Automatic Simulation-based Testing of Autonomous Ships using Gaussian Processes and Temporal Logic

Journal Title
XX(X):1–19

DOI: 10.1177/ToBeAssigned
www.sagepub.com/
SAGE

Tobias Rye Torben[1], Jon Arne Glomsrud[2], Tom Arne Pedersen[2], Ingrid B. Utne[1], Asgeir J. Sørensen[1]

Abstract

A methodology for automatic simulation-based testing of control systems for autonomous vessels is proposed. The work is motivated by the need for increased test coverage and formalism in the verification efforts. It aims to achieve this by formulating requirements in the formal logic Signal Temporal Logic (STL). This enables automatic evaluation of simulations against requirements using the STL robustness metric, resulting in a robustness score for requirements satisfaction. Furthermore, the proposed method uses a Gaussian Process (GP) model for estimating robustness scores including levels of uncertainty for untested cases. The GP model is updated by running simulations and observing the resulting robustness, and its estimates are used to automatically guide the test case selection towards cases with low robustness or high uncertainty. The main scientific contribution is the development of an automatic testing method which incrementally runs new simulations until the entire parameter space of the case is covered to the desired confidence level, or until a case which falsifies the requirement is identified. The methodology is demonstrated through a case study, where the test object is a Collision Avoidance (CA) system for a small high-speed vessel. STL requirements for safety distance, mission compliance and COLREG compliance are developed. The proposed method shows promise, by both achieving verification in feasible time and identifying falsifying behaviours which would be difficult to detect manually or using brute-force methods. An additional contribution of this work is a formalization of COLREG using temporal logic, which appears to be an interesting direction for future work.

Keywords

Verification, Autonomous vessels, Gaussian Processes, Temporal Logic, COLREG

Introduction

With the rapid advances in the field of information and communication technology (ICT) in the recent years, the tasks that are automated gradually become more complex. Space and underwater operations with limited means of communication, as well as autonomous transportation solutions have been driving forces for this development. The autonomy trend has also reached the maritime sector, where several commercial and academic projects aiming towards autonomous maritime vessels have been announced recently[1].

Building a compelling argument for the safety of autonomous systems has proven to be a challenge[2]. In the maritime domain, extensive use of *simulation-based testing* has been proposed as a possible approach[3]. Simulation-based testing refers to creating a simulation model, often termed a *digital twin*, of a system together with its operative environment and performing testing on the digital twin instead of the actual system in the real world. Simulation-based testing is attractive due to its scalability, that is, it is possible to assess system level behaviours for highly complex systems. Autonomous vessel concepts are characterized by high levels of complexity in their hardware and software systems, as well as in their interaction with the operative environment. Combined with the intrinsic challenges related to verification of autonomous functions, such as the use of machine learning components and hard-to-predict emergent behaviours, simulation-based testing stands out as a promising and key way forward. Simulation-based testing, in particular Hardware-in-the-Loop (HiL) testing, has strong traditions in the maritime industry already[4–7].

While simulation-based testing offers a great platform for verification, it is paramount that it is used in combination with valid processes for test case selection, evaluation of results and test coverage assessment. Traditionally, the selection of test cases has been a manual process based on risk analyses and experience with incidents and typical pitfalls in development and implementation. For emerging technologies, such as autonomous vessels, this approach is challenging, as the necessary experience, regulations, class notations and best-practices are not yet available. Moreover, the operative environment for autonomous

[1]Centre for Autonomous Marine Operations and Systems, Norwegian University of Science and Technology (NTNU)
[2]Group Research and Development, Det Norske Veritas (DNV), Veritasveien 1, 1363 Høvik, Norway

Corresponding author:
Tobias Rye Torben, Norwegian University of Science and Technology (NTNU), Otto Nielsens vei 10, 7052 Trondheim, Norway.

Email: tobias.torben@ntnu.no

systems is characterized as highly dynamic, unstructured and uncertain, which gives a wide span of possible situations and failure combinations[8]. This necessitates a large number of simulations to obtain sufficient test coverage, which calls for automation of the test case selection and corresponding evaluation of the results. Also, since simulation-based testing, in contrast to more formal methods, is almost never able to test all possible scenarios due to practical computation time constraints, the notion of *confidence level* in the verification becomes important. By confidence level, we refer to the probablilty that a verified system actually contains no requirement violations. Since autonomous ships are safety-critical, it is crucial to have methods to assess the confidence level in the verification efforts to build sufficient trust.

The main scientific contribution of our work is the development of a methodology for simulation-based testing which attempts to address the needs specified above. We propose to formulate the requirements to test against in the formal logic STL. This enables automatic quantitative evaluation of simulations against the given requirements. Furthermore, we define parametric test cases, where a more general parametric case is defined manually, and we verify that all concrete subcases meet the requirements. We use a Gaussian Process (GP) model to predict the performance and uncertainty level over the entire parameter space. The GP model is updated by running simulations and observing the resulting performance, and its estimates used to adaptively guide the test case selection towards cases with low performance or high uncertainty. The proposed testing method incrementally runs new simulations until the entire parameter space of the test case is covered to the desired confidence level, or until a case which falsifies the requirement is identified.

The proposed method has several advantages. We get an assessment of the coverage and quantification of the confidence in the verification effort. Due to the adaptive test case selection, it is expected to reduce the number of simulations required to obtain a sufficient test coverage compared to a regular grid search. Efficient falsification is also achieved if the system does not meet the requirements. After an initial setup, the testing is completely automatic which enables the execution of a large number of simulations. This also integrates well with agile development and continuous deployment, where nightly builds can run automated simulation-based testing along with the standard unit and integration software tests. We also highlight that the proposed approach is not limited to using STL requirements and STL robustness evaluations. Any quantitative evaluation of simulations may be used instead. The core methodology is also by no means restricted to the testing of autonomous vessels, although that is the focus of this paper.

There exist several previous works on the topic of using STL and STL robustness in simulation-based testing. Prominent frameworks are Breach[9], Taliro[10] and RRT-REX[11]. The Taliro framework is used in Tuncali et al. (2020)[12] to do simulation-based falsification for autonomous driving. To our knowledge, no previous work exists which uses STL in combination with a GP model for verification. However, the methods have previously been combined to achieve data-driven synthesis of requirements[13].

There also exist previous works in automatic evaluation of maritime Collision Avoidance (CA) systems for autonomous ships, including Woerner et al. (2016)[14], which uses tailored penalty functions. This approach is further developed by Pedersen et al. (2020)[3]. Stankiewicz et al. (2019)[15] propose a different approach by running a large number of simulations and mapping decision boundaries using clustering methods. Lee et al. (2020)[16] present a falsification method called Adaptive Stress Testing for aircraft collision avoidance, which uses an adversary reinforcement learning based agent for the environment to identify falsifying interactions. The use of formal methods for specification and verification of autonomous systems has seen some adoption in the sectors of aerospace, automotive and mobile robotics[17]. During the last year, the maritime sector has also seen some use of formal methods for autonomous vessels. Shokri-Manninen et al. (2020)[18] have created a formal automata-based model of single-vessel encounters and synthesized a correct-by-construction navigation strategy in the tool UPPAAL STRATEGO, which uses a combination of model checking and machine learning. Park and Kim (2020)[19] has synthesized a correct-by-construction controller for automatic docking of marine vessels based on reachability analysis. Finally, Foster et al. (2020)[20] present a controller for autonomous marine vessels in the form of a hybrid dynamical system and use the Isabelle theorem prover to verify some invariant propoerties.

This paper is organized as follows. First, the background and mathematical preliminaries are presented. This includes an overview of state-of-the-art in verification of cyber-physical systems (CPSs) and an introduction to temporal logic and GPs. Next, the main contribution, a methodology for automatic testing, is developed, and an implementation is presented in algorithmic form in Algorithm 1. A case study which demonstrates the use of the proposed methodology for a CA system is conducted thereafter. A discussion on methodical issues with the proposed approach follows next before concluding remarks are given.

Preliminaries and Background

Terminology and definitions

In this section we define the main terminology and definitions used in the paper. A list of symbols that are used extensively throughout the paper is given in Table 1. Bold symbols are used for vectors or matrices.

Suppose that we have a simulator of the system under test together with its operational environment. The result of a particular simulation depends on a number of parameters describing for instance the initial conditions, input signals and configurations. For a simulator with n parameters, each parameter p_i, where $i \in [1, 2, ..., n]$ is an index, has an associated *parameter set* $\mathcal{P}_i$ which specifies the values that the parameter can take. The set $\mathcal{P}$, defined by the cartesian product $\mathcal{P} = \mathcal{P}_1 \times \mathcal{P}_2 \times ... \times \mathcal{P}_n$ contains all possible combinations for the simulator. It is usually not feasible nor

useful to test all simulations in $\mathcal{P}$. Instead, we select some subsets where most of the parameters are fixed but some are allowed to vary within a set. We formalize this idea with the notion of a *case*. A case Σ is defined by a collection of k parameters $p_1, p_2, ..., p_k$ with corresponding parameter sets $\mathcal{P}_1, \mathcal{P}_2, ...\mathcal{P}_k$. The parameter set of a case Σ is defined as the cartesian product of the parameter sets of all of the parameters in the case, $\mathcal{P}_\Sigma = \mathcal{P}_1 \times \mathcal{P}_2 \times ... \times \mathcal{P}_k$. Cases are arranged hierarchically according to their parameter sets. We say that case Σ_1 is a *sub-case* of Σ_2 if $\mathcal{P}_{\Sigma_1} \subset \mathcal{P}_{\Sigma_2}$. Similarly, Σ_2 is a *super-case* of Σ_1 if $\mathcal{P}_{\Sigma_1} \subset \mathcal{P}_{\Sigma_2}$.

Each point $\boldsymbol{p} \in \mathcal{P}$ represents a list of parameter values. This defines the input to a simulation. For a particular simulator, we define a simulation as a function $sim : \mathcal{P} \mapsto \mathbf{W}$, which maps a vector of parameter values, $\mathbf{p} \in \mathcal{P}$ to a time stamped output signal $\mathbf{w} = (\mathbf{y}_0, t_0), (\mathbf{y}_1, t_1), ..., (\mathbf{y}_N, t_N)$. The output signal vector $\mathbf{y}$ has m components and its domain is the set $\mathcal{Y} = \mathcal{Y}_1 \times \mathcal{Y}_2 \times ... \times \mathcal{Y}_m$.

We demonstrate these definitions in the following example.

Example 1. Consider a simulator describing an autonomous marine vessel at open sea. Such a simulator will likely have a large number of configurable parameters. Suppose that we want to verify that the autonomous vessel can handle all situations where another is on direct collision course at different speeds. This can be described by a case Σ_1 with two parameters, the course of the other vessel, $\theta \in [-180, 180]°$ and the speed of the other vessel, $U \in [0, 20] m/s$, and all other parameters remain fixed. The parameter set of Σ_1, $\mathcal{P}_{\Sigma_1} = [-180, 180] \times [0, 20]$ is a two-dimensional section of $\mathcal{P}$. Next, suppose that we want to examine head-on situations with high speed more closely. This can be described by a sub-case $\Sigma_2 \subset \Sigma_1$, where the course is fixed at $\theta = 180°$ and $U \in [10, 20] m/s$. This is illustrated in Figure 1, which shows $\mathcal{P}_{\Sigma_1}$ as a two-dimensional subset of $\mathcal{P}$, and $\mathcal{P}_{\Sigma_2}$ as a one-dimensional subset of $\mathcal{P}_{\Sigma_1}$.

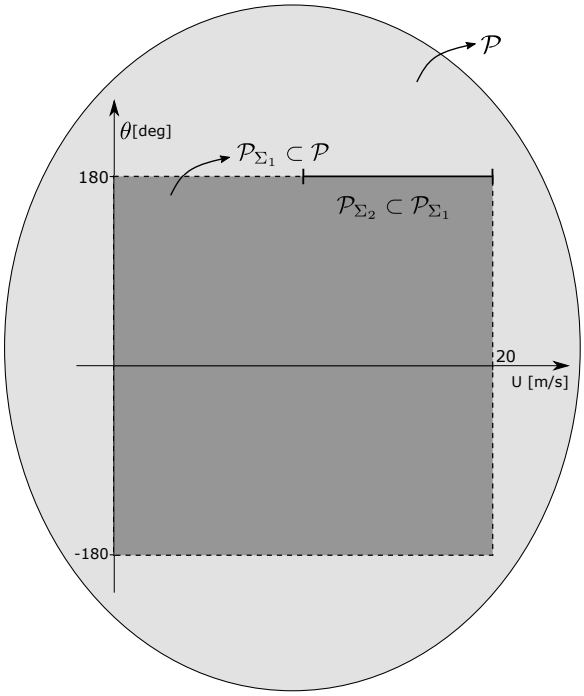

Figure 1. Illustration of the parameters sets in Example 1. $\mathcal{P}$ is the full parameter set for the simulator, here projected in 2D for visualization. $\mathcal{P}_{\Sigma_1}$ is the 2D subset of $\mathcal{P}$ corresponding to the case Σ_1, and $\mathcal{P}_{\Sigma_2}$ is the 1D subset of $\mathcal{P}_{\Sigma_1}$ corresponding to the case Σ_2.

Verification methods for Cyber-Physical Systems

Cyber-Physical Systems (CPSs) are comprised of physical, digital and networking components, Typically a physical plant is controlled by digital computers. Autonomous vessels are clearly an example of this. Since the invention of the micro processor, CPSs have become ubiquitous. The dual nature of these systems, including both hardware and software, introduces challenges in the verification of them, and this has been an active area of research and development. Here, we give a brief overview of existing methods and their merits and shortcomings to build a context around the proposed method of this paper.

