Green Urban Mobility with Autonomous Electric Ferries: Studies of Simulated Maritime Collisions using Adaptive Stress Testing

Jan-Marius Vatle,* Bjørn-Olav Holtung Eriksen,⋆ and Ole Jakob Mengshoel°

***Abstract*— With 90% of the world's goods transported by sea vessels, it is crucial to investigate their safety. This is increasingly important as autonomy is being introduced into sea vessels, which transport goods and people. To study the safety of an autonomous ferry's collision avoidance system, we consider the Adaptive Stress Testing (AST) method in this work. AST uses machine learning, specifically reinforcement learning, along with a simulation of a system under test—in our case, an autonomous and electric ferry—and its environment. Whether that simulation is fully or partially observable has implications for the integration into existing engineering workflows. The reason is that the fully observable simulation induces a more complex interface than the partially observable simulation, meaning that the engineers designing and implementing AST need to acquire and comprehend more potentially complex domain knowledge. This paper presents maritime adaptive stress testing (MAST) methods, using the world's first autonomous, electric ferry used to transport people as a case study. Using MAST in multiple scenarios, we demonstrate that AST can be productively utilized in the maritime domain. The demonstration scenarios stress test a maritime collision avoidance system known as Single Path Velocity Planner (SP-VP). Additionally, we consider how MAST can be implemented to test using both fully observable (gray box) and partially observable (black box) simulators. Consequently, we introduce the Gray-Box MAST (G-MAST) and Black-Box MAST (B-MAST) architectures, respectively. In simulation experiments, both architectures successfully identify an almost equal number of failure events. We discuss lessons learned about MAST including the experiences with both the Gray-Box and Black-Box approaches.**

I. INTRODUCTION

Among the 17 sustainable development goals (SDGs) of the United Nations, we find SDG 11:[1] "Make cities and human settlements inclusive, safe, resilient and sustainable." To meet this SDG, there is a move towards more sustainable transportation, for example by means of electric vehicles [30]. A recent development is the concept of small, electric, autonomous passenger ferries for urban areas. Such ferries can operate in networks on urban waterways, connecting cities across rivers, canals, harbor basins, and lakes. With growing cities in need of safe and sustainable transport for all, this new mobility model can solve challenges with increasing road congestion and emissions, enabling more citizens to walk or bike and combine those transportation modes with other forms of micro-mobility. Autonomous operation will be necessary to make this new mobility mode—the small, electric passenger ferry—truly scalable and enable ubiquitous availability.

Fig. 1: The world's first autonomous electric passenger ferry, milliAmpere 2 (mA2), in operation in Trondheim, Norway.

With this backdrop, researchers at NTNU have since 2016 been developing supporting concepts and algorithms and deployed two operational ferry prototypes: the milliAmpere 1 and 2 [24], [3]. In September 2022, the milliAmpere 2 (mA2) was put into trial operation in Trondheim, Norway, transporting passengers across a canal in the city center as shown in Figure 1. This became the world's first autonomous passenger ferry in public operation, completing 400 crossings and transporting about 1,500 passengers over a period of three weeks. This again leads to the maritime transport company Torghatten and the NTNU spin-off company Zeabuz launching the world's first commercial autonomous passenger ferry, MF Estelle, in June 2023 in Stockholm, Sweden.[2]

A previous version of the autonomous navigation system for mA2 is the system under test (SUT) in this work. Autonomous electric ferries like mA2 operate in complex stochastic environments. As a consequence, it is very hard to entirely eliminate their failures. Furthermore, real-world testing can be too dangerous or too time-consuming to perform during development, and the use of formal verification such as model checking [5] may be too complex. Simulation-based techniques resorting to statistical considerations can address these issues, and simulation of autonomous vehicles and

*Jan-Marius Vatle is currently with KodeWorks, Trondheim, Norway. This work was done while he was at NTNU. janmarius.vatle@outlook.com

⋆Bjørn-Olav Holtung Eriksen is with Zeabuz, Trondheim, Norway. bjorn.olav.eriksen@zeabuz.com

°Ole Jakob Mengshoel is with the Department of Computer Science, NTNU, Trondheim, Norway. ole.j.mengshoel@ntnu.no

[1] https://sdgs.un.org/goals/goal11

[2] https://www.zeabuz.com/torghatten-and-zeabuz-make-history-in-stockholm/

vessels is well-established [26], [15], [23], [10]. Moreover, work done with airborne collision avoidance systems and autonomous vehicles shows successful safety validation by applying a stress testing framework called Adaptive Stress Testing (AST). The framework is based on reinforcement learning (RL) techniques and adaptively finds the most likely path to a failure event for the SUT in a simulated environment [15], [12], [17], [19], [10].

Among failure events, collisions are prominent in the maritime sector. The European Maritime Safety Agency (EMSA) reports the following in 2023 [1]: "From 2014 to 2022, there was a total of 6,781 injuries in 5,941 marine casualties and incidents, the average of injuries in that period was 753 injuries per year. [...] The main events resulting in injuries from 2014 to 2022 were 'slipping / stumbling and fall' for occurrences with persons and 'collision' for occurrences with ships."

To test a collision avoidance system in the maritime domain, this work[3] proposes an architecture called Maritime Adaptive Stress Testing (MAST). MAST extends the existing AST architecture [15], [17] for the purpose of testing maritime autonomous collision avoidance systems, focusing on the Single Path Velocity Planner (SP-VP) used by mA2. Two MAST architecture variants are developed and studied, namely Gray-Box Maritime Adaptive Stress Testing (G-MAST) and Black-Box Maritime Adaptive Stress Testing (B-MAST). They are used, respectively, for fully observable (Gray-Box) and partially observable (Black-Box) simulations of mA2's performance when encountering other vessels in 1,000s of simulation runs. Using the Gray-Box simulator with G-MAST requires more domain knowledge than testing with a Black-Box simulator with B-MAST. On the other hand, G-MAST gives more control than B-MAST. Given this trade-off, investigating empirically the performance of G-MAST versus B-MAST for the purpose of adversarially generating failure events is of interest. Empirically, we find that using B-MAST, approximately 8.1% of the simulations resulted in failure events, while approximately 9% of the G-MAST simulations gave failure events. This demonstration suggests that both the G-MAST and B-MAST approaches can be used to induce failure events in the maritime setting. Furthermore, these results increase the confidence in mA2's behavior in a range of situations when using SP-VP.

II. BACKGROUND

In this section, we introduce the maritime setting and, in particular, the dynamics of marine vessels including mA2. We then cover mA2's motion planning method before discussing previous research on stress testing, including AST. Much previous work related to AST focuses on airborne collision avoidance systems and autonomous systems, and we consider key differences to our research in this paper.

