A Method for Designing Autonomous Robots that Know Their Limits

Alvika Gautam[1], Tim Whiting[2], Xuan Cao[2], Michael A. Goodrich[2] and Jacob W. Crandall[2]

***Abstract*— While the design of autonomous robots often emphasizes developing proficient robots, another important attribute of autonomous robot systems is their ability to evaluate their own proficiency and limitations. A robot should be able to assess how well it can perform a task before, during, and after it attempts the task. Thus, we consider the following question: How can we design autonomous robots that know their own limits? Toward this end, this paper presents an approach, called *assumption-alignment tracking* (AAT), for designing autonomous robots that can effectively evaluate their own limits. In AAT, the robot combines (a) measures of how well its decision-making algorithms align with its environment and hardware systems with (b) its past experiences to assess its ability to succeed at a given task. The effectiveness of AAT in assessing a robot's limits are illustrated in a robot navigation task.**

I. Introduction

The design of autonomous robot systems has understandably emphasized the development of *proficient* robots – i.e., robots that can effectively carry out tasks in a variety of different environments. While developing proficient robots is an ultimate goal, we argue that an autonomous robot should have an additional, related capability: the ability to identify when it can and cannot successfully carry out a task.

Proficiency self-assessment is valuable for several reasons. First, safety concerns dictate that autonomous systems should identify when they will fail, are failing, and have failed to accomplish a task [1], [2] and to be able to explain those failures [3], [4], [5], [6]. Second, because autonomous robots typically operate in the context of a team, the ability to know one's competence and limits can help in the creation and adaptation of synergistic plans [7], [8]. Third, knowing one's limits can be used to improve robot behavior. For example, proficiency self-assessment can be used to determine when to continue learning and when to exploit one's current knowledge [9], [10]. As such, autonomous robot design should focus on both proficiency and self-evaluation of proficiency.

We propose that proficiency self-assessment requires at least three evaluations (Fig. 1). First, the robot must determine whether it has a set of decision-making algorithms (or *behavior generators*) that can be combined to perform the desired task [11]. Second, the robot must have a performance standard, which gives it an understanding of what constitutes desirable (or acceptable) performance on that task. Finally, the robot must estimate its performance and compare this estimate to the performance standard.

*This work was supported in part by the U.S. Office of Naval Research (ONR) under Grant #N00014-18-1-2503.

[1]Department of Mechanical Engineering, Texas A&M University, College Station, TX, USA. alvikag@tamu.edu

[2]Computer Science Department, Brigham Young University, Provo, UT, USA. Contact: crandall@cs.byu.edu

Generator Completeness: The robot has algorithms (or **behavior generators**) necessary to perform the task

Expectation Awareness: The robot has a **performance standard** that represents acceptable/effective performance

Performance Evaluation: The robot has a **performance estimate**, which it compares to the performance standard

Fig. 1. Three components necessary for an autonomous robot to self-assess its ability to perform a task.

Although each of the three components shown in Fig. 1 is necessary for performing proficiency self-assessment, this paper focuses on the third component: How can a robot effectively estimate its performance at any time during task execution? In particular, we propose and analyze an approach based on the following view of proficiency self-assessment: *Proficiency self-assessment is awareness of how well one's 'generators' (i.e., decision-making algorithms) interact and align with the environment(s), robot hardware, and task(s) under consideration.* Empirical results demonstrate that this view can yield useful proficiency estimates.

II. Estimating Performance using AAT

Estimating performance in arbitrary environments and scenarios is challenging because it is difficult to determine how environmental characteristics, the robot's hardware, and the robot's decision-making algorithms (or generators) will combine to impact the robot's performance. This is difficult for *post hoc* estimates, but even more difficult for *a priori* and *in situ* estimates. Past work has conjectured that a robot's performance is a function of the complexity of the environment in which the autonomous system operates (e.g., [12], [13]). Unfortunately, quantifying how complexity impacts a robot's ability to perform a task remains an unsolved problem.

Alternative to a complexity approach, we assert that robot performance is sensitive to how well its generators align with its environment and hardware. This alignment can be determined via a set of metrics that do not require a direct assessment of the complexity of the environment, but rather is made by *tracking the veracity of the assumptions* upon which the robot's generators rely. We call this method *assumption-alignment tracking* (AAT).