In Figure 2, a classification of the different verification methods is shown. The methods are ranked informally according to their scalability, that is, how large or complex systems they can verify, and the exhaustiveness, which is an indication of how thoroughly the possible behaviours of a system are assessed. Figure 3 shows the corresponding system models used in the verification. The models are ranked according to how closely they can resemble the real system that is verified. The methods can be classified into three distinct classes; formal, analytical and simulation-based.

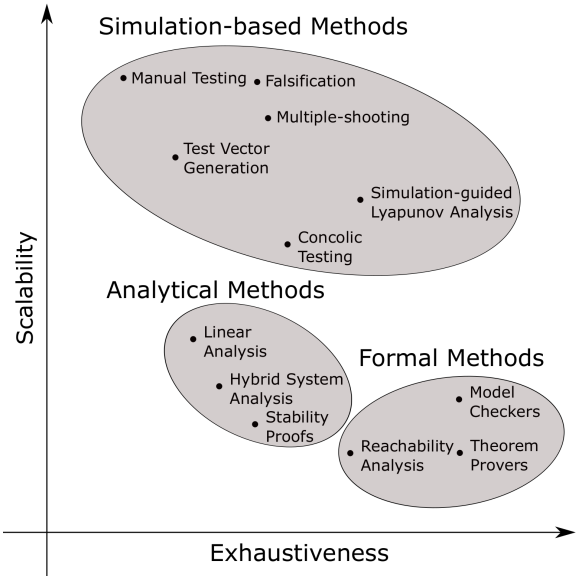

Figure 2. Spectrum of verification methods for CPSs. The horizontal axis indicates the level of exhaustiveness/formality. The vertical axis indicates the scalability of the method, that is, how large or complex systems it can verify. Inspired by Kapinski et al. (2016)[21].

Table 1. Explanation of symbols that are used extensively throughout the paper.

$\boldsymbol{p}$	List of parameter values which defines a simulation
$\boldsymbol{w}$	Output signal from a simulation
φ	STL formula
$[\![\varphi]\!](\boldsymbol{w}, t)$	STL robustness of signal $\boldsymbol{w}$ at time t with respect to formula φ
f_ρ	The function which maps a list of parameters to an STL robustness score
$\mathbf{P}$	List of test points for the GP
$\boldsymbol{\rho}$	List of random variables for the STL robustness at the test points
$\mathbf{P}_{obs}$	List of observed points for the GP
$\boldsymbol{\rho}_{obs}$	List of random variables for the STL robustness at the observed points
$\bar{\rho}(\mathbf{p})$	GP mean estimate of the STL robustness score at point $\mathbf{p}$
$var(\rho(\mathbf{p}))$	GP variance estimate of the STL robustness score at point $\mathbf{p}$

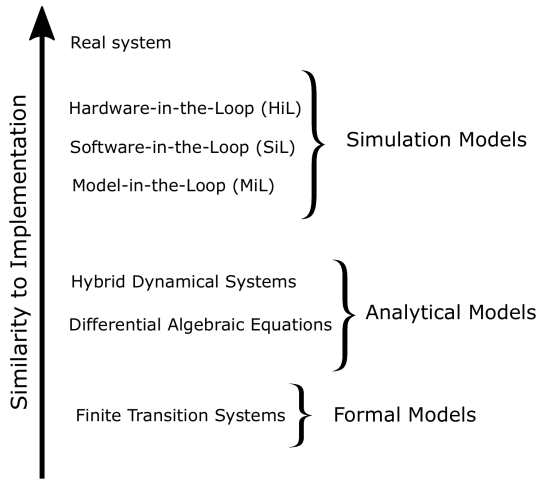

Figure 3. Spectrum of models used in the verification process, ranked according to how close the models can come to the actual implementation of the CPS.

Formal methods are characterized by a high level of formality and exhaustiveness. They are usually based on a finite transition system model, which is a discrete model containing a finite number of states and a finite number of transitions between states. Formal methods are usually used in combination with a formal specification language or logic, such as temporal logic, to specify the desired behaviour. Model checking is the most common formal method, which does an exhaustive search of all of the possible executions of a finite transition system to verify that it satisfies a temporal property[22]. Reachability analysis approximates the set of behaviours that a system can exhibit, and can for instance be used to verify that a set of unsafe behaviours is never visited[23]. Theorem provers are computer tools which aid the user in building mathematical proofs that a model meets a property[24]. There exist both interactive theorem provers, which require input from the user for each step of the proof, and fully automatic theorem provers.

Model checkers and reachability tools have seen limited scalability as they have been limited by the so-called *state-space explosion* problem. This refers to the fact that the set of states grows exponentially as the number of variables increases. However, with the huge increase in computational power and advances in the model checking algorithms, modern model checkers can solve highly complex verification problems.

Most formal methods are based on some form of finite-transition system, which also needs to be limited in size due to the state-space explosion problem[17]. Therefore it is sometimes required to create a separate verification model suitable for formal verification, which is an abstraction of the implementation. Since the model which is verified is an abstraction of the implementation, this creates a *reality gap*[17], which may miss some implementation-level errors. Building an implementation based on a formally verified design has nevertheless proven to be a successful strategy for uncovering fundamental errors in the system. The reality gap is particularly evident for systems with continuous dynamics, which need to be discretized into a finite-transition model to facilitate model checking. Theorem provers are often better able to verify properties of systems with continuous dynamics. For software systems, there exist formal verification tools which can work directly on implementation-level code. There is clearly a wide spectrum of modelling power at play here. However, compared to e.g. simulation-based methods, the models suitable for formal verification generally have stronger limitations with respect to the type and complexity of systems that they can model, and therefore score lower on similarity to implementation[21].

Formal methods have also proven to offer great advantages in the development process, by enabling the development of correct-by-construction designs built on formal specifications. Taking on a formal approach early in the development process can also improve the verification process later, by eliminating many classes of errors and by having a formal specification to verify against[25]. For an in-depth survey of this topic the reader is referred to Luckuck et al. (2019)[17].

Analytical methods perform manual mathematical analysis on a set of symbolic equations. For continuous systems, the models are Differential Algebraic Equations (DAEs), whereas systems which exhibit both continuous and discrete behaviors are modelled as hybrid dynamical systems. These models rank quite low on similarity to implementation due to the sheer difficulty in creating analytical models of complex CPSs. Stability proofs refer to using mathematical analysis, such as Lyapunov methods to prove e.g. stability or invariance of a set for continuous systems[26]. Linear Analysis is similar, but here the system is first linearized about an operational point. This is quite trivial to do even for complex models however, the results are only local and

thus not very exhaustive[27]. Linear analysis is also possible to do numerically for some simulation models. Hybrid system analysis refers to a set of mathematics studying the properties of hybrid dynamical systems, and is able to handle more complex systems than pure DAEs, since it supports discrete dynamics[28]. The mathematical theorems mostly resemble those of DAEs. However, the results which can be proved for hybrid dynamical systems are often not as strong as those of DAEs because they are more complex mathematically, and their theoretical foundation is still relatively immature.

Simulation-based methods use numerical simulation models which can be stepped forward in time using numerical solvers. We have divided simulation models for CPSs into three classes. In a Model-in-the-Loop (MiL) simulation, both the controller and plant are simulation models. In a Software-in-the-Loop (SiL) simulation, the plant is controlled by the real control software, and in HiL simulation the control system runs on the real hardware platform and communicates with the simulated plant through a HiL interface. It is possible to create detailed simulation models of large and complex CPSs, even entire ships or aircraft, which together with the HiL and SiL opportunities make it both highly scalable and close to the real implementation[21]. The reality gap is present also for simulation-based methods, as they operate on models of reality. In particular, complex environments can be hard to model accurately. Nevertheless, due to the strong modelling power and flexibility of simulation-models, this is less pronounced than for other analytical and formal methods.

Simulation is inherently a testing approach, as a single simulation only assesses behaviour for a single test case. Manual simulation-based testing therefore scores low on exhaustiveness. The other simulation-based methods in Figure 2 represent different approaches to improve on this by systematically managing the test case selection and evaluation of the results in various ways. Test vector generation is an automated process for selecting the test cases such that certain coverage criteria are met. Concolic testing combines simulations of the plant with a formal analysis of the decision branching of the controller software. Falsification methods use a parametrization of test cases and take an optimization approach to search for cases with low performance. Multiple-shooting approaches also attempt to achieve falsification by running many partial simulations and splicing together the results to identify a falsifying case. Finally, simulation-gudied Lyapunov analysis refers to a method for searching for a Lyapunov function using the results of simulations. From a Lyapunov function, stability, invariant sets and performance bounds can be derived. For a more detailed description of these methods, the reader is referred to Kapinski et al. (2016)[21].

The method proposed in this paper falls into the class of simulation-based methods which aim to increase the exhaustiveness and formality. It may be used with any of the simulation models in Figure 3.

Temporal Logic

Temporal logics are extensions of propositional logic which also capture temporal aspects. A temporal logic formula specifies a behaviour, and there exists effective algorithms to evaluate a signal against a temporal logic formula to see if it satisfies the specified behaviour. Linear Temporal Logic (LTL), was introduced in Pnueli (1977)[29]. LTL operates on boolean signals in discrete time. An extension of LTL is Metric Temporal Logic (MTL) which operates on boolean signals in continuous time[30]. Signal Temporal Logic (STL) was proposed as a syntactic addition to MTL, where the formulas operate on real-valued signals[31].

During the last decade, many have realized the power and possibilities when specifying behaviours in STL, see Kapinski et al. (2016)[21] and the references therein. In addition to formal specification and verification, it can be used as a runtime monitor[32], where the online behaviour is continuously evaluated against an STL requirement. Applications of this include encoding traffic rules as STL and using this to monitor road safety online[33]. Moreover, it has also seen several applications as a design methodology in planning[34] and control synthesis[35].

The basic building block of STL formulas are *predicates*. A predicate π is a function which maps a signal y to a boolean. In STL, the predicates take the form

$$\pi ::= f(y) \leq c, \tag{1}$$

where $f(y)$ is a scalar, real-valued function which maps the input y signal to a real-valued scalar, and c is a real-valued scalar. The signals which satisfy a predicate, π, represent a subset in the space $\mathcal{Y}$. In the following, we represent the set that corresponds to the predicate π using the notation $\mathcal{O}(\pi)$. We note that STL predicates are only defined for non-strict inequalities. However, since we will evaluate the formulas using the quantitative STL robustness semantics, there is no difference between strict and non-strict inequalities.

STL formulas are built by combining predicates with the operators of propositional logic and temporal operators. An informal description of the various operators is given before the formal syntax and semantics of STL are presented.

- *Conjunction*: $\varphi_1 \wedge \varphi_2$ is true if both φ_1 and φ_2 are true.
- *Disjunction*: $\varphi_1 \vee \varphi_2$ is true if either φ_1 or φ_2 are true.
- *Negation*: $\neg\varphi$ is true if φ is false.
- *Implication*: $\varphi_1 \rightarrow \varphi_2$ is true if φ_2 follows from φ_1, i.e. $\varphi_1 \rightarrow \varphi_2$ is false if and only if φ_1 is true and φ_2 is false.
- *Eventually*: $\Diamond\varphi$ is true if φ is true at some time.
- *Always*: $\Box\varphi$ is true if φ is true at all times.
- *Next*: $\bigcirc\varphi$ is true if φ is true at the next discrete time step.
- *Until*: $\varphi_1 U \varphi_2$ is true if φ_1 is true until φ_2 first becomes true.
- *Release*: $\varphi_1 R \varphi_2$ is true if φ_2 is true until φ_1 first becomes true.

Definition 1. STL Syntax[31]:
Let Π be the set of predicates and $\mathcal{I}$ be any non-empty

connected interval of $\mathbb{R}_{\geq 0}$. The set of well-formed STL formulas is defined by the grammar

$$\varphi ::= \top \mid \pi \mid \neg\varphi \mid \varphi_1 \vee \varphi_2 \mid \bigcirc \varphi \mid \varphi_1 U_{\mathcal{I}} \varphi_2, \tag{2}$$

where φ, φ_1 and φ_2 are STL formulas, $\top$ is the True constant and $\pi \in \Pi$ is an STL predicate.

Note that all the logical and temporal operators can be derived from these basic operators:

$$\varphi_1 \wedge \varphi_2 \equiv \neg(\neg\varphi_1 \vee \neg\varphi_2) \tag{3}$$
$$\varphi_1 \rightarrow \varphi_2 \equiv \neg\varphi_1 \vee \varphi_2 \tag{4}$$
$$\Diamond_{\mathcal{I}} \varphi \equiv \top U_{\mathcal{I}} \varphi \tag{5}$$
$$\Box_{\mathcal{I}} \varphi \equiv \neg\Diamond_{\mathcal{I}} \neg\varphi \tag{6}$$
$$\varphi_1 R_{\mathcal{I}} \varphi_2 \equiv \neg(\neg\varphi_1 U_{\mathcal{I}} \neg\varphi_2) \tag{7}$$

Example 2. An example of a simple STL formula, which will be used extensively in this paper, is the requirement for a vessel to always keep a safe distance from other vessels. In STL this can be expressed as

$$\varphi_{safety} = \Box\neg\left(d(t) \leq d_{min}\right) \tag{8}$$

where $d(t)$ is the distance to another vessel at time t and d_{min} is a constant specifying the minimum required safety distance.