A. Marine Vessel Dynamics

The dynamics of a marine vessel are often described using Six Degrees of Freedom (6DOF), which are the set of independent displacements and rotations that define the displaced position and orientation of the vessel. For marine vessels that do not have actuation in all 6DOF and operate under certain conditions, it is possible to simplify the simulation and use reduced-order models [8]. For mA2, the following assumptions apply [8]: First, marine vessels operating at relatively low speeds can neglect the Earth's rotation, and thereby the Earth-centered, Earth-fixed coordinate system ECEF-frame can be considered to be inertial. Second, for marine vessels operating in a local area with approximately constant longitude and latitude, an Earth-fixed tangent plane on the surface of the Earth is used for navigation.[4] Third, for marine vessels that operate in the calm sea one can assume that the displaced orientations in roll and pitch are to be arbitrarily small. Therefore, the components corresponding to heave, roll, and pitch can be neglected.[5]

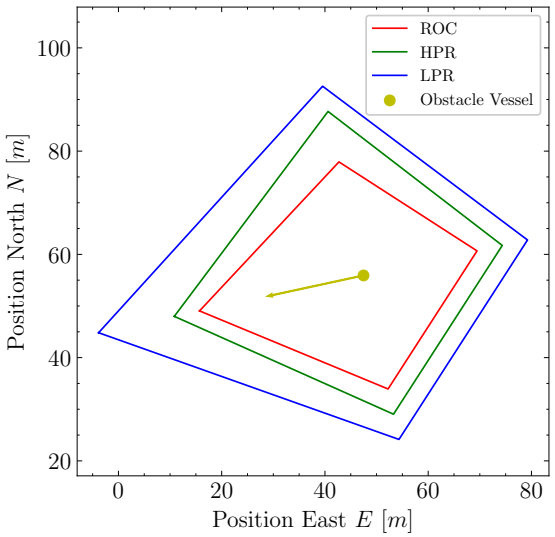


Fig. 2: This is SP-VP's obstacle representation. The obstacle, a point, and its heading, a line segment, are in yellow. In red is the Region of Collision (ROC), in green is the High Penalty Region (HPR), and in blue is the Low Penalty Region (LPR).

The above three assumptions make it possible to describe the dynamics of a marine vessel such as mA2 using Three Degrees of Freedom (3DOF). Equation 1 expresses the positions and orientations of the marine vessels in 3DOF in vector form, represented by $\boldsymbol{\eta}$:

$$\boldsymbol{\eta} = [N, E, \psi]^T. \tag{1}$$

Here, N and E represent the marine vessel's displaced positions in the reference frame, and ψ represents the displaced orientation.

Equation 2 shows the mathematical notation for the velocity components, which is divided into linear and angular

[3]This paper builds upon the MS Thesis of Jan-Marius Vatle [31].

[4]Due to this, the NED-frame can be assumed to be inertial.

[5]*Roll*, *pitch*, *yaw*, *surge*, *sway* and *heave* describe different ship motions. *Roll*, *pitch*, and *yaw* are rotaional motions while *surge*, *sway*, and *heave* are translational motions.

velocities, represented by $\boldsymbol{\nu}$:

$$\boldsymbol{\nu} = [u, v, r]^T. \tag{2}$$

Here, u and v represents the linear velocities in *surge* and *sway*, respectively, while r represents the angular velocity in *yaw*. For horizontal plane models, the kinematic equations can, when assuming calm sea and no weather such as wind, be expressed as:

$$\begin{aligned} \dot{\boldsymbol{\eta}} &= \boldsymbol{R}(\psi)\boldsymbol{\nu}, \\ \boldsymbol{M}\dot{\boldsymbol{\nu}} + \boldsymbol{C}(\boldsymbol{\nu})\boldsymbol{\nu} + \boldsymbol{D}(\boldsymbol{\nu})\boldsymbol{\nu} &= \boldsymbol{\tau}. \end{aligned} \tag{3}$$

Here, $\boldsymbol{M}$ is the mass matrix, $\boldsymbol{C}(\boldsymbol{\nu})$ is the centripetal and Coriolis matrix, and $\boldsymbol{D}(\boldsymbol{\nu})$ is the damping matrix. Since the only rotation is about the z-axis (yaw), we get $\boldsymbol{R}(\psi)$ in Equation 4 expressed as:

$$\boldsymbol{R}(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}. \tag{4}$$

The marine vessels in the simulator, including mA2, use Equation (3) and Equation (4) for their dynamics.

B. Single Path Velocity Planner (SP-VP)

With the aim of ensuring collision-free maneuvers from the start to its goal waypoint, mA2 uses SP-VP as its motion planning system. SP-VP behaves as a maritime autonomous collision avoidance system [28], [29], [2]. SP-VP is developed for autonomous passenger ferries operating in confined waters, which is an area of the sea with a relatively narrow waterway relative to the marine vessel's ability to maneuver. SP-VP is provided with a predefined waypoint mission, which must be collision-free with respect to static obstacles, such as islets and breakwaters. The collision avoidance problem thus becomes a velocity planning problem, which means that mA2 only plans a velocity profile with speed change maneuvers and does not apply course change maneuvers.

The SP-VP method does not fully align with the International Regulations for Preventing Collision at Sea (COLREGs) when maneuvering in sight of other vessels [3]. The reason is that COLREGs rules rely more on course change maneuvers than speed change maneuvers to avoid collisions. SP-VP tracks dynamic obstacles such as other vessels with an update rate of 0.25 Hz, and applies a simplified obstacle representation for robustness and ease of computation (see Figure 2). Three diamond-shaped safety regions, each a Region of Collision (ROC), are shown. The ROCs surround the point considered to be the obstacle represented in a North-East-Down frame. The ROCs are slightly asymmetric with an increased size on the starboard side. This is a common approach for "motivating" collision avoidance algorithms to choose more COLREGs-compliant maneuvers.

An ROC includes an obstacle's dimensions and the dimensions of mA2. Consequently, mA2 can be considered as a point when constructing SP-VP's visibility graph. The regions can be calculated using the method of Thyri *et al.* [29]. The obstacle vessels are transformed into a path-time space and are then constructed as a conditioned visibility graph and traversed with Dijkstra's algorithm [7] in order to compute a collision-free velocity profile. A similar decomposition method is proposed by Kant *et al.* [11].

SP-VP is summarized here to provide the domain knowledge necessary to read this paper. We take SP-VP for granted as it is used for the mA2 ferry. However, we do not test every detail of SP-VP. Other methods to control the ferry exist, such as model-predictive control [10], but SP-VP is an important method for mA2.

C. Adaptive Stress Testing (AST)

AST was introduced by Lee *et al.* [15] in 2015 to test airborne collision avoidance systems. AST uses Reinforcement Learning (RL) to stress test a prototype of the next-generation Airborne Collision Avoidance System (ACASX). The goal is to find, using a simulator, the most likely path to a near mid-air collision [15], [17]. This aerospace setting induces a large search space, in which exhaustive search is unrealistic and failure states can be hard to find.

