Temporal mission planning for autonomous ships: Design and integration with guidance, navigation and control

M.A. Hinostroza[a], A.M. Lekkas[a,*]

[a]*Department of Engineering Cybernetics, Faculty of Information Technology and Electrical Engineering, Norwegian University of Science and Technology (NTNU), NO-7034, Trondheim, Norway*

ARTICLE INFO

Keywords:
Mission planning
Maritime autonomous surface ships
Temporal AI planning
Guidance navigation and control
High-level planning
Intelligent navigation

ABSTRACT

Mission planning constitutes an important feature of autonomy for Maritime Autonomous Surface Ships (MASS). Nevertheless, this research topic remains largely unexplored, as the majority of academic and industry projects primarily focus on developing low-level systems, such as control, collision avoidance, and situational awareness. The main contribution of this paper is to address this problem by developing a high-level decision-making system capable of generating an efficient and feasible temporal sequence of high-level actions, which is then sent to the ship control systems responsible for execution. The mission planner is based on the *simultaneous temporal planner* (STP), which in our case considers temporal actions related to, for example, moving to a specific location, activating docking mode or starting the process of container (un)loading, which are then executed by their respective control systems. Contrary to classical artificial intelligence (AI) planning algorithms, Temporal AI planning algorithms, such as STP, can consider duration of actions, which allows more realistic representation of the mission. We connect the high-level mission planner with the ship's guidance, navigation and control (GNC) system, which has path-planning, path-following control and fuzzy logic-based collision avoidance capabilities. The efficiency of our approach is demonstrated through a series of simulations of a MASS operating in a realistic marine environment including other ships and static obstacles.

1. Introduction

Following recent advances in communication and sensing technology, as well as an overall trend toward autonomous transportation, there has been an increased interest in the development of maritime autonomous surface ships (MASS) for academic and industry applications. Thus, several key results related to MASS for maritime transportation have been presented the last years, including: 1) Rolls-Royce Marine, now a part of Kongsberg Maritime, accomplished a successful test in 2018 with the "FALCO," the world's first fully autonomous ferry (Rolls-Royce, 2018), 2) Kongsberg Maritime has initiated trials aboard the "Yara Bikerland," the world's first electric autonomous cargo vessel (Konsberg-Maritime, 2021), and recently successfully demonstrates autonomous vessel operations on Belgium's inland waterway network (Konsberg-Maritime, 2023) 3) NTNU "milliAmpere" Autonomous passenger ferry (Brekke et al., 2022; NTNU, 2022), is a prototype ferry designed for urban transportation in Norway. The milliAmpere ferry was launched for testing with passengers in September 2022 in Trondheim, Norway. Other notable developments in the field of autonomous ferries can be found in Zeabuz (2023), which has been operating the world's first commercial autonomous passenger ferry since June 8, 2023, in Stockholm, Sweden, and in Enevoldsen et al. (2023). Additionally, Japan (Executive, 2022) and South-Korea (Splash, 2023) are taking important steps towards the era of unmanned ships. On the other hand, the International Maritime Organization (IMO) recently concluded its first regulatory scoping exercise on Maritime Autonomous Surface Ships (IMO, 2021), a set of rules designed to assess existing IMO instruments and determine how they can be applied to ships with varying degrees of automation.

The potential advantages of this new generation of MASS, i.e., enhanced cargo safety, optimized fuel consumption, reduced pollution and costs, and alleviation of traffic congestion in urban areas, has incentivized technology suppliers and researchers to develop robust and efficient algorithms to operate this intelligent maritime vehicles, for example, path-planning (Lekkas et al., 2016; Liu and Bucknall, 2016; Sans-Muntadas et al., 2019), path-following (Lekkas and Fossen, 2013), collision avoidance (Eriksen et al., 2020; Brekke et al., 2019), automatic docking (Bitar et al., 2021), navigation, integration of multi-agent platforms in marine operation (Ludvigsen and Sørensen, 2016) and control systems (Fossen, 2011). However, an important aspect of marine autonomy that has not yet been investigated is mission planning, with few exceptions such as in Thompson and Guihen (2019), where the application of high-level planning is suggested for the cooperative operation of MASS, or in Hinostroza and Lekkas (2022), where a rudimentary mission planning system for MASS was presented but lacked reactive capabilities and a realistic representation of tasks. Frequently, in the ship control literature, mission planning in reality pertains to dynamic path-planning, or waypoint selection, based on real-time information about criteria such as weather and energy consumption (Thompson and Galeazzi, 2020). Nevertheless, MASS missions are expected to require more complex decision-making processes, which

*Corresponding author
miguel.hinostroza@ntnu.no (M.A. Hinostroza); anastasios.lekkas@ntnu.no (A.M. Lekkas)
ORCID(s): 0000-0002-8505-7051 (M.A. Hinostroza); 0000-0001-6885-6372 (A.M. Lekkas)

include considerations such as fallbacks for extraordinary situations and switching operational modes, for example, transitioning from transit mode to docking mode and then to container (un)loading mode. Such planning is of a higher level and can be best expressed in different domains compared to traditional state-space control approaches. The field of AI planning provides algorithms capable of addressing this aspect of mission planning, and has made substantial contributions to advanced fields such as space exploration (Rabideau et al., 1999), Autonomous underwater vehicles (McGann et al., 2007) and Martian Rovers, i.e., Rovers used in Mars exploration were controlled by planning and scheduling software developed by NASA's Jet Propulsion Laboratory (Estlin et al., 2003).

AI planning, also known as automated planning and scheduling, is a sub-field of artificial intelligence which studies the deliberation and decision-making processes in a computational manner, making it applicable to robots and intelligent agents (Ghallab et al., 2004). In the classical AI planning formulation, with a pre-defined problem domain, initial states and a goal as inputs, the planner is capable of computing a plan, which is a sequence of actions serving as a solution for the problem domain.

The first AI planning algorithm, *STRIPS*, short for Stanford Research Institute Problem Solver, was developed in 1971 to solve a basic planning problem for a mobile ground robot named "Shaky" (Fikes and Nilsson, 1971). Since then, the field has made significant progress and has contributed to applications such as space exploration (Rabideau et al., 1999; Bernard et al., 1999), prevention of disasters (Nau et al., 1999), rescue operations (Currie and Tate, 1991) and operations with robots and autonomous systems (Pinto et al., 2012; Xue and Lekkas, 2020). In addition to STRIPS, other fundamental AI planning algorithms are Hierarchical task network (HTN) (Erol et al., 1994) and GraphPlan (Blum and Langford, 1999). The primary distinction among these classical methods lies in the space where the plan is searched. STRIPS and GraphPlan employ state-space search techniques, whereas HTN is a task-network based search.

The fields of Artificial Intelligence (AI) and Robotics were strongly connected in the early days of AI but have since diverged (Rajan and Saffiotti, 2017). Nowadays, on one side, we have highly optimized and complex AI algorithms applied in computer science, and on the other hand, we have robust and reliable robots, such as industrial robotic arms, capable of performing tasks even faster and better than humans. Recently, there is a renewed interest in bringing both fields together. The scientific goal of this presented work is to contribute to the integration of AI and robots, particularly in the context of maritime robots.

In recent years, with the new developments in microchips and, consequently, increasing of the computational power, a new generation of AI planners have been proposed, see Fig. 1, for example to include temporal constrains EUROPA (Barreiro et al., 2012) and system dynamics T-REX (McGann et al., 2007), to be connected to the Robotic Operation System ROSplan (Cashmore et al., 2015), a planner that solve problems that require concurrent actions CRIKEY (Coles et al., 2009), an unified planning and execution framework IDEA (Muscettola et al., 2002), a planner that allows the inclusion of complex system dynamics DiNo (Piotrowski et al., 2016), and a planner capable of generating plans with non-linear dynamics (Cashmore et al., 2020). However, many of these frameworks quickly become obsolete due to a lack of maintenance, resulting in a lack of updates and support. As a result, these techniques were overcome by more recent algorithms such as *simultaneous temporal planner* (STP) (Furelos Blanco et al., 2018). The STP planner depends on a process that transforms temporal planning into classical planning problem and builds a temporal plan by finding a sequence of classical actions that solve the problem while satisfying a given set of temporal constraints. These features make STP a faster and more robust algorithm that allows it to be implemented in real-time applications as demonstrated in Hinostroza et al. (2023), where a system for inspection and maintenance operations on an oil and gas facility using unmanned ground vehicles was presented. Another approach in AI planning is *conformant planning*, which refers to when the agent is uncertain about the current state of the system and is unable to make any observations. In more recent works, authors have proposed intelligent navigation methods to enhance the level of autonomy in Maritime Autonomous Surface Ships, with a focus on decision-making methods. For instance, in Cui et al. (2023), an intelligent planning and decision-making method for MASS based on Rapidly-exploring Random Trees star (RRT-star) and an improved Proximal Policy Optimization (PPO) algorithm was presented. These works show great promise; however, the process of building an artificial neural network and collecting data for its training could represent a limitation to an effective implementation in real-world maritime applications.

The main objective of this paper is to develop a high-level mission planning system for a MASS and connect it to a traditional GNC system in order to investigate how this fusion can enhance the current autonomoy capabilities of marine control systems. We chose a temporal planner since it allows to take action durations into account and tackles concurrency, allowing us to test a more realistic scenario. In this work, the STP planner was chosen over other well-known planners such as ROSplan, Europa, and T-rex because of its feasibility in being integrated into a GNC system. The STP planner is coded in a recent version of Python and Linux, making it possible to run simultaneously with a GNC system programmed in Matlab. This is not the case with the other planners, i.e., ROSplan is a black box that cannot be extended easily to add new blocks or extra code, and the Europa planner has been discontinued since 2012 and is not compatible with current versions of Linux and Python. The GNC system we consider is composed of a path-planning based on fast-marching (Hinostroza et al., 2020), line-of-sight (LOS) path-following (Hinostroza et al., 2017), and collision avoidance based on fuzzy logic (Hinostroza and Soares, 2018). To evaluate the performance of the proposed

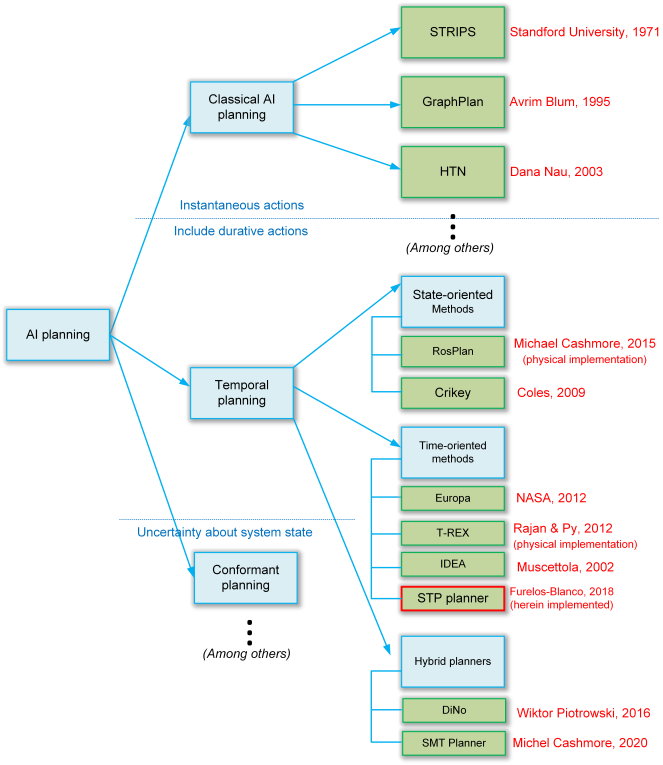

Figure 1: Types of AI planning algorithms.

system, numerical simulations were conducted for a realistic maritime environment, with static obstacles and other ships. The principal contributions of this paper can be outlined in the following list:

- We formulate a MASS mission as an AI planning problem, which allows to include a larger spectrum of potential actions compared to traditional control engineering schemes. The mission we consider in this paper has higher complexity than the one we presented in Hinostroza and Lekkas (2022), since it also includes reactive planning, durative action, better domain representation and a set of simulations.
- We implement STP, which is a state-of-the-art temporal planning algorithm, to generate a sequence of high-level actions for achieving the mission. STP allows to take action duration into account, as well as concurrency (actions taking place at the same time), which are clear advantages compared to the GraphPlan algorithm we used in Hinostroza and Lekkas (2022).
- We integrate STP with a MASS simulator, that includes guidance, navigation and control capabilities, and demonstrate the efficiency of the fusion via a mission in a relevant maritime environment.