STL robustness metric

Much of the popularity of STL is due to the *STL robustness metric*, introduced by Fainekos et al. (2009)[36]. In contrast to the normal Boolean semantics, which only give a true/false evaluation of whether a signal satisfies a formula, the STL robustness metric defines the semantics for quantitative evaluation of how robustly a signal satisfies an STL formula. This has proven to be a powerful combination, as STL both provides a language to describe behaviours and a metric to measure conformance to these behaviours.[37].

Let $[\![\varphi]\!](\boldsymbol{w}, t)$ denote the STL robustness of a signal $\mathbf{w}$ against the formula φ at a discrete time t. The STL robustness metric has the soundness property that $[\![\varphi]\!](\boldsymbol{w}, t) \geq 0$ if $\mathbf{w}$ satisfies φ at time t and $[\![\varphi]\!](\boldsymbol{w}, t) < 0$ otherwise. Informally, the magnitude of the robustness metric indicates how much the signal can change without violating the requirement.

Before we formally define the STL robustness metric we must first define *metrics* and the *signed distance*.

Definition 2. Metric[36]:
A metric on a set S is a positive function $d : S \times S \rightarrow \mathbb{R}_{\geq 0}$ such that
1) $\forall s, s' \in S, \quad d(s, s') = 0 \Leftrightarrow s = s'$
2) $\forall s, s' \in S, \quad d(s, s') = d(s', s)$
3) $\forall s, s', s'' \in S, \quad d(s, s'') \leq d(s, s') + d(s', s'')$

In this paper the *Euclidean metric* $d(s, s') = \|s - s'\|$; has been used.

Next, we define the *signed distance* to a set. Intuitively, this captures how robustly a point belongs to a set. If the point is in the set, the signed distance is positive. If the point is on the boundary then it is zero and if the point its outside the set it is negative. The magnitude represents how far away the point is from the boundary. A formal definition follows.

Definition 3. Signed distance[36]:
Consider a point $s \in S$ a set $A \subseteq S$ and a metric d on S. The signed distance from s to A is defined as

$$Dist_d(s, A) := \begin{cases} -\inf\{d(s, s') \mid s' \in A\} & \text{if } s \notin A \\ \inf\{d(s, s') \mid s' \notin A\} & \text{if } s \in A \end{cases} \tag{9}$$

With the definitions of metrics and the signed distance, a formal definition of the STL robustness semantics is stated next.

Definition 4. STL robustness semantics[36]:
For a metric d and a signal $\mathbf{w} = (\mathbf{y}_0, t_0), (\mathbf{y}_1, t_1), ..., (\mathbf{y}_N, t_N)$, the STL robustness $[\![\varphi]\!]_d(\mathbf{w}, t)$ of $\mathbf{w}$ w.r.t φ at time instance $t \in [0, 1, \ldots, N]$ is defined as:

$$[\![\top]\!]_d(\mathbf{w}, t) := +\infty \tag{10a}$$
$$[\![\pi]\!]_d(\mathbf{w}, t) := Dist_d(\boldsymbol{y}_t, \mathcal{O}(\pi)) \tag{10b}$$
$$[\![\neg\varphi]\!]_d(\mathbf{w}, t) := -[\![\varphi]\!]_d(\mathbf{w}, t) \tag{10c}$$
$$[\![\varphi_1 \vee \varphi_2]\!]_d(\mathbf{w}, t) := \tag{10d}$$
$$\max\left([\![\varphi_1]\!]_d(\mathbf{w}, t), [\![\varphi_2]\!]_d(\mathbf{w}, t)\right)$$
$$[\![\bigcirc\varphi]\!]_d(\mathbf{w}, t) := \tag{10e}$$
$$\begin{cases} [\![\varphi]\!]_d(\mathbf{w}, t+1) & \text{if } t+1 \in N \\ -\infty & \text{otherwise} \end{cases}$$
$$[\![\varphi_1 U_{\mathcal{I}} \varphi_2]\!]_d(\mathbf{w}, t) := \max_{j \text{ s.t. } (t_j - t_t) \in \mathcal{I}} \tag{10f}$$
$$\left(\min\left([\![\varphi_2]\!]_d(\mathbf{w}, j), \min_{t \leq k < j} [\![\varphi_1]\!]_d(\mathbf{w}, j)\right)\right),$$

where $\pi \in \Pi$ is a predicate, and $\mathcal{O}(\pi)$ is the corresponding set in $\mathcal{Y}$. For short, $[\![\varphi]\!](\boldsymbol{w})$ refers to the robustness at time $t = 0$ using the Euclidean norm as the metric.

Example 3. As an example, we illustrate the use of the STL robustness metric for the safety distance requirement of (8), as shown in Figure 4. A sample signal for the distance between two vessels $d(t)$ has been generated by a random Wiener process. In the top plot, $d(t)$ is plotted together with the minimum safety distance limit $d_{min} = 50m$. This is the input signal to an STL monitor, whose output is shown in the lower plot. To build an intuitive understanding of the resulting STL robustness score, we split the evaluation of the signal into two steps. First, we look at the inner subformula, $\varphi = \neg\left(d(t) \leq d_{min}\right)$. This STL formula is time-independent, that is, the value of $[\![\varphi]\!](d(t), t)$ only depends on the value of $d(t)$ at time t. This robustness value, shown in yellow, intuitively represents how far the distance to the other vessel is from violating the requirement. This means that when $d(t) = 50m$, $[\![\varphi]\!](d(t), t) = 0m$. When $d(t) > 50m$, $[\![\varphi]\!](d(t), t) > 0$ and when $d(t) < 50m$, $[\![\varphi]\!](d(t), t) < 0$.

When adding an operator on this subformula, the robustness score of the subformula becomes the input signal for this operator. Since $\Box$ is a temporal operator, the

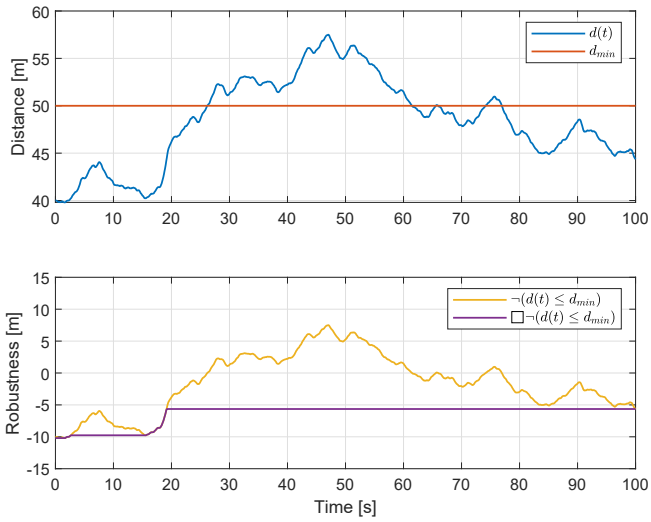

Figure 4. STL robustness evaluation against safety distance requirement. The upper plot shows a sample distance signal, $d(t)$ together with the required minimum distance d_{min}, plotted against time. The lower plot shows the corresponding STL robustness score for the subformula $\varphi = \neg\left(d(t) \leq d_{min}\right)$ and the full formula $\varphi_{safety} = \square\neg\left(d(t) \leq d_{min}\right)$.

robustness score at any time depends on the entire time series from that time to the end. In the case of the always operator, it can be shown that the output robustness score is the minimum of the input signal over time. The robustness score for the full formula is shown in violet in the lower plot of Figure 4. This demonstrates that the value of the violet curve at time t corresponds to the minimum of the yellow curve from t to the end.

Gaussian Processes

Consider an unknown function, $y = f(\mathbf{x})$, which we want to estimate by making observations $(\mathbf{x_i}, y_i)$. The standard regression approach is to assume a model for $f(\mathbf{x})$, and try to find the parameters of this model such that it fits the observations well. GPs take a different approach to this. A GP models the function values $y_i = f(\mathbf{x_i})$ at points $\mathbf{x_i}$ as random variables which are jointly Gaussian distributed[38]. The covariance between function values at points $\mathbf{x_p}$ and $\mathbf{x_q}$ is denoted

$$\operatorname{cov}\left(f\left(\mathbf{x}_p\right), f\left(\mathbf{x}_q\right)\right) = k\left(\mathbf{x}_p, \mathbf{x}_q\right), \tag{11}$$

where $k\left(\mathbf{x}_p, \mathbf{x}_q\right)$ is a *covariance function* of choice, also known as a *kernel function*. A common choice of covariance function is the squared exponential

$$k\left(\mathbf{x}_p, \mathbf{x}_q\right) = \sigma^2 \exp\left(-\frac{1}{2l^2}\left\|\mathbf{x}_p - \mathbf{x}_q\right\|^2\right), \tag{12}$$

which is built on the assumption that points that are close to each other are more strongly correlated than points which are far apart. This function has two parameters: the variance σ^2, and the length scale l. Note that for $\mathbf{x_p} = \mathbf{x_q}$ the covariance reduces to σ^2 which is the variance of the function value at this point. Informally, l describes how much $\|\mathbf{x}\|$ needs to change in order to significantly change the value of $f(\mathbf{x})$.

Further, assume that we can make uncertain observations of $f(\mathbf{x})$ by the measurement equation

$$y_{obs} = f(\mathbf{x}) + \epsilon, \tag{13}$$

where the measurement noise ϵ is normally distributed with zero mean and variance σ_ϵ^2, and is independent from the noise of other observations.

Suppose that we have made observations of f at n_{obs} points, given by the random variables $\mathbf{y_{obs}} \in \mathbb{R}^{n_{obs}}$ and wish to predict the value of f at n test points, given by the random variables $\mathbf{y} \in \mathbb{R}^n$. It can be shown that the random variables $\left[\mathbf{y_{obs}}^\top, \mathbf{y}^\top\right]^\top$ are jointly Gaussian distributed[38]. Their probability distribution is

$$\mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}(\mathbf{X}_{obs}, \mathbf{X}_{obs}) + \sigma_\epsilon^2\mathbf{I} & \mathbf{K}\left(\mathbf{X}_{obs}, \mathbf{X}\right) \\ \mathbf{K}\left(\mathbf{X}, \mathbf{X}_{obs}\right) & \mathbf{K}\left(\mathbf{X}, X\right) \end{bmatrix}\right), \tag{14}$$

where $\boldsymbol{I}$ is the identity matrix. $\mathbf{K}\left(\mathbf{X}_{obs}, \mathbf{X}\right)$ denotes the $n_{obs} \times n$ matrix of the covariances evaluated at all pairs of observations and test points, and similarly for the other entries $\mathbf{K}(\mathbf{X}_{obs}, \mathbf{X}_{obs}), \mathbf{K}\left(\mathbf{X}, \mathbf{X}\right)$ and $\mathbf{K}\left(\mathbf{X}, \mathbf{X}_{obs}\right)$.

To make predictions at the test points we take a bayesian inference approach, and calculate the conditional probability distribution of $\mathbf{y}|\mathbf{y_{obs}}$. It can be shown[38] that this results in another Gaussian distribution with mean

$$\overline{\mathbf{y}} = \mathbf{K}\left(\mathbf{X}, \mathbf{X}_{obs}\right)\left[\mathbf{K}(\mathbf{X}_{obs}, \mathbf{X}_{obs}) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{y_{obs}}, \tag{15}$$

and covariance matrix

$$\mathbf{COV}\left(\mathbf{y}\right) = \mathbf{K}\left(\mathbf{X}, \mathbf{X}\right) - \mathbf{K}\left(\mathbf{X}, \mathbf{X}_{obs}\right)\left[\mathbf{K}(\mathbf{X}_{obs}, \mathbf{X}_{obs}) + \sigma_\epsilon^2\mathbf{I}\right]^{-1}\mathbf{K}\left(\mathbf{X}_{obs}, \mathbf{X}\right). \tag{16}$$

Hence, for each test point we have now obtained a Gaussian distribution with an associated mean and covariance. The mean value is used as the predicted values of the unknown function f at the test points. The diagonal elements of the covariance matrix represent the variance of the estimates, and can be used to establish confidence intervals on the predictions. This is illustrated in Figure 5.

Automatic testing using Gaussian Processes and Temporal Logic

In this section, we develop the main scientific result of the paper, a new method for automatic and adaptive simulation-based testing of autonomous vessels. An overview of how the simulator, STL monitor and GP model interact in our proposed method is given in Figure 6.

Problem statement and scope

Consider a case Σ with parameters in $\mathcal{P}_\Sigma$, and a requirement in the form of an STL formula. Recall that a simulation satisfies φ if and only if the STL robustness score is greater than 0. We define the confidence level of a verification as the probability that the STL robustness score is greater than zero for all points $\mathbf{p} \in \mathcal{P}_\Sigma$. The objective of the proposed method

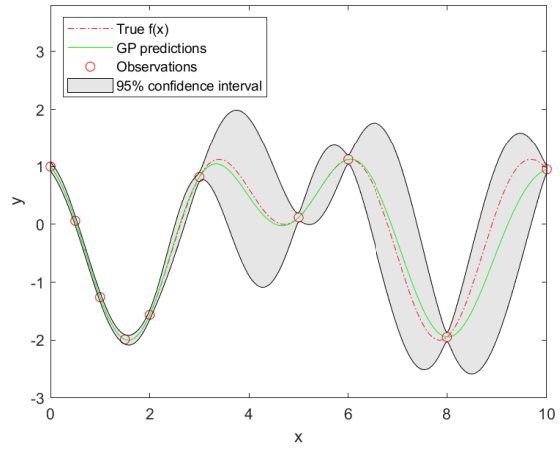

Figure 5. An example of a GP with 10 observations of $y = f(x)$. The GP mean estimates at the test points are shown by the green curve together with their 95% confidence intervals. The figure shows how the GP fit is tighter for small values of x, where the observations are denser. For large values of x, observations are sparser. This gives a less accurate fit, which is reflected by the increased uncertainty bounds.

is to produce evidence that φ is satisfied with a probability greater than the desired confidence level p_{conf} for all points $\mathbf{p} \in \mathcal{P}_\Sigma$.