Lee *et al.* discuss different AST architectures, with an important consideration being whether the simulator is fully observable or partially observable; this is what we refer to as Gray-Box and Black-Box simulation respectively. A variant of Monte Carlo Tree Search (MCTS), Monte Carlo Tree Search for Seed-Action simulators (MCTS-SA), is proposed. MCTS-SA only requires access to the pseudorandom number generator of the simulator to overcome partial observability and uses progressive widening. Progressive widening is introduced due to the large action space consisting of all possible pseudorandom seeds. This simulator has deterministic behavior since the same pseudorandom seed always leads to the same next state from the previous state. In other words, the transition behavior of the simulator is deterministic [17].

Lee *et al.* [18] extend the AST framework with regression testing to find failures that occur in one system but not in another. This extended framework is called Differential Adaptive Stress Testing (DAST). DAST is used to compare ACASX with Traffic Alert and Collision Avoidance (TCAS), to test the performance of ACASX relative to TCAS. DAST works by searching two simulators simultaneously and maximizing the difference between their outcomes [18].

It is essential to understand how failures occur to be able to design, evaluate, and certify safety-critical systems. In this context, AST and DAST contributed to the certification case of the ACASX, which led to the technical acceptance of ACASX [17].

Lipkis *et al.* [19] use AST to test the Airborne Collision Avoidance System for smaller UASs (ACAS sXu). Their work aims to provide detect-and-avoid capability for small unmanned aircraft operating beyond line-of-sight. They use a different approach compared to Lee *et al.* [17], in that they apply Deep Reinforcement Learning (DRL) with a Proximal Policy Optimization (PPO) algorithm [25]. The goal is to search more efficiently through the large and continuous state space. Using this approach they found several failure events, which were useful for the refinement of ACAS sXu.

An autonomous vehicle needs to be equipped with a decision-making system. Koren *et al.* [12] present a method for testing the decision-making system of autonomous vehicles. They formulate the problem as a Markov decision process and use RL algorithms to find the most likely failures. They show that extending AST to use DRL improves the efficiency of the original AST, which uses an MCTS variant. Koren *et al.* simulate autonomous vehicle scenarios involving pedestrians approaching a crosswalk. They conclude that DRL can find more likely failure scenarios than MCTS in addition to finding them more efficiently [12].

We now discuss this work's relationship to previous work. First, we note that most previous AST research has focused on aircraft [15], [16], [17], [19] or cars [12]. That said, there is some AST research in the maritime setting [10]. Similar to us, Hjelmeland *et al.* study AST as applied to an autonomous small passenger ferry [10]. They demonstrate that AST can be used to find failures, specifically collisions with adversary vessels. Different from us, they do not consider the interaction of mA2 using SP-VP with multiple obstacles nor the question of Black-Box versus Gray-Box simulation for AST. In fact, we are not aware of any previous work that empirically studies the pros and cons of Black-Box versus Gray-Box simulation for AST.

III. Maritime Vessel Simulator

Our simulator is a continuous-space and discrete-time simulator that uses the assumptions presented in Section II-A and flat earth navigation with 3DOF. It is purpose-made for testing collision avoidance systems like SP-VP. The simulator provides two types of vessels, mA2 and obstacle vessels. The mA2 ferry operates with complex dynamics and is always equipped with the SP-VP controller. The obstacles are first-order control systems using first-order differential equations for their transfer function and proportional–integral–derivative (PID) controllers for speed and heading control [32]. The marine vessels' heading ψ operates in the unit circle; their dynamics are based on the kinematic equations discussed in Section II-A.

Figure 3 shows an example simulation frame consisting of mA2 and two first-order obstacle vessels. The positions of the three vessels are represented by solid circles with different colors. Diamond-shaped dashed lines around the obstacles are true ROCs. The true ROCs differ from the ROCs that SP-VP uses to represent obstacles by not being affected by noise or delayed due to SP-VP's update rate of 0.25 Hz.

A. Failure Events

A **failure event** is defined as a collision between mA2 and one or more obstacles in the simulation. More formally, a collision occurs if mA2 intersects one of the obstacle vessels' true ROC. The failure events can, at an intuitive level, be classified into: *side collisions*, which are when a vessel is struck on the side by another vessel; *bow-on collisions*, which is when two vessels strike each other head-on; and *stern collisions* which take place when one vessel runs into the aft of another. Other factors such as the speed and number of obstacles also play important roles, as will be seen in Section V.

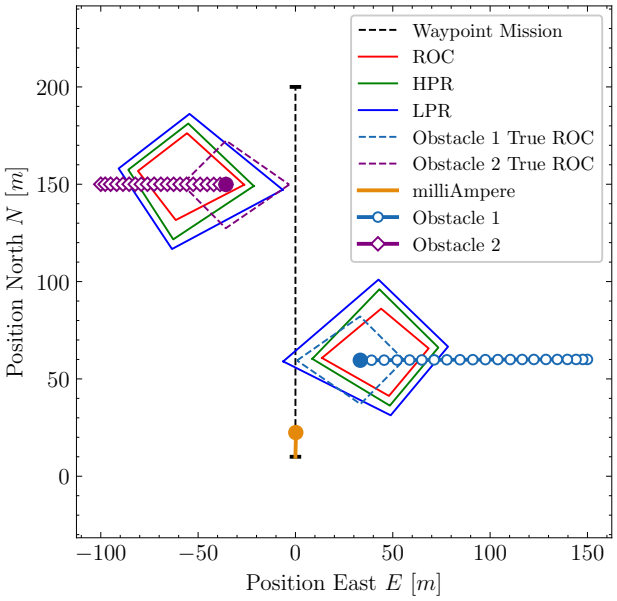


Fig. 3: Two obstacles and mA2 are shown. The mA2 vessel and its course are shown with an orange dot and solid line segment, respectively. The dashed black line shows mA2's waypoints. The solid red, green and blue lines, are ROC, HPR and LPR of SP-VP, respectively. The obstacles are represented by markers for every second of the simulation. Obstacle 1 is in light blue with circles as markers and Obstacle 2 is in purple with diamonds as markers. True ROC is an obstacle's region that indicates an actual collision, it differs from the SP-VP ROC due to noise given to the collision avoidance system.

While using AST to search for failure events, we discovered that most found events were caused by obstacle vessels colliding with mA2 while mA2 was stationary, similar to a kamikaze attack on a stationary ship. However, collisions when mA2 is not moving are not really a failure in its maritime collision avoidance system. Thus we redefined a failure event as a collision when mA2 was also in motion. After the search was finished, we manually reviewed the failure events found by AST. Since AST assigns the highest scores to the most likely failure events, we selected those. We then looked at the failure events that we found most interesting in terms of their realism and analyzed them further. The overall approach is illustrated in Figure 4.

B. Simulator Interface

To make it possible for AST's RL-agent to interact with the simulator, the following functions are defined. These functions are fundamental for both the G-MAST and B-MAST architectures investigated in this work.

