Improving Competence for Reliable Autonomy

Connor Basich
University of Massachusetts
Amherst, Massachusetts, USA
cbasich@cs.umass.edu

Justin Svegliato
University of Massachusetts
Amherst, Massachusetts, USA
jsvegliato@cs.umass.edu

Shlomo Zilberstein
University of Massachusetts
Amherst, Massachusetts, USA
shlomo@cs.umass.edu

Kyle Hollins Wray
Alliance Innovation Lab Silicon Valley
Santa Clara, California, USA
kyle.wray@nissan-usa.com

Stefan J. Witwicki
Alliance Innovation Lab Silicon Valley
Santa Clara, California, USA
stefan.witwicki@nissan-usa.com

Given the complexity of real-world, unstructured domains, it is often impossible or impractical to design models that include every feature needed to handle all possible scenarios that an autonomous system may encounter. For an autonomous system to be reliable in such domains, it should have the ability to improve its competence online. In this paper, we propose a method for improving the competence of a system over the course of its deployment. We specifically focus on a class of semi-autonomous systems known as *competence-aware systems* that model their own competence—the optimal extent of autonomy to use in any given situation—and learn this competence over time from feedback received through interactions with a human authority. Our method exploits such feedback to identify important state features missing from the system's initial model, and incorporates them into its state representation. The result is an agent that better predicts human involvement, leading to improvements in its competence and reliability, and as a result, its overall performance.

1 Introduction

Recent advances in artificial intelligence and robotics have enabled the deployment of autonomous systems in domains of increasing complexity and over long durations. Examples include autonomous navigation [8, 17, 9], extraterrestrial exploration [25, 20], and personal assistance [24, 21]. However, despite substantial progress in these areas, it is still often either impossible or impractical to design models with every feature needed to handle all possible scenarios that may be encountered by an autonomous system [37]. This can be due to the complexity of the state space, a lack of information, or unknown personal needs and preferences of stakeholders. As a result, many autonomous systems still rely on human assistance in various capacities to successfully accomplish their tasks. Ideally, to diminish their reliance on a human operator in unanticipated situations, an autonomous system should have the ability to identify and introduce missing features to their models during deployment, increasing the extent of their autonomous operation.

Competence-aware systems (CAS) have been recently proposed as a planning framework to reduce unnecessary reliance on human assistance [6]. However, these systems operate on a fixed model, making it hard to successfully operate in situations where the system lacks key features in its state representation. In general, a CAS is a semi-autonomous system [45] that operates in, and plans for, different levels of autonomy, each of which corresponds to unique constraints on its capabilities and unique interactions with a human authority [6]. For example, a service robot may request that the human operator supervise or even take control when an unrecognized obstacle blocks its way. Reliance on a human operator in these situation stems from limited competence on the part of the autonomous system. Competence-aware systems (CAS) were developed to provide a formal model for enabling the system to learn its

Submitted to:
ECAI AREA Workshop 2020

© C. Basich, J. Svegliato, K. Wray, S. Witwicki, and S. Zilberstein

competence, optimize its autonomy given its learned competence, and use this knowledge to improve the efficiency and reliability of its decision making while reducing unnecessary reliance on humans.

While the CAS model enables a semi-autonomous system to optimize its autonomy over time, it is still limited by the features in its fixed model. For example, consider a robot that is deployed on a campus with the task of delivering packages to various offices in different buildings. As a baseline, the robot can detect crosswalks and knows that it must ask for approval or supervision before crossing them. However, the robot only uses the number of vehicles in close proximity to reason about any crosswalk, while the human authority who approves or supervises the robot may use more detailed features, such as the visibility at the crosswalk, the time of day, or the weather conditions. Since the CAS model does not have the ability to add features to its state representation, the human feedback may appear inconsistent to the robot and lead to low competence, poor performance, and extra burden on the human.

In this paper, we propose a method for providing competence-aware systems the ability to improve their competence over time by increasing the granularity of the state representation through online model updates, leading to a more nuanced drawing of the boundaries between regions with different levels of competence. To do this, we leverage existing human feedback acquired through the agent's interactions with the human authority and identify instances where the feedback is inconsistent and appears random. As the system expects human feedback to be consistent up to small noise, instances where this expectation is not met are situations in which the agent's model is likely lacking some feature(s) that the human's feedback depends on. By adding these features to the model, the system can improve its predictive capabilities of the human's involvement, better modeling the value of operating in each level of autonomy. This leads to an increase in the system's competence and, consequently, the system's overall performance.

To validate our approach, we test both a standard CAS without the ability to modify its feature space and a modified CAS with the ability to modify its feature space, on a simulated domain where a robotic agent is tasked with delivering packages to various locations throughout the map. To complete its task, the agent must navigate multiple obstacles that it initially represents with only a few of the features used by the human authority. We show that the modified CAS correctly identifies missing causal features used by the human authority and adds them to its feature space, leading to a higher competence, a more accurate model, and a lower cost of operation compared to the standard CAS.

2 Background on Competence Awareness

We begin by reviewing the primary model used in this approach. A competence-aware system (CAS) is a semi-autonomous system [45] that operates in and plans for multiple levels of autonomy, each of which is associated with different restrictions on autonomous operation and distinct forms of human involvement [6]. A CAS combines three different models—a *domain model*, an *autonomy model*, and a *human feedback model*—into a single decision making framework.

2.1 Domain Model

The *domain model* (DM) models the environment in which the agent operates as a stochastic shortest path (SSP) problem. An SSP is a general-purpose model for sequential decision making in stochastic domains with an objective of finding the least-cost path from a start state to a goal state. This model has been used in a wide range of applications, including exception recovery [37], electric vehicle charging [32], search and rescue [27], and autonomous navigation [45].

Formally, an SSP is represented by the tuple $\langle S, A, T, C, s_0, s_g \rangle$, where S is a finite set of states, A is a finite set of actions, $T : S \times A \times S \rightarrow [0,1]$ represents the probability of reaching state $s' \in S$ after performing action $a \in A$ in state $s \in S$, $C : S \times A \rightarrow \mathbb{R}^+$ represents the expected immediate cost of performing action $a \in A$ in state $s \in S$, s_0 is a start state, and s_g is a goal state such that $\forall a \in A, T(s_g, a, s_g) = 1 \wedge C(s_g, a) = 0$.

A solution to an SSP is a policy $\pi : S \rightarrow A$ that indicates that action $\pi(s) \in A$ should be taken in state $s \in S$. A policy π induces the value function $V^\pi : S \rightarrow \mathbb{R}$ that represents the expected cumulative cost $V^\pi(s)$ of reaching the goal state s_g from state s following policy π. An optimal policy π^* minimizes the expected cost of reaching the goal from the start state, $V^*(s_0)$.