For simplicity, we restrict the parameter set $\mathcal{P}_\Sigma$ to be a range set in $\mathbb{R}^k$. This can trivially be extended to cover simply connected sets, while the non-connected case can be handled by treating each connected subset separately.

Proposed approach

For a particular STL requirement, φ we propose to model the variations in the STL robustness for different choices of case parameters as a GP. Hence, we consider $f_\rho : \mathcal{P}_\Sigma \mapsto \mathbb{R}$ as an unknown function which maps a parameter vector $\mathbf{p} \in \mathcal{P}_\Sigma$ to a robustness value $\rho \in \mathbb{R}$. In reality, f_ρ is a known function defined by

$$f_\rho(\boldsymbol{p}) := [\![\varphi]\!](sim(\boldsymbol{p})). \tag{17}$$

However, since evaluating this function requires running a simulation, it is often computationally intractable to evaluate it for all points $\mathbf{p} \in \mathcal{P}_\Sigma$, which makes GP estimation an attractive option. We estimate f_ρ at n test points in $\mathcal{P}_\Sigma$. These points are collected in the matrix $\mathbf{P} \in \mathbb{R}^{k\times n}$, such that each column in $\mathbf{P}$ is a parameter vector $\mathbf{p}$, that is, a point in $\mathcal{P}_\Sigma$. The choice of points in $\mathbf{P}$ is typically a uniform grid over $\mathcal{P}_\Sigma$ with the desired resolution. To each test point we assign a random variable representing the unknown STL robustness value at this point. These are collected in the vector $\boldsymbol{\rho} \in \mathbb{R}^n$. We model our prior beliefs in the unknown function $f_\rho(\boldsymbol{p})$ by a joint Gaussian probability distribution with zero mean and covariance matrix $\mathbf{K} \in \mathbb{R}^{n\times n}$. The (i, j)th entry in $\mathbf{K}$ is given by some covariance (kernel) function $k(\mathbf{p_i}, \mathbf{p_j})$.

Next, we wish to observe values of the STL robustness by running simulations and evaluating the resulting output signals against φ using the semantics of (10). We assume that a simulation gives us an uncertain observation of the robustness, given by (13). In reality, the outcome of a simulation is completely deterministic, however, adding a small uncertainty has some technical advantages. This is discussed more closely in the discussion section.

Taking a bayesian inference approach, the posterior predictions of the robustness values at the points $\mathbf{P}$ are updated based on observed robustness values. Suppose that n_{obs} observations have been made at points $\mathbf{P}_{obs} \in \mathbb{R}^{k\times n_{obs}}$. The observed robustness values are collected in the vector $\boldsymbol{\rho}_{obs} \in \mathbb{R}^{n_{obs}}$. Given the assumptions made this far, the robustness values at the points $\mathbf{P}$ and the observed points at $\mathbf{P}_{obs}$ are jointly Gaussian with distribution

$$\begin{bmatrix} \boldsymbol{\rho}_{obs} \\ \boldsymbol{\rho} \end{bmatrix} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} \mathbf{K}_{obs} + \sigma_\epsilon^2 I & \mathbf{K}_{cross} \\ \mathbf{K}_{cross}^\top & \mathbf{K} \end{bmatrix}\right), \tag{18}$$

where $\mathbf{K}_{obs} \in \mathbb{R}^{n_{obs}\times n_{obs}}$ is the covariance matrix for the observation points and $\mathbf{K}_{cross} \in \mathbb{R}^{n\times n_{obs}}$ is the cross covariance between points in $\mathbf{P}$ and $\mathbf{P}_{obs}$.

Using bayesian inference, the conditional probability distribution of $\boldsymbol{\rho}$ given $\boldsymbol{\rho}_{obs}$ is given by

$$\begin{aligned} \boldsymbol{\rho}|\boldsymbol{\rho}_{obs} \sim \mathcal{N}(&\mathbf{K}_{cross}[\mathbf{K}_{obs} + \sigma_\epsilon^2\mathbf{I}]^{-1}\boldsymbol{\rho}_{obs}, \\ &\mathbf{K} - \mathbf{K}_{cross}[\mathbf{K}_{obs} + \sigma_\epsilon^2\mathbf{I}]^{-1}\mathbf{K}_{cross}^\top) \end{aligned} \tag{19}$$

Hence, at each point in $\mathbf{P}$ we now have an associated expected value and uncertainty. In the following, let the operators $\bar{\rho}(\mathbf{p})$ and $var(\rho(\mathbf{p}))$ refer to the expected value and variance of the STL robustness at the point $\mathbf{p}$.

The main loop of our proposed method consists of iteratively running new simulations and updating the GP. This is repeated until a desired confidence level is achieved. The criterion for a sufficient confidence level is that the minimum of the p_{conf} probability confidence interval of f_ρ is greater than 0. That is, we terminate the search in a *Verified* state when

$$min_{\boldsymbol{p}\in\boldsymbol{P}} \quad \bar{\rho}(\mathbf{p}) - n_{conf}\sqrt{var(\rho(\boldsymbol{p}))} > 0. \tag{20}$$

Here, n_{conf} is the number of standard deviations associated with the confidence level p_{conf}.

Achieving this, we have shown that the probability, with which the system satisfies the requirement φ, is greater than p_{conf}, given the assumptions above. If the search process finds a case with robustness lower than 0, the search will terminate prematurely in a *Falsified* state, with an associated counterexample that can be used for debugging and improving the system.

Since observing the robustness at a point involves running a simulation, it is usually a time consuming operation. Thus, it is paramount that sample points are well chosen as to only explore interesting regions of $\mathcal{P}_\Sigma$ and thereby minimizing the number of simulation runs. We propose to choose the next sampling point based on two criteria:

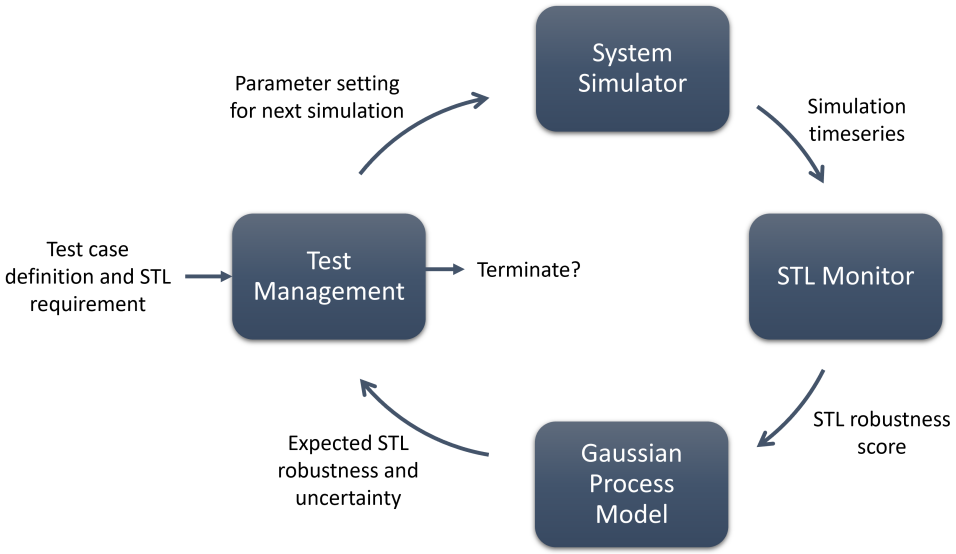

Figure 6. An overview of the main components of the proposed testing methodology and how they interact. The input to the method is a parametric test case definition and an STL requirement to test against. The test management selects a concrete test case from the test space and sends its parameter setting to the simulator. The resulting simulation timeseries is given as input to the STL monitor which calculates the STL robustness score. The GP model, which estimates the STL robustness score as a function of the test case parameters, is updated using the observed STL robustness score. The expected STL robustness and its uncertainty are used for smart selection of the next test case to simulate, and to determine if the test coverage is sufficient to terminate.

- Explore points with low expected robustness.
- Explore points with large uncertainty.

This can be combined in the following selection process

$$\mathbf{p}_{next} = argmin_{\mathbf{p}\in\mathbf{P}} \quad \bar{\rho}(\mathbf{p}) - \kappa\sqrt{var(\rho(\mathbf{p}))}, \qquad (21)$$

where $\kappa \in \mathbb{R}_{\geq 0}$ is a parameter deciding the trade-off between visiting areas with low predicted robustness and visiting areas where there is large uncertainty. This is commonly referred to as the *exploration vs. exploitation trade-off*.

Before starting the search, we first draw n_{seed} *seed points* from $\mathbf{P}$, run simulations and build an initial GP from these observations. This serves to build an initial model of the robustness landscape before starting the adaptive search using (21). To obtain the maximum amount of information about f_ρ from n_{seed} samples, a design of experiments approach is taken, using *latin hypercube sampling* (LHS)[39]. In essence, LHS divides each parameter into n_{seed} slots, and thereby creates a grid of hypercubes over $\mathcal{P}_\Sigma$. Along each dimension, LHS chooses samples such that no two samples occupy a hypercube in the same row, column, height and so forth. Within each hypercube, the sampling is random.

This concludes the development of the proposed testing approach. The method is stated more concisely in Algorithm 1. For a MATLAB/Simulink implementation, the reader is referred to the open-source online repository[40].

Validation of Gaussian Process Model

One of the most attractive properties of the proposed methodology is that it gives a quantification of the confidence level in the verification. This comes in the form of a probability of exceeding a determined robustness limit. However, this probability is based on a statistical model with several assumptions and parameters. It is paramount that these are justified or validated in order to trust the resulting probabilities. The most notable assumption is that the robustness values are jointly Gaussian distributed with covariance given by the selected kernel function. This is hard to justify à priori, since little is known about the robustness function in advance. It is therefore necessary to validate this assumption after running the algorithm. For cases with only one parameter, this can to a large extent be done by visual inspection of the predicted robustness function. However, for higher-dimensional cases, such visual intuition is more difficult. We therefore need quantitative metrics to validate the GP model. A common choice for this is the *marginal likelihood* $p(\boldsymbol{\rho}_{obs}|\boldsymbol{P}_{obs})$. This represents the likelihood of observing the robustness values $\boldsymbol{\rho}_{obs}$ for the points $\boldsymbol{P}_{obs}$ under the given GP model, giving an indication of the goodness of the model fit.

The marginal likelihood is calculated by marginalizing over all values for $\boldsymbol{\rho}$

$$p(\boldsymbol{\rho}_{obs}|\boldsymbol{P}_{obs}) = \int p(\boldsymbol{\rho}_{obs}|\boldsymbol{\rho}, \boldsymbol{P}_{obs})p(\boldsymbol{\rho}|\boldsymbol{P}_{obs})d\boldsymbol{\rho} \qquad (22)$$

As all probability distributions in (22) are Gaussian, this integral has an analytical solution given by the log marginal likelihood

$$\log p(\boldsymbol{\rho}_{obs}|\boldsymbol{P}_{obs}) = -\frac{1}{2}\boldsymbol{\rho_{obs}}^\top(\boldsymbol{K}_{obs} + \sigma_\epsilon^2\boldsymbol{I})^{-1}\boldsymbol{\rho_{obs}} \qquad (23)$$
$$-\frac{1}{2}\log|\boldsymbol{K}_{obs} + \sigma_\epsilon^2\boldsymbol{I}| - \frac{n_{obs}}{2}\log 2\pi$$

Next, we propose a more complete validation method for the GP model. After achieving a verification, the normalized error e_n for each observation $(\mathbf{p}, f_\rho(\mathbf{p}))$ can be calculated as