To the best of the authors' knowledge, this is the first implementation where a temporal mission planning system has been designed to collaborate with the GNC system of a MASS. In addition, there have not been so far other works where the STP algorithm has been used in combination with a robotic maritime application. The ultimate goal of this paper is to emphasize the importance of incorporating high-level planning, which appears to be a natural advancement in current trends in autonomous ship technologies that can contribute to increasing the level of autonomy of MASS.

2. Background

2.1. Classical AI planning

In the classical AI planning formulation, when given a predefined planning problem, $P = (\Sigma, s_i, g)$, where s_i represents the initial state, Σ is the domain and g signifies the goal, the planner computes a plan, denoted as $\pi = (a_1, \ldots, a_k)$, which represents a solution for the problem domain if $\gamma(s_i, \pi)$ satisfies the goal g, and $\gamma(s_k, \pi) = \gamma(\gamma(s, a_1), (a_2, \ldots, a_k))$; if $k > 0$ and a_1 applicable in the state s, where γ denotes the intermediate state resulting from the application of action a_i, also called an "effect".

In a general overview, classical planning can be defined as illustrated in Fig. 2, where given an initial state, allowed actions, goals and description of the system, the high-level planner computes a plan which is subsequently transferred to the low-level controller module, where the plan is parsed and then sent to the physical layer where is executed. Subsequently, sensor readings are collected, providing feedback to the planner. Depending on these observations, the planner may decide to modify, adjust, or recalculate the plan.

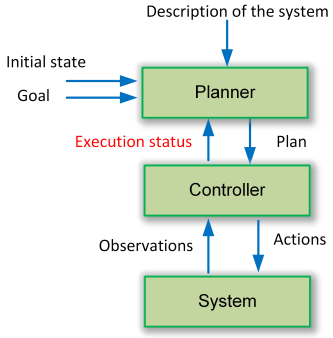

Figure 2: Representation of classical AI planning algorithm.

The majority of classical planners rely on the Planning Domain Definition Language (PDDL), which is a language used to represent the agents, allowed actions and goals in a format suitable for computational processing. PDDL is an action-centered language, inspired by the well-known STRIPS formulations of planning problems. At its core there is a simple standardization of the syntax for representing these familiar semantics of actions, using pre- and post-conditions to describe the applicability and outcomes of actions.

2.2. Temporal planning

An important aspect related to actions and their applications is the element of time. When a robotic system is acting based on a plan, the actions and effects take place over a period of time, and the possibility of taking actions may depend on events and other actions taking place simultaneously. Thus, in high-level temporal planning, the actions do necessarily follow each other sequentially, but may temporally interfere and overlap. The feasibility of carrying out an action may depend on whether some other actions have been taken first. In classical (non-temporal) planning, actions are executed sequentially, and the feasibility of taking an action remains unaffected by previous or subsequent actions, given a current state. Furthermore, in temporal planning, the effects of an action can be an intricate function of the current state and concurrent actions, whereas in classical planning, they remain unaffected by other actions. Also, temporal planners must cope with the fact that actions may start at any point in time. To address this, many temporal planners mitigate this challenge by constraining the initiation of actions to a limited set of decision epochs. This restriction facilitates state-space exploration and harnesses robust state-based reachability heuristics, which were initially designed for classical planning.

When time is introduced into the modelling of a domain it is possible for concurrent activity to occur in a plan. The PDDL 2.1 or PDDL+ is an extension to PDDL language for expressing temporal planning domains. It begins to bridge the gap between basic research and applications-oriented planning by providing the expressive power necessary to capture real problems. The PDDL2.1 language has the expressive power to represent a class of deterministic mixed discrete continuous domains as planning domains. The PDDL2.1 language introduces a form of durative action based on three connected parts: the initiation of an interval in which numeric change might occur and its explicit termination by means of an action that produces the state corresponding to the end of the durative interval. This form of action allows the modelling of both discrete and continuous behaviours. The language provides solutions to the critical issues of concurrency, continuous change and temporal extent. The semantics of the language are derived from the familiar state transition semantics of STRIPS, extended to interpret invariants holding over intervals in which continuous functions might also be active.

For instance, a typical temporal problem domain for a DriverLog problem (Coles et al., 2009) can be expressed in PDDL2.1 syntax as follows:

```
(define (domain Driverlog-L252-6)
  (:requirements :typing :durative-actions)
  (:types      location locatable - object
               driver truck obj - locatable)
  (:predicates
    (at ?obj - locatable ?loc - location)
        (in ?obj1 - obj ?obj - truck)
        (driving ?d - driver ?v - truck)
        (link ?x ?y - location)
        (path ?x ?y - location)
        (empty ?v - truck)
(:durative-action load-truck
  :parameters
   (?obj - obj ?truck - truck ?loc - location)
  :duration (= ?duration 2)
  :condition
   (and (over all (at ?truck ?loc))
   (at start (at ?obj ?loc)))
  :effect
  (and ( (not (at ?obj ?loc)))
  (at end (in ?obj ?truck))))
```

3. Temporal mission planning for MASS

In this section, the proposed high-level mission planning for MASS based on temporal AI planning is presented. The overall view of the system is presented in Fig. 3. The system is composed of a 3-Degree-of-freedom (DOF) ship simulator, guidance and control, path-following, collision avoidance and AI planning module. Each module of the system will be described in detail, the inputs, outputs and relation between blocks. The proposed formulation is based on a modular and hierarchical organization of the MASS's systems, starting from a low-level system (control and observer) until the upper-layer (AI-planner). Note that this architecture is not rigid, and slightly different architectures or additional layers, such as reactive collision avoidance, could be considered.

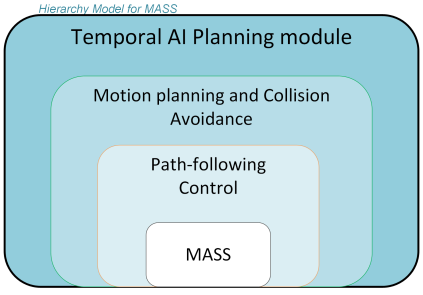

Figure 3: Proposed architecture for operation of MASS.

Fig. 4 illustrates the integration of the temporal AI planning module into the traditional GNC system. In this diagram, the AI planning module is positioned above the Motion planning module. The inputs to the AI planning module include the problem domain, actions, goals, and feedback from the motion planning block. The output of the AI planning module is the temporal plan, which is feed to the motion-planning. The guidance and control blocks are responsible for executing the tasks. This system is an extension of the work presented in Hinostroza and Lekkas (2022).

3.1. Modelling, guidance, navigation and control of MASS

This subsection introduces the modelling, guidance, navigation and control modules used in this work. The MASS

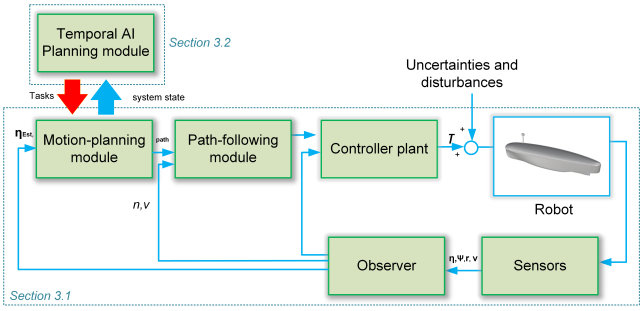

Figure 4: Temporal AI planning into a traditional GNC system.

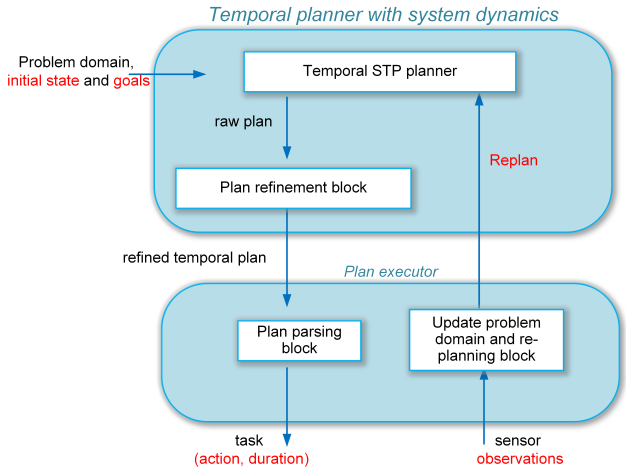

Figure 5: High-level mission planner based on AI planning.

dynamics are formulated as presented in Fossen (2011) and the GNC system is based on the work presented in Hinostroza et al. (2020).

The Guidance navigation and control system is composed of path-planning, path-following, collision avoidance and control modules. The path-planning is based on the *fast-marching* method, known for its ability to compute smooth and fast paths for the MASS (Hinostroza et al., 2020). The path-following algorithm relies on the Line-of-sight (LOS) algorithm which has consistently shown robust performance in tracking waypoints. The COLAV, collision avoidance module, is implemented using a fuzzy logic algorithm (Hinostroza and Soares, 2018). It is important to mention that the GNC system implemented in this work includes a non-linear mathematical model for the ship dynamics obtained from system identification of free-running model tests (Xu et al., 2018). Furthermore, this GNC system is able to deal with environmental disturbances (Hinostroza et al., 2021).

Extensive numerical and experimental results of the GNC (Guidance, Navigation, and Control) system employed in this study have been previously presented by the authors. These formulations are not presented here to avoid diverting attention from the novelties of the proposed system and to prevent the paper from becoming excessively long. Thus, numerical simulations and experimental results for the path-planning algorithm can be found in Hinostroza et al. (2020); Lekkas et al. (2016), collision avoidance simulations in Hinostroza and Soares (2018); Eriksen et al. (2020), the path-following algorithm in Hinostroza et al. (2017); Lekkas and Fossen (2014), and the control system in Lekkas and Fossen (2013).