2.2 Autonomy Model

The *autonomy model* (AM) models the extent of autonomous operation the system can perform, i.e. the forms of operation and the external constraints imposed on when each form is allowed. *AM* is represented by the tuple $\langle \mathscr{L}, \kappa, \mu \rangle$. $\mathscr{L} = \{l_0, ..., l_n\}$ is the set of *levels of autonomy*, where each l_i corresponds to a set of constraints on the system's autonomous operation. Although not required, each level of autonomy generally involves a form of human involvement that reflects, or compensates for, the constraints imposed on the autonomy. For instance, it is common in autonomous service robots to have some level of *supervised autonomy* which allows for autonomous operation conditioned on the requirement that a human is ready and available to monitor the system and override it if they deem an action to be unsafe or undesirable. $\kappa : S \times A \rightarrow \mathscr{P}(\mathscr{L})$ is the *autonomy profile* that indicates the allowed levels of autonomy when performing action $a \in A$ in state $s \in S$. κ may model external constrains that could represent legal or ethical considerations. κ constrains the full policy space so that the system can never follow a policy that violates κ. $\mu : S \times \mathscr{L} \times A \times \mathscr{L} \rightarrow \mathbb{R}$ is the *cost of autonomy* that represents the cost of performing action $a \in A$ at level $l' \in \mathscr{L}$ in state $s \in S$ having just operated in level $l \in \mathscr{L}$.

2.3 Human Feedback Model

The *human feedback model* (HM) models the autonomous agent's current knowledge and belief about its interactions with the human agent. *HM* is represented by the tuple $\langle \Sigma, \lambda, \rho, \tau \rangle$, where $\Sigma = \{\sigma_0, ..., \sigma_n\}$ is the set of possible feedback signals the agent can receive from the human, $\lambda : S \times \mathscr{L} \times A \times \mathscr{L} \rightarrow \Delta^{|\Sigma|}$ is the *feedback profile* that represents the probability of receiving signal σ when performing action $a \in A$ at level $l' \in \mathscr{L}$ given that the agent is in state $s \in S$ and just operated in level $l \in \mathscr{L}$, $\rho : S \times \mathscr{L} \times A \times \mathscr{L} \rightarrow \mathbb{R}^+$ is the *human cost function* that represents the cost to the human of performing action $a \in A$ at level $l' \in \mathscr{L}$ given that the agent is in state $s \in S$ and just operated in level $l \in \mathscr{L}$, and $\tau : S \times A \rightarrow \Delta^{|S|}$ is the *human state transition function* that represents the probability of the human taking the agent to state $s' \in S$ when the agent attempted to perform action $a \in A$ in state $s \in S$ but the human took over control. In general, the human's true feedback profile, $\lambda^{\mathscr{H}}$, and transition function, $\tau^{\mathscr{H}}$, are likely unknown to the agent, which instead operates on some, possibly data-driven, estimate of each.

2.4 Competence-Aware Systems

Definition of a CAS

A competence-aware system combines the three models defined above into one planning model by augmenting the base domain model with information from the autonomy model and the human feedback model. Solving a CAS then produces a policy that exploits its information about the human feedback to

best utilize the levels of autonomy that are allowed by its autonomy profile. Formally, we define a CAS as the following augmented SSP:

Definition 1. *A **competence-aware system** $\mathscr{S}$ is represented by the tuple $\langle \overline{S}, \overline{A}, \overline{T}, \overline{C}, \overline{s}_0, \overline{s}_g \rangle$, where:*

- $\overline{S} = S \times \mathscr{L}$ *is a set of factored states, with each defined by a domain state and a level of autonomy.*
- $\overline{A} = A \times \mathscr{L}$ *is a set of factored actions, with each defined by a domain action and a level of autonomy.*
- $\overline{T} : \overline{S} \times \overline{A} \to \Delta^{|\overline{S}|}$ *is a transition function comprised of* $T : S \times A \to \Delta^{|S|}$, $\lambda : \overline{S} \times \overline{A} \to \Delta^{|\Sigma|}$, *and* $\tau : S \times A \to \Delta^{|S|}$.
- $\overline{C} : \overline{S} \times \overline{A} \to \mathbb{R}^+$ *is a cost function comprised of* $C : S \times A \to \mathbb{R}^+$, $\mu : \overline{S} \times \overline{A} \to \mathbb{R}$, *and* $\rho : \overline{S} \times \overline{A} \to \mathbb{R}^+$.
- $\overline{s}_0 \in \overline{S}$ *is the initial state* $\overline{s}_0 = \langle s_0, l \rangle$ *for some* $l \in \mathscr{L}$.
- $\overline{s}_g \in \overline{S}$ *is the goal state* $\overline{s}_g = \langle s_g, l \rangle$ *for some* $l \in \mathscr{L}$.

The objective of a CAS is to find an optimal policy $\pi^*_\kappa \in \Pi$ that minimizes the value function V^π subject to the condition that, for every state $\overline{s} = (s, l') \in \overline{S}$, the policy $\pi(\overline{s})$ *never* indicates an action $\overline{a} = (a, l) \in A$ for which the level of autonomy l is *not* allowed for state s and action a by $\kappa(s, a)$.

Example of a CAS

To illustrate the role of the various functions, we describe the CAS model used in our experiments in Section 4, which is the same CAS defined in Section 4.2 of [6].

The CAS can operate in one of the four levels, $\mathscr{L} = \{l_0, l_1, l_2, l_3\}$, corresponding to (1) no autonomy, (2) verified autonomy, (3) supervised autonomy, and (4) unsupervised autonomy. Level 1 requires a human to complete the action for the agent, either through direct control, tele-operation, or simply manually performing the action themselves. Level 2 requires the agent to query for explicit approval from the human prior to executing the action. Level 3 requires a human to be available in a supervisory capacity while the agent executes the action, with the ability to override and take control of the system if deemed necessary. Level 4 requires no human involvement and is performed fully autonomously.

The system can receive the the following four feedback signals: approval ($\oplus$), disapproval ($\ominus$), override ($\oslash$), and no signal ($\emptyset$). We furthermore assume that approval and disapproval can only be received in l_1, and the system always receives a feedback signal in l_1, and can only receive $\oslash$ in l_2.

We can now specify the **state transition function** of this CAS. Given $\overline{s}, \overline{a}$, and $\overline{s}'$, we define $\overline{T}$ as follows:

$$\overline{T}(\overline{s}, \overline{a}, \overline{s}') = \begin{cases} \tau(s, a, s'), & \text{if } l = l_0, \\ \lambda(\oplus) T(s, a, s') + \lambda(\ominus)[s = s'], & \text{if } l = l_1, \\ \lambda(\emptyset) T(s, a, s') + \lambda(\oslash) \tau(s, a, s'), & \text{if } l = l_2, \\ T(s, a, s'), & \text{if } l = l_3, \end{cases} \tag{1}$$

where $\lambda(\cdot) = \lambda(\cdot | \overline{s}, \overline{a})$ and $[\cdot]$ denotes Iverson brackets.

Intuitively, in l_0, the system acts according to its estimate of the human's transition function. In l_1, the system operates according to its own transition when it receives approval, and stays in the same state when it receives disapproval. In l_2, the system acts according to its estimate of the human's transition function when it is overridden, and otherwise acts according to its own transition function. In l_3, it simply acts according to its own transition function.

We define $\overline{C}$ as follows:

$$\overline{C}(\overline{s},\overline{a}) = g\big(C(s,a),\mu(\overline{s},\overline{a}),\rho(\overline{s},\overline{a})\big), \tag{2}$$

where g is any cost aggregation function on C,μ, and ρ. The simplest case of which is a a linear combination of the three factors.