Algorithm 1: Automatic simulation-based testing

input: Test case Σ, Requirement φ, Kernel function $k(\mathbf{p_1}, \mathbf{p_2})$ and Hyperparameters n_{conf}, κ, σ_ϵ
Select n points uniformly from $\mathcal{P}_\Sigma$, store in $\boldsymbol{P}$;
Calculate covariance matrix $\mathbf{K}$ for points in $\boldsymbol{P}$;
// Find robustness for seed points
Sample n_{seed} seed points from $\boldsymbol{P}$ using Latin Hypercube Sampling;
for i = *1 to* n_{seed} **do**
  Run simulation $\mathbf{w_i} = sim(\mathbf{p_i})$;
  Calculate robustness $[\![\varphi]\!](\mathbf{w_i})$;
  Append $(\mathbf{p_i}, [\![\varphi]\!](\mathbf{w_i}))$ to list of observed points $(\mathbf{P}_{obs}, \boldsymbol{\rho}_{obs})$;
end
// Main loop
for i = $(n_{seed}+1)$ *to maxNumberOfSimulations* **do**
  Calculate covariance matrices $\boldsymbol{K}_{obs}$ and $\boldsymbol{K}_{cross}$;
  Calculate expected value for each point in $\boldsymbol{P}$: $\bar{\boldsymbol{\rho}} = \boldsymbol{K}_{cross}[\boldsymbol{K}_{obs} + \sigma_\epsilon^2 I]^{-1}\boldsymbol{\rho}_{obs}$;
  Calculate variance for each point in $\boldsymbol{P}$: $var(\boldsymbol{\rho}) = \boldsymbol{K} - \boldsymbol{K}_{cross}[\boldsymbol{K}_{obs} + \sigma_\epsilon^2 \boldsymbol{I}]^{-1}\boldsymbol{K}_{cross}^\top$;
  if $min_{\boldsymbol{p}\in\boldsymbol{P}} \quad \bar{\rho}(\mathbf{p}) - n_{conf}\sqrt{var(\rho(\boldsymbol{p}))} > 0$ **then**
    return Verified;
  end
  Select next sample point: $\mathbf{p_i} = argmin_{\mathbf{p}\in\mathbf{P}} \quad \bar{\rho}(\mathbf{p}) - \kappa\sqrt{var(\rho(\mathbf{p}))}$;
  Run simulation $\mathbf{w}_i = sim(\mathbf{p}_i)$;
  Calculate robustness $[\![\varphi]\!](\mathbf{w_i})$;
  if $\rho(\varphi, \mathbf{w}_i) < 0$ **then**
    return Falsified with counterexample $\mathbf{p}_i$;
  end
  Append $(\mathbf{p_i}, [\![\varphi]\!](\mathbf{w_i}))$ to list of observed points $(\mathbf{P}_{obs}, \boldsymbol{\rho}_{obs})$;
end
// Max number of simulations reached
return Inconclusive;

$$e_n(\mathbf{p}) = \frac{\mathbf{f}_\rho(\mathbf{p}) - \bar{\rho}(\mathbf{p})}{\sqrt{var(\rho(\mathbf{p}))}} \tag{24}$$

It can be shown that if the assumptions of the GP hold, then $e_n \sim \mathcal{N}(0,1)$. We propose to validate the method by calculating a reference robustness function by simulating a large number of cases covering the parameter space. Then, the automatic testing algorithm is run, and we calculate the resulting mean and variance for each point on the reference function. This enables the calculation of a large number of normalized errors. The distribution of the normalized errors can then be compared to its theoretical distribution $\mathcal{N}(0,1)$. The normality assumption can be assessed using a normal probability plot.

The method above is useful for validating the method for this paper. However, it is also important for users of the method to validate their verification results. In this case, the validation scheme defeats the whole purpose of the method, as it requires a large number of simulations when creating the reference surface. However, it may be used in a cross-validation setting. Observations can be split into n folds, and the GP can then be trained on $n-1$ of the folds and the normalized error can be calculated for the remaining fold. This process can be repeated such that each fold is left out of training, and we therefore obtain a normalized error for each observation. For cases with two or more parameters, this should give enough simulations to enable a statistical study of the normalized errors. Two important indicators are the mean and the standard deviation of normalized errors. If the mean is significantly less than zero, it indicates that the GP is over-estimating the robustness. If the standard deviation is greater than one, it indicates that the GP is overly confident in its predictions. If such problems appear, this can give input to adjust the hyper parameters and rerun the automatic testing algorithm. Since most of the observations have already been made, this adds little extra time as the time for the GP inference is usually negligible compared to running the simulations.

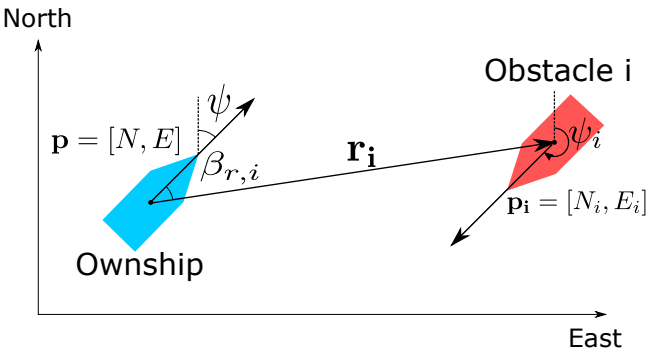

Figure 7. Definition of the symbols and notation used in the case study. The figure shows the position vector $\boldsymbol{p}$, heading ψ, relative position of obstacle i, $\boldsymbol{r_i}$ and relative bearing to obstacle i, $\beta_{r,i}$.

Case study: Open-sea Collision Avoidance

To demonstrate the use of the proposed method a case study has been conducted for CA in open sea. This section first presents the setup for the case study and the definition and evaluation of requirements. The results from applying the automatic testing method and STL evaluation to two test cases are presented and a statistical validation is performed.

Setup

The test object of the case study is the Branching Course Model Predictive Control (BC-MPC) algorithm. The system is implemented on a small high-speed vessel, details of the algorithm and the vessel are given in Eriksen et al. (2019)[41]. For the sake of simplicity, perfect situational awareness is assumed, that is, the CA system has perfect tracks on obstacles.

To evaluate simulations, certain signals are calculated based on position, heading and speed of the vessels. The symbols and notation used in the case study are shown in Figure 7. It is assumed that the heading and course are identical, that is, no side-slipping or cross currents. The *distance at closest point of approach* (d_{CPA}) is defined as the smallest separation between two vessels if they continue with the current speed and course. The *time to closest point*

Table 2. Parameters in the STL formulas and the normalization factors for the requirements. The subscript in the normalization factor corresponds to the signal that it normalizes.

Parameter	Value	Unit
$t_{CPA.turn}$	15	s
$d_{CPA.min}$	50	m
d_{min}	50	m
e_{max}	400	m
$\nu_{t_{CPA}}$	10	s
$\nu_{d_{CPA}}$	100	m
ν_{β_r}	10	deg
ν_d	100	m
ν_e	400	m

Table 3. Requirements and subformulas in the case study.

$\varphi_{safety} = \square\neg\left(d_i \leq d_{min}\right)$
$\varphi_{mission} = \square\left(e \leq e_{max}\right)$
$\varphi_{colreg} = \square\left(HO \rightarrow \varphi_{HO} \wedge GW \rightarrow \varphi_{GW} \wedge OT \rightarrow \varphi_{OT}\right)$
$\varphi_{HO} = t_{CPA} \leq t_{CPA,turn} \rightarrow \beta_r \in [-170°, -10°]$
$\varphi_{GW} = t_{CPA} \leq t_{CPA,turn} \rightarrow d_{CPA} \geq d_{CPA,min}$
$\varphi_{OT} = t_{CPA} \leq t_{CPA,turn} \rightarrow d_{CPA} \geq d_{CPA,min}$
$\varphi^C_{OT} = \varphi^C_{OV} = \varphi^C_{GW} = \varphi^C_{SO} = \varphi^C_{HO} = t_{CPA} \leq 0$
$\varphi^C_{NC} = d_{CPA} \leq d_{CPA,min} \wedge t_{CPA} \leq t_{CPA,min}$

of approach (t_{CPA}) is defined as the time until d_{CPA} occurs.

The STL robustness values for the requirements are saturated and normalized. This rationale for saturation is that very large robustness values are not interesting, but they make it harder to fit a GP to them. Normalization aims to ease the process of hyper parameter selection in the GP by having all requirements operate on the same scale, such that a set of hyper parameters are likely to work for a broad range of requirements. The normalization and saturation are defined by one extra parameter for each predicate, a normalization factor ν. The robustness of this predicate is calculated by first saturating the robustness to $[-\nu, \nu]$ and then dividing by ν such that all robustness values are mapped to the interval $[-1, 1]$.

Requirements

Automatic quantitative evaluation of simulations are achieved by testing against three STL requirements: COLREG, safety distance and mission compliance. These are detailed in the following. All parameters used in the requirements are given in Table 2. We also emphasize that the automatic testing method scales well for testing against multiple requirements, as the GP for a particular requirement can be initialized with the observed simulations from previous runs against other requirements. It is also possible to test against all requirements in a single run, by testing against a conjunction of all requirements. The robustness for the individual requirements can be easily decomposed during post processing. For simplicity and clarity these optimizations are not utilized in this case study. The STL formulas for all requirements are summarized in Table 3.

COLREG requirement: COLREG are the international regulations for preventing collisions at sea[42]. Automatic evaluation of COLREG compliance is a challenging topic due to the complexity and wide span of the possible scenarios combined with the inherent reliance on human judgement of the current COLREG. This section presents a new take on this problem, by using a formal language (STL) to express COLREG. Using a formal language enables systematic construction of modular requirements which can be subject to mathematical analysis such as consistency checking and formal verification. While this is by no means a complete system for COLREG evaluation, this aims to illustrate and motivate a new possible direction for automatic COLREG evaluation.

We propose to formulate the STL COLREG as a conjunction of reactive subformulas. In each subformula, the antecedent is a boolean variable expressing a COLREG situation, and the consequent is a set of required behaviours which must be satisfied in the corresponding COLREG situation. Five COLREG situations are defined, corresponding to COLREG Rules 13-17: *overtaking* (OT), *overtaken* (OV), *give way* (GW), *stand on* (SO) and *head-on* (HO). Here, only the situations which require explicit action by ownship are included in the requirement, given in (25).

$$\varphi_{colreg} = \square\left(HO \rightarrow \varphi_{HO} \wedge GW \rightarrow \varphi_{GW} \wedge OT \rightarrow \varphi_{OT}\right) \tag{25}$$

In a head-on situation, COLREG require a port-to-port passing. This is enforced by requiring that the relative bearing β_r is between $-170°$ and $-10°$ when t_{CPA} is lower than the threshold value $t_{CPA,turn}$:

$$\varphi_{HO} = t_{CPA} \leq t_{CPA,turn} \rightarrow \beta_r \in [-170°, -10°] \tag{26}$$

For give way and overtaking situations, COLREG do not specify a required maneuver, but require that ownship makes early and substantial action to avoid a collision. We enforce this by requiring that d_{CPA} is larger than the threshold $d_{CPA,min}$ when the t_{CPA} is lower than the threshold $t_{CPA,turn}$:

$$\varphi_{GW} = \varphi_{OT} = t_{CPA} \leq t_{CPA,turn} \rightarrow d_{CPA} \geq d_{CPA,min} \tag{27}$$

The COLREG requirements of (25) use boolean signals which specify the type of COLREG situation. We adopt the COLREG situation selection of Tam & Bucknall (2016)[43], which classifies the situations based on sectors for relative heading and bearing. One key difference in our approach is that we distinguish between two different classes of overtaking situations. In an *overtaking* situation, ownship is overtaking an obstacle, which is deemed to exist when the obstacle is ahead of ownship. In an *overtaken* situation, an obstacle overtakes ownship, which is deemed to exist when the obstacle is aft of ownship. This separation is necessary when designing reactive requirements, because the two situations have different required behaviours by COLREG.

When selecting the COLREG situation, it is of great importance to have persistence throughout an encounter. For instance, using the rules of Tam & Bucknall alone, a head-on situation could change into a give way situation

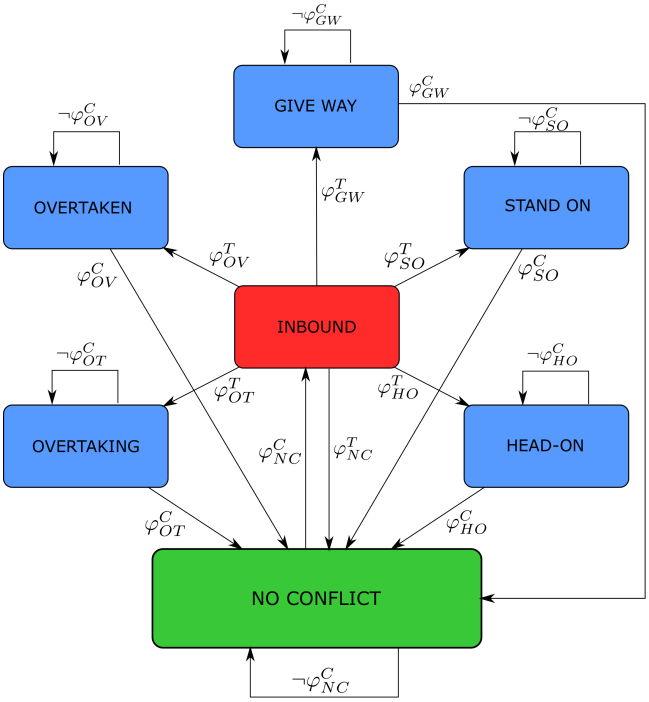

Figure 8. Finite-state Machine for persistent selection of COLREG situation. Transitions are defined by complementary STL formulas. The formulas can be divided into trigger conditions (superscript T) and cease conditions (superscript C).

due to the course change of the avoidance maneuver. We propose to avoid this problem by using a finite-state machine (FSM), as shown in Figure 8. The FSM has seven states, and the transitions between them are determined by the satisfaction of complementary STL formulas. Five of the states correspond to the five COLREG situations described above. The FSM starts in a *no conflict* state. It exits this state if an obstacle is on collision course, and the collision is sufficiently close in time. This is represented by the STL formula

$$\varphi^C_{NC} = d_{CPA} \leq d_{CPA,min} \wedge t_{CPA} \leq t_{CPA,min} \quad (28)$$

After exiting the no conflict state, the FSM enters an intermediate *inbound* state. From here the type of the encounter is determined, where it will transition to one of the five COLREG situations or back to the no conflict state. Each COLREG situation has a *trigger* condition (superscript T) which corresponds to the sectors for relative heading and bearing outlined above, and a *cease* condition (superscript C). The FSM will remain in the same state until the cease condition is satisfied and thus achieves persistent selection of the COLREG situation throughout an encounter. The cease condition is equal for all situations, and is represented by the t_{CPA} being negative:

$$\varphi^C_{OT} = \varphi^C_{OV} = \varphi^C_{GW} = \varphi^C_{SO} = \varphi^C_{HO} = t_{CPA} \leq 0 \quad (29)$$

The interpretation of this is that a negative t_{CPA} means that the closest point of approach is in the past, and hence the ships have passed each other. The FSM will then transition to a *No Conflict* state.