3.2. Temporal AI planning module

The proposed mission planning system architecture is presented in Fig.5. This figure illustrates the combination of high-level temporal planning and traditional low-level GNC. The mission planner module is composed of 4 sub-blocks, which will be described in detail in the next subsection: (1) temporal STP planner, (2) Plan refinement, (3) Plan parsing, and (4) Update problem domain and re-planning.

3.2.1. *Temporal STP planner*

The proposed high-level mission planning system is based on the *simultaneous temporal planner* (STP) (Furelos Blanco et al., 2018). This algorithm was the runner-up algorithm in the 2018 International Planning Competition (IPC). It relies on a transformation process that converts temporal planning into classical planning problem, and constructs a temporal plan by identifying a sequence of classical actions which not only resolve the problem but also adhere to a specified set of temporal constraints. The main novelty of STP, as it was presented in Furelos Blanco et al. (2018), is its capability to address problems that necessitate simultaneous events, for example, the temporal actions have to be organized in way that allows two or more of their effects occur concurrently. To facilitate this, STP breaks down each event into three distinct phases: the first phase in which temporal actions are scheduled to the end, the second phase handles the occurrence of simultaneous effects, and the third phase focuses on temporal actions are scheduled to the start. Thus, a temporal planning problem is a tuple:

$$P =< F, A, I, G > \tag{1}$$

where the fluent set F, initial state I and goal condition G are defined as for classical planning. The action set A consists of temporal or durative actions $a \in A$ composed of:

$$\begin{cases} \text{d(a):} & \text{duration of } a \\ \text{pre(a):} & \text{preconditions of } a \\ \text{add(a):} & \text{add effects} \\ \text{del(a):} & \text{delete effects} \end{cases} \tag{2}$$

Although a has a duration, its effects apply instantaneously at the start and end of a, respectively. The preconditions $pre(a)$ is also checked instantaneously.

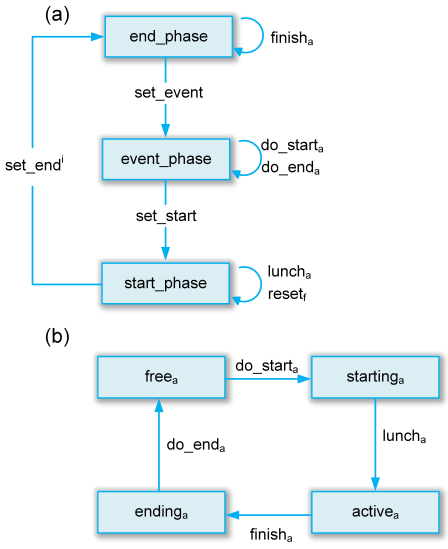

Figure 6: STP planner algorithm.

Note that STP applies a modified version of the Fast Downward (FD) planning system (Helmert, 2006) to generate temporal plans. The modified version of FD incorporates simple temporal networks to represent temporal constraints. During the search process, a branch is pruned if the temporal constraints are violated. The STP imposes a bound K on the number of active temporal actions, that started but did not end yet. Hence no more than K temporal actions can be executed concurrently. STP works by protecting the contexts of temporal actions in case a native execution of events using classical planning would produce inconsistent results. Thus, the compilation divides each concurrent event into three phases:

- End phase (immediately before the event). This is where active actions are scheduled to end, and in doing so, the corresponding counters of fluent in context are decreased.
- Event phase (concurrent event itself). This is where simultaneous events take place, both ending and starting actions. Here we check preconditions and apply effects and verify that the concurrent event is valid.
- Start phase (immediately after the event). Here we check that the contexts of active actions that just started are satisfied (possibly as a result of being added during the concurrent event itself) and increment the corresponding counters of fluent in context.

Fig. 6a shows the interconnection between the phases and actions. The actions $set-event$, $set-start$ and $set-end$ change the current phase, whereas the other actions can only be applied (if their preconditions hold) in the corresponding phase. Actions $do-start$ and $do-end$ correspond to the semantic events. Execution begins in the end-phase and ends in the $start-phase$.

Fig. 6b shows the cycle each action a. Between $do-start_a$ and $do-end_a$, actions $launch_a$ and $finish_a$ execute the start phase and end phase, respectively. Each time we transition from one state to another, we delete the auxiliary fluent of the state, and add the next, thus obtaining a mutex invariant.

Comprehensive details on the temporal logic equations, lemmas, and proofs for the STP planner are available in Furelos Blanco et al. (2018).

3.2.2. *Plan refinement block*

The initial plan, calculated by the AI planning module, is based on preliminary data, data from Kalman filter estimations, AIS information (ship position), weather routing system, etc. The plan refinement block is responsible for revising action by action if the plan is feasible. Thus, this module analyzes each action of the plan in combination with the current status and system dynamics to determine if the action is feasible to be executed or needs to be refined. The refinement process is done iteratively until a convergence value is reached.

3.2.3. *Plan parsing block*

The contribution of the present work is the integration of AI planning into a traditional maritime GNC system. The overall system must operate in real-time with the ship sensors and navigation system. The output of the temporal plan module is a plan. The plan parsing block organized this plan into a sequence of actions with a determinate duration.

3.2.4. *Update problem domain and re-planning block*

The update problem domain and re-planning block is responsible for reading the feedback from the system responses and verifying if the plan needs to be modified, changed or replanned. This module is responsible for recomputing a plan in case of a new event. The event could be introduced by an operator or by an unexpected situation or error in sensors or equipment. So, in case of one of these events the planner is responsible to re compute a new plan, having as an input the current status of the mission, the remaining goals and the actual system dynamics from Kalman filter.

3.3. Algorithm flowchart of the whole system

The algorithm of the mission planning system for MASS based on temporal AI planning is presented in Fig. 7. In this flowchart, the algorithm starts reading information about the maritime domain, actions, initial states and goals of the overall mission. Then, the temporal AI planning module computes an initial temporal plan. This temporal is refined by the block of plan refinement. This new temporal plan is parsed into a simple sequence of task and duration. These individual tasks are translated to the GNC system of MASS to be executed by the low-level control system. When a task is finished the algorithm updates the actual state of the system, problem domain, remaining goals, etc. in the system identification module. Then in case of any failure or unexpected event the plan is revised and, if necessary,

a re-planning process is initiated. Note that, in case of a re-planning, the updated problem domain and remaining goals are transferred to the temporal AI planner and the algorithm is restarted. Additionally, it is important to notice that the tasks mentioned here, such as moving between two locations, involve a set of sub-procedures that include algorithms and equations for low-level control, such as motion, planning, guidance, and control, to achieve these actions. The details of these sub-systems can be found in Hinostroza et al. (2020).

3.4. Strengths and limitations of the system

The presented work aims to develop a high-level mission planning system for maritime autonomous surface ships and integrate it with a traditional Guidance, Navigation, and Control (GNC) system to study how this fusion can enhance the autonomy of MASS. As the topic is relatively new in literature, there are some points that require further research and improvement. The principal strengths (✓) and limitations (✗) of the paper are listed below:

✓ The implementation of a mission planning system based on STP allows the MASS to perform more complex missions.

✓ The algorithm is efficient, demonstrating a low computational time.

✓ The system allows for the inclusion of temporal actions.

✓ The proposed algorithm can be easily integrated into a classical guidance, navigation, and control system.

✓ This work represents the first implementation of a temporal mission planning system specifically designed for MASS.

✗ Experimental validation is required to demonstrate the concept.

✗ More scenarios need to be studied, especially those involving re-planning.

✗ Pre-processing is required to build the input PDDL domain file.

✗ The temporal planner does not allow the inclusion of complex system dynamics in the PDDL domain formulation.

In summary, the main strength of the study lies in its novelty and the potential benefits of applying it to autonomous ships. The implementation of temporal planners can increase the autonomy of MASS. At the current stage of research, the primary limitation of the study is the absence of presented experimental results; however, the numerical simulations have shown good performance. Another limitation of the proposed system is the impossibility of including dynamic equations into the PDLL domain of the STP planner. To solve this type of problem, a hybrid planner is required (Piotrowski et al., 2016).

4. Numerical simulations

In order to assess the performance of the overall system in a maritime scenario, a case-study for a real maritime domain is presented. The physical location for this study is a Norwegian fjord, and it considers three dynamics obstacles (ships) moving along predefined trajectories. The tasks for the ships include moving, loading and unloading containers and performing repairs at five different ports. The MASS used in simulations is based on the model presented in Hinostroza et al. (2021).

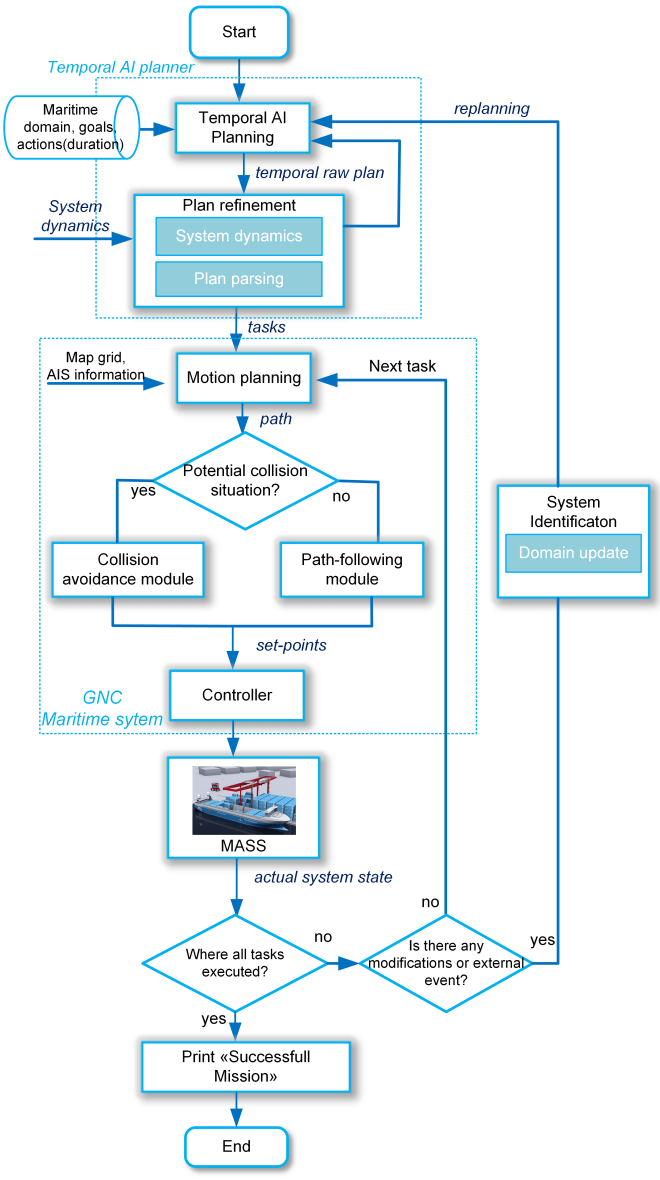

Figure 7: Diagram flux of the overall system.