The **steer-obstacles** function makes it possible for the RL-agent to steer the obstacle vessels. A reference surge speed u_* and vessel reference heading ψ_* is chosen by the RL-agent and given to the obstacle controllers, which in turn

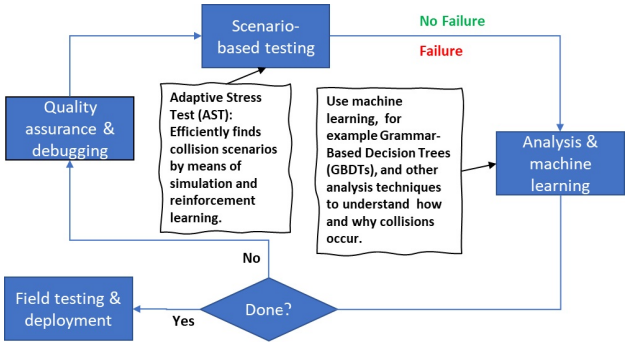


Fig. 4: An overview of how MAST, which is the key part of "Scanario-based testing," fits into the overall engineering workflow. Two MAST variants are considered, namely G-MAST (Figure 5) and B-MAST (Figure 6).

controls the obstacles to achieve the reference speed and heading.

The **control-SP-VP-noise** function is used to add noise to the obstacle estimates that SP-VP uses. This is intended to replicate the behavior of a sensor-based tracking system which produces estimates with a certain amount of noise. To model this, the simulator uses a Gauss-Markov process.

The **is-failure-event** function checks whether a failure event, as defined in Section III-A, has occurred.

The **calculate-distance** function computes the distance between two vessels in the NED-frame, using Euclidean distance:

$$d(p,q) = \sqrt{(q_1 - p_1)^2 + (q_2 - p_2)^2}. \tag{5}$$

Here, p and q are points with Cartesian coordinates (p_1, p_2) and (q_1, q_2), respectively. When there are multiple obstacle vessels in the simulation, the average of all distances between mA2 and the obstacle vessels is returned.

The **calculate-transition-likelihood** function calculates the simulator's transition likelihood $p(x|s)$. This function is only needed for the B-MAST approach; the G-MAST approach calculates the transition likelihood in the reward function. In the simulator, a state transition occurs every $0.1s$ second. In each transition, mA2 is controlled by SP-VP, and the obstacle vessels are controlled by the RL-agent. During this transition, the vessels are moved to their next position in the NED-frame based on their given $\boldsymbol{\nu}$ and $\boldsymbol{\eta}$, as described in II-A. The simulator transitions are deterministic when using the Gray-Box approach, and deterministic given a pseudo-random seed input when using the Black-Box approach.

To reset the simulator to its initial conditions and re-set the initial seed, a **reset** function is implemented.

A **step** function is also implemented. In G-MAST, the step function takes in disturbances x while B-MAST uses pseudorandom seeds. These inputs are provided by the RL agent and given to the simulator. The step function interacts with the simulator at a rate of $1s$, but the simulator is updated at a rate of $0.1s$. However, the state s is only returned to the RL agent from the simulator with a rate of $1s$. More details about states and disturbances can be found in IV-C.

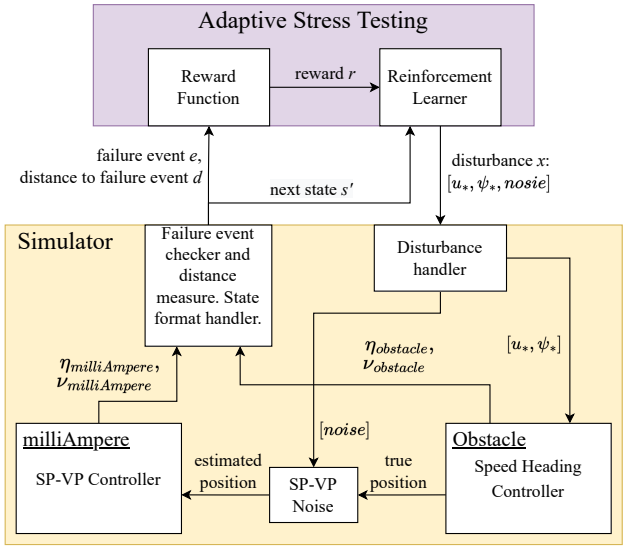


Fig. 5: This illustrates the G-MAST architecture. The RL-agent chooses disturbances that are sent to the simulator's disturbance handler. The simulator's disturbance handler sends the disturbances to the right parts of the simulator. The reference surge speed u_* and heading ψ_* are sent to the Speed Heading Controller. And the noise is given to the SP-VP tracker. The simulator then updates and transitions into its next state, and the state of the vessels is given to the failure event checker, distance measure, and state format handler. Then the simulator returns a boolean indicating if the state is a failure state or not e, the distance to failure d, and the next state of the simulator. The reward function calculates the transition likelihood in this architecture and the reward r is sent to the RL-agent.

IV. Adaptive Stress Testing Methods

We propose two adaptive stress testing architectures, G-MAST and B-MAST, for use in fully observable and partially observable simulators, respectively. The proposed architectures enable MAST usage with both fully and partially observable simulators. The G-MAST architecture (see Section IV-A) is designed for fully observable simulators. However, many simulators restrict access to some or all state information for confidentiality reasons, due to privacy concerns, or to make them more accessible to testers without domain knowledge. In other words, such simulators are not fully observable. Therefore, we introduce the B-MAST architecture (see Section IV-B) to be used with partially observable simulators.

A. Gray-Box Architecture

G-MAST extends the existing AST architecture by tailoring it to the maritime domain. G-MAST is a suitable solution when the simulator makes its environment variables and state available. The RL agent then samples the variables, or disturbances, directly from probability distributions that vary between disturbances (see Section IV-C).

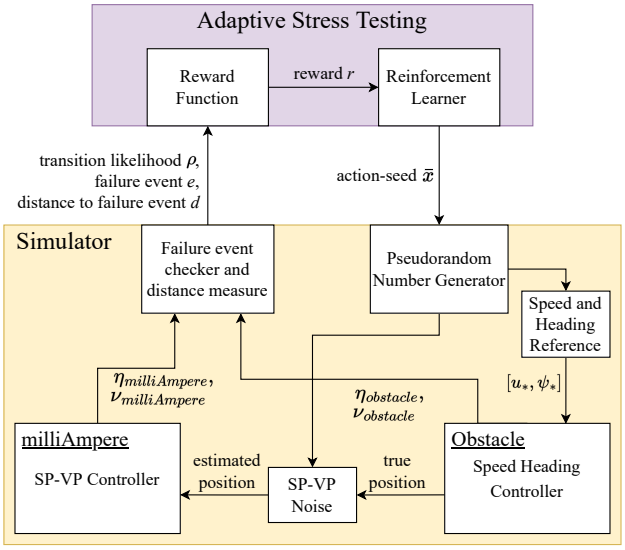


Fig. 6: This shows the B-MAST architecture. The RL-agent sends an action-seed to the simulator in each simulator step. The pseudorandom number generator in the simulator is used to sample environment disturbances internally. The disturbances are the reference surge speed u_* and heading ψ_* and noise to SP-VP. The transition likelihood is calculated internally in the simulator for B-MAST.