Safety distance requirement: The safety distance requirement is perhaps the most important requirement. There exist different approaches to define the safe distance, depending on, for instance, the speed, relative bearing and size of the vessels involved[41,44]. For simplicity, the safety distance requirement only specifies a limit on the minimum separation between vessels in this case study. The separation is the Euclidean norm of the relative position vector $d = \|\boldsymbol{r}_i\|$. The STL formula for the safety distance requirement is

$$\varphi_{safety} = \Box\neg\,(d_i \leq d_{min}) \quad (30)$$

Mission requirement: It is also important to verify that the system completes the task it was set out to do. This can, for instance, uncover deadlocks where the system is safe and COLREG compliant but useless. Lack of mission compliance can also create dangerous situations by bringing the system outside its operational design domain (ODD). For this case study, a simple mission requirement is used, which states that the vessel should not deviate significantly from its pre-planned path. The deviation from the path is captured from the cross-track error. A piece wise linear path is assumed, which gives the following expression for the cross-track error[45]

$$e(t) = -[N(t) - N^k_{wp}]\sin(\alpha_k) + [E(t) - E^k_{wp}]\cos(\alpha_k) \quad (31)$$

where N^k_{wp} and E^k_{wp} are the north and east position of waypoint k, and $\alpha_k = atan2(E^{k+1}_{wp} - E^k_{wp}, N^{k+1}_{wp} - E^k_{wp})$ is the angle between path segment k and the north axis.

The STL mission requirement states that the cross-track error should always be lower than the threshhold e_{max}, which enforces the vessel not to deviate too far from the preplanned path.

$$\varphi_{mission} = \Box\,(|e| \leq e_{max}) \quad (32)$$

where $|e|$ denotes the absolute value of e.

Case 1: Obstacle on direct collision course with varying course

The first test case is a situation where ownship is travelling in a straight line, and encounters an obstacle on a direct collision course. This case has only one parameter, θ, describing the relative course of the obstacle. This is illustrated in Figure 9. The range for θ is set to $[-160°, 160°]$. Courses in $[-180°, -160°] \cup [160°, 180°]$ are not included as they would cause the obstacle to start unreasonably close to ownship. The hyper parameters used in the automatic testing method are given in Table 4.

Figure 10 shows the results from running the automatic testing method on Case 1 against the safety distance requirement. To validate the method, 321 tightly spaced simulations were also run to get a ground truth robustness curve for comparison. The results show that this case is verified in 27 simulations with a minimum robustness of 0.6. The estimated robustness curve from the GP predicts the ground truth robustness well, and is almost always on the conservative side when there is a deviation. This is as

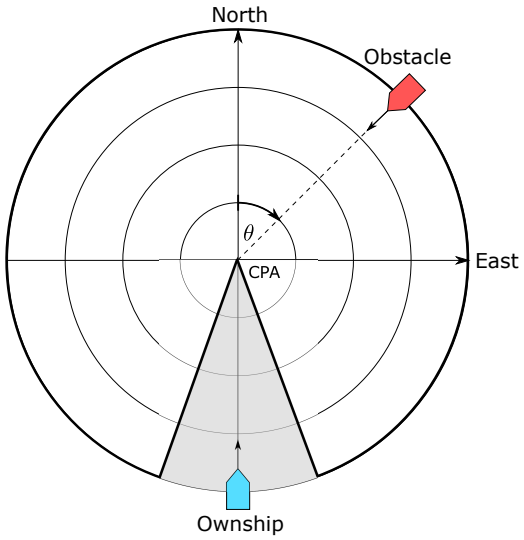

Figure 9. Definition and parametrization of Case 1. The figure depicts an obstacle on direct collision course with varying course given by the parameter $\theta \in [-160°, 160°]$.

Table 4. Hyper parameters used in the case study.

Parameter	Case 1	Case 2
n_{conf}	3	3
κ	2	2
σ_ϵ	0.02	0.02
σ	0.5	0.5
l	$10°$	$[6°, 2m/s]$

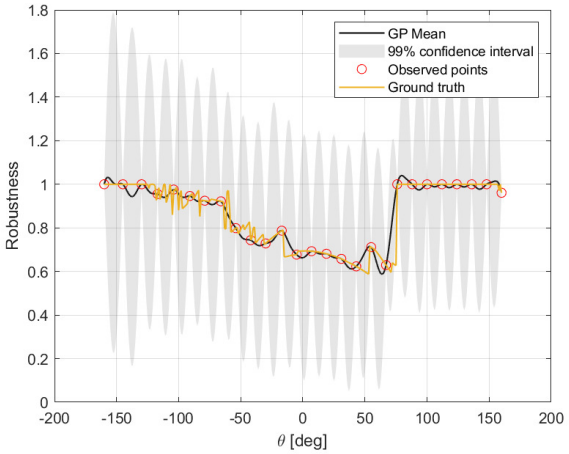

Figure 10. Robustness of the safety distance requirement in Case 1. The figure shows that the requirement is verified as the lower uncertainty bound is above zero for the entire parameter space.

expected, as the GP estimate will be pulled towards the prior assumption of zero robustness in regions with sparse observation. A sharp jump can be observed at $\theta = 75°$. The jump corresponds to a decision boundary where the CA system transitions from maneuvering aft of the obstacle to maneuvering in front of the obstacle, which results in a sudden increase in the safety margins. Such decision boundaries are common in autonomous navigation systems and it is therefore important that the GP is able to react properly to them.

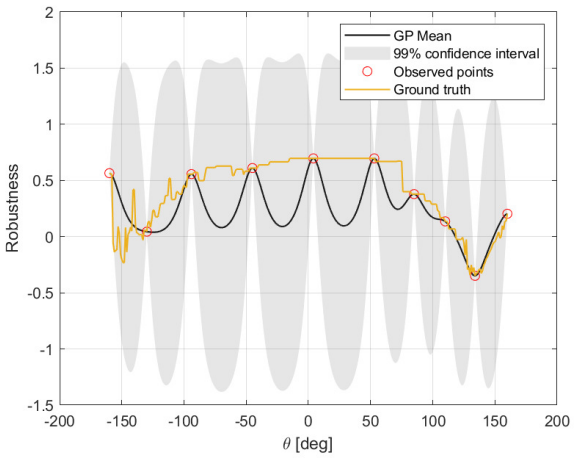

Figure 11. Robustness of the mission requirement in Case 1. The figure shows that the requirement is falsified after 10 simulations as a parameter setting with robustness less than zero is observed at $\theta = 130°$.

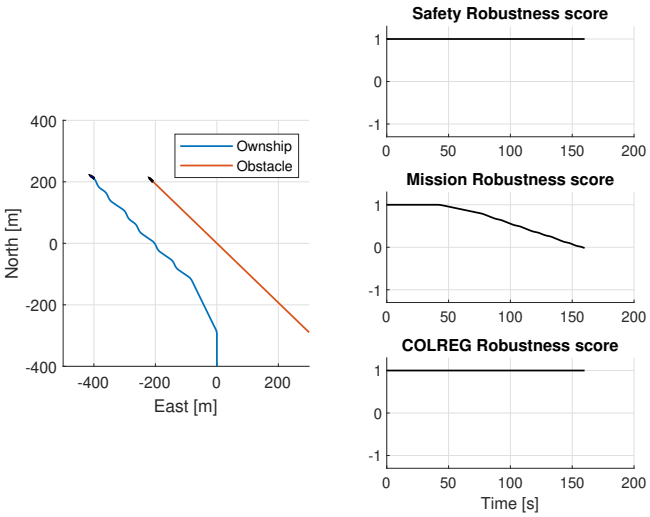

Figure 12. Falsifying behaviour for the mission requirement in Case 1. The figure shows that the autonomous navigation system on ownship enters a deadlocked state, where it is intercepted by the obstacle and is not able to return to its planned trajectory

Figure 11 shows the results for the mission requirement. This run resulted in a falsification after 10 simulations, as a case with robustness less than zero was observed at $\theta = 130°$. By going back and rerunning the simulation with the falsified parameter setting, the falsifying behaviour can be uncovered. The falsifying behaviour is visualized in Figure 12. This clearly shows that the autonomous navigation system on ownship enters a deadlocked state, where it is intercepted by the obstacle and is not able to return to its planned trajectory. This highlights the importance of completeness in the requirements. Although this case passed the safety distance requirement, it still had unwanted and potentially dangerous behaviours.

Finally, Figure 13 shows the results from running the automatic testing method against the COLREG requirement. This resulted in a verification after 27 simulations. The results show good robustness over the entire parameter space. There is a step down in robustness in the interval $\theta =$

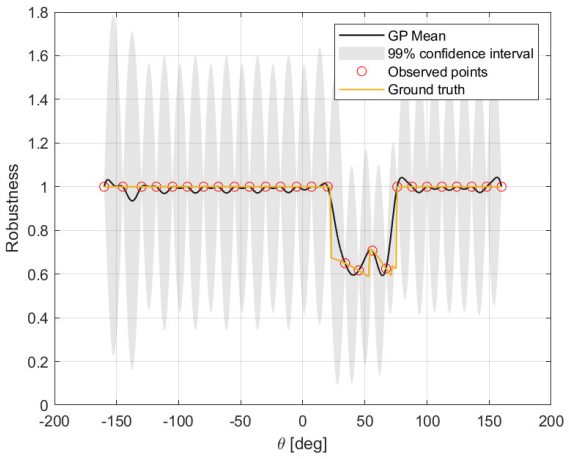

Figure 13. Robustness of the COLREG requirement in Case 1. The figure shows that the requirement is verified after 27 simulations.

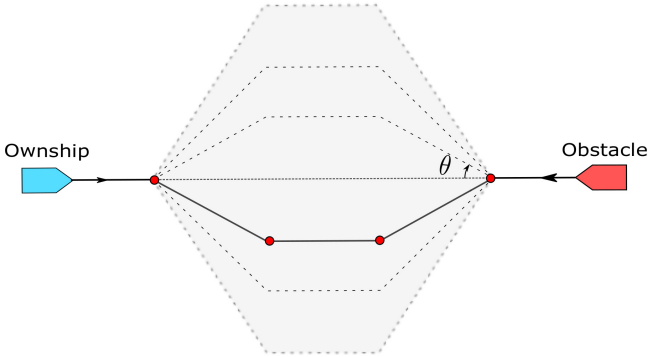

Figure 14. Illustration of Case 2. This case has two parameters, the angle of the avoidance maneuver, $\theta \in [-60°, 60°]$ and the speed of the obstacle $U \in [0, 20]m/s$.

$[22.5°, 75°]$. The step at $22.5°$ is due to a change in COLREG evaluation from *Head-on* to *Give way* and the step at $75°$ is again due to a decision boundary of the CA system, where it transitions from maneuvering aft of the obstacle to maneuvering in front of the obstacle.

Case 2: Head-on encounter with avoidance maneuver by obstacle

The second case is a head-on situation where the obstacle performs a predefined avoidance maneuver. This aims to test the CA systems ability to handle dynamic maneuvers by the obstacle. The case has two parameters, as illustrated in Figure 14. They are the angle of the maneuver, $\theta \in [-60°, 60°]$ and the speed of the obstacle $U \in [0, 20]m/s$. As the parameter space is now two-dimensional, there will be a robustness surface instead of a robustness curve as in Case 1.

We begin again by testing against the safety distance requirement. The results are shown in Figure 15. The automatic testing method resulted in a falsification after 273 simulations, as a case with robustness of -0.38 is observed at $U = 18m/s$ and $\theta = -24°$. We also note that Figure 15 illustrates the adaptivity in the test case selection well, as the

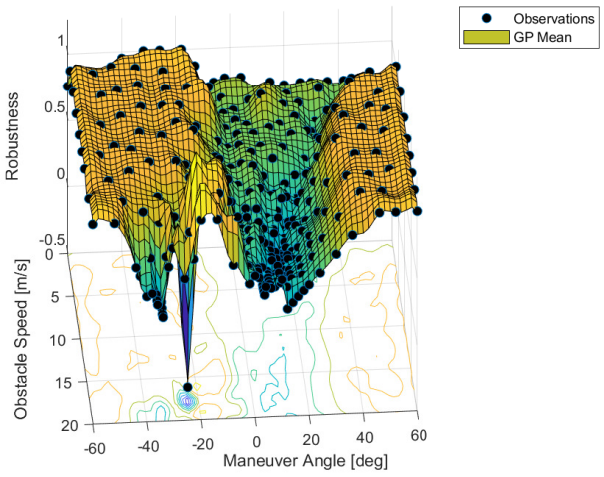

Figure 15. Inferred robustness surface and observations from running the automatic testing method against the safety distance requirement. The results show a falsification after 273 simulations.

simulations are much denser in areas of low robustness.