4.1. Place for simulations: Trondheim fjords

The place chosen for simulations is the entrance of the Trondheim Fjord in Norway, Fig. 8. The dimension of the map is 2.3 x 2.0 Km^2.

Figure 8: Top view of place for simulations (Courtesy of Google Maps).

In order to perform numerical simulations, the physical map was discretized into a grid map, Fig. 9, where 1 [pixel] is equal to 200 m. This figure also shows the output of the fast-marching method, where blue colors are the static obstacles, and the yellow nodes are free space.

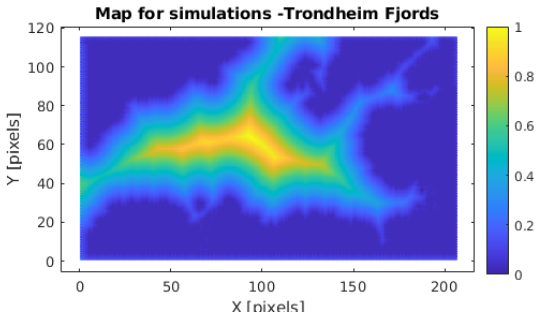

Figure 9: Grid map from FMM algorithm.

In order to introduce more realistic characteristics for numerical simulations and evaluate the reactive capabilities of the proposed system, we simulated a maritime traffic involving three vessels. A Passenger Ferry 1, which operates on the route between Trondheim and Vannik, a container ship 2 which loads and unloads containers between ports "B" and "E", and a leisure yacht which is sailing in the fjord. The ship's trajectories are plotted in Fig. 10. These trajectories were obtained from an online website for maritime AIS information (https://www.marinetraffic.com), the ferry 1 is the "Lagatun", IMO 9820398, the second ship is the "Trondheimsfjord II" a high-speed craft, IMO 9432189 and the Yacht is a small leisure craft.

4.2. Vehicle: Aurora Autonomous surface vehicle

The autonomous surface vehicle chosen for simulations is a scaled-down chemical tanker constructed in single skin of glass-reinforced polyester and plywood framing, Fig. 11. Detailed information about the scale of hydrodynamic coefficients, dynamics, kinematics and controller parameters

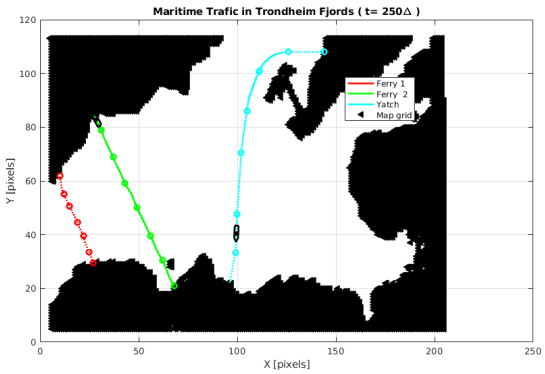

Figure 10: Maritime traffic simulation.

of the model for numerical simulations can be found in Hinostroza et al. (2021).

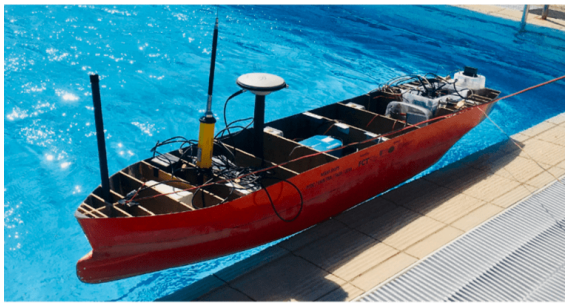

Figure 11: Model for simulations: "Aurora" ASV.

4.3. Problem Domain definition

This subsection presents the problem domain definition for the case study is presented. The locations, actions and agents are presented in Fig. 12. The defined tasks mainly consist of five: moving the ship, loading and unloading containers, performing repairs, and refueling the ship. We have considered five ports: A, B, C, D, and E. The (un)load, reparations and refueling tasks are actions with a fixed time, however, the action "Moving of the ship" from one port to another has variable duration. It is because this task is subjected to uncertainties. Due to complexity of the system, the ship fuel consumption, which depends on ship resistance, efficiency of the engine and proper modelling of the ship engine, was not included in this study and will be addressed in future work. For now, it is assumed that the ship has enough fuel to execute all the desired tasks and only needs to refuel by the end, in port E (last port).

4.3.1. Agents

The agents are shown in Table 1. One ship, four containers, five ports, a reparation team and one refueling station are considered. Additionally, the status of the ship is defined

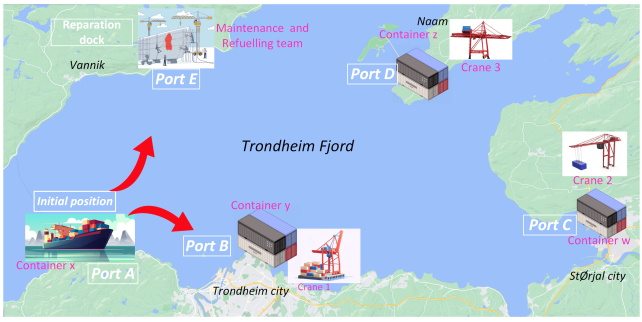

Figure 12: Problem domain definition for a marine environment.

Table 1
STP planner domain variables.

Agent	Variable	Notation
ship	ship0	?V-vehicle
container	cont-x, cont-y, cont-w, cont-z	?P-cont
port	port A, port B, port C, port D, port E	?O-location
ship loading status	busy, available	predicates
ship reparation status	repaired, no rep, teamR0	predicates
ship fuel status	fuel, no-fuel, teamF0	predicates

Table 2
Initial state and goal for simulation case A and B.

Case	Initial state		Goal
A (No simulated event)	(at ship0 port A)	→	(at ship0 port E)
	(at cont x port A)	→	(at cont x port B)
	(at cont y port B)	→	(at cont y port C)
	(at cont w port C)	→	(at cont w port d)
	(at cont z port D)	→	(loaded cont z ship0)
	(no rep ship0)	→	(repaired ship0)
	(no fuel ship0)	→	(refuel ship0)
	(available ship0)	→	(busy ship0)
	(at teamR0 port E)		-
	(at teamF0 port E)		-
	((speed ship0) 0.18)		-

Table 3
Computed plan - simulation case A.

n.	Time(s)	Action	Dur.(s)
1:	0.0	load-cont port-a cont-x	60.0
2:	60.2	move-ship port-a port-b	288.8
3:	348.8	unload-cont cont-x port-b	50.0
4:	398.8	load-cont port-b cont-y	60.0
5:	458.8	move-ship port-b port-c	566.6
6:	1025.5	unload-cont cont-y port-c	50.0
7:	1075.5	load-cont port-c cont-w	60.0
8:	1135.5	move-ship port-c port-d	594.4
9:	1730.0	unload-cont cont-w port-d	50.0
10:	1780.0	load-cont port-d cont-z	60.0
11:	1840.0	move-ship port-d port-e	483.3
12:	2323.3	repair-ship port-e teamr0	60.0
13:	2323.3	refuel-ship port-e teamf0	80.0

as "loaded", "available", "repaired" and "fueled". An important parameter in the Planning domain is the link between locations, in the case of open sea, all the connections are assumed.

4.3.2. Actions

The allowed actions for the MASS sailing in a marine domain are presented in Table 5. The planning domain includes five actions: moving the ship (*durative-action-move-ship*), loading (*durative-action-load-cont*) and unloading containers (*durative-action-unload-cont*), performing maintenance (*durative-action-repair-ship*), and refueling the ship (*durative-refuel-ship*), along with their associated preconditions and effects.

To evaluate the performance of the proposed temporal STP planner for MASS operating in a maritime domain, three cases, A, B and C, were considered for simulations, as shown in Table 2. The case A is a simple scenery where the MASS has arrived at the Fjord in *portA* and need to perform a maintenance, refuel and (un)load the containers *x, y, z* and *w* in ports *B*, *C*, *D* and *E* and departure from Trondheim fjord. The case B is a bit more complex scenery, where an unexpected event is simulated to test the reactive capabilities of the STP planner. The case C presents a re-planning scenario. For this scenario, additional assumptions were made in order to create a re-planning situation. Thus, the ship fuel tank percentage serves as a trigger for re-planning, and each action consumes a certain percentage of the fuel tank.

4.4. Numerical results case A

In this first numerical simulation, case A, we present a simple scenario in which the MASS needs to perform a calculation of a plan, plan refinement, and then execute the plan. The goal of the mission consists of moving containers from ports and performing maintenance and refueling. The plan calculated for simulation case A is presented in Table 3.

In order to demonstrate the ability of the system to execute the computed plan presented in Table 3, numerical simulations based on the simulator for the Aurora MASS were carried out. Fig. 14 presents the satellite view of the overall mission execution and Fig. 15 presents a black and white map plot for Case A. From these plots, the MASS trajectory is marked in magenta, the ferry in red, the high-speed craft in green, and the leisure yacht in cyan. Waypoints for each vessel are represented as small circles. It is important to mention that these results must be interpreted together with the events that occur during the mission execution, addressed in Table 4. The simulation begins by executing Action 1: loading the container *x* in port A, followed by Action 2: moving the ship between ports A and B, and so on, action by action. Action 5 is to move the ship between C and D, Action

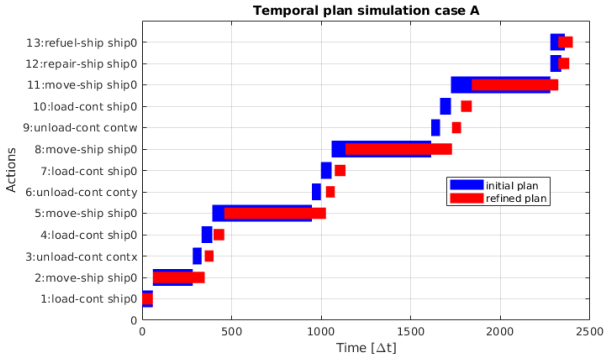

Figure 13: Temporal plan - case A.

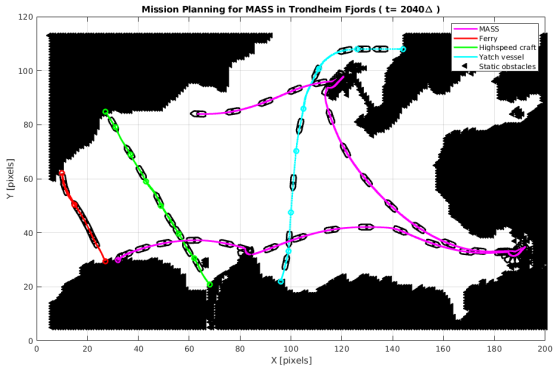

Figure 15: Mission execution - map view case A.

8 is to move the ship between ports C and D, and Action 11 is to move the ship between D and E. From these plots, it is evident that the system demonstrates strong capabilities in executing a simple mission in a maritime environment. The algorithm successfully computed and executed smooth trajectories for the MASS.