With AAT, a robot estimates its performance via a two-step process. First, the robot evaluates the veracity of the assumptions made in the implementation of its generators (Fig. 2). To give the robot this capability, the robot designer must identify the assumptions made in the construction of the generators and then create tools that allow the robot to assess

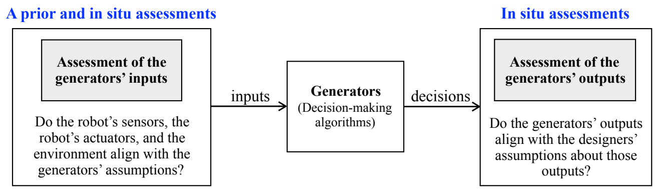

A prior and in situ assessments
Assessment of the generators' inputs
Do the robot's sensors, the robot's actuators, and the environment align with the generators' assumptions?
inputs
Generators (Decision-making algorithms)
decisions
In situ assessments
Assessment of the generators' outputs
Do the generators' outputs align with the designers' assumptions about those outputs?

Fig. 2. In assumption-alignment tracking (AAT), the robot continually monitors how well its generators (or decision-making algorithms) align with its hardware and its current environment. These evaluations involve determining how well the generators' inputs and outputs meet assumptions.

the veracity of these assumptions. Second, the outcomes of the evaluations over time are used, in conjunction with the robot's past experiences, to predict the robot's performance on the task. We discuss each step in turn.

A. *Evaluating Generator Assumptions*

As asserted in many No-Free-Lunch Theorems (e.g., [14]), all decision-making algorithms (or generators) for autonomous robot systems are based on assumptions or biases that dictate their performance [15]. When these assumptions and biases are correct, the robot's behavior and its impact on the world are predictable and satisfactory. However, when the assumptions are not met, the robot's behavior and its impact on the world are both less predictable and less likely to be satisfactory. Thus, the first step in AAT is to identify and evaluate these assumptions and biases.

Fig. 2 illustrates two important forms of assumptions related to a robot's generators: assumptions about generator (1) inputs and (2) outputs. Assumptions about generator inputs include assumptions about the robot's sensors, the robot's actuators, and the properties of the environment. For example, a mapping system for a robot performing a navigation task might assume that sensor readings have low variance and that the sensor can detect the objects in the environment. On the other hand, assumptions about generator outputs relate to the properties of the outcomes of generator decisions. For example, a robot navigation system might assume that the robot's mapping system will compute consistent positions of stationary objects in the environment.

Once generator assumptions are identified, the system designer should create *assumption checkers* which continually check *assumption veracity*. Initial assessments begin before the robot starts to operate in the environment (i.e., *a priori*; these assessments are mainly limited to input assumptions) and then continue throughout the mission (i.e., *in situ* assessments). The resulting time-series of assessments over each of the assumptions forms the *alignment profile*, which can then be used to evaluate the robot's ability to perform the given task in the current environment.

Formally, let $\Phi = \{\phi_1, \cdots, \phi_M\}$ be the set of M assumptions made by the robot's generators, and let $x^t_{\phi_i}$ be the assessment of the veracity of assumption ϕ_i at time t, produced from some subset of observations made up to time t. Also, let $\mathbf{x}(t) = (x^t_{\phi_1}, \cdots, x^t_{\phi_M})$ be the vector of veracity assessments made at time t. Then, the *alignment profile* at time t, denoted $\mathbf{X}(t)$, is the following time series:

$$\mathbf{X}(t) = \langle \mathbf{x}(1), \mathbf{x}(2), \cdots, \mathbf{x}(t) \rangle \tag{1}$$

Note that veracity assessments share similarities with other prior work. For example, Ramesh *et al.* [7] advocate for tracking so-called *robot vitals*, which appear to be assessments of assumptions made about generator outputs. Similarly, Das *et al.* [4] use assessments of generator outputs to produce explanations to assist users in fault recovery of a robot system. Finally, Béné and Doyen create viability tests that compare performance to pre-defined thresholds and use the result to estimate the resilience of an agent [16].

B. *Estimating Performance from Prior Experiences*

The alignment profile $\mathbf{X}(t)$ provides detailed information about a robot's performance on a task. When the alignment profile identifies no or few assumption violations, the robot assumes it performs at its *purported performance level*, denoted as $\hat{p}(t)$ at time t. $\hat{p}(t)$ encodes the robot's expected performance under normal circumstances. However, assumption violations mean that the robot is possibly operating in a scenario for which its generators are not designed, and thus its performance is likely to vary from $\hat{p}(t)$. In such cases, we propose that $\mathbf{X}(t)$ composes a scenario-independent feature set that serves as input to prediction algorithms that estimate the robot's performance. Thus, AAT estimates the robot's performance at time t, denoted $p(t)$, as a function of the purported performance $\hat{p}(t)$ and the alignment profile $\mathbf{X}(t)$:

$$p(t) = f(\hat{p}(t), \mathbf{X}(t)). \tag{2}$$

This paper advocates neither for a specific means for determining or estimating the function f, nor a specific structure of $p(t)$; many are possible. However, in the next section, we demonstrate a simple prediction function f for a robot navigation task that represents $p(t)$ as a probability distribution. Empirical results show that this prediction function allows a robot to effectively estimate its performance in previously unseen environments, thus illustrating the usefulness of AAT.