The identified safety violation appears to lie on a very sharp spike of low robustness. To verify this, and assess the overall prediction of the GP, a reference robustness surface was generated by running 2501 simulations in a 61x41 grid over the parameter space. This corresponds to steps of $\theta = 2°$ and $U = 0.5m/s$. The reference robustness surface is shown in Figure 16. The reference surface shows a good overall match with the predicted surface by the GP and shows that indeed there is a very narrow spike of low robustness. In fact, of the 2501 simulations, only five resulted in a safety violation. This is clearly a very difficult or time consuming safety violation to detect by manual or brute force approaches. Furthermore, it illustrates that even for such a simple case with only one obstacle in open water and perfect situational awareness, safety violations that are difficult to anticipate and detect can emerge.

To examine the robustness landcape around the safety violation, 2911 simulations were ran in a dense 71x41 grid in the range $\theta = [-26°, -22°]$ and $U = [13, 20]m/s$, the resulting robustness surface is shown in Figure 17. This shows a sharp and narrow cleft. The vertical walls of the cleft indicate that the safety violation occurs in a region between two decision boundaries. In these boundaries, the CA system goes from full robustness to a safety violation in an arbitrarily small change in the case parameters. This type of falsification is particularly challenging to detect, because there is no gradient information in the robustness surface to guide the adaptive search towards the safety violation.

As before, we replay the simulation with the falsifying behaviour to examine the cause of the safety violation. This can be very useful input for debugging and fixing the control software. The course of events is illustrated by the time-lapse in Figure 18. Ownship (blue trace) first turns starboard to do a port-port passing in accordance with COLREG Rule 14 for head-on situations. As the obstacle breaks COLREGS and turns port, ownship turns more steeply starboard to avoid a collision while still being COLREG compliant. Finally,

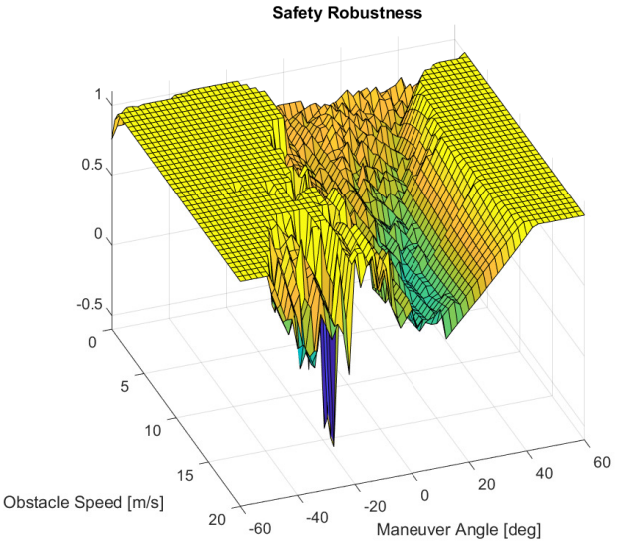

Figure 16. Ground truth robustness surface for the safety distance requirement, obtained by running 2501 simulations on a 61x41 grid over the parameter space. The surface shows a very narrow spike with low robustness.

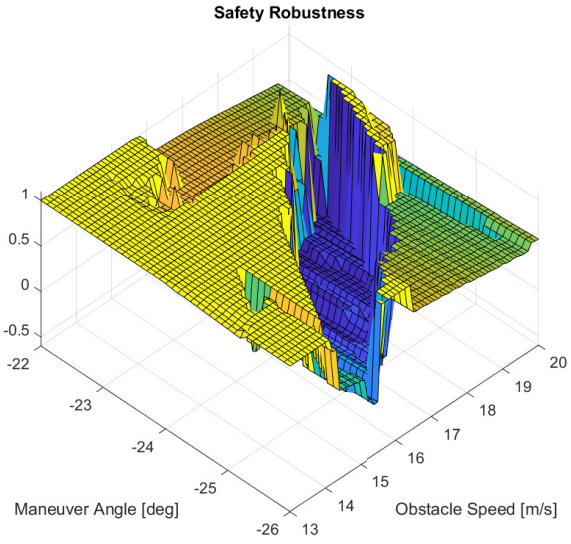

Figure 17. A closer look at the narrow spike with a safety violation, obtained by running 2911 simulations on a 71x41 grid in the range $\theta = [-26°, -22°]$ and $U = [13, 20] m/s$.

ownship determines that a collision can not be avoided by a port-port passing and decides to break Rule 14 and turn steeply port. At the same time, the obstacle turns starboard and a collision occurs.

To better understand the safety violation and see what the decision boundaries represent, simulations from both sides of the cleft were also replayed. The selected cases all had $U = 16 m/s$ and $\theta = [-26°, -25°, -24°]$. This corresponds to points on the right, in the middle and to the left of the cleft in Figure 17. The simulations showed that for $\theta = -24°$, the CA system decided to be compliant to Rule 14 throughout the encounter. For $\theta = -26°$, on the other hand, it decided early to break Rule 14 and turn port.

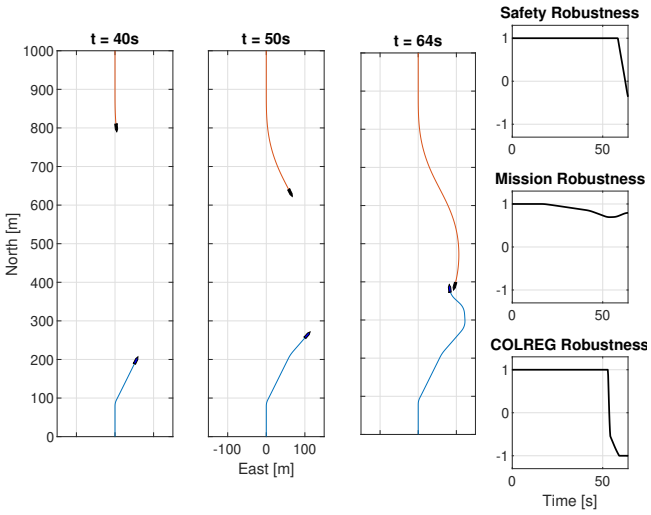

Figure 18. Time-lapse which shows the falsifying behaviour identified by the automatic testing method. Ownship is shown with a blue trace, and the obstacle with an orange trace. The online STL robustness is plotted against time in the right column, which shows that both the safety distance requirement and the COLREG requirement are violated at the end of the simulation.

Both of these decisions resulted in large safety margins. However, in the very narrow region in between, the CA system is indecisive and switches from the first strategy to the second at the critical moment, which results in a collision.

For brevity, the results when testing against the COLREG and mission requirements are not presented in detail, but the code and data needed to generate these results are available in an open-source online repository[40]. The key results are that testing against the mission requirement led to a verification in 267 simulations with a minumum observed robustness of 0.29. Not surprisingly, testing against the COLREG requirement resulted in a falsification after 5 simulations. This is not necessarily a problem, as there exists situations where it is necessary to break the explicit COLREG rules in order to navigate safely. It may nevertheless be instructive to calculate a robustness surface to get an overview of which situations the CA system decides not to be COLREG compliant and to map possible decision boundaries.

Statistical Validation

Next, we apply the proposed validation method to the results from the case study. Since we are only interested in validating runs which resulted in a verification, the safety distance requirement from Case 1 and the mission requirement from Case 2 are chosen.

The reference robustness function for the safety distance requirement in Case 1, shown in Figure 10 contains 321 observations, hence we have a sample of 321 normalized errors. The distribution and normal probability plot for the normalized errors are given in Figure 19. The mean on the normalized errors is -0.036, which shows that the distribution is close to centered about zero. The standard deviation is 0.59, which is less than the theoretical value of 1. This indicates that the GP is conservative in its

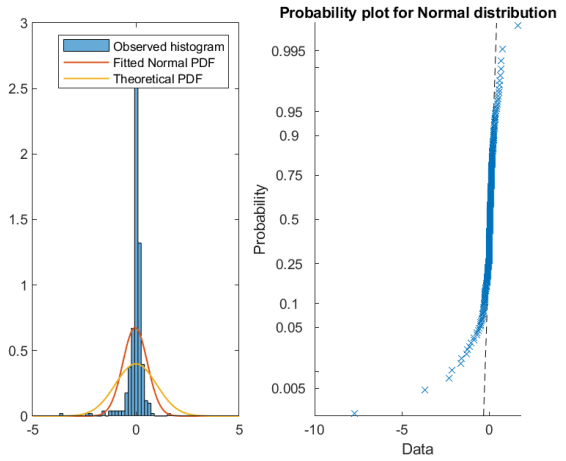

Figure 19. Statistical validation of the safety verification in Case 1. The left plot shows the histogram of the observed normalized errors together with the fitted and theoretical normal distributions. To the right, a normal probability plot is shown for the fitted distribution.

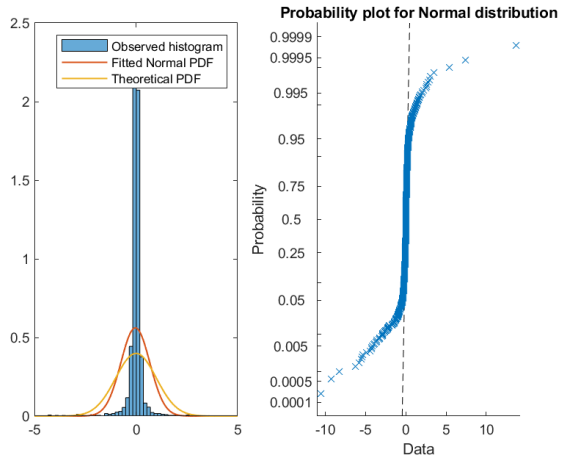

Figure 20. Statistical validation of the mission verification in Case 2. The left plot shows the histogram of the observed normalized errors together with the fitted and theoretical normal distributions. To the right, a normal probability plot is shown for the fitted distribution.

confidence measure. The normal probability plot shows an S-shape, which is characteristic for distributions with light tails. This means that many of the observations are centered around zero, but there are more extreme observations than what would be expected from a normal distribution. This is backed by the histogram of the normalized errors. Further investigations showed that these extreme observations are around the discontinuity at $\theta = 75°$, which is expected as the robustness function clearly disagrees with the smoothness property of the covariance function.

The reference robustness function for the mission verification in Case 2 contains 2501 observations, giving 2501 samples of the normalized error. The distribution and normal probability plot for the normalized errors are given in Figure 20. The mean on the normalized errors is -0.032, which again shows that the distribution is close to centered about zero. The standard deviation is 0.71, again less than the theoretical value of 1. The characteristics of the observed distribution are close to those of the radial case, where again the extreme values reside around discontinuities in the robustness surface. The results from this validation indicate the GP model for the most part fits well with the observations. The extreme values indicate discontinuities which may require further investigation by the verifier.

Discussion

The results from the case study indicate that the proposed methodology has promise as a tool in the verification process for autonomous ships. Some aspects regarding its use in practice and the interpretation of the results are discussed next.

Measurement uncertainty in the Gaussian Process

As shown in (13), the uncertainty when making observations can be modelled as independent Gaussian random variables. In our case, the observations are deterministic simulations which have no uncertainty associated with them. It has, however, shown to be useful to add a small observation uncertainty to relax the model fit. As shown in (19), the inference step involves the inversion of the matrix $\left[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\epsilon^2\mathbf{I}\right]$. If the distance between two observations is small, compared to the length scale of the covariance function, $\mathbf{K}(\mathbf{X},\mathbf{X})$ will have two rows which are very similar, and it will therefore be close to singular. Adding some measurement noise adds a positive number to the diagonal and therefore increases the condition number of the matrix to be inverted. This also makes intuitive sense, as the measurement noise implies that the GP mean does not need to exactly match each observation. The low condition number of $\left[\mathbf{K}(\mathbf{X},\mathbf{X}) + \sigma_\epsilon^2\mathbf{I}\right]$ manifests itself as oscillations in the GP mean surrounding sharp changes in the robustness. Some tendencies for this can be seen at the discontinuity in Figure 10 and at the falsifying downward spike in Figure 15. This has been greatly mitigated by adding an artificial measurement uncertainty.

There also exist other possible approaches to make it easier to fit a GP to the robustness surface, which instead of relaxing the GP fit by adding artificial uncertainty in the measurement, try to regularize the robustness surface. One approach is to use a smooth robustness operator[46] to obtain a smoother surface. Another approach is to use interface aware STL[47], which distinguishes between input and output signals, and only calculates robustness on the output signals. This can remove some discontinuous jumps in the robustness surface. Both of these directions stand out as interesting candidates for future work.

Choice of hyper parameters

The choice of hyper parameters is an important part of using the proposed testing method. Since there are several of them, this gives large flexibility to tweak the verification results to the needs of the user. Therefore, it is important to have good a process for objectively selecting them. We share some experiences and insight for this next.

The hyper parameters are listed in Table 4. n_{conf} can be freely chosen up front by the user, depending on the desired confidence level. This can for instance be based on a risk analysis which determines the consequence of breaking a certain requirement for this test case. κ determines the trade-off between exploration and exploitation when selecting the next simulation to run. This does not have a large effect on the final prediction of the robustness surface, but it may affect how many simulations are needed to obtain it. Generally, small values are favourable for fast falsification, as the search is guided toward areas with low robustness, whereas large values are favourable for fast verification, as the search is guided towards unexplored areas with high uncertainty. We found that $\kappa = 2$ offered a good trade-off in all runs. This corresponds to selecting the point which has the lowest 95% confidence interval. As discussed above, the measurement noise, σ_ϵ, should be close to zero, but a small positive number is preferable to stabilize the inference. We found that $\sigma_\epsilon = 0.02$ gave a stable inference without having a noteworthy effect on the uncertainty level.