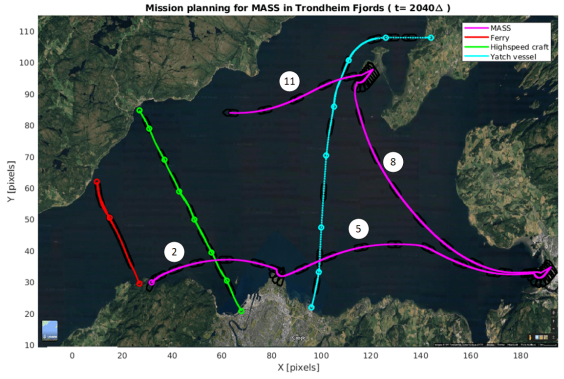

Figure 14: Mission execution - satellite view case A.

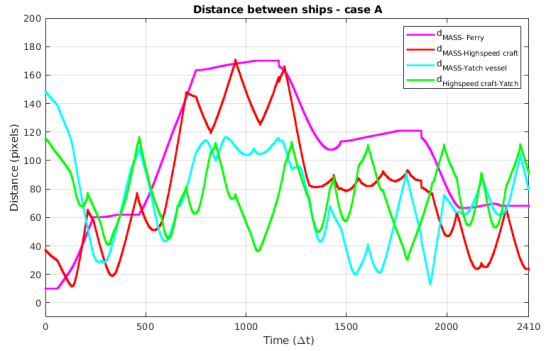

Figure 16: Distance between ships during simulation case A.

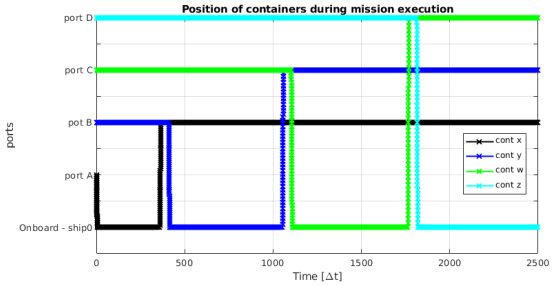

Figure 17: Position of containers during mission execution - Case A.

Fig. 16 presents the time/series plot of distances between the MASS and the ferry, yatch and high-speed craft during the mission execution.

Fig. 17 shows the time-series plot of the container's position during the mission execution. From this plot is possible to identify three different positions of each container, port of origin, on-board of the MASS and in the destination port.

As complementary information regarding the mission execution, Table 4 presents the navigation logs for simulation case A. In this table, all relevant events and their timestamps are presented. From this table, it is possible to observe the events and the timestamps when the ship starts its motion, as well as when containers are loaded and unloaded. From this table, it can be seen that a potential collision situation was detected at *time=1885 [s]* between the MASS and the yacht; however, no avoidance manoeuvre is shown in Fig. 15. This is primarily due to the short duration of the potential collision situation and the value considered for the safety navigation region in the collision avoidance system. These parameters were modified in Simulation Case C and more collision situations were observed.

4.5. Numerical results case B

In this second numerical simulation case B, a more complex scenery case was simulated. Thus, an unexpected event (Table 5) was introduced. The event consists of deviating the ship's trajectory from position 1 to position 2 at certain time stamp, an error of the GNSS receiver. The objective of this

Table 4
Navigation Log - Simulation case A.

Ship navigation logs	
Event	Starting time(s)
starting MASS computational simulation	time:0.000
reading inputs,	time: 0.002
computing the temporal plan,	time: 0.004
refining the plan,	time: 6.5406
parsing the plan,	time: 14.0808
load task started	time: 14.1808
verifying task load	time: 60
verifying task move	time: 360
unLoad task started	time: 360.002
executing move started	time: 470
verifying task move	time: 1.053e+03
potential collision situation	time:1.885e+03
task move failed, need re-planning	time: 2240
replan and refined done	time: 2.246e+03
repair task started	time: 2.267e+03
refuel task started	time: 2.327e+03
end of computational simulation	time:2.407e+03

Table 5
Simulated event.

Initial time	100 [s]
Final time	170 [s]
Ship0 initial position	(38, 35)
Ship 0 final position	(30, 50)

Table 6
Computed plan - simulation B.

n.	Time(s)	Action	Dur.(s)
1:	0.0	load-cont port-a cont-x	60.0
2(a):	60.2	move-ship port-a port-b	288.8
2(b):	349.0	move-ship port-s port-b	66.6
3:	415.6	unload-cont cont-x port-b	50.0
4:	465.6	load-cont port-b cont-y	60.0
5:	525.6	move-ship port-b port-c	566.6
6:	1092.3	unload-cont cont-y port-c	50.0
7:	1142.3	load-cont port-c cont-w	60.0
8:	1202.3	move-ship port-c port-d	594.4
9:	1796.7	unload-cont cont-w port-d	50.0
10:	1846.7	load-cont port-d cont-z	60.0
11:	1906.7	move-ship port-d port-e	483.3
12:	2390.1	repair-ship port-e teamr0	60.0
13:	2390.1	refuel-ship port-e teamf0	80.0

event is to force the system to perform reactive planning, and re-calculate a plan to continue with the mission and achieve the desired mission goals.

Table 6 presents the initial and re-planning plans computed for the numerical simulation case B. In Fig. 18, the initial plan consists of 13 actions, while the second plan comprises 12 actions. This is due to an unexpected event during the execution of Action 2 (moving the ship from port A to B). Consequently, the mission planning system initiated re-planning, resulting in the calculation of a new plan to achieve the remaining goals.

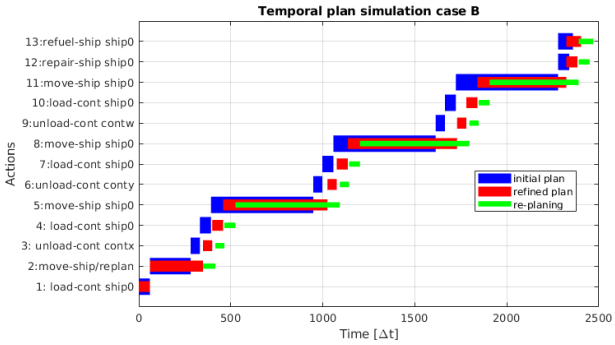

Figure 18: Temporal plan computed for case B.

Similar to simulation case A, the plan computed for case B was executed using the MASS simulator. Fig. 19 presents the mission execution of the plan computed for case B, and Fig. 20 presents a black and white map plot, where is possible to see the trajectories of the three ships for case B. Note that for a better interpretation of these plots, the table of events (Table 7) must be read together with these plots. The trajectory of the MASS is plotted in magenta, the ferry in red, the high-speed craft in green and the leisure yacht in cyan. The way-points of the vessels are plotted as small circles. These figures also plot the simulated unexpected event at time t=99s, see table Navigation logs, where the position of the ship was moved to a position in the centre of the map. In these figures, the symbol (2a) corresponds to the initial action of moving the MASS between port A and port B, (2b) represents the new action computed after the simulated event, (5) corresponds to the action of moving the ship between port B and C, (8) represents the action of moving the ship between C and D, and (11) represents the action of moving the ship between D and E. Additionally, it can be observed that the system is capable of computing and executing safe paths for the MASS. In this plot in particular, there was no possible collision situation, as can be verified in the navigation logs presented in Table 7.

Table 7 presents the navigation logs for simulation case B. In this table all the relevant events and their time during the mission executions are presented. In this table is possible to see a simulated event starting at *time= 99 [s]*, when the unexpected event was introduced, the ship was moved from a pre-defined point to another. The system was able to re-calculate its plan and continue with the mission execution.

4.6. Numerical results case C

In order to assess the performance of the proposed system in a more complex re-planning scenario, simulation case C was computed. For this new scenario, additional variables, actions, and the ship fuel consumption will be considered as triggers for re-planning. To simulate this new scenario, the following considerations were made:

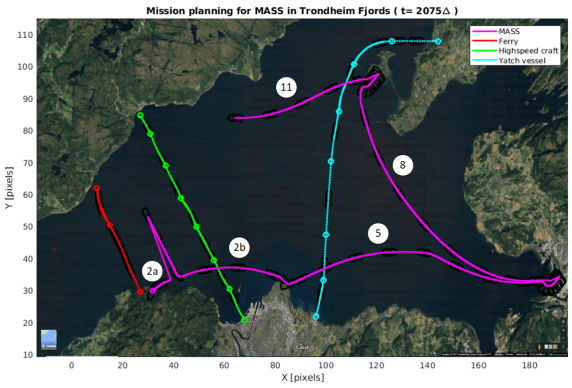

Figure 19: Mission execution - satellite view case B.

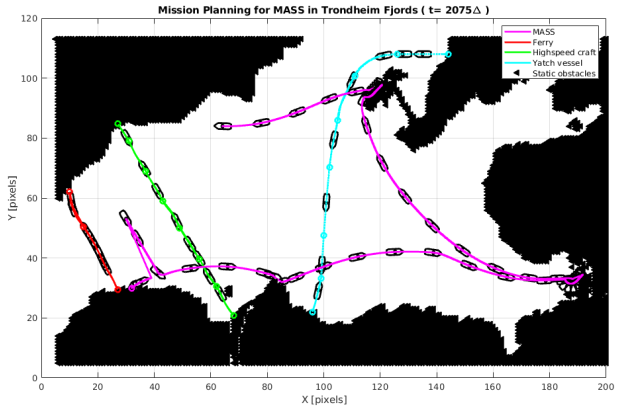

Figure 20: Mission execution - map view case B.

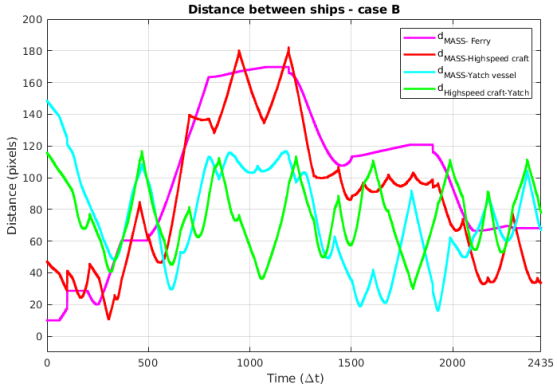

Figure 21: Distance between ships during simulation case B.

- The ship fuel consumption is now modeled as a planning variable. Thus, when there is low fuel, the system will require performing a re-planning.

Table 7
Navigation log - simulation case B.

Ship Navigation logs	
Event	Starting time(s)
starting MASS computational simulation	time:0.000
reading inputs,	time: 0.002
computing the temporal plan,	time: 0.004
refining the plan,	time: 6.5406
parsing the plan,	time: 14.0808
load task started	time: 14.1808
verifying task load	time: 60.004
simulated event at t=[99,169]	time: 99.002
verifying task move	time: 360.002
task move failed, needs re-planning	time: 360.002
re-planning finished	time: 362.004
verifying task move	time: 387.902
load task started	time: 437.904
verifying task load	time: 1.191e+03
executing move started	time: 1.901e+03
repair task started	time: 2.295e+03
refuel task started	time: 2.355e+03
end of computational simulation	time: 2.435e+03

- Two charging stations are defined at locations *Port B* and *D*. These stations will be important for refueling the ship and continuing mission execution.
- A new action is defined: *move ship to refuel*. This action will be executed when the ship is running out of fuel and has a precondition of a low fuel tank level.
- All ship actions now consume a certain amount of fuel. The action *move ship* between ports consumes 15% of the fuel, *(un)loading container* consumes 10%, and *repair ship* consumes 10%.
- A new location, *port S*, is defined to increase the problem's complexity and create more potential collision situations with other ships.