Figure 5 shows the G-MAST architecture and how it interacts with the maritime vessel simulator, using the functions discussed in Section III-B. It is possible to simulate more than one obstacle, but for simplicity, only one obstacle vessel is illustrated. In each iteration, the simulator checks if the current state is a failure event e, computes the distance to failure d, and formats the state to a one-dimensional vector which is sent back to AST. The reward function calculates the transition likelihood in this architecture and returns the reward r to the RL-agent.

B. Black-Box Architecture

The B-MAST architecture, see Figure 6, is suitable for simulators that do not reveal their environment variables and where all the updates of the simulator happen internally. Only a random seed, referred to as an action-seed $\bar{x}$, is chosen by the RL-agent and given to the pseudorandom number generator of the simulator. The pseudorandom number generator is then used by the Speed and Heading Reference handler to sample a random reference surge speed u_* and heading ψ_* for the obstacle's Speed Heading Controller. The pseudorandom number generator is further used to generate random noise in the SP-VP tracking system. The simulator works similar to the G-MAST approach, but the disturbances in the environment are now sampled internally in the simulator. The B-MAST approach is suitable, as an example, for simulators that are provided as software binaries not revealing their internal states [17].

C. States and Disturbances

In the G-MAST architecture, a state is returned to MAST. The state s is represented as:

$$s = [N_m, E_m, \psi_m, N_{o1}, E_{o1}, \ldots, N_{on}, E_{on}, \psi_{on}], \quad (6)$$

where N, E, and ψ are the North position, the East position, and the heading of a vessel, respectively. The subscript m denotes mA2, which is the SUT. The subscript on denotes that the vessel is an obstacle o and the number of the obstacle vessel n. Due to its simulator-internal handling, the B-MAST architecture does not pass the state s from the simulator to AST. The state is considered terminal if the simulation has reached its user-defined maximum number of steps or if it has resulted in a failure event as described in III-A.

The disturbances are sampled from the same types of distributions in both G-MAST and B-MAST. We use three types of disturbances, namely the reference surge speed u_*, the reference heading ψ_* of the obstacle vessel, and the noise to SP-VP. The vessel reference speed u_* and reference heading ψ_* are sampled from a truncated normal distribution [4]. The reference surge speed u_* use the truncated normal distribution with a minimum allowed velocity u_{min} and a maximum allowed velocity u_{max}. The mean of the distribution is set to the velocity that should be treated as most likely in the given scenario. The standard deviation σ is set to sample both the minimum and maximum values with some frequency. The same approach is used for the heading ψ, but here the mean is set to the initial heading. This makes the initial heading of the vessel most likely, while deviation from it are less likely. The noise is sampled using a Gaussian distribution with the mean $\mu = 0$ and standard deviation $\sigma = 1.0$.

D. Reward Functions

The G-MAST and B-MAST architectures use different types of reward functions. Since the simulator in the G-MAST architecture returns the state s, the G-MAST reward function calculates the transition likelihood similar to previous work [15], [17]:

$$R(s,x) = \begin{cases} R_E & \text{if } s \text{ is terminal and } s \in E \\ -d & \text{if } s \text{ is terminal and } s \notin E \\ \log(p(x|s)) & \text{otherwise.} \end{cases} \quad (7)$$

Here, R_E is the reward when a failure event e is found. This is set to be high enough to outweigh the maximum cumulative unlikeliness.

In contrast, the B-MAST reward function is:

$$R(\rho, e, d, \tau) = \begin{cases} R_E & \text{if } \tau \wedge e \\ -d & \text{if } \tau \vee \neg e \\ \log \rho & \text{otherwise.} \end{cases} \quad (8)$$

The B-MAST reward function does not have the state s and the disturbance x available from the partially observable simulator. Instead, the simulator returns the transition likelihood ρ, a boolean indicating if it is a failure event e, a miss distance d, and a boolean indicating if the simulator is in

TABLE I: mA2 SP-VP regions configurations

	SP-VP Obstacle Region Margins			
	Fore l_f	Starboard l_s	Aft l_a	Port l_p
ROC	32.5	22.5	22.5	22.5
HPR	37.5	32.5	27.5	27.5
LPR	52.5	37.5	32.5	32.5

a terminal state τ. The transition likelihood ρ is calculated internally in the simulator for B-MAST.

If we compare the reward function for the G-MAST approach and the B-MAST approach, we see that the reward is calculated in the same way. The difference is that some of the parts needed in the equations are calculated internally in the partially observable simulator for the B-MAST approach.

V. Experimental Results

A. Experimental Setup

We search for failure events by considering three types of scenarios: The first scenario type involves fast-moving obstacles, the second type involves slow-moving obstacles, and the last one contains two fast-moving obstacles.[6] The maximum time for each simulation is 100 seconds for fast-moving obstacles and multiple fast-moving obstacles, and 150 seconds for slow-moving obstacles.

In single-obstacle scenarios, the initial position of the obstacle vessel was chosen on the starboard side of mA2. An additional obstacle was placed on the port side of mA2 in the experiments with multiple obstacles.

The existing SP-VP simulator, previously used for manual testing, was adapted for automated use within MAST. Code implementing the definition of a failure event and the RL-agent's steering of obstacle vessels were among the things added. Both variants of the MAST architecture, namely G-MAST and B-MAST, were implemented.

The specifications of the computer and software used in experiments are as follows. The operating system is *Arch Linux x86_64*, with the kernel version *5.19.5-arch1-1*. The CPU is a *Intel i7-6700K 4.2 GHz*, the GPU is *NVIDIA GeForce GTX 980*, and *32 GB RAM*. The programming languages used were Python 3.9.13 and Julia 1.7.3. The AST software used was the Julia package AdaStress 0.1.0, developed by the Robust Software Engineering technical area, based in the Intelligent Systems Division at NASA's Ames Research Center.

B. Configurations for mA2

The same mA2 configurations are used in all of the experiments. The regions for the SP-VP obstacle representations are shown in Table I. The reason why the margins are so large is due to including the dimensions for the mA2 vessel itself and added 10 meters for the obstacle vessels and some safety factors and perimeter size.

Furthermore, mA2 has an initial position of 10 meters North N and 0 meters East E with a heading straight towards North, in the northeast frame. The initial position is the start waypoint of the waypoint mission given to the vessel, and the goal waypoint is 200 meters straight North N. The SP-VP collision avoidance system is tracking obstacle vessels with an update rate of 0.25 Hz. The max velocity of mA2 is $1.2\ m/s$ and the min velocity is set to $-0.2\ m/s$. The mA2 is also configured with gains and time constants in use by the SP-VP noise model [31].

C. Adaptive Stress Testing Results

Goal. Is MAST able to find interesting failure events for SP-VP in single- and multiple-obstacle settings, and what do failure events look like?