III. Case Study: Robot Navigation

This section illustrates how AAT, defined by Eqs. 1-2, can be used to perform proficiency self-assessment in a navigation task. After describing the task domain and robot system, we then describe an AAT implementation and analyze its ability to perform proficiency self-assessment.

A. *Task Domain*

The (simulated) robot can spin in place in either direction or move straight forward. It is equipped with two sensors: a camera that looks down on the world from above and a sensor that detects when the robot is on a charging pad.

The robot's task is to navigate through the world to its charger within a certain amount of time. This paper considers the four scenarios shown in Fig. 3. These simulation environments include the robot (shown as a blue circle in the figures) and its charger (shown as a red square in the figures). All other entities shown in the figures, including black line

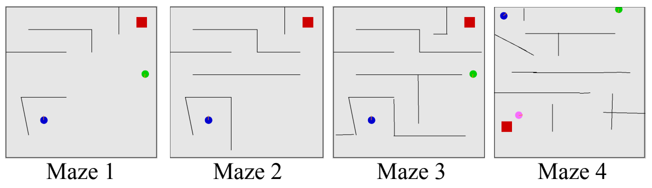


Fig. 3. Four worlds (or mazes) used in our study. The blue (circle) robot is tasked with getting to its charger (the red square). Black line segments and other robots (green and pink circles) are obstacles in the environment.

TABLE I

COMBINATION OF VARIATIONS USED IN THE SCENARIOS OF OUR STUDY

Combination of Variations	**Range of Parameters**
Camera noise-Robot bias	Noise: 0 to 0.3; Bias: -2 to 2
Camera noise-Robot speed	Noise: 0 to 0.3; Speed: 1 to 11m/s
Camera distortion-Robot bias	Distortion: -2 to 2; Bias: -2 to 2
Camera distortion-Robot speed	Distortion: -2 to 2; Speed: 1 to 11m/s

segments and pink and green robots (shown as circles), are obstacles through which the blue robot cannot pass.

Performance is determined by the time taken by the robot to reach the charger. The four worlds shown in Fig. 3 are designed to represent different difficulty levels, determined by the initial distance between robot and charger as well as the number and placement of obstacles. The *performance standards* must be tailored to each world since each world is different. Acceptable performances in Mazes 1-4 are subjectively set such that the robot should reach the charger within 60, 150, 300, and 220 seconds, respectively.

To evaluate the ability of the robot to perform proficiency self-assessment, the simulator is configured to allow a human to act as a *foil* to the robot. The human can modify the robot's environment and its hardware, yielding 41 simulations in each world. We, acting as the foil, varied these scenarios by modifying the robot's camera sensor and actuators in two ways each: the noise and distortion of the camera image, and the speed and bias of the robot's wheels (negative bias causes the robot to drift right when moving straight, while positive bias causes the robot to drift left). Table I describes how these variations were randomized and combined together throughout the scenarios. These combinations were spread out evenly across the simulations in the four worlds.

B. Robot Generators

The autonomous robot system is equipped with three different decision-making algorithms (or generators) to perform the navigation task: (1) a mapper, (2) a path planner, and (3) a controller. The *mapper* takes as input the camera image and creates a map of the environment by detecting and localizing itself and its charger in the world. It also creates an obstacle map of the world. Mapper assumptions are: (1) the camera produces up-to-date images, (2) the camera is in the expected location, is oriented downward, and has the assumed view-angle, (3) the camera sees color according to specification, (4) the camera has low noise and distortion, (5) the robot is blue and the charger red, and no other objects in the environment are those colors, (6) the robot and charger are visible in the camera image, and (7) all obstacles are also

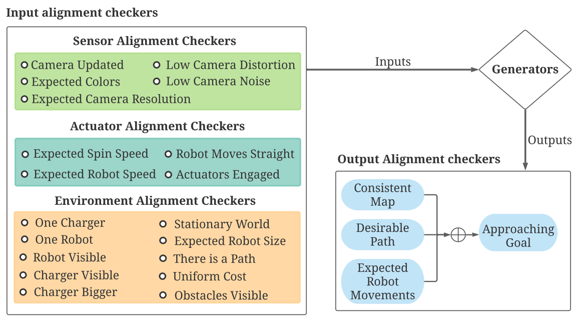


Fig. 4. The list of alignment checkers implemented in our robot system.

visible and are not white in color. The assumed output of the mapper is a consistent and accurate map of the world.