To simulate a re-planning scenario for this Simulation Case C, the PDDL domain needs to be modified slightly to include the constraint of having enough fuel to execute the tasks. Thus, now a condition for ship fuel has been added to the actions: *move ship*, *(un)load container*, and *repair Ship*. The new PDDL domain can be seen in Annex A, this new PDDL domain includes a new pre-condition: *(at-start (fuel ship))*, which requires having enough power to execute the tasks.

The variables and agents for Simulation C are also slightly different from those in Simulations A and B, see Fig. 12. Thus, Table 8 presents the variables for re-planning scenario. In this new domain the charging stations are located at *Port B* and *Port D*, the initial position of the MASS is at *port S* (entrance of the Trondheim Fjord) and a new variable for ship fuel is defined. In summary, one ship, four containers, six ports, a reparation team, and two refueling stations are defined.

Addressing the ship's fuel level in the proposed mission planning system can be done in two ways: (1) Including

Table 8
STP planner variables for re-planning scenario case C.

Agent	Variable	Notation
ship	ship0	?V-vehicle
container	cont-x, cont-y,cont-w, cont-z	?P-cont
port	port A, port B, port C, port D, port E	?O-location
charger station	charger B, charger D	?L-charger
ship fuel tank	fuel-ship	-
location	port S (ship at entrance fjord)	?N-location
ship loading status	busy, available	predicates
ship reparation status	repaired, no rep, teamR0	predicates
ship fuel status	fuel, no-fuel, teamF0, teamF1	predicates

Table 9
Initial state and goal for simulation case C.

Case	Initial state		Goal
C	(at ship0 port S)	→	(at ship0 port E)
	(at cont x port B)	→	(at cont x port C)
	(at cont y port A)	→	(at cont y port B)
	(at cont w port C)	→	(at cont w port D)
	(at cont z port D)	→	(loaded cont z ship0)
	(no rep ship0)	→	(repaired ship0)
	(no fuel ship0)	→	(fuel ship0)
	(available ship0)	→	(busy ship0)
	(at teamR0 port E)		-
	(at teamF0 port B)		-
	(at teamF1 port D)		-
	((speed ship0) 0.18)		-

Table 10
Computed plan - simulation C.

Action	Time(s)	Action	Dur.(s)	fuel (%)
1(a)	0.0	move-fueling-ship port-s port-b	472	-
2(a)	472.2	refuel-ship port-b teamf0	80	100
3(a)	552.2	move-ship port-b port-a	294	85
4(a)	846.6	load-cont port-a cont-y	60	75
5(a)	906.6	move-ship port-a port-b	300	60
6(a)	1206.6	unload-cont cont-y port-b	50	50
7(a)	1256.6	load-cont port-b cont-x	60	40
8(a)	1316.6	move-ship port-b port-c	583	25
9(b)	1900.2	move-fueling-ship port-c port-d	600	-
10(b)	2500.2	refuel-ship port-d teamf1	80	100
11(b)	2580.2	move-ship port-d port-c	594	85
12(b)	3174.4	unload-cont cont-x port-c	50	75
13(b)	3224.2	load-cont port-c cont-w	60	65
14(b)	3284.2	move-ship port-c c port-d	600	50
15(b)	3884.2	unload-cont cont-w port-d	50	40
16(b)	3934.6	load-cont port-d cont-z	60	30
17(c)	3994.8	refuel-ship port-d teamf1	80	100
18(c)	4074.8	move-ship port-d port-e	366	85
19(c)	4440.2	repair-ship port-e teamr0	60	75

the fuel consumption equations in the AI planner layer and compute the plan using a Hybrid Planner (Piotrowski et al., 2016). This approach creates a new variable in PDDL domain and formulate the fuel consumption equation as a precondition for each allowed action of the ship. This approach allows the computation of the entire plan of the system at once; however, it can be computationally heavy due to the possibility of creating several new nodes that will need to be explored to find the optimal plan. (2) Including the fuel consumption equation in the low-level system, specifically in the plan executor, see Fig. 5, as was done in Hinostroza et al. (2023). This second approach defines the ship's fuel consumption equations in the low-level system, and the value of the variable is updated every time an action is executed by the system. This second approach computes partial plans that need to be recomputed every time the ship's fuel is low and will generate the complete plan when the full mission execution is finished. In this numerical simulation, Case C, we are adopting the second approach due to computational performance, and the STP planner does not allow equations in its PDDL domain definition. The ship fuel threshold for re-planning is defined as $ship\text{-}fuel \leq 30\%$. This value is assumed arbitrarily to ensure a backup of power for executing heading to the refuel charging station.

To assess the performance of the proposed temporal mission planning for MASS operating in a maritime domain with re-planning, the mission planning problem presented in Table 9 is simulated. In this problem, the initial conditions are: initial position of MASS at the entrance of the Trondheim Fjord, at *port S*, the location of containers *x, y, z*, and *w* in ports *B,A,C* and *D*, respectively. The goal is to move the ship to port *E* and the containers to a different port, all of it subject to the ship's fuel consumption limitation. The initial value for the ship's fuel is set to $ship\text{-}fuel = 30\%$.

Based on the variables presented in Table 8, initial conditions and goals presented in Table 9, and the PDDL domain presented in Annex 5. The mission planning problem was solved and the results are presented in Table 10 and Fig. 22. In these charts the initial: $plan(a)$, is composed by 15 actions and plotted in color blue, the second plan: $plan(b)$ is composed by 10 actions and plotted in color red, and finally the last plan: $plan(c)$, is composed by 3 actions and plotted in color green. The re-planning is triggered by a low fuel of the ship, thus in Table 10 the first action, *1a* correspond to sending the ship to refuel and it is possible to see that *2a* is to refuel the ship and the actual value of fuel is 100%, then, action from *3a* to *8a* are executed normally, until the fuel level is low again, fuel level 25%, then a re-planning is triggered and the second plan is computed, composed by actions *9b* to *16b*. Finally, when the ship's fuel level drops below 30%, the planner computes the last three actions to execute the remaining goals, as specified in plan (c). This final plan consists of only three actions, namely *17c* to *19c*. Note that the initial plan has a total time duration of 3200 seconds; however, after two re-plannings, the duration of the complete plan is extended to 4501 seconds.

As mentioned earlier, the goal of the present work is not only to compute temporal plans for MASS but also to study

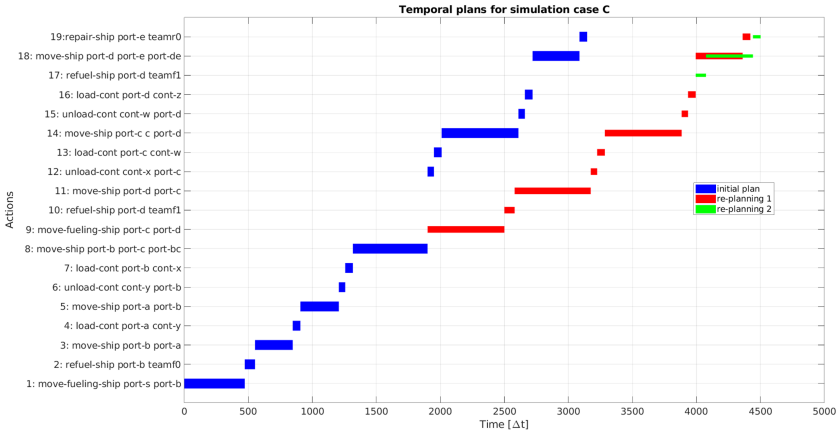

Figure 22: Temporal plan computed for case C.

how to integrate them with the ship GNC system and execute the tasks successfully. Following this objective, the plan was executed for the MASS simulator, and the results of the mission execution are presented in Fig. 23 and Table 11. During the execution of the mission for the MASS, the program reports the actions that are being executed and events during mission execution. Table 11 presents the navigation logs for execution of the mission for simulation case C. In this table all the relevant events that occurs during the mission execution are reported. Fig. 23 presents the time sequence plots of the mission execution for the plan computed in case C. It shows the trajectories of the ships, grid map, potential collision situations, and labels corresponding to the actions in the plan. The trajectory of the MASS is plotted in magenta, the ferry in red, the high-speed craft in green and the leisure yacht in cyan. The way-points of the vessels are plotted as small circles. Fig. 23a displays the initial conditions for simulations, where the position of the MASS is defined at the entrance of the Fjord, and the positions of the target ships are also specified. Fig. 23b displays the plots at *time=700 [s]*. In this plot, it is possible to see the first collision situation between the MASS and Ferry1. The potential collision situation I occurs at *time=49.1 [s]*, and a collision avoidance manoeuvre is performed by the MASS following the algorithm presented in Hinostroza and Soares (2018). This plot also illustrates the execution of actions *1a: move-fueling-ship port-s port-b* and *3a: move-ship port-b port-a*, both of which are executed successfully. Fig. 23c presents the execution of the remaining actions from plan (a) and the initial action of the plan (b). In this plot, the collision situation II is identified at *time=958 [s]*. This second collision situation is smoother compared to collision situation I due to the initial heading angles of both ships, resulting in a soft collision manoeuvre from MASS. In this plot, a small deviation of the ship can be identified at port C. This is due to the presence of many obstacles on the map, causing the path-planning algorithm based on fast-marching-method to encounter challenges in computing a path with unbounded static obstacles. Finally, Fig. 23d presents the execution of the remain part of plan (b) and (c). The actions executed are: *11b: move-ship port-d port-c*, *14b: move-ship port-c c port-d* and *18c: move-ship port-d port-e*. This plot also shows the potential collision situation III at *time=4079 [s]*. From this numerical simulation in case C, it is possible to see that the system has the ability to solve problems through re-planning and execute them in maritime scenarios with static obstacles and other ships. It is important to note that the collision situations between the MASS and other ships were generated by arbitrarily defining the initial positions of the target ships.

4.7. Computational time analysis

The numerical simulations were computed using a hybrid program, coded in MATLAB 2021b, Python 3.8.10 and Ubuntu 20.04. The ship simulator is coded in MATLAB, the STP planner in Python. This integration was possible using the function **pyrun** and **Os**. The entire code was developed on Ubuntu 20.04 running on a Core i7 8th generation processor, 2.6Hz and 16GB of RAM memory.

The simulation results demonstrate the effective performance of the temporal STP planner, it was able to provide a plan in less than 1.0 second for a problem involving several agents, actions and constraints. These positive results lead us to infer a successful real-world application in maritime domains for MASS. In the maritime domain, due to the significant inertia of ships and the slow response of actuators, as well as the higher acquisition frequency of sensors and equipment, system updates typically occur at intervals greater than 1 second. For example, a maritime GPS typically provides position updates every 1 second.

4.8. Discussion

This subsection analyzes the performance of the temporal STP planner within the presented case study, including the computed temporal plan, reactive capabilities, and plan execution.

