to generate a policy for the autonomy to follow. The solver then simulates 1000 different trials with the chaser starting randomly in the road network and the UGV starting at a predefined point. Finally, the solver takes the results of this simulation to generate an outcome distribution for the trial.

Instead of going through all of the outcome distributions and the associated outcome assessments, we will look at four different outcome distributions that are representative of the types of distributions generated. For each of the four distributions, we discuss the factors that lead to the distribution, the result of the outcome assessment metric, and how the other the other factors should effect the autonomy's self-confidence as well. We will specifically discuss the solver quality metric, as tests with a real world simulation for model validity, and command interpretation are still in the works.

The first predicted outcome distribution shown in Figure 6.2 has mostly good outcomes, with


(a) The 'Y' Road Network


(b) The Irregular Road Network

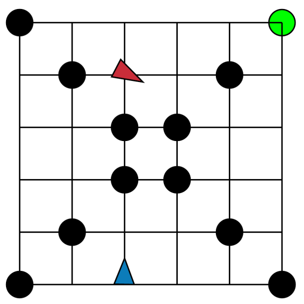

(c) The Grid Road Network

Figure 6.1: Example Road Networks: chaser in red, sensor in black, UGV in blue, and exit in green

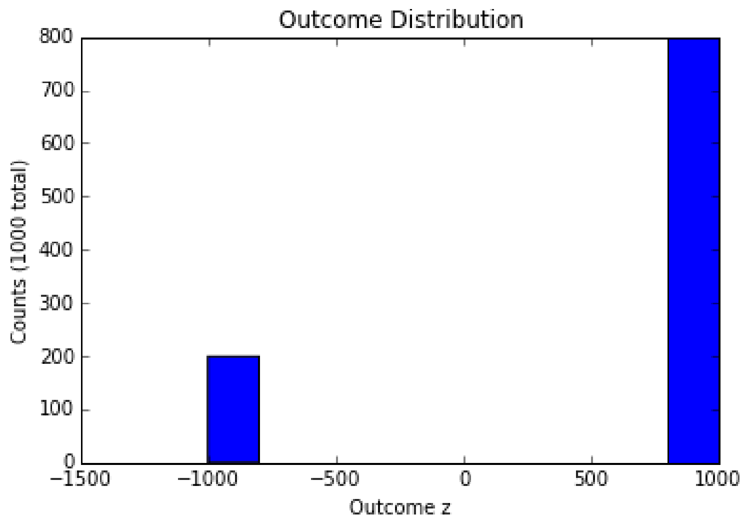


Figure 6.2: A mostly good outcome

the UGV being able to escape the road network. This generally happens in two circumstances. In the first, the chaser is moving randomly, allowing the UGV to avoid the chaser. In the second, the chaser is chasing, but the road network was large enough that the UGV can usually make it out. The actual output assessment metric is $x_4 = 0.606$. In these cases, the solver was able to converge to an expected cumulative reward, meaning the UGV is following a close to optimal strategy.

The second predicted outcome distribution shown in Figure 6.3 has mostly bad outcomes. This particular distribution has an outcome assessment of $x_4 = -0.54$. These mostly happen on

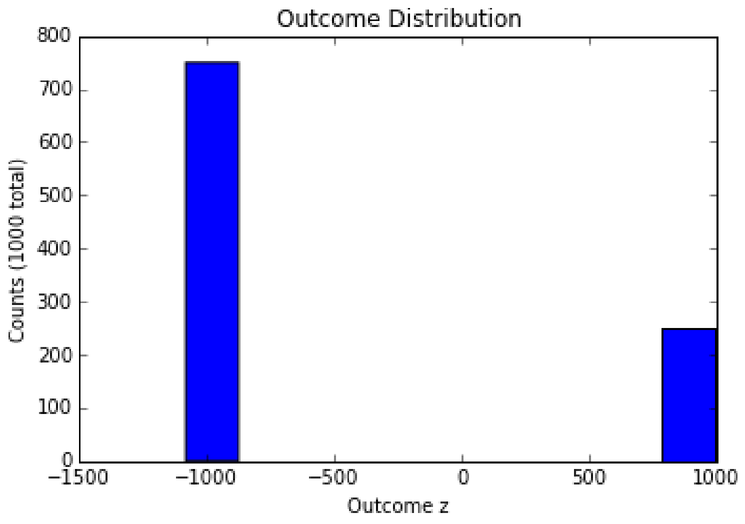


Figure 6.3: A mostly bad outcome

the ‘y’ road network when the chaser is moving randomly. This is because whether the UGV got out or not mainly depended on where the chaser started. When the chaser starts on either the UGV’s starting branch or the exit branch, the UGV usually ends up getting caught. However, if the chaser starts on the other branch, then the UGV usually escapes. As with the first outcome distribution, the solver was able to converge to an expected cumulative reward. This means that the autonomy’s policy is very close to the optimal policy in this scenario.