The *planner* takes as input the map, created by the mapper every time a new camera image is received, to plan a path from the robot to the charger using RRT* [17]. In addition to assuming the map created by the mapper is correct, the planner assumes: (1) the robot is of the assumed size and shape, (2) the world is stationary (other than robot movement), (3) there is a path to the charger that can be found within 1500 iterations of RRT*, and (4) the open space in the world has uniform cost (i.e., the robot as easily through one open space as another). The assumed planner output is a planned path with a consistent path length.

The *controller* takes as input the planned path and outputs commands to the robot's actuators, which move the robot along the chosen path to the charger. In addition to assuming the path selected by the planner is desirable, the controller assumes: (1) the robot's actuators are engaged (and react to the controller's commands), (2) the robot spins at the expected speed, (3) the robot moves straight when going forward, and (4) the robot moves at the expected speed. The expected output of the controller are wheel movements that move the robot in the expected manner through the world. Together with the expected output of the other two generators, the generators are assumed to cause the robot to approach the charger at an expected rate.

C. Evaluating Generator Assumptions

The AAT process requires identifying the assumptions made in the creation of the generators and then assessing the veracity of these assumptions. To do this, we designed *alignment checkers* for each of the assumptions listed in the previous subsection. (As an exception, we did not create an alignment checker to verify that the camera was located and oriented as expected due to the difficulty.) These alignment checkers are listed in Fig. 4. Each alignment checker returns a boolean value at each time step depending on whether the alignment checker deemed the corresponding assumption to be true or false. These valuations create the alignment profile $\mathbf{X}(t)$ that is used by the performance estimator.

D. Estimating Performance

When all assumptions are met, the robot's behaviors are generally understood, and the decisions made by its generators lead it to consistently navigate to its charger. In such situations, the robot can effectively estimate its performance on the task as the purported performance $\hat{p}(t)$, defined as the total time it takes to reach the goal:

$$\hat{p}(t) = t + g(t). \tag{3}$$

Here, t is the time elapsed in the mission so far and $g(t)$ is the estimated remaining time to complete the task. Experiments conducted under nominal circumstances showed that:

$$g(t) = 4|P| + \frac{1.2 \cdot \mathrm{dist}(P)}{\mathrm{v}_{\max}}, \tag{4}$$

where P is a set of segments defining the planned path from the robot's current position to its charger, $|P|$ is the number of segments in the path, $\mathrm{dist}(P)$ is the length of the path P, and $v_{\max}$ is the robot's assumed maximum speed. The first portion of Eq. (4) (i.e, $4|P|$) indicates that it takes the robot approximately 4 seconds on average to orient to the new direction after each segment. The second half of this equation specifies the average time required to traverse the segments of the path when assumptions are met.

When generator assumptions are violated, Eq. (3) is unlikely to be an accurate estimate of the robot's performance. Thus, AAT estimates the robot's performance on the task using a function of the form specified in Eq. (2). The robot estimates this function f from its prior experiences using a k-nearest neighbor algorithm. Specifically, the algorithm computes a set of data samples Y taken at periodic time stamps throughout past operations. Each sample $y \in Y$, is a tuple denoted by $y = (\tau, \bar{s}, \zeta, z_1^\tau, \cdots, z_M^\tau, \eta)$. Here, τ is the time in the mission at which the sample was taken, $\bar{s}$ is the average speed the robot has traveled so far up to sample time τ, ζ is the proportion of the distance the robot has traveled to the charger from its initial position, $\eta = \frac{T}{\hat{p}(\tau)}$ is a multiplier specifying the amount of time beyond the nominally predicted time the robot actually required to reach the goal in that past experience, T is the realized time-to-completion in that scenario, and $z_i^\tau = \lambda z_i^{\tau-1} + (1-\lambda) x_i^\tau$ (where $z_i^0 = 0$ and $\lambda \in [0,1]$; we used $\lambda = 0.3$) is an exponential average of the boolean values of assumption i as computed by the alignment checker in question.