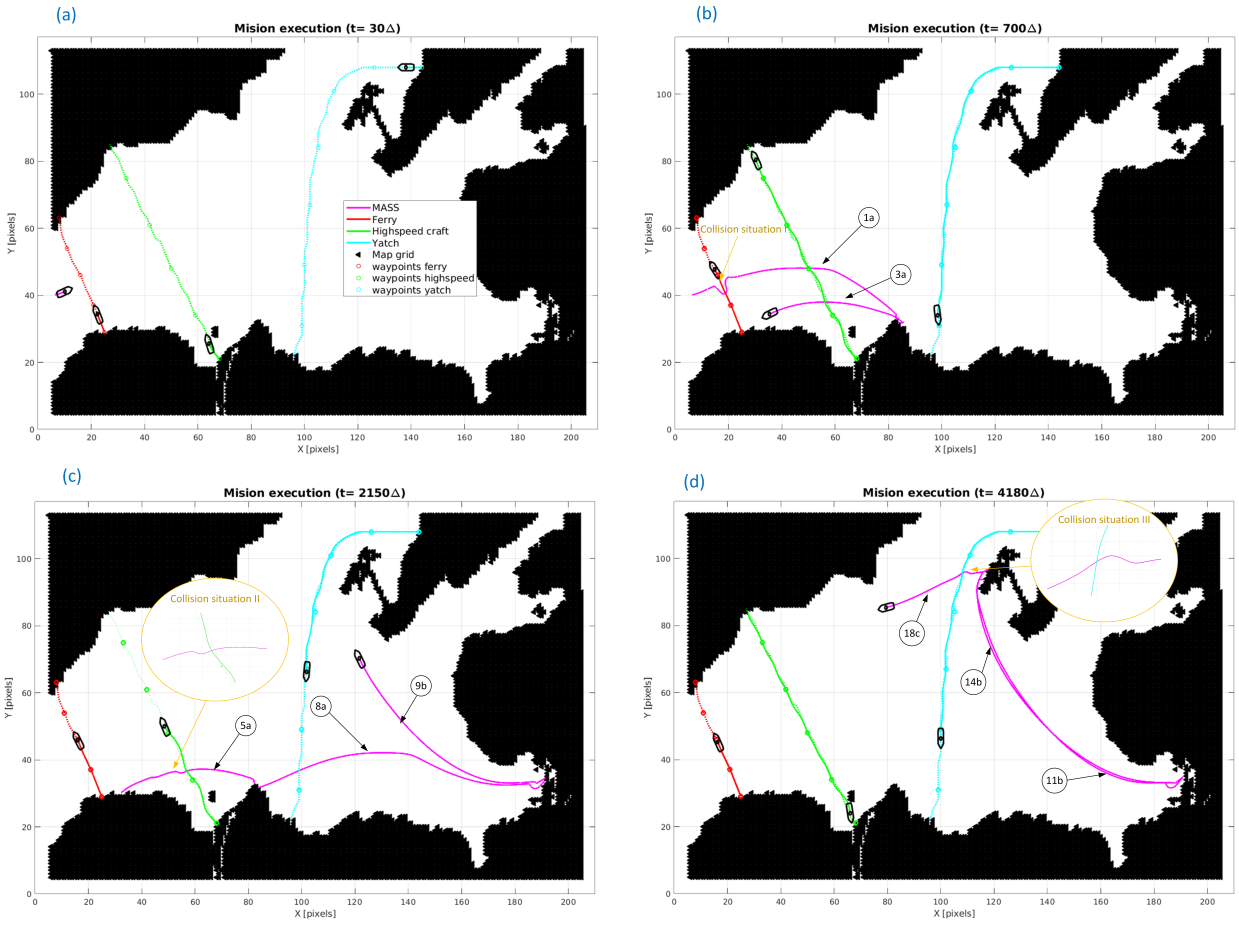

Figure 23: Mission execution of simulation case C, (a) Initial conditions for simulation (b) Executing initial plan, (c) Executing plan after the first re-planning, (d) Executing plan after the second re-planning.

In the numerical results of case A, the STP planner was able to generate a temporal plan based on a pre-defined maritime domain, see Fig. 12. This initial plan is revised by the Refinement block (Fig. 5) and an improved plan is calculated, this plan is slightly different from the original one, see Fig.13. This plan is transferred to the executor module and begins to be executed by the control module. The overall performance of the system is presented in Fig. 14-15. The events in this overall mission execution are presented in Logs of navigation (Table 4). Additionally, in order to complement the results, the distance between the MASS and the other ships is presented in Fig. 16.

In order to test the re-planning capabilities of the high-level mission planning module, a more complex scenario was presented in numerical simulations case B. Thus, an unexpected event is simulated at time = [100-170] seconds. The event consists in moving to an aleatory position, from this new state the full system must be capable of computing a new plan and execute it. The new plan is presented in Fig. 18, this new plan is different from the original. Once this new plan is computed the control system is able to execute the plan, as can be verified in Fig. 19-21. The navigation log of this mission execution is presented in Table 7.

In a maritime domain scenery, where the time of response of each action is slow, primarily due to the significant inertia of the ships, and the number of actions is finite, it was possible to see good synergy between the AI planning and the traditional GNC system. The STP planner was able to compute a feasible plan which can be executed by the control system. However, there are practical difficulties which will need to be overcome before a real-world implementation, 1) Decrease the STP planner computational time, 2) Incorporate the collision avoidance module into the AI planning. The second difficulty is particularly challenging because the collision avoidance system for autonomous surface vehicles is a complex problem itself, an interesting work addressing this problem can be found in Rothmund et al. (2022). Moreover, integrating COLAV rules into the Planning Domain Definition Language (PDDL) domain will significantly increase the number of variables and constraints. Consequently, this will lead to an increase in the computational time of the AI planning module, which is undesirable for real experiments

Table 11
Navigation log - simulation case C.

Ship Navigation logs	
Event	Starting time(s)
starting MASS computational simulation	time:0.000
reading PDDL domain inputs,	time: 0.002
computing the temporal plan,	time: 0.004
refining the plan,	time: 0.806
executing move-fueling-ship port-s port-b	time: 0.908
potential collision I detected with ferry1	time:4.91e+01
end of collision avoidance manoeuvre	time:5.45e+01
refuel-ship port-b teamf0 task started	time: 4.72e+02
executing move port-b port-a started	time: 5.52e+02
load-cont port-a cont-y started	time: 8.46e+02
executing move-ship port-a port-b started	time: 9.06e+02
potential collision II detected craft	time:9.58e+02
end of collision avoidance manoeuvre	time: 9.88e+02
unload cont-y port-b started	time: 1.206e+03
load-cont port-b cont-x started	time:1.256e+03
executing move-ship port-b port-c started	time: 1.316e+03
re-planning needed, Ship low fuel	time: time: 1900
executing move-fueling-ship port-c port-d	time: 1.902e+03
refuel-ship port-d teamf1 task started	time: 2.500e+03
executing move-ship port-d port-c started	time: 2.580e+03
unload-cont cont-x port-c task started	time: 3.174e+03
load-cont port-c cont-w task started	time: 3.224e+03
executing move-ship port-c port-d started	time: 3.284e+03
unload-cont cont-w port-d task started	time: 3.884e+03
load-cont port-d cont-z	time: 3.934e+03
re-planning needed, Ship low fuel	time: 3.994e+03
refuel-ship port-d teamf1 task started	time: 3.999e+03
executing move-ship port-d port-e started	time: 4.074e+03
potential collision III detected with yacht	time:4.079e+03
end of collision avoidance manoeuvre	time:4.087e+03
repair-ship port-e teamr0 task started,	time: 4.441e+03
end of computational simulation	time: 4.501e+03

with MASS. It's important to note that in this paper, the collision avoidance module is based on the study presented in Hinostroza and Soares (2018), and is integrated into the low-level system.

In the current early stage of the research, the results are based on numerical simulations. However, experimental validation is feasible and will be addressed in a future publication. These future experimental validations will be conducted onboard the full-scale autonomous ferry "milliAmpere 1" from the Norwegian University of Science and Technology Brekke et al. (2022), where the authors are currently engaged. This autonomous ferry is equipped with a fully operative low-level control system, including path planning, collision avoidance, path following, and control. The hardware comprises azimuth thrusters, GPS, electrical batteries, IMU, Lidar, and 5G communication. The software runs on Linux and Robotic Operative System (ROS), which is compatible with the proposed mission planner. The chosen location for experiments will be "Vestre Kanalkai" in Trondheim Fjord, offering ideal conditions for testing as it provides a real scenario with other boats navigating through.

Regarding the scenarios that could be tested, there will be two cases. The first involves a straightforward scenario where the system must compute a plan for the ferry to travel between two ferry stops, transporting passengers, navigating between two sides of the rivers, and entering docking mode. The second scenario entails a re-planning case, triggered by the ferry's battery consumption, which requires the system to re-plan when a low battery is detected, including reserving some time for charging the batteries.

Table 12
Comparison between STP, TP, TSHE, and Rosplan planners.

Planner	Number of actions plan	Computational time (s)	Number of nodes explored	Memory used (MB)
STP	52	6.5	28403	38.3
TP	50	7.6	22133	42.5
SEQ	52	8.3	6324	-
TPSHE	53	9.3	4138	-
Rosplan	77	24.8	-	50

4.9. Comparison of the STP planner with other planning algorithms

In order to assess the performance of the temporal STP planner in comparison with other temporal planners such as ROSplan (Cashmore et al., 2015) and "tempo" (TP), TPSHE planner and Sequential planner (SEQ) from Jiménez et al. (2015); Furelos Blanco et al. (2018), numerical simulations were conducted using the PDDL2.1 Satellite domain from the International Planning Competition (IPC) (Briel et al., 2008).

The temporal plan generated by the STP planner consists of 52 actions with a total duration of 132 seconds. Due to the large number of actions in the computed plan, it is not feasible to be present it here. However, the main characteristics of the plan, number of actions, computational cost, number of nodes (potential states of the system) explored and memory used, are detailed in Table 12. From this table is possible to see the superior performance of the STP planner compared with others temporal planners. However, the number of actions in the plan calculated with STP planner is higher than with the TP planner, it is because the STP planner is an extension of the TP planner (Furelos Blanco et al., 2018). It is important to emphasize that this PDDL domain was chosen because it has been widely studied, and results for other planners are available in the literature.

5. Conclusions

A mission planning system for maritime autonomous surface vehicles based on high-level AI temporal planning was presented. To the authors' best knowledge, this is the first research result connecting temporal planning with guidance, navigation and control for autonomous ship operations. Our results demonstrate that such integration increases significantly the situations a MASS can deal with autonomously without resorting to a remote operation. The temporal STP planner is an adequate method to compute fast and reliable plans for a MASS operating in a real maritime environment

including dynamic and static obstacles. The GNC system, composed of a path-planning, path-following and control system has demonstrated a positive synergy with the new planning module and has shown a good performance executing a calculated plan. Finally, it is important to remark that the proposed mission planning is not only about solving motion planning or collision avoidance for autonomous ships but also dealing with additional functionalities, normally not solvable by classical guidance and control approaches.