Properties of a CAS

We now review two key notions of a CAS: *competence* and *level-optimality*. First, the *competence* of a CAS for executing action a in state $\overline{s}$ is the most cost-effective level of autonomy given perfect knowledge of the human's feedback model. If the human is likely to deny or override the system autonomously carrying out action a, the competence will likely be low. Similarly, if the human is likely to allow the action to be carried out autonomously, we expect the competence to be high, although this is not formally required. It is worth emphasizing that this is a definition on the human-agent system as a whole, and not simply the underlying autonomous agent, as the competence is directly affected not just by the technical capabilities of the agent, but also the human's perception of the agent's capabilities.

Definition 2. *Let $\lambda^{\mathscr{H}}$ be the stationary distribution of feedback signals that the human authority follows. The* ***competence*** *of CAS $\mathscr{S}$, denoted $\chi_{\mathscr{S}}$, is a mapping from $\overline{S}\times A$ to the optimal (least-cost) level of autonomy given perfect knowledge of $\lambda^{\mathscr{H}}$. Formally:*

$$\chi_{\mathscr{S}}(\overline{s},a) = \operatorname*{argmin}_{l\in L} Q(\overline{s},(a,l);\lambda^{\mathscr{H}})$$

where $Q(\overline{s},(a,l);\lambda^{\mathscr{H}})$ is the expected cumulative reward when taking action $\overline{a}=(a,l)$ in state $\overline{s}$ conditioned on knowing the human's true feedback distribution, $\lambda^{\mathscr{H}}$.

Second, we say that a CAS is *level-optimal* in state $\overline{s}$ if the system operates at its competence in that state. In other words, it is a measure of how often the system operates at its competence. The higher the system's level-optimality is, the better it has learned to interact with and exploit the capabilities of the human authority as well as its own capabilities.

Definition 3. *A CAS $\mathscr{S}$ is* ***level-optimal*** *if*

$$\pi^*(\overline{s}) = (a,\chi_{\mathscr{S}}(\overline{s},a)) \quad \forall \overline{s}\in\overline{S}$$

Similarly, $\mathscr{S}$ is ***γ-level-optimal*** *if this holds for an γ portion of states.*

3 Improving Competence

While our recent work on competence awareness proves that under a set of assumptions a CAS will converge to be level-optimal in the limit, this result says nothing about the quality of the CAS's competence [6]. In particular, if a CAS is missing the features necessary to correctly represent its domain in a way that aligns with the human, even it converges to be level-optimal, its competence may be quite low.

Hence, the objective of this work is to **provide a CAS with the ability to improve its competence over time** by leveraging the existing human feedback available to the agent. Formally, this means that the CAS will increase both its competence in various situations and the total level-optimality of the system. Our approach relies on the assumption that humans are *ε-consistent* in their feedback; that is, given the same $(\overline{s},\overline{a})$, the feedback signal returned will be the same up to some small noise ε. We address practical concerns regarding this assumption in Section 3.3. Under this assumption, the system can identify cases

where feedback appears inconsistent or random, indicating a potential *missing* feature—a feature used by the human authority to make their decisions, but currently not used by the system. By identifying the most likely feature or combination of features missing from the system's domain model, the agent can update its model to better align with the internal model of the human, enabling it to improve its overall competence by discriminating between situations where it can and cannot act in any given level of autonomy.

3.1 Definitions

Before presenting the general algorithm of our approach, we begin with several important definitions. Let $\mathscr{S} = \langle AM, HM, DM \rangle$ be a competence-aware system.

The *complete feature space* available to $\mathscr{S}$, e.g. from its sensors or other external sources, can be partitioned into an *active feature space* that is used by $\mathscr{S}$ and an *inactive feature space* that is not yet used by $\mathscr{S}$. As $\mathscr{S}$ receives additional feedback over time, $\mathscr{S}$ will learn to exploit inactive features in order to more effectively align with the features used by the human authority.

Definition 4. *Given the* ***complete feature space*** $F = F_1 \times F_2 \times \cdots \times F_n$ *produced by the set of sensors available to* $\mathscr{S}$*, the* ***active feature space*** *used by* $\mathscr{S}$ *is* $\hat{F} = \hat{F}_1 \times \cdots \times \hat{F}_m \subset F$ *while the* ***inactive feature space*** *not used by* $\mathscr{S}$ *is* $\breve{F} = F \setminus \hat{F}$.

In order to ensure that level-optimality can be improved, we assume that the human authority produces feedback that remains *consistent* during the operation of $\mathscr{S}$. This means that when the agent performs the same action in the same state at the same level of autonomy, it expects to receive, with high probability, the same feedback each time; observing consistent violation of this assumption is central to our approach.

Definition 5. *A human's feedback is* ε**-consistent** *if with probability at least* $1-\varepsilon$*, for some small noise* ε*, the feedback provided for some* $(\overline{s}, \overline{a})$ *is the same each time* $(\overline{s}, \overline{a})$ *is encountered. If this fails to hold for any sufficiently small* ε*, the feedback is said to be* ***inconsistent****.*

We now present the two central concepts of our paper. (1) A state $\overline{s}$ is *indiscriminate* if it is missing information, leading to the feedback profile having a low predictive confidence. Intuitively, the condition states that for at least one action, there is *no* feedback signal that the system predicts with sufficiently high confidence. (2) A *discriminator* is a feature (or set of features) in $\breve{F}$ that enables the feedback profile to better discriminate the feedback of an indiscriminate state. For example, consider a state that represents a closed door. With no additional features, the agent may have received, say, 50% approvals and 50% disapprovals. After adding the feature *door size*, the agent may observe that it receives 100% approval for light doors, and 100% disapproval for heavy doors.

Definition 6. *We say that a state* $\overline{s}$ *is* ***indiscriminate*** *if it satisfies the following condition:*

$$\exists\, \overline{a} \in \overline{A} \;\mid\; \forall \sigma \in \Sigma : \hat{\lambda}(\sigma | \overline{s}, \overline{a}) \leq 1 - \delta, \;\; \delta \in \left(\varepsilon, 1 - \frac{1}{|\Sigma|}\right)$$

where the human is ε*-consistent.*

Note that δ is a parameter that is *chosen*. The lower that δ is, the looser the state's condition is, meaning that a higher predictive confidence is needed for $(\overline{s}, \overline{a})$ to not be considered indiscriminate. In practice, we also require that we have seen $(\overline{s}, \overline{a})$ a sufficient number of times, m, which is also a parameter that is chosen a priori.

Definition 7. *Given an active feature space $\hat{F}$ and inactive feature space $\breve{F}$, a* ***discriminator****, D, is a cross product of feature sets in $\breve{F}$ that when added to the active feature space, improves the accuracy of the feedback profile by at least $\alpha \in (0,1)$, and does not cause any state that was previously not indiscriminate to become indiscriminate.*

These three concepts form the basis for our approach. Intuitively, our algorithm identifies indiscriminate states, finds the best possible discriminator for that indiscriminate state, and if that discriminator satisfies a set of criteria, augments the active feature space $\hat{F}$ with the discriminator.