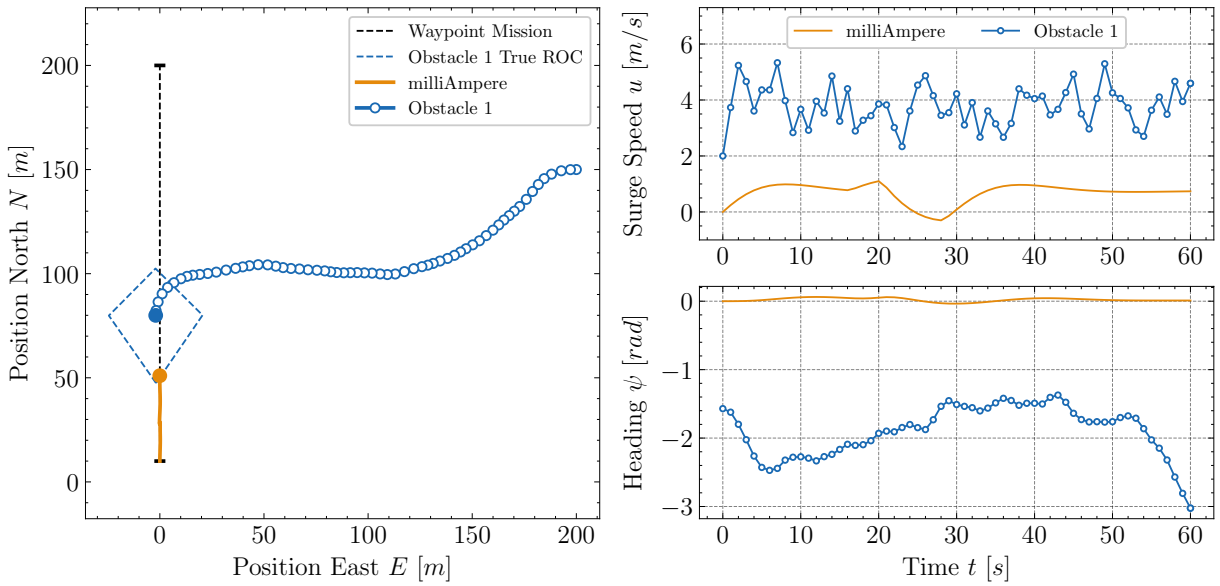
Method and Data. To study this question, 1,000 single- and 1,000 multiple-obstacle simulations were generated using G-MAST and B-MAST respectively. Both fast-moving and slow-moving obstacles were simulated. But due to their higher risk and more complex behavior we discuss a few manually selected, fast-moving scenarios in detail.

Results and Discussion. One interesting failure event found with fast-moving obstacles is the bow-on collision shown in Figure 7. In this case, mA2 is, in fact, moving when the collision occurs at time t = 60 seconds, with a speed of $0.73m/s$. Interestingly, it looks like the obstacle vessel "tricks" mA2 into crashing by changing its heading from almost straight west to almost straight south and towards mA2 at time $t = 53$ seconds. The graph for the surge speed u for mA2 shows that it has an almost constant speed over 10 seconds before the collision occurs.

Figure 8 shows a failure event found with multiple obstacles. The first image shows the simulation at time $t = 28$. Both obstacles have a course toward mA2's path. The SP-VP tracking system was updated at time $t = 28$. The next image shows the simulation frame at time $t = 52$. The SP-VP tracking system was updated at time $t = 52$ as well. In this frame, mA2 is trapped between the two obstacles and tries to speed away from Obstacle 1 and pass behind Obstacle 2. The last image shows the simulation frame at time $t = 55.6$, when SP-VP's tracking system is not updated. Because of Obstacle 2's rapid course change, it blocks mA2 from being able to pass behind it and causes a collision instead. Figure 9 shows the surge speed u and heading ψ at each simulation time step. We observe how "confused" mA2 is by looking at the big variations in mA2's surge speed in Figure 9. The figure also shows the course change of Obstacle 2 from time $t = 48$ until the end of the simulation.

In general, both G-MAST and B-MAST are able to find many interesting failure events similar to those discussed above (see also Section V-D). The degree of realism varies between the events, and their correspondence to real-world collisions can be debated, due to the often observed "irrational behavior" of an obstacle. This is similar to previous

[6] While collisions with moving obstacles are an important maritime safety concern [1], it would be very interesting to consider other hazards. This includes, for example, stationary objects in the path of the mA2 vessel. In this case, unless there is a collision, mA2 using SP-VP adequately deals with that situation by stopping up. However, this is an example of a potential problem that would not be found with the current RL problem formulation in MAST. Handling such potential deadlocks would be an interesting area of future research.



(a) Northeast plot.

(b) Surge speed u and heading ψ.

Fig. 7: This shows a bow-on collision between mA2 and a fast-moving obstacle vessel, found with G-MAST. Figure 7a shows how the obstacle's trajectory changes right before the collision at time $t = 60$ seconds. The change in the obstacle's heading ψ is shown clearly in Figure 7b where the heading changes from almost straight west to almost straight south and towards mA2 at time $t = 53$. Further, mA2 has an almost constant speed for over 10 seconds before the collision occurs. The speed of the mA2 and the obstacle vessel is 0.73 m/s and 4.59 m/s, respectively, when the collision occurs.

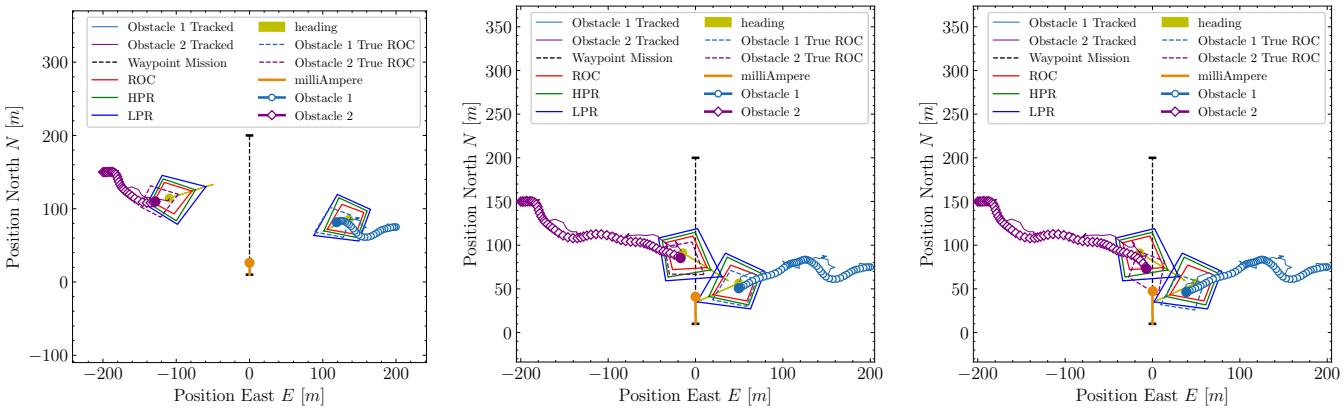


Fig. 8: This scenario illustrates the behavior of mA2 when two fast-moving obstacles are present. The three images show the simulation frames at time $t = 28$, $t = 52$, and $t = 55.6$ seconds. Here, the mA2 is trapped between the two obstacles and is unable to resolve the situation. Towards the end, mA2 plans to pass behind Obstacle 2, but this is blocked by Obstacle 2 applying a rapid course change. Corresponding surge speeds and headings are in Figure 9.