In the predicted outcome distribution shown in Figure 6.4, the UGV never escapes the road network. The only difference in outcome is how quickly the chaser catches the UGV. The outcome assessment for this was obviously $x_4 = -1$, showing that the autonomy is sure it would never get a good outcome. This only happened on the ‘y’ road network when the chaser was chasing the UGV. This was not unexpected, as no matter where the chaser starts, if it chases the UGV it is impossible for the UGV to escape. Like the previous two distributions, the solver was able to converge quickly to a solution under the 10 minute time limit.

In the final predicted outcome distribution, show in Figure 6.5, the autonomy anticipates mixed results. While the UGV rarely gets caught, it also never makes it to the exit. Again, this gives an outcome assessment of $x_4 = -1$, but the majority of the outcomes come from the UGV

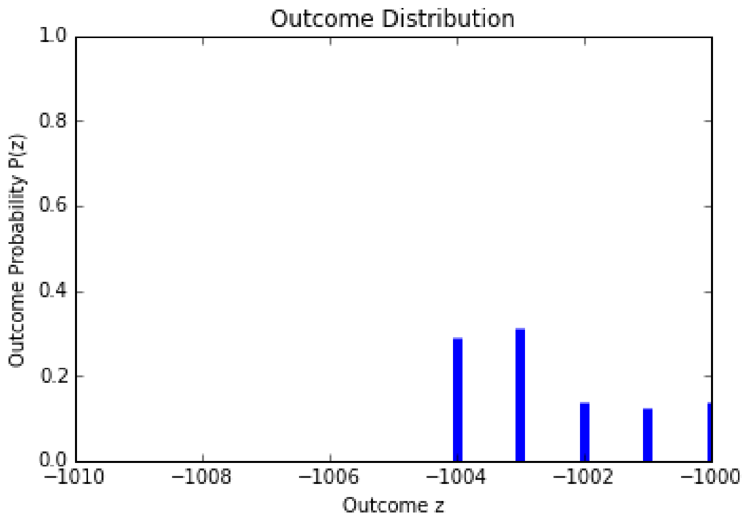


Figure 6.4: A completely bad outcome

simply wandering around for 200 steps, which was the maximum simulation time. By that time, the autonomy would have had plenty of time to both avoid the chaser and to make it to the exit, so it implies that the autonomy was not actually moving towards the exit. The reason for this unusual behavior is due to the nature of the SARSOP algorithm. As mentioned before, the SARSOP algorithm attempts to approximate the optimal policy by exploring only the reachable belief space, or the set of possible beliefs that could be achieved from a specific initial belief under the optimal policy. However, this sampling algorithm never reached a belief where the UGV reached the exit.

This is due to two factors: the state space size/distance to the exit from the UGV's starting position, and the fact that in every case, the chaser was moving randomly. Because the state space was so large, the distance between the UGV's starting position and the exit was also very large. The sampling algorithm never explored far enough to realize that there was an exit that could provide a large positive reward. Since the chaser was not chasing the UGV, the UGV also had very little motivation to move away from its starting position. Even though the output assessment was completely negative, in this case the solver should not have been confident that it reached the optimal policy (and it did not).

In these scenarios, just knowing the state space size was not enough to tell if the solver would

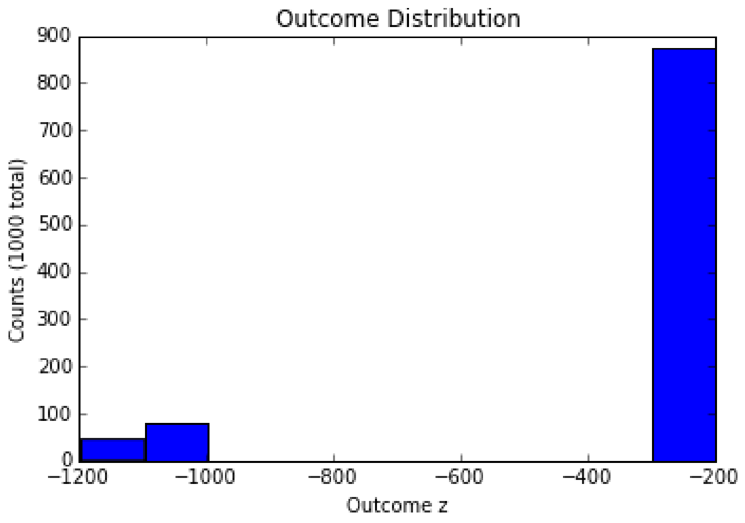


Figure 6.5: A completely bad outcome as a result of a poor solution returned by the solver

converge. Convergence also depended on the chaser's behavior. Although this can be remedied by having the solver consider all possible starting positions for the UGV instead of the designated one, it does show how assumptions in the solver can actually effect the efficacy of the solver metric.