3.2 Algorithm

Algorithm 1 performs the following steps.

1. An indiscriminate state $\bar{s}$ is sampled with the additional requirement that $(\bar{s}, \bar{a})$ has been visited at least m times, where $\bar{a}$ refers to the action in Definition 6. If no such state is found, the algorithm exits.
2. The full feedback dataset is split into a training set and a validation set. An element of the dataset is comprised of an action, a level of autonomy, a set of state features, and a label corresponding to the feedback received.
3. We identify the top k potential discriminator, the process for which is described below.
4. For each discriminator D found, we train a new feedback profile on the training dataset using the feature space $\hat{F} \times D$.
5. Each feedback profile λ_D is then tested on the validation set, and the best performing discriminator, D^*, is selected for validation.
6. We validate D^* by checking if λ_D performs at least α better than the current $\hat{\lambda}$.
7. Finally, if the validation is successful, the active feature space is augmented with the discriminator D^*, and the CAS $\mathscr{S}$ is updated.

Algorithm 1: A general method for improving level-optimality in competence-aware systems

Input: A dataset $\mathscr{D}$, a CAS $\mathscr{S}$, a tolerance δ, and a count k
Result: An updated CAS $\mathscr{S}$

$\bar{s} \leftarrow$ SampleIndiscriminateState($\mathscr{S}, \mathscr{D}, \delta, \tau$)
if $\bar{s} = \emptyset$ **then**
  Quit()
$\mathscr{D}_{train}, \mathscr{D}_{val} \leftarrow$ Split($\mathscr{D}$)
discriminators $\leftarrow$ GetDiscriminators($\mathscr{D}_{train}$, k, $\bar{s}$)
for D **in** *discriminators* **do**
  $\lambda_D \leftarrow$ TrainClassifier($\hat{F} \times D, \mathscr{D}_{train}$)
$D^* = \text{argmax}_D$ EvaluateClassifier(λ_D, $\mathscr{D}_{val}$)
if ValidateDiscriminator($D^*, \mathscr{S}$) **is** *True* **then**
  $\hat{F} \leftarrow \hat{F} \times D^*$
  Update($\mathscr{S}$)

Potential discriminators are determined as follows. First, we isolate all instances of $(\overline{s},\overline{a})$ in the dataset $\mathscr{D}_{train}$ and build a correlation matrix of the inactive features to the feedback that has been received for $(\overline{s},\overline{a})$. Although different types of correlation coefficients may be used in principle, we observed that the Pearson correlation coefficient was most effective for our tasks. In particular, rank-based correlation coefficients, such as Spearman's or Kendall, did not perform well in this setting as the notion of *rank* has little meaning with respect to the feedback signals or the input.

Next, we use the correlation matrix to build a discrimination matrix. The discrimination matrix provides a value, or score, for each feature set in $\check{F}$, which is calculated by summing over the maximum absolute value for each row (feature value) in the correlation matrix associated with the feature set in question, and then normalizing over the size of the feature set. We take the absolute value because a highly negative correlation is as informative as a highly positive correlation; what matters is the magnitude of the correlation. We max over each row to prioritize having a high correlation with at least one feedback signal, rather than medium correlation with numerous feedback signals.

A final note on the algorithm is that the validation step is intended to be flexible enough to capture any requirements on adding a discriminator to the active feature space. While we only use the two stated above, there may be other factors, including domain-specific ones, that can further improve the robustness of the approach for any given domain.

3.3 Assumptions

In this work, we make a few key assumptions. We briefly describe these assumptions, why we made them, and why we believe they are reasonable.

First, we make the assumption that the initial transition function provided in the domain model is *sufficiently correct* for any scenario where the agent is allowed, under κ, to act in an autonomous capacity. We are not concerned in this work with agents whose base domain model is poor and prone to failure, but with improving the competence of already capable systems operating in difficult domains (e.g., autonomous service robots or extraterrestrial rovers). In general, we are aiming to improve the robustness of deployed systems where the designers cannot hope to account for every possible scenario a priori, but where the scenarios that are considered are well-designed. While our approach investigates improving the competence of a CAS by augmenting the feature space to increase the granularity of the state representation, it may also be possible to increase the competence by updating the transition function itself and replanning as the human improves *its* understanding of the sytem's capabilities. However, such an approach falls outside the scope of this work.

Second, we assume that the human authority has a sufficient understanding of the agent's capabilities to (1) prevent the execution of an action that the agent cannot perform successfully, and (2) provide largely consistent feedback. We make this assumption for two reasons. First, there are different ways to improve the human's understanding of the system's capabilities so that it has the appropriate trust [22], or reliance, on the system. These include pre-deployment training, standardized feedback criteria, or simply expert knowledge of the system. Second, recognizing potential failure and fault recovery are separate areas of active research that is orthogonal to what we are examining in this paper.

It is worth emphasizing that, operating under these assumptions, *we do not need to update the underlying domain model's transition or reward functions directly at any point*. It suffices for the agent to simply be able to discriminate between actions that it has the competence to perform autonomously and actions that require human involvement. In instances where the agent is not competent to act autonomously, the agent will either have to do something else, or have the human take control in some capacity, and the transitions will reflect this fact. Hence the system does not need to reason about the

true transition dynamics of these actions since it cannot execute them in the first place.

3.4 Theoretical Results

The following proposition states that if the agent and the human share the same finite set of possible features for modeling the domain, the human is consistent up to small noise in their feedback, and every state-action pair is encountered sufficiently often, then in the limit the agent will have no indiscriminate states. Intuitively, this follows from the fact that a causal feature (or more generally set of features) will always have as high of correlation as any non-causal feature, and so the algorithm will select them to be added to the model. A state can only be indiscriminate if there is some action that the feedback profile cannot confidently predict any feedback. However, if the causal features used by the human for that state-action pair are added to the model, and enough feedback is received in the limit, then eventually the feedback profile will converge sufficiently to predict the human's feedback with high probability. Hence no state will be indiscriminate in the limit.

Proposition 1. *If the full feature space is finite, the human and agent share the same feature space, and the human is ε-consistent, then if no $(\overline{s},\overline{a}) \in \overline{S} \times \overline{A}$ is starved, in the limit for every $(\overline{s},\overline{a}) \in \overline{S} \times \overline{A}$, there will be some $\sigma \in \Sigma$ for which it holds that $\lambda(\sigma|\overline{s},\overline{a}) > 1-\delta$, for any $\delta \in \left(\varepsilon, 1-\frac{1}{|\Sigma|}\right)$.*