The kNN algorithm for estimating f estimates the robot's performance at any time using the $k = 15$ samples in the set Y that are deemed to be the closest to the current mission state. We compare three different distance functions, each utilizing a different set of predictors. The first prediction function is based, not on AAT, but on the *rate-of-work* (RoW) the robot has accomplished in the mission so far. Formally, let s be the sample that denotes the current mission state. The distance between s and any sample $y \in Y$ is then:

$$\mathcal{D}_{\mathrm{RoW}}(s, y) = 2|s_{\bar{s}} - y_{\bar{s}}| + 3|s_\zeta - y_\zeta|. \tag{5}$$

Here, the tuple variable for each subscript is given as a subscript to the sample (e.g., $y_{\bar{s}}$ denotes the average speed in sample y). Thus, in this distance function, past experiences are selected based on similarities in how fast the robot was approaching the charger as well as the robot's current progress on the mission. The coefficients used in Eq. 5

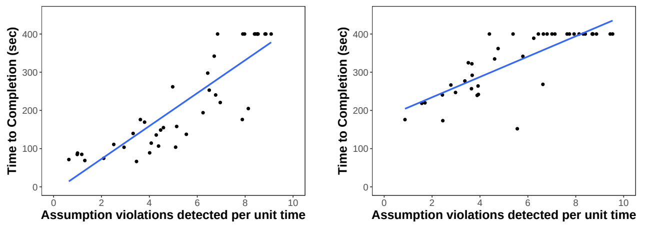


Fig. 5. The average number of detected violations of assumptions per unit time compared with task performance (completion time) in Mazes 2 (left) and 3 (right). Similar trends were found in Mazes 1 and 4. Blue lines depict linear fits to the data. Simulations were automatically terminated after 400 seconds, meaning that completion times marked as 400 seconds could have potentially been much higher had the simulation been run to completion.

were chosen to weight the two aspects of RoW used in this study, though these choices were not carefully scrutinized for optimality, suggesting better fit could be obtained.

The second distance function is based on the assessments made by alignment checkers and is given by:

$$\mathcal{D}_{\mathrm{Checkers}}(s, y) = \sum_{i=1}^{M} |s_{z_i^\tau} - y_{z_i^\tau}|. \tag{6}$$

In this distance function, the robot seeks past experiences for which its assessments of its assumptions are similar.

Finally, the third distance function combines the first two distance functions, and is given by:

$$\mathcal{D}_{\mathrm{Checkers+RoW}}(s, d) = \mathcal{D}_{\mathrm{RoW}}(s, d) + \mathcal{D}_{\mathrm{checkers}}(s, d). \tag{7}$$

Let $N \subseteq Y$ be the subset of k samples from the robot's set of past experiences Y deemed to be the closest to the current state s. Then, each sample $n \in N$ predicts the performance of the robot to be:

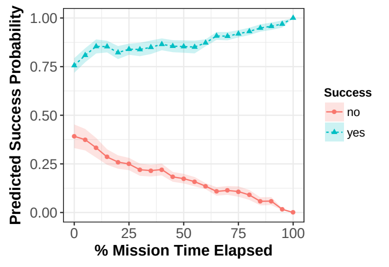
$$p_n(t) = \hat{p}(t) \cdot n_\eta. \tag{8}$$

In this way, the k samples in N provide a distribution of the robot's performance, where each sample $n \in N$ is weighted inversely proportional to its distance from s.

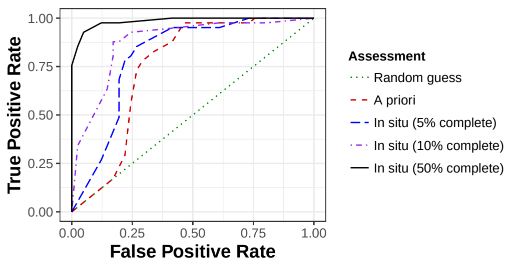
Combined with Eq. (8), the distance functions specified in Eqs. 5-7 specify prediction functions with different predictors. All three prediction functions use the purported performance as a predictor. However, only the prediction functions that use Eqs. 6 and 7 utilize AAT. The other prediction function is used as a point of comparison to assess the value of the alignment profile in predicting performance.