In these scenarios we do not consider how the factors of command interpretation, model validity, or past performance could be generated or effected. This is because the real world implementation of this scenario has not been implemented yet.

Chapter 7

Conclusions and Future Work

7.1 Future Work

The work is meant to guide the development of self-confidence for autonomous systems. There is still much work that needs to be done in terms of both development and assessment. Some areas of future research include testing current metrics with human trials to see if they actually do influence trust in the way they are designed to, generating other factors like command interpretation, and expanding self-confidence to other types of autonomy.

7.1.1 Application Studies with Human Users

The core goal of self-confidence is to effect the user's trust in a way that calibrates it to the systems actual capabilities. While self-confidence has been designed with this goal in mind, human trials are still necessary to see how it will actually effect a user's trust. Trust is a very complicated subject, involving both rationality and emotions, and self-confidence is not expected to be the end-all solution for managing the user's trust. It is, however, expected to make a measurable impact on the user's trust and their overall interaction with autonomy.

There are two main methods for understanding a user's trust. The first uses qualitative surveys, and the second uses observations of user's actions. For the first method, user's are asked to report feelings of trust and beliefs about the subject. References [31, 32, 33, 34] are examples of studies measuring trust using different survey techniques. In particular, Reference [35] provides a well-researched questionnaire consisting of twelve Likert Scale questions for evaluating the trust between people and autonomous systems.

The second method focuses on measuring user behavior to observe the user's trust. Trust actions are a specific set of actions or behaviors that can be used to observe trust. References [5, 36, 37] provide examples of studies that use observations about the user's actions to indicate trust. Both of these methods require careful experimental setup so that the effect of trust can be differentiated from other factors that influence the user's reliance on the autonomy.

7.1.2 Self-Confidence for Other Systems

This work mainly focuses on the development of the outcome assessment factor of self-confidence. While some ideas for the other self-confidence factors have been proposed, there is still work done in developing the actual metrics. Command interpretation is a challenge, due to the variety of possible user interactions as well as the opaqueness of humans in general, but understanding the alignment of the user's and the autonomy's intent is important. Model validity has many developed techniques to draw from, but translating those metrics (like the previously mentioned chi-square test and surprise index) into a confidence factor remains to be done. Solver quality is very specific to the algorithms used, and can be hard to assess (for instance, how does the autonomy recognize convergence to a local minimum?). Output assessment is also dependent on the algorithm and its ability to predict possible outcomes (for instance, what does a non-probabilistic planner report?), and like command interpretation is very dependent on the users intentions and interpretations. Finally, past performance requires an understanding of similarity in previous circumstances, as well as a way of translating previous outcomes and their similarity to the current task into a confidence metric of the current task.

It will be important to explore how these metrics can be used between systems so that designers do not have to create autonomous system specific algorithms. This means all of the previously mentioned challenges will need to be approached from both an autonomy-specific and general perspective. Similarly, how self-confidence can be applied to other autonomous system architectures is another important idea to explore. For instance, how would self-confidence work in behavioral/reactive autonomous systems? Can it even be applied to them? How would self-confidence work in hierarchical systems, or in a swarm of agents? For instance, if one autonomous agent controls a group of agents who report their individual self-confidence to the leader, how would the leader report it's self-confidence to the user? These are all open questions that should be addressed, so that self-confidence is not misapplied to systems where it does not make sense.

7.1.3 Reporting

Another unaddressed issue is the problem of communicating self-confidence to the user. There are many ways to present information to humans, and the way it is presented can make a large difference in how it is interpreted. Framing effects and other cognitive biases can affect how a user interprets a self-confidence assurance, which will change how the assurance affects the users trust [38]. There are several related questions that should be explored.

The first is how self-confidence should be reported; whether it is just a number, an indicator bar, a Chernoff face, or something else will effect how the user interprets it. Secondly, how does the presence of other assurances or communications effect the user's interpretation of self-confidence? For instance, does it matter if it is reported alongside an expected utility, or some representation of the autonomy's plan?