Proof Sketch. For every $(\overline{s},\overline{a})$, there is some subset $F^{\mathscr{H}}_{\overline{s},\overline{a}} \subseteq F$ that the human's feedback is dependent on for $(\overline{s},\overline{a})$. If $F^{\mathscr{H}}_{\overline{s},\overline{a}} \subseteq \hat{F}$, then in the limit as no $(\overline{s},\overline{a})$ is starved, it will be the case that $\lambda(\sigma|\overline{s},\overline{a}) \geq 1-\varepsilon \geq 1-\delta$, for any $\delta \in \left(\varepsilon, 1-\frac{1}{|\Sigma|}\right)$. So, suppose there exists some $\overline{s} \in \overline{S}$ that is indiscriminate on some action $\overline{a} \in \overline{A}$. It must be the case that some subset of $F^{\mathscr{H}}_{\overline{s},\overline{a}}$ is not in $\hat{F}$. However, such a subset of features, for some $\sigma \in \Sigma$, in the limit given that the human is ε-consistent, will have correlation near 1 and at least as high of correlation as any non-causal feature. By our algorithm, such a discriminator would be selected for evaluation and would pass the validation criteria, and so would be added to the active feature space. This is a contradiction that it is not in $\hat{F}$, so $\overline{s}$ cannot be an indiscriminate state. □

4 Experiments

To validate our technique, we implemented our framework in a simulated domain in which a robotic agent is tasked with delivering packages to various rooms on a college campus. An illustration of the map used in our experiments can be seen in Figure 1. To accomplish its task, the agent may be required to deal with two categories of obstacles. The first is crosswalks that can have differing levels of traffic conditions that change stochastically, different visibility conditions that are fixed through time (i.e. a blind corner), and can be on either one-way streets or two-way streets, also fixed through time. The second category of obstacle is doors which the robot must open to get into buildings, through buildings, and into offices. Doors have different sizes—light, medium, and heavy—are painted various colors, and can be either push or pull doors. Trees, walls, and roads are all avoided by the system completely. Initially, $\kappa(s,a) = \{0,1\}$ for all actions in obstacle states. For states with no obstacles, the system is allowed to operate in unsupervised autonomy. In all experiments, there is a 5% chance that the human's feedback signal is chosen uniformly at random from the possible feedback signals.

Throughout all of our experiments, we use the CAS model defined in Section 2.4. We consider the following four levels of autonomy: (1) no autonomy, (2) verified autonomy, (3) supervised autonomy, and (4) unsupervised autonomy. Level 1 requires a human to complete the action for the agent. Level 2

requires the agent to query the human for explicit approval prior to executing its intended action. Level 3 requires a human to be available in a supervisory capability while the agent executes the action, with the ability to override and take over control if deemed necessary. Level 4 requires no human involvement and is performed fully autonomously. The agent can receive the following four feedback signals: approval ($\oplus$), disapproval ($\ominus$), override ($\oslash$), and none ($\emptyset$). We initialize the feedback profile to a uniform prior over the feedback signals, and initialize the autonomy profile to be level 1 for any state with an obstacle in it, and level 3 otherwise.

To validate our approach, we initialize the system to only use a subset of the above features. In particular, the CAS initially only considers the traffic condition at crosswalks, and whether a door is open or closed. To increase its competence, reducing unnecessary reliance on the human, the CAS must learn which additional features it needs to add to its model to correctly predict human feedback with high confidence. In this case, the additional causal features that the human authority uses to determine their feedback are the visibility condition of the crosswalk, and both the size and opening mechanism of doors. To obfuscate things further for the system, the non-causal features are intentionally highly correlated with certain causal features. For example, in one building, the door color is perfectly consistent with the door size, and with one exception, the street type is consistent with the visibility condition. This is to test whether our approach correctly identifies causal features over correlative features.

We implement our feedback profile λ as a GA^{2}M [23]. This model works well on low dimension feature spaces where pairwise feature interactions are particularly important, and has nice interpretability properties, making it desirable for robotic decision making domains. In particular, we train our λ on all pairwise feature tensors using the native gridsearch function for hyperparameter tuning.

In Algorithm 1, δ is set to 0.95, the threshold m is set to 30, k is set to 1, and the train/validation split is 75/25.

4.1 Results

In this domain, we conducted two different experiments. In the first, we set the start and goal states to be fixed throughout all episodes. In the second, the start and goal states are randomly drawn from the set of

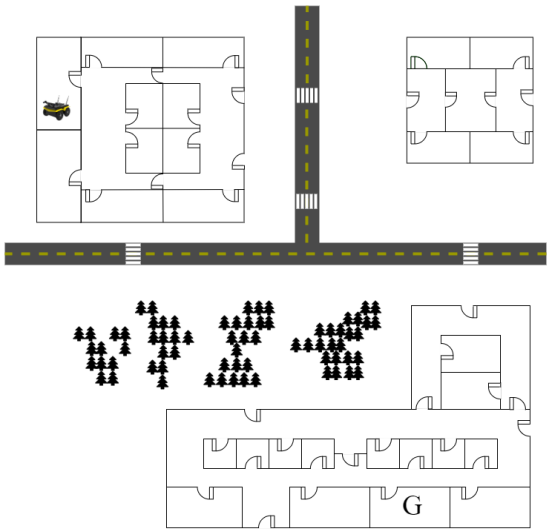
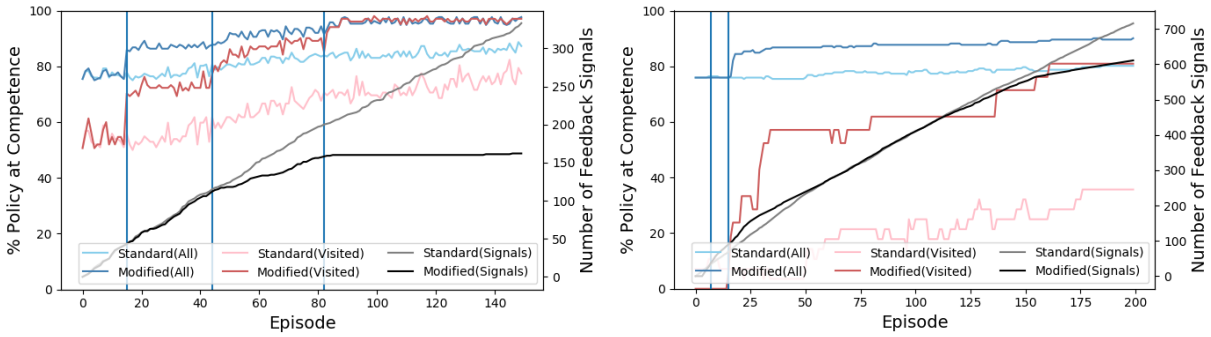


Figure 1: A depiction of the domain used for our experiments.



(a) Start and goal states randomly assigned every episode.

(b) Start and goal states stay fixed throughout all episodes.

Figure 2: Level-optimality of the CAS across different subsets of the state space. *All states* refers to the entire state space. *Visited states* are states that the system entered at least once during any episode. *Cumulative signals* are the total feedback signals received after each episode. Results shown are the mean and standard error over 10 trials.