E. Results

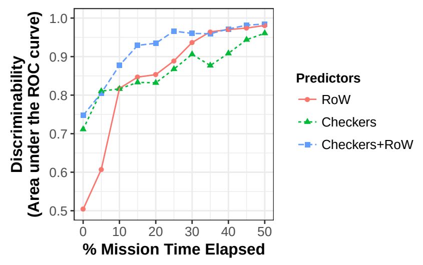
We first observe the correlation between robot performance (measured as time-to-completion) and the number of detected assumption violations per unit time. This correlation is visualized in Fig. 5 for two of the worlds. In each world, task performance is negatively correlated with the number of violations in assumptions. As the number of detected assumption violations per unit time increases, completion time also tends to increase. Pearson correlations confirm statistically significant ($p < 0.001$) and strong correlations between the number of alignment violations and completion times ($r = 0.778$, 0.874, 0.802, and 0.890 for Mazes 1-4,



(a)



(b)



(c)

Fig. 6. AAT effectively discriminates between success and failure. (a) The average predicted probability of success over time (using the distance function in Eq. 7) for cases in which the robot completed and did not complete the mission on time. Error ribbons show the standard error of the mean. (b) ROC curves showing the ability of the performance estimator to determine whether the robot's performance meets an acceptable threshold at various points in time (using the distance function in Eq. 7). (c) The discriminabilty of the predictor (as measured by the area under the ROC curve–0.5 indicates no discriminability) over time using the three different distance functions.

respectively). Thus, tracking the veracity of assumptions can be useful to understanding robot performance.

Next, we evaluate the ability of the prediction functions created using Eqs. 5-8 to produce predictions that can be used in proficiency self-assessment. To do this, we used data collected in the scenarios conducted in Mazes 1 and 2 as training data to estimate the prediction functions, and then tested the robot on scenarios performed in Mazes 3 and 4. In particular, we used the prediction functions to estimate the probability that the robot would reach its charger within an acceptable amount of time (defined as 300 and 220 seconds in Maze 3 and 4, respectively) in each scenario.

Fig. 6a shows that when both rate-of-work and assumption tracking are used (Eq. 7), the prediction function effectively differentiates between scenarios where the robot succeeds and scenarios where the robot does not succeed. In instances in which the robot fails to complete the task on time, the algorithm estimates that the success probability is much lower than the estimates of the success probability for a robot that eventually succeeded. These differences in predicted probability of success began in the early stages of the mission, suggesting that the predictor can make effective *a priori* assessments (before beginning to move). Additional analysis is provided in Fig. 6b, which shows receiver operating characteristic (ROC) curves for predicting success or failure at various times throughout the mission.

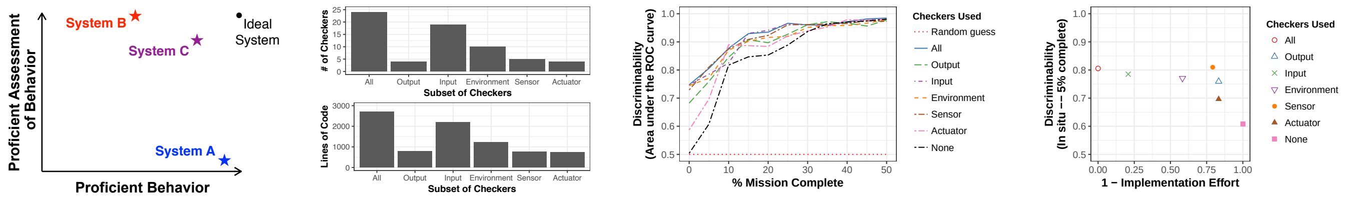
Finally, Fig. 6c compares the effectiveness of the three distance functions (Eqs. 5-7) by plotting the area under the ROC curve over time. The RoW predictor (Eq. 5), which make the assumption that the robot's performance in the task up to the current time t is indicative of future performance, are less effective early in the task, though this predictor eventually has high discriminability. On the other hand, alignment checkers produce reasonably good predictions early in the mission (though discriminability rises more slowly over time), thus producing more effective assessments earlier. The combined predictor (which uses both RoW and alignment checkers) is the most successful over all.

IV. Trade-offs in the Design Space

While the case study described in the previous section illustrates that AAT can effectively be used in the design of autonomous robot systems to effectively perform proficiency self-assessment in previously unseen environments, we observed that it took more time and effort for us to implement AAT in these systems than it took for us to create the generators that determined the robots' behaviors. For example, the generators for the robot system in the prior section (including sensing, planning, and acting) required just 1100 lines of computer code. However, the current implementation of the AAT capabilities (including assessing input and output assumptions and defining the prediction function f) required over 2700 lines of code.

This highlights a potential trade-off in the design and development of autonomous robot systems. Should system designers focus more on developing proficient autonomous behavior or on creating proficient assessments of behavior? This trade-off is visualized by Fig. 7a, which compares several hypothetical systems. System A represents a system in which designers spent all of their time developing proficient robot behaviors and ignored the problem of proficiency self-assessment. Such a system would be acceptable in scenarios in which either the robot never fails or when it is unnecessary to identify failures. However, in situations in which failing to identify failures has large implications, System B (in which system designers focus more extensively on developing proficiency self-assessment capabilities) would be a better choice. System C illustrates a potential *middle ground*. In System C, the robot has neither ideal behavior generation nor ideal behavior assessment, but efficient investment in both directions leads to a robot that is reasonably proficient in both behavior generation and behavior assessment.