7.2 Conclusion

The main contributions of this work are the design and evaluation of a framework for computing the self-confidence assurance and the specific design, implementation, and analysis of the outcome assessment factor of self-confidence. Self-confidence is designed to elicit appropriate trust from the user by providing the user with the autonomous system's assessment of its own capabilities.

In this thesis we have factored the computation into several different parts, corresponding to the elements of the autonomous system. These factors attempt to capture the accuracy of the autonomous system's assessment of the task as well as its understanding of its expected performance. While we show how self-confidence can be computed from the outcome assessment factor if the autonomy is completely certain in its assessment, how self-confidence is generated in general from the individual factors remains an open question.

One of the difficulties of computing self-confidence is that the user is the one who determines the goals and evaluates the outcomes of the task. In order for the autonomy to compute its self-confidence in its ability to complete a task, it must understand the users intentions as well as the user's assessment of different outcomes. This is evident in the command interpretation, outcome assessment, and past performance factors discussed in this thesis. Since there are many modes of interaction between users and autonomous systems, this is often problem specific and requires careful thought to capture the differences between the autonomy's and user's communication and understanding of the problem.

Another difficulty in computing self-confidence is that it asks the autonomous system to recognize where it may fail. Because this answer is generated from the autonomous system's own reasoning, it may fail to accurately assess its abilities and under- or over-report its self-confidence. Even though self-confidence will not be a perfect understanding of the system's capabilities, it should still allow the user to calibrate their trust in the autonomous system more quickly than without self-confidence.

Overall, this framework provides a definition of an autonomous system's self-confidence and a method for computing it from the evaluation of different elements of the system. While there are still many open questions to address, this work provides a deeper understanding of the issues in computing self-confidence, and should assist further research to improve the reporting of the self-confidence assurance.

Bibliography

[1] Committee on Autonomy Research for Civil Aviation; Aeronautics, Space Engineering Board; Division on Engineering, and Physical Sciences; National Research Council. Autonomy Research for Civil Aviation: Toward a New Era of Flight. The National Academies Press, 2014.

[2] Austin Lillard, Eric Frew, Brian Argrow, and Dale Lawrence. Assurances for enhanced trust in autonomous systems. In 2015 AAAI Fall Symposium Series, 2015.

[3] John D Lee and Katrina A See. Trust in automation: designing for appropriate reliance. Human factors, 46(1):50–80, 1 2004.

[4] Raja Parasuraman and Victor Riley. Humans and automation: Use, misuse, disuse, abuse. Human Factors: The Journal of the Human Factors and Ergonomics Society, 39(2):230–253, 1997.

[5] Paul Robinette, Wenchen Li, Robert Allen, Ayanna M Howard, and Alan R Wagner. Overtrust of robots in emergency evacuation scenarios. In 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 101–108. IEEE, 2016.

[6] Poornima Kaniarasu, Aaron Steinfeld, Munjal Desai, and Holly Yanco. Robot confidence and trust alignment. In ACM/IEEE International Conference on Human-Robot Interaction, pages 155–156, 2013.

[7] Yue Wang, Zhenwu Shi, Chuanfeng Wang, and Fumin Zhang. Human-robot mutual trust in (semi) autonomous underwater robots. In Cooperative Robots and Sensor Networks 2014, pages 115–137. Springer, 2014.

[8] Andrew R Hutchins, ML Cummings, Mark Draper, and Thomas Hughes. Representing autonomous systems self-confidence through competency boundaries. In Proceedings of the Human Factors and Ergonomics Society Annual Meeting, volume 59, pages 279–283. SAGE Publications, 2015.

[9] Ugur Kuter and Chris Miller. Computational mechanisms to support reporting of self confidence of automated/autonomous systems. In 2015 AAAI Fall Symposium Series, 2015.

[10] Krishnanand N Kaipa, Akshaya S Kankanhalli-Nagendra, and Satyandra K Gupta. Toward estimating task execution confidence for robotic bin-picking applications. In 2015 AAAI Fall Symposium Series, 2015.

[11] Nicholas Sweet, Nisar R Ahmed, Ugur Kuter, and Christopher Miller. Towards self-confidence in autonomous systems. In AIAA Infotech@ Aerospace, page 1651. 2016.

[12] Nisar Ahmed, David Casbeer, Yongcan Cao, and Derek Kingston. Bayesian hidden markov models for uav-enabled target localization on road networks with soft-hard data. In SPIE Defense+ Security, pages 94640Q–94640Q. International Society for Optics and Photonics, 2015.

[13] Sylvie CW Ong Shao Wei Png and David Hsu Wee Sun Lee. Pomdps for robotic tasks with mixed observability. 2009.