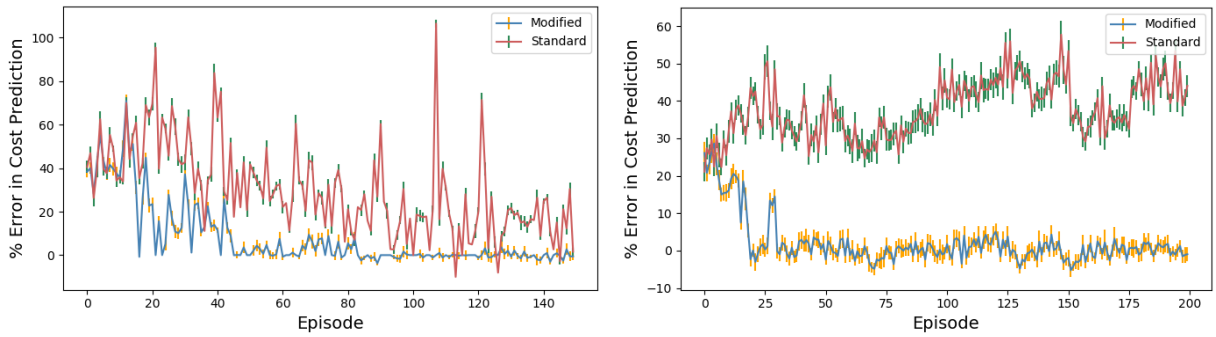
rooms through the campus map. For each experiment, we simulated both a modified CAS and a standard CAS for comparison.

We first observe that in the modified CAS, the level-optimality reached nearly 100% level-optimality in the random task experiment across all states, and all visited states. The vertical lines in Figure 2 indicate where discriminators were added, and we can observe that after each discriminator was added the competence starts to climb up before stagnating. While we do not expect to see this with all states in the single task, the single task experiment failed to identify and add any missing feature for crosswalks, as it only ever traversed a single crosswalk, and hence did not get enough variation in its feedback data. However, both results for the modified CAS are in stark contrast to the standard CAS which was not able to increase its level-optimality nearly as much—only a few percent across all states over the course of both experiments. In the single task domain, the standard CAS was unable to reach even 40% level-optimality across all visited states, more than 40% lower than the modified CAS with over 100 feedback signals more at the same point. Furthermore, the modified CAS used significantly fewer feedback signals than the standard CAS which never decreased in rate, demonstrating that the modified CAS is also less burdensome on the human. Overall, these graphs demonstrate that our method improves the CAS's competence over the course of its activity.

The second important result can be observed in Figure 3. In both experiments, the expected cost converges to be almost identical to the incurred cost, averaged over 10 trials. This result demonstrates that adding the new features improves the model's accuracy and leads to better overall performance. In the standard CAS, we can see that the agent incurs costs significantly higher than predicted throughout its lifespan in both experiments, although this is most notable in the single task experiment. This indicates that the standard CAS's performance stays stagnant, while the modified CAS is able to significantly improve its performance.

5 Related Work

Markov Decision Processes with Uncertain Parameters. There is a substantial body of literature on Markov decision processes (MDPs) with uncertain model parameters that has been investigated over the course of the last five decades. *MDPs with imprecise transitions* (MDP-IPs) were introduced as early as 1973 by Satia and Leve [33], and seminally by White and Eldeib in 1994 [42]. In an MDP-IP, rather



(a) Start and goal states randomly assigned every episode.

(b) Start and goal states stay fixed throughout all episodes.

Figure 3: Percent difference in the cost incurred averaged over 10 episodes, and the expected cost.

than modeling the transition dynamics as a single function, it is represented by a distribution over a set of possible transition functions. Since then, numerous expansions to the problem have been investigated including *factored* MDP-IPs [16] and MDPs with set-valued transitions (MDP-STs) [39], as well as novel solution approaches, including a multi-linear and integer programming approach [34], and a real-time dynamic programming approach [15].

Similarly, MDPs with uncertain rewards (sometimes referred to as IR-MDPs, or *imprecise reward MDPs*) were introduced as early as 1986 by White and Eldeib [41], and are similar in concept to the MDP-IP. More recently, Regan and Boutilier have proposed both an offline [28] and online [29] approach to solving IR-MDPs, which are, to our knowledge, the state of the art in computational solutions to this class of problems.

However, more generally, the concept of an *uncertain MDP* (UMDP) was first formally described by Chen and Bowling [10] to capture MDPs in which both the transitions and rewards may be uncertain or imprecisely specified. Note that this is not to be confused with MDP-IPs referred to as uMDPs [5, 44] or IR-MDPs referred to as uMDPs [46]), and has since been expanded upon [2, 1, 3].

All of these models are intended to deal with domains where producing a fully precise and accurate model of the domain a priori is hard or infeasible, or where the dynamics are simply non-stationary. While the motivation of these lines of work are similar to ours, namely the fact that precisely modeling complex, real-world domains is often hard or impossible to do prior to deployment, our work differs starkly in that we do not directly modify the transition or reward functions, but rather we are concerned with modifying the factored state representation to better match that of a separate entity (e.g. the human).

Learning from Human Interaction. There are several areas of research that have investigated learning from interaction with a human, namely *learning from human input* [31, 36, 38] and *learning from demonstration* [11, 12, 14, 30]. In the former, the agent must learn some parameter of the model, generally the q-value of an action or the best action itself at a given state, from interacting with a human either through advice or questions. In the latter, the agent attempts to learn its policy by mimicking the actions of another agent (generally a human).

However, while our work is concerned with learning from interacting with a human, both the focus and scope of what we are attempting to have the agent learn is very different as we are limiting the learning to be only over missing features *used by the human in their feedback*.

Model-Based Reinforcement Learning. Our problem is related to the work on model-based RL in which

the agent actively builds an estimate of the transition and reward function for the domain it is operating in without relying on a model to be defined a priori [43, 40]. By bypassing the need for an a priori model entirely, model-based RL as a decision-making framework does not suffer from the primary problem investigated in this paper, i.e. imprecise and incomplete models. In fact, there is large body of literature that has investigated ways to use reinforcement learning for robot decision making that addresses some of its primary challenges such as safety and explainability [4, 7, 35]. However, our work is geared towards improving robotic systems which use stochastic model-based planning methods, rather than reinforcement learning.

Feature Selection. Given that a primary element of our work is identifying useful features, by some metric, from some large set of possible features, our work is highly related to that of unsupervised feature selection where the objective is to select from large space of possible features the one(s) that maximize some given objective function [13]. Feature selection has most commonly been applied to data analysis where the data is high dimensional, warranting a need to select the features that optimize the objective function while preserving the underlying structure of the data [19, 26]. It has also been used in reinforcement learning as a technique to more efficiently determine the most relevant sensory features for capturing the transition dynamics of the domain [18]. Largely, because the *method* of identifying the features is not the focus of our work, feature selection is an orthogonal problem to what is studied here. However, we believe that many ideas from this area are symbiotic with our work, and we will be examining how we may incorporate these methods to improve the performance of our approach.

6 Discussion

This paper proposes a method for providing competence-aware systems the ability to improve their competence over time by identifying key features missing from their model of the domain and adding those features into their state representation. This method works by identifying *indiscriminate states*, states missing information needed to predict human feedback, and for such states determining the most likely *discriminators*, features that improve the confidence of the feedback profile by enabling the system to better discriminate the feedback for indiscriminate states. To validate our approach, we provide empirical results in a simulated delivery robot domain that demonstrate that our approach (1) identifies the missing features, (2) only adds the missing *causal* features, (3) significantly improves the competence of the system when compared to an unmodified CAS, and (4) leads to major improvements in performance. However, there are several areas of potential improvement for our approach that we are actively addressing for future work.