This tradeoff space suggests that modest proficiency assessment capabilities might be sufficient in many situations. For example, limiting the number of assumptions that are tracked in AAT would theoretically reduce assessment coverage therefore decrease the accuracy of proficiency assessments. However, the case studies showed that the assumption veracities appear to be correlated with each other. For example, changes in the veracity of assumptions made about the robot's sensors and actuators tend to correlate with (and likely cause) violations in assumptions made about generator outputs. In this example, violated assumptions about generator output are symptoms of violated input assumptions.



(a) (b) (c) (d)

Fig. 7. An illustration of the trade-off between implementation effort and the discriminability of AAT. (a) Hypothetical trade-off for designing robots that have both proficient behavior and are proficient at assessing their behavior. (b) Implementation effort to create various subsets of assumption checkers as measured by the number of the number of checkers (top) and lines of computer code (bottom). (c) The discriminability of AAT as a function of mission time based on the area under the ROC curve for various subsets of assumption checkers. (d) The tradeoff between implementation effort and discriminability after 5% of the mission was complete. *1 - Implementation Effort* is computed as one minus the percentage of checkers used. An ideal outcome would be in the upper right-hand corner – which would indicate perfect discriminability achieved without any implementation effort.

Additionally, assessments of some input assumptions tend to correlate with each other. For example, violating assumptions about the amount of sensor noise often causes the robot to believe that its actuators are not behaving according to assumptions as well. These observations give credence to the idea that not all assumptions need to be tracked for the robot to self-assess its proficiency.

The assessment-behavior tradeoff space was explored by evaluating how well AAT estimated robot performance in the first case study (a robot navigation task) given various subsets of checkers identified in Fig. 4. Since these subsets require different amounts of effort to implement, as measured by both lines of code and number of alignment checkers (Fig. 7b), it is informative to evaluate how they impact the ability of the robot to assess its performance.

Fig. 7c uses the area under the ROC curve to compare the ability of each subset of checkers to accurately discriminate between success and failure over the initial stages of the robot's mission. The figure shows that all subsets of checkers differentiate between success and failure substantially better than random guesses even at the start of the task. After 10% of the mission is completed, all subsets or checkers produce similarly high discriminability. However, the subsets of Output and Actuator alignment checkers (Fig. 4) initially produced lower discriminability than the other subsets of checkers, suggesting that assessment accuracy is affected by which checker subset is chosen but that some subsets can be very accurate. These results indicate that not all generator assumptions need to be tracked in order for the robot to do a reasonably good job of assessing its own proficiency, but that the subset of checkers chosen is important.

The trade-off between the implementation effort and discriminability of AAT in the first case study presented in the previous section is plotted in Fig. 7d. Ideally, the system would produce full discriminability at no effort to the system creator, which would produce a point in the upper right-hand corner of this figure. In practice, the figure shows that set of Sensor alignment checkers (Fig. 4) seems to provide a good trade-off in producing high discriminability with relatively low implementation effort in the scenarios tested.

We are not suggesting that Sensor alignment checkers will always produce an ideal trade-off between implementation effort and discriminability. Different systems and scenarios would likely produce different results with respect to these trade-offs. However, these results do demonstrate that not all assumption checkers need to be implemented for AAT to do a reasonably good job at proficiency self-assessment. Identifying effective subsets of checkers to implement can substantially reduce the effort required to implement AAT.

The tradeoff between time spent generating quality behaviors and performing quality assessment can be misleading. The metaphor of writing test cases for computer code is applicable. Frequently, doing unit testing in software engineering produces better code in less time. We hypothesize that something similar will likely occur with creating alignment checkers in parallel with creating behavior generators. Explicitly identifying and tracking assumptions made in the implementation of decision-making algorithms (or generators) can help to produce more effective robot behavior.

V. Conclusions and Future Work

This paper formalizes a method for designing robot systems that perform proficiency self-assessment. This method, called assumption-alignment tracking (AAT), is developed from the perspective that proficiency self-assessment is awareness of how one's generators (i.e., decision-making algorithms) interact and align with the environment(s), robot hardware, and task(s) under consideration. In AAT, the robot continually monitors the veracity of input and output assumptions made in the construction of its generators, and then uses these assessments to estimate the robot's ability to perform the task at hand. A robot navigation study demonstrated that AAT can perform informative proficiency self-assessment.