Finally comes the choice of kernel function and its parameters. We used the ARD Matèrn 5/2 Kernel[48], which is a generalization of the simple Sqauared Exponential introduced in (12), with separate length scales for each case parameter. This kernel offered better handling of sharp edges, where the Squared Exponential tended to introduce oscillations in the robustness surface. The Matèrn kernel has the same parameters as the Squared Exponential, the variance σ, and the length scales l for each case parameter.

The σ hyper parameter is a scaling factor which determines the scale of the assumed variations in the robustness. We chose it to give a reasonable prior distribution based on the fact that all robustness values are in the interval $[-1, 1]$. The prior distribution is zero-mean with variance of σ^2 over the entire parameter space. By choosing $\sigma = 0.5$ we get a prior which assumes that 95% of the robustness values are in the interval $[-1, 1]$. This choice gave good results for all runs.

The length scales l describes how smooth the robustness function is. Our experience is that this is the hyper parameter which has the greatest effect on the verification result and which requires the most attention when creating the GP model. This parameter determines how quickly the uncertainty increases away from observations, and therefore it essentially determines how many simulations are needed to cover the parameter space with the desired uncertainty. If the length scales are chosen too large, the verification can miss sharp spikes, such as the one in Figure 15. A practical approach to this is to first select the length scales based on how many simulations are feasible to run. We found that using a ratio between the width of the parameter range and the length scale of around 20 offered a good starting point, resulting in about 30 simulations for the 1-parameter runs and 300 simulations for the 2-parameter runs. Then, the results can be validated using the proposed method described above. If the length scales are chosen too large, this should be reflected in the distribution of the normalized errors by a standard deviation greater than 1 and possibly a non-zero mean. The length scales should then be decreased.

Usage in the approval process for autonomous vessels

Finally, we discuss the context in which this method can be used in the approval process for autonomous vessels. Both the classification society DNV[49] and Norwegian Maritime Directorate[50] have proposed approval processes which by large contain the same main steps. Starting with establishing the Concept of Operations (CONOPS) which gives input to a Preliminary Hazard Analysis (PHA/HAZID) at an early stage, and later a detailed risk analysis. The output of the risk analysis will be a set of identified risks. To get an approval, evidence needs to be produced that all of the identified risks are adequately mitigated. This evidence can come in many forms, such as documentation, expert judgement, analytical results and physical tests. For autonomous vessels it has, however, become evident that the complexity makes it necessary to do extensive simulation-based testing to verify some claims. The testing method proposed in this paper can produce evidence for some the safety claims that are difficult to prove otherwise. This can for instance be done as shown in this case study, where some traffic situations which are considered important or critical are selected and tested against a predefined set of requirements.

Another possible use is to link it more directly to the risk analysis. Systems Theoretic Process Analysis (STPA) has been proposed as a risk analysis tool for autonomous vessels[51]. An outcome of an STPA analysis is a set of loss scenarios with corresponding safety constraints for preventing such losses. The safety constraints could be expressed as STL requirements and the loss scenarios could be parametrized as is done in this paper, enabling automatic verification by the proposed testing methodology. How to build trust in autonomous vessels is a challenging topic and an active area of research. How to integrate the proposed methodology in an approval process and in a design process is an important area of future research.

Conclusions

We have developed a methodology for automatic simulation-based testing of control systems for autonomous vessels. The work was motivated by the need for increased test coverage and formality in the verification efforts for autonomous vessels. It aimed to achieve this by formulating requirements in the formal logic STL, which enabled automatic evaluation of simulations against requirements using the STL robustness metric. Furthermore, the work used a GP model to estimate the robustness score and uncertainty level for untested cases. The GP model was used to automatically guide the case selection towards cases with low robustness or high uncertainty. The main contribution of our work was an automatic testing method which incrementally runs new simulations until the entire parameter space of the case is covered to the desired uncertainty level, or until a case which falsifies the requirement is identified. The methodology was demonstrated through a case study,

where the test object was a CA system for a small high-speed vessel. Requirements for safety distance, mission compliance and COLREG compliance were developed. The results showed promise, by both achieving verification in feasible time and identifying falsifying behaviours which would be difficult to detect manually or using brute-force methods. An additional contribution of this work was a formalization of COLREG using temporal logic, which appears to be an interesting direction for future work.

Acknowledgements

This work was supported by the Research Council of Norway through the Centres of Excellence funding scheme, project number 223254 - NTNU AMOS, and the KPN ORCAS project, project number 280655. We wish to thank Bjørn-Olav Holtung Eriksen for providing an implementation and simulator for the BC-MPC collision avoidance algorithm for use as a test subject in the case study. We sincerely thank the anonymous reviewers for their constructive feedback which has improved the quality and clarity of the paper.

References

1. NFAS. Projects, 2020. URL https://nfas.autonomous-ship.org/resources_page/projects-page/.
2. Koopman P, Ferrell U, Fratrik F et al. A Safety Standard Approach for Fully Autonomous Vehicles. In *Computer Safety, Reliability, and Security*. Springer, pp. 326–332.
3. Pedersen TA, Glomsrud JA, Ruud EL et al. Towards simulation-based verification of autonomous navigation systems. *Safety Science* 2020; 129(December 2019): 104799.
4. Skjetne R and Egeland O. Hardware-in-the-loop testing of marine control systems. *Modeling, Identification and Control* 2006; 27(4): 239–258.
5. Johansen TA, Sørensen AJ, Nordahl OJ et al. Experiences from Hardware-in-the-loop (HIL) Testing of Dynamic Positioning and Power Management Systems. In *2nd International Conference on Technology & Operation of Offshore Support Vessels*. 2007, RINA IMAREST, pp. 41–50.
6. Smogeli OM. Experiences From Five Years of DP Software Testing. In *Dynamic Positioning Conference*. 2010, MTS, pp. 1–12.
7. Smogeli O and Skogdalen JE. Third party HIL testing of safety critical control system software on ships and rigs. In *Offshore Technology Conference*. 2011, One Petro, pp. 839–845.
8. Sørensen AJ and Ludvigsen M. Underwater Technology Platforms. *Encyclopedia of Maritime and Offshore Engineering* 2018; : 1–11.
9. Donzé A. Breach, A Toolbox for Verification and Parameter Synthesis of Hybrid Systems. In *Computer Aided Verification*. 2010, Springer, pp. 167–170.
10. Hoxha B, Bach H, Abbas H et al. Towards Formal Specification Visualization for Testing and Monitoring of Cyber-Physical Systems. In *International Workshop on Design and Implementation of Formal Tools and Systems*. 2014, ASU, pp. 1–10.
11. Dreossi T, Dang T, Donzè A et al. Efficient guiding strategies for testing of temporal properties of hybrid systems. In *NASA Formal Methods*. Springer, pp. 127–142.
12. Tuncali CE, Fainekos G, Prokhorov D et al. Requirements-Driven Test Generation for Autonomous Vehicles With Machine Learning Components. *IEEE Transactions on Intelligent Vehicles* 2020; 5(2): 265–280.
13. Bartocci E, Bortolussi L and Sanguinetti G. Data-driven statistical learning of temporal logic properties. *Lecture Notes in Computer Science* 2014; 8711(600708): 23–37.
14. Woerner K. *Multi-Contact Protocol-Constrained Collision Avoidance for Autonomous Marine Vehicles*. PhD Thesis, Massachusetts Institute of Technology, 2016.
15. Stankiewicz PG and Mullins GE. Improving Evaluation Methodology for Autonomous Surface Vessel COLREGS Compliance. In *OCEANS Marseille*. 2019, pp. 1–7.
16. Lee R, Mengshoel OJ, Saksena A et al. Adaptive stress testing: Finding likely failure events with reinforcement learning. *Journal of Artificial Intelligence Research* 2020; 69(2020): 1165–1201.
17. Luckcuck M, Farrell M, Dennis LA et al. Formal specification and verification of autonomous robotic systems: A survey. *ACM Computing Surveys* 2019; 52(5): 100.
18. Shokri-Manninen F, Vain J and Waldén M. Formal Verification of COLREG-Based Navigation of Maritime Autonomous Systems. In *Software Engineering and Formal Methods*. Springer, pp. 41–59.
19. Park J and Kim J. Autonomous docking of an unmanned surface vehicle based on reachability analysis. In *International Conference on Control, Automation and Systems*. 2020, IEEE, pp. 962–966.
20. Foster S, Gleirscher M and Calinescu R. Towards Deductive Verification of Control Algorithms for Autonomous Marine Vehicles. In *Proceedings of the IEEE International Conference on Engineering of Complex Computer Systems*. 2020, pp. 113–118.
21. Kapinski J, Deshmukh JV, Jin X et al. Simulation-Based Approaches for Verification of Embedded Systems. *IEEE Control Systems Magazine* 2016; 36(November).
22. Clarke Jr EM, Grumberg O, Kroening D et al. *Model checking*. MIT Press, 2018. ISBN 9780262038836.
23. Asarin E, Dang T, Frehse G et al. Recent progress in continuous and hybrid reachability analysis. In *Proceedings of the 2006 IEEE Conference on Computer Aided Control Systems Design, CACSD*. IEEE, pp. 1582–1587.
24. Duffy DA. *Principles of Automated Theorem Proving*. John Wiley & Sons, 1991. ISBN 0471927848.
25. Nuzzo P, Huan Xu, Ozay N et al. A Contract-Based Methodology for Aircraft Electric Power System Design. *IEEE Access* 2014; 2: 1–25.
26. Khalil HK. *Nonlinear Systems*. 2 ed. Prentice-Hall, 2002. ISBN 9780201190380.
27. Chen CT. *Linear System Theory and Design*. Third ed. Oxford University Press, 1999. ISBN 0195117778.
28. Goebel R, Sanfelice RG and Teel AR. Hybrid dynamical systems. *IEEE Control Systems Magazine* 2009; 29(2): 28–93.
29. Pnueli A. The temporal logic of programs. In *Proceedings - Annual IEEE Symposium on Foundations of Computer Science*. 1977, IEEE, pp. 46–57.
30. Alur R and Henzinger TA. Real-Time Logics: Complexity and Expressiveness. *Information and Computation* 1993; 104: 35–77.
31. Maler O and Nickovic D. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and*

Analysis of Timed and Fault-Tolerant Systems. Springer, pp. 152–166.
32. Deshmukh JV, Donzé A, Ghosh S et al. Robust online monitoring of signal temporal logic. *Formal Methods in System Design* 2017; 51(1): 5–30.
33. Hekmatnejad M, Yaghoubi S, Dokhanchi A et al. Encoding and monitoring responsibility sensitive safety rules for automated vehicles in signal temporal logic. In *17th ACM-IEEE International Conference on Formal Methods and Models for System Design*. 2019, ACM, pp. 1–11.
34. Wongpiromsarn T, Topcu U and Murray RM. Receding horizon temporal logic planning for dynamical systems. In *Proceedings of the IEEE Conference on Decision and Control*. 2009, pp. 5997–6004.
35. Raman V, Donzé A, Sadigh D et al. Reactive synthesis from signal temporal logic specifications. In *Proceedings of the 18th International Conference on Hybrid Systems: Computation and Control*. 2015, ACM, pp. 239–248.
36. Fainekos GE and Pappas GJ. Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science* 2009; 410(42): 4262–4291.
37. Varnai P and Dimarogonas DV. On Robustness Metrics for Learning STL Tasks. *Proceedings of the American Control Conference* 2020; : 5394–5399.
38. Rasmussen CE and Williams CKI. *Gaussian Processes for Machine Learning*. MIT Press, 2006. ISBN 026218253X.
39. McKay MD, Beckman RJ and Conover WJ. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* 2000; 42(1): 55–61.
40. Torben TR. AutoSimTest, 2021. URL https://github.com/tobiastorben/AutoSimTest.
41. Eriksen BOH, Breivik M, Wilthil EF et al. The branching-course model predictive control algorithm for maritime collision avoidance. *Journal of Field Robotics* 2019; 36(7): 1222–1249.
42. IMO. COLREGS - International Regulations for Preventing Collisions at Sea. *Convention on the International Regulations for Preventing Collisions at Sea, 1972* 1972; : 1–74.
43. Tam C and Bucknall R. Collision risk assessment for ships. *Journal of Marine Science and Technology* 2010; 15(3): 257–270.
44. Bakdi A, Glad IK, Vanem E et al. AIS-based multiple vessel collision and grounding risk identification based on adaptive safety domain. *Journal of Marine Science and Engineering* 2020; 8(1).
45. Fossen TI. *Handbook of Marine Craft Hydrodynamics and Motion Control*. Wiley, 2011. ISBN 9781119991496.
46. Pant YV, Abbas H and Mangharam R. Smooth operator: Control using the smooth robustness of temporal logic. In *1st Annual IEEE Conference on Control Technology and Applications*. 2017, IEEE, pp. 1235–1240.
47. Ferrère T, Nickovic D, Donzé A et al. Interface-aware signal temporal logic. In *Proceedings of the 2019 22nd ACM International Conference on Hybrid Systems: Computation and Control*. 2019, pp. 57–66.
48. Neal RM. *Bayesian Learning for Neural Networks*. Springer-Verlag, 1996. ISBN 0387947248.
49. DNV GL. Autonomous and remotely operated ships - class guideline. Technical Report September, DNV GL, 2018.
50. NMD. Føringer i forbindelse med bygging eller installering av automatisert funksjonalitet, med hensikt å kunne utføre ubemannet eller delvis ubemannet drift. Technical report, NMD, 2020.
51. Rokseth B, Haugen OI and Utne IB. Safety Verification for Autonomous Ships. *MATEC Web of Conferences* 2019; 273(ICSC-ESWC 2018): 02002.