Human Understanding of System Ability. Our method relies on the assumption that the human has a sufficient understanding of the system's underlying capabilities to consistently provide accurate feedback, and to know when to disallow the agent to perform certain actions autonomously that it is not capable of doing successfully. In practice, these assumptions may not be the case. To address this issue, we are investigating the use of new feedback signals that specifically indicate a lack of knowledge or confidence on the part of the human, prompting the system to gather additional information rather than to act immediately. This is similar to the recently proposed approach where an agent must monitor its own ability and consequently determine where to ask for help from a human (which has a cost), and learn from the provided demonstration [30].

Feedback from Multiple Human Authorities. Similarly, we make the assumption that feedback comes from a single source—namely, a single human authority. In practice it may often be the case that the system interacts with, and receives feedback from, different human authorities each with different perspectives, preferences, or simply different understanding of the system's abilities. In practice, this problem can be addressed in some cases by requiring a training phase for any authority that interacts with the system to provide a rigorous understanding of the system's capabilities, and a set of standardized feedback criteria to enforce consistency of feedback. Hence, we could extend the CAS model to not have a single feedback profile λ, but instead to maintain a distribution over feedback profiles, or learn a different profile for each human that interacts with the system to better personalize the response.

Incongruous Domain Models. In general, it may not be the case that every, or even any, feature that the human authority's feedback is conditioned on is available to the agent. In these cases, it may still be beneficial to identify and add correlative features to the model as proxies for the causal features. However, these features may cause problems elsewhere in the domain where the correlation does hot hold, so addressing this issue is key in making this method more robust.

Practical Models. The preliminary results presented in this paper are on a simple domain with only a small number of (highly abstract) features. We are testing our approach on a real mobile robot where our method must operate directly on the high dimensional *sensory features* available to the agent, rather than semantically interpretable features as presented in the paper. By replacing the GA^2M used for the feedback profile in this work with a more sophisticated CNN, we may be able to identify these discriminators directly from the perceptual data.

Feature Selection. As noted earlier, there is a large body of literature on the problem of feature selection which our approach largely does not utilize as presented here. We will investigate ways to incorporate techniques from feature selection to improve both the efficiency and effectiveness of our algorithm with respect to identifying missing causal features. This problem is particularly relevant in high dimensional feature spaces like the sensory features of a mobile robot, where the computational overhead will be much higher, and there is more noise in the data.

Acknowledgments

This work was supported in part by the National Science Foundation Grants IIS-1724101, IIS-1813490, and DGE-1451512, and in part by the Alliance Innovation Lab Silicon Valley.

References

[1] Yossiri Adulyasak, Pradeep Varakantham, Asrar Ahmed & Patrick Jaillet (2015): *Solving uncertain MDPs with objectives that are separable over instantiations of model uncertainty*. In: *AAAI Conference on Artificial Intelligence*, pp. 3454–3460.

[2] Asrar Ahmed, Pradeep Varakantham, Yossiri Adulyasak & Patrick Jaillet (2013): *Regret based robust solutions for uncertain Markov decision processes*. In: *Advances in Neural Information Processing Systems (NeurIPS)*, pp. 881–889, doi:10.1613/jai.5242.

[3] Asrar Ahmed, Pradeep Varakantham, Meghna Lowalekar, Yossiri Adulyasak & Patrick Jaillet (2017): *Sampling based approaches for minimizing regret in uncertain Markov decision processes (MDPs)*. Journal of Artificial Intelligence Research (JAIR) 59, pp. 229–264, doi:10.1613/jair.5242.

[4] Sule Anjomshoae, Amro Najjar, Davide Calvaresi & Kary Främling (2019): *Explainable agents and robots: Results from a systematic literature review*. In: International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS), pp. 1078–1088.

[5] J. Andrew Bagnell, Andrew Y. Ng & Jeff G. Schneider (2001): *Solving uncertain Markov decision processes*. Technical Report, Carnegie Mellon University.

[6] Connor Basich, Justin Svegliato, Kyle Hollins Wray, Stefan Witwicki, Joydeep Biswas & Shlomo Zilberstein (2020): *Learning to Optimize Autonomy in Competence-Aware Systems*. In: International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS), pp. 123–131.

[7] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig & Andreas Krause (2017): *Safe model-based reinforcement learning with stability guarantees*. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 908–918.

[8] Alberto Broggi, Massimo Bertozzi, Alessandra Fascioli, C. Guarino Lo Bianco & Aurelio Piazzi (1999): *The ARGO autonomous vehicle's vision and control systems*. International Journal of Intelligent Control and Systems 3(4), pp. 409–441.

[9] Alberto Broggi, Pietro Cerri, Mirko Felisa, Maria Chiara Laghi, Luca Mazzei & Pier Paolo Porta (2012): *The VisLab Intercontinental Autonomous Challenge: An extensive test for a platoon of intelligent vehicles*. International Journal of Vehicle Autonomous Systems 10(3), pp. 147–164, doi:10.1504/IJVAS.2012.051250.

[10] Katherine Chen & Michael Bowling (2012): *Tractable objectives for robust policy optimization*. In: Advances in Neural Information Processing Systems (NeurIPS), pp. 2069–2077.

[11] Sonia Chernova & Manuela Veloso (2009): *Interactive policy learning through confidence-based autonomy*. Journal of Artificial Intelligence Research (JAIR) 34, pp. 1–25, doi:10.1613/jair.2584.

[12] Jeffery A. Clouse (1996): *On integrating apprentice learning and reinforcement learning*. Ph.D. thesis, University of Massachusetts Amherst.

[13] Manoranjan Dash & Huan Liu (1997): *Feature selection for classification*. Intelligent Data Analysis 1(3), pp. 131–156, doi:10.1016/S1088-467X(97)00008-5.

[14] Francesco Del Duchetto, Ayse Kucukyilmaz, Luca Iocchi & Marc Hanheide (2018): *Do not make the same mistakes again and again: Learning local recovery policies for navigation from human demonstrations*. IEEE Robotics and Automation Letters (RA-L) 3(4), pp. 4084–4091.

[15] Karina V. Delgado, Leliane N. De Barros, Daniel B. Dias & Scott Sanner (2016): *Real-time dynamic programming for Markov decision processes with imprecise probabilities*. Artificial Intelligence (AIJ) 230, pp. 192–223, doi:10.1016/j.artint.2015.09.005.

[16] Karina Valdivia Delgado, Scott Sanner & Leliane Nunes De Barros (2011): *Efficient solutions to factored MDPs with imprecise transition probabilities*. Artificial Intelligence (AIJ) 175(9-10), pp. 1498–1527, doi:10.1016/j.artint.2011.01.001.

[17] Ernst D. Dickmanns (2007): *Dynamic vision for perception and control of motion*. Springer Science & Business Media, doi:10.1007/978-1-84628-638-4.

[18] Carlos Diuk, Lihong Li & Bethany R. Leffler (2009): *The adaptive k-meteorologists problem and its application to structure learning and feature selection in reinforcement learning*. In: International Conference on Machine Learning (ICML), pp. 249–256, doi:10.1145/1553374.1553406.