Future work is needed to better establish AAT's usefulness as a systematic approach to performing proficiency self-assessment. This includes evaluating AAT in a broader set of applications, better understanding the design trade-offs between generator quality and alignment checker coverage, and using assumption assessments to develop explanations [18] about the system's proficiency assessments. Further work is also needed to establish benchmarks for proficiency self-assessment [19]. Through these and other related effort, we believe that we can learn to systematically create autonomous robots and systems that can assess their own capabilities and limitations in previously unseen environments and scenarios.

References

[1] R. Dearden, T. Willeke, R. Simmons, V. Verma, F. Hutter, and S. Thrun, "Real-time fault detection and situational awareness for rovers: report on the mars technology program task," in *Proceedings of the IEEE Aerospace Conference Proceedings*, vol. 2, 2004, pp. 826–840.

[2] P. Zhang, J. Wang, A. Farhadi, M. Hebert, and D. Parikh, "Predicting failures of vision systems," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 3566–3573.

[3] S. Honig and T. Oron-Gilad, "Understanding and resolving failures in human-robot interaction: Literature review and model development," *Frontiers in psychology*, p. 861, 2018.

[4] D. Das, S. Banerjee, and S. Chernova, "Explainable AI for robot failures: Generating explanations that improve user assistance in fault recovery," in *Proceedings of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 351–360.

[5] Z. Han, D. Giger, J. Allspaw, M. S. Lee, H. Admoni, and H. A. Yanco, "Building the foundation of robot explanation generation using behavior trees," *To Appear in ACM Transactions on Human-Robot Interaction*, 2021.

[6] A. Alvanpour, S. K. Das, C. K. Robinson, O. Nasraoui, and D. Popa, "Robot failure mode prediction with explainable machine learning," in *2020 IEEE 16th International Conference on Automation Science and Engineering (CASE)*. IEEE, 2020, pp. 61–66.

[7] A. Ramesh, M. Chiou, and R. Stolkin, "Robot vitals and robot health: An intuitive approach to quantifying and communicating predicted robot performance degradation in human-robot teams," in *Companion of the 2021 ACM/IEEE International Conference on Human-Robot Interaction*, 2021, pp. 303–307.

[8] D. Schreckenghost, T. Fong, T. Milam, E. Pacis, and H. Utz, "Real-time assessment of robot performance during remote exploration operations," in *2009 IEEE Aerospace conference*. IEEE, 2009, pp. 1–13.

[9] M. Zillich, J. Prankl, T. Mörwald, and M. Vincze, "Knowing your limits-self-evaluation and prediction in object recognition," in *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011, pp. 813–820.

[10] A. Jauffret, C. Grand, N. Cuperlier, P. Gaussier, and P. Tarroux, "How can a robot evaluate its own behavior? a neural model for self-assessment," in *The 2013 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2013, pp. 1–8.

[11] T. Frasca, E. Krause, R. Thielstrom, and M. Scheutz, ""can you do this?" self-assessment dialogues with autonomous robots before, during, and after a mission," in *Proceedings of the HRI 2020 Workshop on Assessing, Explaining, and Conveying Robot Proficiency for Human-Robot Teaming)*, 2020, pp. 1–6.

[12] H. Huang, E. Messina, and J. Albus, "Autonomy level specification for intelligent autonomous vehicles: Interim progress report," in *Proceedings of the 2003 Performance Metrics for Intelligent Systems Workshop*, Gaithersburg, MA, 2003.

[13] J. W. Crandall, M. A. Goodrich, D. R. Olsen, Jr., and C. W. Nielsen, "Validating human-robot interaction schemes in multi-tasking environments," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 35(4), pp. 438–449, 2005.

[14] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[15] L. S. Fletcher, S. Teller, E. B. Olson, D. C. Moore, Y. Kuwata, J. P. How, J. J. Leonard, I. Miller, M. Campbell, D. Huttenlocher, A. Nathan, and F.-R. Kline, *The MIT – Cornell Collision and Why It Happened*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 509–548.

[16] C. Béné and L. Doyen, "From Resistance to Transformation: A Generic Metric of Resilience Through Viability," *Earth's Future*, vol. 6, no. 7, pp. 979–996, 2018.

[17] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," in *Proceedings of Robotics Science and Systems*, 2010, pp. 34–41.

[18] T. Chakraborti, S. Sreedharan, and S. Kambhampati, "The emerging landscape of explainable automated planning and decision-making," in *Proceedings of 2019 Workshop on Explainable Planning*, Berkeley, Ca, 2010.

[19] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018.