AST results for autonomous cars and pedestrians [12]. However, we believe that these are interesting results that increase the confidence in mA2's behavior in a range of situations. Further, the results provide a basis for using MAST to test other scenarios, which can potentially find other and more realistic failure events.

D. Black Box versus Gray Box Testing

Goal. If an SUT simulator contains complex parts that are very difficult to understand without substantial domain knowledge, the simulator does not reveal its internal variables or state, or the tester is not the same person as the one designing the simulator, the Black-Box approach of B-MAST can be used with great benefit. Specifically, the B-MAST architecture only requires the tester to provide an action-seed to step the simulator. The simulator designer needs to provide: a boolean indicating if the simulator is in a failure state or not, the transition likelihood from one state to another, and a failure distance measure. On the other hand, the fact that the Black-Box simulator in B-MAST does not reveal its internals may lead to limited configuration

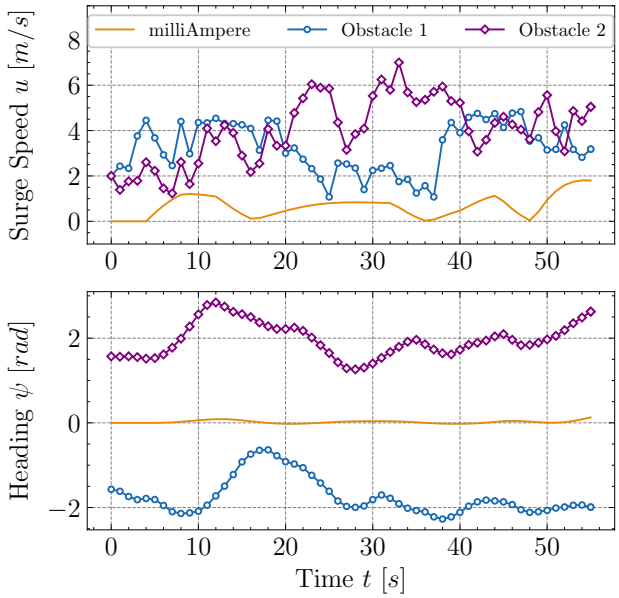


Fig. 9: Surge speed u (top) and heading ψ (bottom) for multiple obstacles. Corresponding simulation frames are in Figure 8.

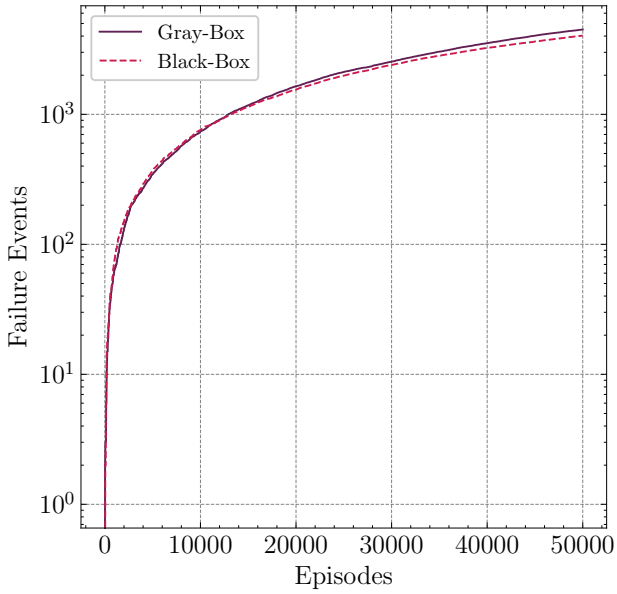


Fig. 10: G-MAST and B-MAST results. The figure shows the number of failure events found over 50,000 simulation episodes. Black-Box found 4035 failure events, and Gray-Box found 4493. The Fast-moving obstacle scenario was used.

options, which may negatively impact performance. How do the B-MAST and G-MAST architectures perform for mA2, and does performance differ much in light of the above differences?

Method and Data. Implementations of the B-MAST and G-MAST architectures with mA2 were both tested in 50,000 simulation episodes for each setup, and the number of failure events was recorded.

Results and Discussion. Figure 10 contains a comparison of the resulting performance of the B-MAST and G-MAST approaches. Using B-MAST, approximately 8.1% of the episodes were failure events, while approximately 9% of the simulation episodes were failure events with G-MAST.

In other words, both architectures successfully found a relatively high number of failure events and the number of failure events was quite similar between the two architectures. While a detailed study of the failure events is on-going, these results suggest that B-MAST can be recommended. This is due to its ease-of-use for testers, along with its similar failure event-finding performance to G-MAST.

VI. CONCLUSION AND DISCUSSION

In light of the world's need for autonomous and sustainable transportation at sea, we investigate the problem of stress-testing a navigation system for the world's first autonomous, electric ferry used to transport people. Specifically, a new architecture coined MAST is proposed. The architecture uses AST [15], [17] for testing a maritime autonomous collision avoidance system, in particular the SP-VP method implemented in the prototype passenger ferry mA2. The MAST architecture is able to find interesting failure events in the system, some of which are discussed here. Which type of architecture, G-MAST or B-MAST, is best suited for testing maritime autonomous collision avoidance systems with AST really depends on how complex the different parts of the simulator are and how much domain knowledge the tester has. The Black-Box approach is preferred if the simulator is complex to understand and if the tester does not have much domain knowledge. In this case, the simulator should be built by application professionals. The Gray-Box approach might be the best solution if the simulator does not consist of too complex parts and the tester has sufficient domain knowledge.

From a tester's practical point of view, the results in this paper suggest that the choice of architecture depends on whether a simulator is already available or not. If a simulator is available (i.e., has been implemented) and is fully observable, the tester may want to consider G-MAST (as it is slightly better in finding failure events). If a simulator is available and is partially observable, B-MAST can be used without much concern for substantially worse performance than with G-MAST. If a simulator has not been implemented yet, this paper provides guidelines on how to implement it for AST in a maritime environment. This can make the testing process easier for the tester.