[19] Liang Du & Yi-Dong Shen (2015): *Unsupervised feature selection with adaptive structure learning*. In: ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 209–218, doi:10.1145/2783258.2783345.

[20] Yang Gao & Steve Chien (2017): *Review on space robotics: Toward top-level science through space exploration*. Science Robotics 2(7), doi:10.1126/scirobotics.aan5074.

[21] Nick Hawes, Christopher Burbridge, Ferdian Jovan, Lars Kunze, Bruno Lacerda, Lenka Mudrova, Jay Young, Jeremy Wyatt, Denise Hebesberger, Tobias Kortner et al. (2017): *The STRANDS project: Long-term autonomy in everyday environments*. IEEE Robotics & Automation Magazine 24(3), doi:10.1109/MRA.2016.2636359.

[22] Kevin Anthony Hoff & Masooda Bashir (2015): *Trust in automation: Integrating empirical evidence on factors that influence trust*. Human Factors 57(3), pp. 407–434, doi:10.1177/0018720814547570.

[23] Yin Lou, Rich Caruana, Johannes Gehrke & Giles Hooker (2013): *Accurate intelligible models with pairwise interactions*. In: ACM International Conference on Knowledge Discovery and Data Mining (SIGKDD), pp. 623–631, doi:10.1145/2487575.2487579.

[24] Wim Meeussen, Eitan Marder-Eppstein, Kevin Watts & Brian P. Gerkey (2011): *Long term autonomy in office environments*. In: Robotics: Science and Systems (RSS) ALONE Workshop.

[25] John F. Mustard, D. Beaty & D. Bass (2013): *Mars 2020 science rover: Science goals and mission concept*. In: AAS/Division for Planetary Sciences Meeting Abstracts, 45.

[26] Feiping Nie, Wei Zhu & Xuelong Li (2016): *Unsupervised feature selection with structured graph optimization*. In: AAAI Conference on Artificial Intelligence, pp. 1302–1308.

[27] Luis Pineda, Takeshi Takahashi, Hee-Tae Jung, Shlomo Zilberstein & Roderic Grupen (2015): *Continual planning for search and rescue robots*. In: IEEE/RAS International Conference on Humanoid Robots (Humanoids), IEEE, pp. 243–248.

[28] Kevin Regan & Craig Boutilier (2010): *Robust policy computation in reward-uncertain MDPs using non-dominated policies*. In: AAAI Conference on Artificial Intelligence, pp. 1127–1133.

[29] Kevin Regan & Craig Boutilier (2011): *Robust online optimization of reward-uncertain MDPs*. In: International Joint Conference on Artificial Intelligence (IJCAI), pp. 2165–2171, doi:10.5591/978-1-57735-516-8/IJCAI11-361.

[30] Marc Rigter, Bruno Lacerda & Nick Hawes (2020): *A Framework for learning From demonstration with minimal human effort*. IEEE Robotics and Automation Letters (RA-L) 5(2), pp. 2023–2030, doi:10.1109/LRA.2018.2861080.

[31] Michael T. Rosenstein & Andrew G. Barto (2004): *Supervised actor-critic reinforcement learning*. In J. Si, A. G. Barto, W. B. Powell & D. Wunsch, editors: Handbook of Learning and Approximate Dynamic Programming, chapter 7, IEEE Press, pp. 359–380.

[32] Sandhya Saisubramanian, Shlomo Zilberstein & Prashant Shenoy (2017): *Optimizing electric vehicle charging through determinization*. In: ICAPS Workshop on Scheduling and Planning Applications.

[33] Jay K. Satia & Roy E. Lave Jr. (1973): *Markovian decision processes with uncertain transition probabilities*. Operations Research 21(3), pp. 728–740, doi:10.1287/opre.21.3.728.

[34] Ricardo Shirota Filho, Fabio Gagliardi Cozman, Felipe W. Trevizan, Cassio Polpo de Campos & Leliane Nunes De Barros (2007): *Multilinear and integer programming for Markov decision processes with imprecise probabilities*. In: International Symposium on Imprecise Probability: Theories and Applications (ISTPTA), pp. 395–404.

[35] William D. Smart & Leslie Pack Kaelbling (2002): *Effective reinforcement learning for mobile robots*. In: IEEE International Conference on Robotics and Automation (ICRA), 4, pp. 3404–3410, doi:10.1109/ROBOT.2002.1014237.

[36] Halit Bener Suay & Sonia Chernova (2011): *Effect of human guidance and state space size on interactive reinforcement learning*. In: IEEE Conference on Robot and Human Interactive Communication (RO-MAN), IEEE, pp. 1–6, doi:10.1109/ROMAN.2011.6005223.

[37] Justin Svegliato, Kyle Hollins Wray, Stefan J. Witwicki, Joydeep Biswas & Shlomo Zilberstein (2019): *Belief Space Metareasoning for Exception Recovery*. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 1224–1229, doi:10.1109/IROS40897.2019.8967676.

[38] Lisa Torrey & Matthew Taylor (2013): *Teaching on a budget: Agents advising agents in reinforcement learning*. In: *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*, pp. 1053–1060.

[39] Felipe W. Trevizan, Fabio Gagliardi Cozman & Leliane Nunes de Barros (2007): *Planning under risk and knightian uncertainty*. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, 2007, pp. 2023–2028.

[40] Thomas J. Walsh, Sergiu Goschin & Michael L. Littman (2010): *Integrating sample-based planning and model-based reinforcement learning*. In: *AAAI Conference on Artificial Intelligence*, pp. 612–617.

[41] Chelsea C. White III & Hany K. El-Deib (1986): *Parameter imprecision in finite state, finite action dynamic programs*. *Operations Research* 34(1), pp. 120–129, doi:10.1287/opre.34.1.120.

[42] Chelsea C. White III & Hany K. Eldeib (1994): *Markov decision processes with imprecise transition probabilities*. *Operations Research* 42(4), pp. 739–749, doi:10.1287/opre.42.4.739.

[43] Grady Williams, Nolan Wagener, Brian Goldfain, Paul Drews, James M. Rehg, Byron Boots & Evangelos A. Theodorou (2017): *Information theoretic MPC for model-based reinforcement learning*. In: *IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1714–1721, doi:10.1109/ICRA.2017.7989202.

[44] Eric M. Wolff, Ufuk Topcu & Richard M. Murray (2012): *Robust control of uncertain Markov decision processes with temporal logic specifications*. In: *Conference on Decision and Control (CDC)*, IEEE, pp. 3372–3379, doi:10.1109/CDC.2012.6426174.

[45] Kyle Hollins Wray, Luis Enrique Pineda & Shlomo Zilberstein (2016): *Hierarchical Approach to Transfer of Control in Semi-Autonomous Systems*. In: *International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 517–523.

[46] Huan Xu & Shie Mannor (2009): *Parametric regret in uncertain Markov decision processes*. In: *Conference on Decision and Control (CDC)*, IEEE, pp. 3606–3613, doi:10.1109/CDC.2009.5400796.