[14] Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. Planning and acting in partially observable stochastic domains. Artificial intelligence, 101(1):99–134, 1998.

[15] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic robotics. MIT press, 2005.

[16] David Silver and Joel Veness. Monte-carlo planning in large pomdps. In Advances in neural information processing systems, pages 2164–2172, 2010.

[17] Nicholas Roy and Geoffrey J Gordon. Exponential family pca for belief compression in pomdps. In Advances in Neural Information Processing Systems, pages 1635–1642, 2002.

[18] Hanna Kurniawati, David Hsu, and Wee Sun Lee. Sarsop: Efficient point-based pomdp planning by approximating optimally reachable belief spaces. In Robotics: Science and Systems, volume 2008. Zurich, Switzerland, 2008.

[19] Gareth R Jones and Jennifer M George. The experience and evolution of trust: Implications for cooperation and teamwork. Academy of management review, 23(3):531–546, 1998.

[20] RRE. Clearpath husky, 2014. [Online; accessed July 22, 2016].

[21] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In Proceedings of the twenty-first international conference on Machine learning, page 1. ACM, 2004.

[22] Y Bar-Shalom, X Li, and T.Kirubarajan. Estimation with Applications to Navigation and Tracking. Wiley, New York, 2001.

[23] Adam Zagorecki, Marcin Kozniewski, and Marek Druzdzel. An approximation of surprise index as a measure of confidence. In 2015 AAAI Fall Symposium Series, 2015.

[24] Mahdi Milani Fard, Joelle Pineau, and Peng Sun. A variance analysis for pomdp policy evaluation. In AAAI, pages 1056–1061, 2008.

[25] Stephane Ross, Masoumeh T Izadi, Mark Mercer, and David L Buckeridge. Sensitivity analysis of pomdp value functions. In ICMLA, pages 317–323, 2009.

[26] Hengle Jiang, Sebastian Elbaum, and Carrick Detweiler. Reducing failure rates of robotic systems though inferred invariants monitoring. In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems, pages 1899–1906. IEEE, 2013.

[27] James L Carroll and Kevin Seppi. Task similarity measures for transfer in reinforcement learning task libraries. In Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005., volume 2, pages 803–808. IEEE, 2005.

[28] Daniel Kahneman and Amos Tversky. Prospect theory: An analysis of decision under risk. Econometrica: Journal of the econometric society, pages 263–291, 1979.

[29] Amos Tversky and Daniel Kahneman. Advances in prospect theory: Cumulative representation of uncertainty. Journal of Risk and uncertainty, 5(4):297–323, 1992.

[30] Simone Farinelli and Luisa Tibiletti. Sharpe thinking with asymmetrical preferences. Technical report, Technical Report presented at European Bond Commission, 2003.

[31] Eui Park, Quaneisha Jenkins, and Xiaochun Jiang. Measuring trust of human operators in new generation rescue robots. In Proceedings of the JFPS International Symposium on Fluid power, volume 2008, pages 489–492. The Japan Fluid Power System Society, 2008.

[32] Poornima Kaniarasu, Aaron Steinfeld, Munjal Desai, and Holly Yanco. Potential measures for detecting trust changes. In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pages 241–242. ACM, 2012.

[33] Deborah R Billings, Kristin E Schaefer, Jessie YC Chen, and Peter A Hancock. Human-robot interaction: developing trust in robots. In Proceedings of the seventh annual ACM/IEEE international conference on Human-Robot Interaction, pages 109–110. ACM, 2012.

[34] Clifford Nass and Youngme Moon. Machines and mindlessness: Social responses to computers. Journal of social issues, 56(1):81–103, 2000.

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

[36] Raja Parasuraman, Scott Galster, Peter Squire, Hiroshi Furukawa, and Christopher Miller. A flexible delegation-type interface enhances system performance in human supervision of multiple robots: Empirical studies with roboflag. IEEE Transactions on systems, man, and cybernetics-part A: Systems and Humans, 35(4):481–493, 2005.

[37] Terrence Fong, Charles Thorpe, and Charles Baur. Robot, asker of questions. Robotics and Autonomous systems, 42(3):235–243, 2003.

[38] Amos Tversky and Daniel Kahneman. The framing of decisions and the psychology of choice. In Environmental Impact Assessment, Technology Assessment, and Risk Analysis, pages 107–129. Springer, 1985.

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

[40] Adelardo AD Medeiros. A survey of control architectures for autonomous mobile robots. Journal of the Brazilian Computer Society, 4(3), 1998.