Due to limited space, we have omitted several important topics that have received attention in the AST literature. These topics may also deserve future research. First, when simulations are used for AST, computational time and cost can be a serious problem [15], [23], [13], [10], [21]. In fact, the problem of fitness function evaluation cost, where "cost" may refer to computational cost, energy cost, engineering cost, or other costs, is a more general problem in AI [27], [9], [14], [20], [22]. Second, failure events other than collisions

are of great interest. Such other failure events could for example be deadlock situations, and at least some of them can be formalized by changing AST's reward function [6], [13], [10]. Third, there is the discussion of which changes does the SUT need to undergo to be corrected in order to reduce the number and probability of failure events? Here, the answer is highly application-dependent and typically involves data analysis, such as clustering, of the time series that result from 100s or 1,000s of simulation runs with AST [16], [13], [10] along with engineering judgment from the maritime domain.

REFERENCES

[1] European Maritime Safety Agency. Annual overview of marine casualties and incidents 2023. Technical report, European Maritime Safety Agency, June 2023.

[2] H. Berget. An area-time trajectory planning approach to collision avoidance for confined-water vessels. Master's thesis, NTNU, 2021.

[3] E. F. Brekke, E. Eide, B.-O. H. Eriksen, E. F. Wilthil, M. Breivik, E. Skjellaug, Ø. K. Helgesen, A. M. Lekkas, A. B. Martinsen, E. H. Thyri, et al. milliampere: An autonomous ferry prototype. In *Journal of Physics: Conference Series*, volume 2311, page 012029. IOP Publishing, 2022.

[4] J. Burkardt. The truncated normal distribution. *Department of Scientific Computing Website, Florida State University*, 1:35, 2014.

[5] E. M. Clarke, T. A. Henzinger, H. Veith, R. Bloem, et al. *Handbook of model checking*, volume 10. Springer, 2018.

[6] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validatio. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019.

[7] E. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[8] T. I. Fossen. *Handbook of marine craft hydrodynamics and motion control*. John Wiley & Sons, 2011.

[9] D. Guirguis, N. Aulig, R. Picelli, B. Zhu, Y. Zhou, W. Vicente, F. Iorio, M. Olhofer, W. Matusik, C. A. Coello Coello, and K. Saitou. Evolutionary black-box topology optimization: Challenges and promises. *IEEE Transactions on Evolutionary Computation*, 24(4):613–633, 2020.

[10] H. W. Hjelmeland, O. J. Mengshoel, B.-O. H. Eriksen, and A. M. Lekkas. Identification of failure modes in the collision avoidance system of an autonomous ferry using adaptive stress testing. In *14th IFAC Conference on Control Applications in Marine Systems, Robotics, and Vehicles*, September 2022.

[11] K. Kant and S. W. Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The international journal of robotics research*, 5(3):72–89, 1986.

[12] M. Koren, S. Alsaif, R. Lee, and M. J. Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.

[13] M. Koren, A. Corso, and M. J. Kochenderfer. The adaptive stress testing formulation. *arXiv preprint arXiv:2004.04293*, 2020.

[14] E. H. Lee, D. Eriksson, V. Perrone, and M. W. Seeger. A nonmyopic approach to cost-constrained Bayesian optimization. *CoRR*, abs/2106.06079, 2021.

[15] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, G. P. Brat, and M. P. Owen. Adaptive stress testing of airborne collision avoidance systems. In *2015 IEEE/AIAA 34th Digital Avionics Systems Conference (DASC)*, pages 6C2–1. IEEE, 2015.

[16] R. Lee, M. J. Kochenderfer, O. J. Mengshoel, and J. Silbermann. Interpretable categorization of heterogeneous time series data. In *Proceedings of the 2018 SIAM International Conference on Data Mining*, pages 216–224. SIAM, 2018.

[17] R. Lee, O. J. Mengshoel, A. Saksena, R. W. Gardner, D. Genin, J. Silbermann, M. Owen, and M. J. Kochenderfer. Adaptive stress testing: Finding likely failure events with reinforcement learning. *Journal of Artificial Intelligence Research*, 69:1165–1201, 2020.

[18] R. Lee, O. J. Mengshoel, An. Saksena, R. Gardner, D. Genin, J. Brush, and M. J. Kochenderfer. Differential adaptive stress testing of airborne collision avoidance systems. In *2018 AIAA Modeling and Simulation Technologies Conference*, page 1923, 2018.

[19] R. Lipkis, R. Lee, J. Silbermann, and T. Young. Adaptive stress testing of collision avoidance systems for small UASs with deep reinforcement learning. In *AIAA SCITECH 2022 Forum*, page 1854, 2022.

[20] P. Luong, D. Nguyen, S. Gupta, S. Rana, and S. Venkatesh. Adaptive cost-aware Bayesian optimization. *Knowledge-Based Systems*, 232, 2021.

[21] B. Lytskjold and O. J. Mengshoel. Speeding up adaptive stress testing: Reinforcement learning using monte carlo tree search with neural networks and memoization. In *The 37th AAAI Conference on Artificial Intelligence*, 2023.

[22] O. J. Mengshoel, E. L. Flogard, T. Yu, and J. Riege. Understanding the cost of fitness evaluation for subset selection: Markov chain analysis of stochastic local search. In *Proc. GECCO*, page 251–259, 2022.

[23] I. Porres, S. Azimi, and J. Lilius. Scenario-based testing of a ship collision avoidance system. In *46th Euromicro Conference on Software Engineering and Advanced Applications (SEAA)*, pages 545–552, 2020.

[24] N. P. Reddy, M. K. Zadeh, C. A. Thieme, R. Skjetne, A. J. Sørensen, S. Aa. Aanondsen, M. Breivik, and E. Eide. Zero-emission autonomous ferries for urban water transport: Cheaper, cleaner alternative to bridges and manned vessels. *IEEE Electrification Magazine*, 7(4):32–45, 2019.

[25] J. Schulman, F. Wolski, Pr. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[26] A. C. Schultz, J. J. Grefenstette, and K. A. De Jong. Adaptive testing of controllers for autonomous vehicles. In *Proceedings of the 1992 Symposium on autonomous underwater vehicle technology*, pages 158–164. IEEE, 1992.

[27] J. Snoek, H. Larochelle, and R. P. Adams. Practical Bayesian optimization of machine learning algorithms. In *Proc. NeurIPS*, pages 2951–2959, 2012.

[28] E. H. Thyri. A path-velocity decomposition approach to collision avoidance for autonomous passenger ferries. Master's thesis, NTNU, 2019.

[29] E. H. Thyri, M. Breivik, and A. M. Lekkas. A path-velocity decomposition approach to collision avoidance for autonomous passenger ferries in confined waters. *IFAC-PapersOnLine*, 53(2):14628–14635, 2020.

[30] United Nations. Sustainable transport, sustainable development. Interagency report for second global sustainable transport conference, 2021.

[31] J.-M. Vatle. Adaptive stress testing for safety validation of maritime autonomous collision avoidance systems. Master's thesis, Norwegian University of Science and Technology (NTNU), 2022.

[32] L. Wang. Basics of PID control. In *PID Control System Design and Automatic Tuning using MATLAB/Simulink*, pages 1–30. Wiley-IEEE Press, 2020.