Regarding future work, it is important to expand the mission planner to also include elements related to the power system and available fuel, which are decisive in scheduling actions. Taking into account these new parameters involve new challenges, such as the inclusion of ship dynamics into the problem domain. To solve these new type of problems two options could be investigated, the use of hybrid planners and discretization of fuel level in a finite number of symbolic variables. Once these new challenges have been addressed, conducting field experiments onboard ships to evaluate the performance of the proposed methodology will constitute a significant contribution to the field of MASS.

Acknowledgments

This work was funded by the Research Council of Norway (RCN) under the projects "Autonomous robot missions with AI-based planning and acting"(ROBPLAN), Norway, project number 32274, and "SFI Autonomous ships for safe and sustainable operations" (SFI AutoShip), Norway, project number 309230. The authors are grateful to the Centre for Marine Technology and Ocean Engineering (CENTEC) for providing the ship's mathematical model for simulations.

References

Barreiro, J., Boyce, M., Do, M., Frank, J., Iatauro, M., Kichkaylo, T., Morris, P., Ong, J., Remolina, E., Smith, T., et al., 2012. Europa: A platform for ai planning, scheduling, constraint programming, and optimization. 4th International Competition on Knowledge Engineering for Planning and Scheduling (ICKEPS) .

Bernard, D., Dorais, G., Gamble, E., Kanefsky, B., Kurien, J., Man, G., Millar, W., Muscettola, N., Nayak, P., Rajan, K., et al., 1999. Spacecraft autonomy flight experience: The ds1 remote agent experiment .

Bitar, G., Eriksen, B.O.H., Lekkas, A.M., Breivik, M., 2021. Three-phase automatic crossing for a passenger ferry with field trials, in: 2021 European Control Conference (ECC), pp. 2271–2277.

Blum, A.L., Langford, J.C., 1999. Probabilistic planning in the graphplan framework, in: European Conference on Planning, Springer. pp. 319–332.

Brekke, E.F., Eide, E., Eriksen, B.O.H., Wilthil, E.F., Breivik, M., Skjellaug, E., Helgesen, Ø.K., Lekkas, A.M., Martinsen, A.B., Thyri, E.H., 2022. milliampere: An autonomous ferry prototype, in: Journal of Physics: Conference Series, p. 012029.

Brekke, E.F., Wilthil, E.F., Eriksen, B.H., Kufoalor, D., Helgesen, Ø.K., Hagen, I.B., Breivik, M., Johansen, T.A., 2019. The autosea project: Developing closed-loop target tracking and collision avoidance systems, in: journal of physics: conference series, IOP publishing. p. 012020.

Briel, M.H.L., Vossen, T., Kambhampati, S., 2008. Loosely coupled formulations for automated planning: An integer programming perspective. Journal of Artificial Intelligence Research 31, 217–257.

Cashmore, M., Fox, M., Long, D., Magazzeni, D., Ridder, B., Carrera, A., Palomeras, N., Hurtos, N., Carreras, M., 2015. Rosplan: Planning in the robot operating system, in: Proceedings of the International Conference on Automated Planning and Scheduling, pp. 333–341.

Cashmore, M., Magazzeni, D., Zehtabi, P., 2020. Planning for hybrid systems via satisfiability modulo theories. Journal of Artificial Intelligence Research 67, 235–283.

Coles, A., Fox, M., Halsey, K., Long, D., Smith, A., 2009. Managing concurrency in temporal planning using planner-scheduler interaction. Artificial Intelligence 173, 1–44.

Cui, Z., Guan, W., Luo, W., Zhang, X., 2023. Intelligent navigation method for multiple marine autonomous surface ships based on improved ppo algorithm. Ocean Engineering 287, 115783.

Currie, K., Tate, A., 1991. O-plan: the open planning architecture. Artificial intelligence 52, 49–86.

Enevoldsen, T.T., Blanke, M., Galeazzi, R., 2023. Autonomy for ferries and harbour buses: a collision avoidance perspective. arXiv preprint arXiv:2301.02711 .

Eriksen, B.O.H., Bitar, G., Breivik, M., Lekkas, A.M., 2020. Hybrid collision avoidance for asvs compliant with colregs rules 8 and 13–17. Frontiers in Robotics and AI 7.

Erol, K., Hendler, J., Nau, D.S., 1994. Htn planning: Complexity and expressivity, in: AAAI, pp. 1123–1128.

Estlin, T., Castano, R., Anderson, R., Gaines, D., Fisher, F., Judd, M., 2003. Learning and planning for mars rover science .

Executive, T.M., 2022. Japan demonstrates long distance autonomous ship operations. https://maritime-executive.com/article/japan-demonstrates-long-distance-autonomous-ship-operations , (accessed: 09.01.2024).

Fikes, R.E., Nilsson, N.J., 1971. Strips: A new approach to the application of theorem proving to problem solving. Artificial intelligence 2, 189–208.

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

Furelos Blanco, D., Jonsson, A., Palacios Verdes, H.L., Jiménez, S., 2018. Forward-search temporal planning with simultaneous events, in: 13th Workshop on Constraint Satisfaction Techniques for Planning and Scheduling, Jun 24-29; Delft, the Netherlands.

Ghallab, M., Nau, D., Traverso, P., 2004. Automated Planning: theory and practice. Elsevier.

Helmert, M., 2006. The fast downward planning system. Journal of Artificial Intelligence Research 26, 191–246.

Hinostroza, M., Lekkas, A.M., 2022. A rudimentary mission planning system for marine autonomous surface ships. IFAC-PapersOnLine 55, 196–203.

Hinostroza, M., Lekkas, A.M., Transeth, A.A., Luteberget, B., de Jonge, C., Sagatun, S.I., 2023. Automated planning for inspection and maintenance operations using unmanned ground vehicles. IFAC-PapersOnLine 56, 7873–7879.

Hinostroza, M., Soares, C.G., 2018. Collision avoidance, guidance and control system for autonomous surface vehicles in complex navigation conditions, in: Progress in maritime technology and engineering. Taylor & Francis Group London, UK, pp. 121–132.

Hinostroza, M., Xu, H., Guedes Soares, C., 2017. Experimental and numerical simulations of zig-zag manoeuvres of a self-running ship model, Taylor & Francis Group, London. pp. 563–570.

Hinostroza, M., Xu, H., Soares, C.G., 2021. Experimental results of the cooperative operation of autonomous surface vehicles navigating in complex marine environment. Ocean Engineering 219, 108256.

Hinostroza, M.A., Xu, H., Guedes Soares, C., 2020. Motion Planning, Guidance, and Control System for Autonomous Surface Vessel. Journal of Offshore Mechanics and Arctic Engineering 143.

IMO, 2021. Autonomous shipping. https://www.imo.org/en/About/Conventions/Pages/International-Convention-on-Maritime-Search-and-Rescue-(SAR).aspx , (accessed: 31.03.2022).

Jiménez, S., Jonsson, A., Palacios, H., 2015. Temporal planning with required concurrency using classical planning, in: Proceedings of the International Conference on Automated Planning and Scheduling, pp. 129–137.

Konsberg-Maritime, 2021. World's first fully electric and autonomous container ship. https://www.kongsberg.com/maritime/support/themes

/autonomous-ship-project-key-facts-about-yara-birkeland/ , (accessed: 31.03.2022).

Konsberg-Maritime, 2023. Trial of autonomous shipping. https://www.kongsberg.com/maritime/about-us/news-and-media/news-archive/2023/trial-of-autonomous-shipping/ , (accessed: 09.01.2024).

Lekkas, A.M., Fossen, T.I., 2013. Line-of-sight guidance for path following of marine vehicles. Advanced in marine robotics , 63–92.

Lekkas, A.M., Fossen, T.I., 2014. Integral los path following for curved paths based on a monotone cubic hermite spline parametrization. IEEE Transactions on Control Systems Technology 22, 2287–2301.

Lekkas, A.M., Roald, A.L., Breivik, M., 2016. Online path planning for surface vehicles exposed to unknown ocean currents using pseudospectral optimal control. IFAC-PapersOnLine 49, 1–7.

Liu, Y., Bucknall, R., 2016. The angle guidance path planning algorithms for unmanned surface vehicle formations by using the fast marching method. Applied Ocean Research 59, 327–344.

Ludvigsen, M., Sørensen, A.J., 2016. Towards integrated autonomous underwater operations for ocean mapping and monitoring. Annual Reviews in Control 42, 145–157.

McGann, C., Py, F., Rajan, K., Thomas, H., Henthorn, R., McEwen, R., 2007. T-rex: A model-based architecture for auv control, in: 3rd Workshop on Planning and Plan Execution for Real-World Systems.

Muscettola, N., Dorais, G.A., Fry, C., Levinson, R., Plaunt, C., Clancy, D., 2002. Idea: Planning at the core of autonomous reactive agents, in: Sixth International Conference on AI Planning and Scheduling.

Nau, D., Cao, Y., Lotem, A., Munoz-Avila, H., 1999. Shop: Simple hierarchical ordered planner, in: Proceedings of the 16th international joint conference on Artificial intelligence-Volume 2, pp. 968–973.

NTNU, 2022. Ntnu trials world's first urban autonomous passenger ferry. https://norwegianscitechnews.com/2022/09/ntnu-trials-worlds-first-urban-autonomous-passenger-ferry/ .

Pinto, J., Sousa, J., Py, F., Rajan, K., 2012. Experiments with deliberative planning on autonomous underwater vehicles, in: IROS 2012 Workshop on Robotics for Environmental Monitoring, Vilamoura, Portugal.

Piotrowski, W., Fox, M., Long, D., Magazzeni, D., Mercorio, F., 2016. Heuristic planning for hybrid systems, in: Proceedings of the AAAI Conference on Artificial Intelligence.

Rabideau, G., Knight, R., Chien, S., Fukunaga, A., Govindjee, A., 1999. Iterative repair planning for spacecraft operations using the aspen system, in: Artificial Intelligence, Robotics and Automation in Space, p. 99.

Rajan, K., Saffiotti, A., 2017. Towards a science of integrated ai and robotics.

Rolls-Royce, 2018. Rolls-royce and finferries demonstrate world's first fully autonomous ferry. https://www.rolls-royce.com/media/press-releases/2018/03-12-2018-rr-and-finferries-demonstrate-worlds-first-fully-autonomous-ferry.aspx , (accessed: 31.03.2022).

Rothmund, S.V., Tengesdal, T., Brekke, E.F., Johansen, T.A., 2022. Intention modeling and inference for autonomous collision avoidance at sea. Ocean Engineering 266, 113080.

Sans-Muntadas, A., Kelasidi, E., Pettersen, K.Y., Brekke, E., 2019. Path planning and guidance for underactuated vehicles with limited field-of-view. Ocean Engineering 174, 84–95.

Splash, 2023. South korea takes the next step forward in autonomous shipping. https://splash247.com/south-korea-takes-the-next-step-forward-in-autonomous-shipping/ , (accessed: 09.01.2024).

Thompson, F., Galeazzi, R., 2020. Robust mission planning for autonomous marine vehicle fleets. Robotics and Autonomous Systems 124, 103404.

Thompson, F., Guihen, D., 2019. Review of mission planning for autonomous marine vehicle fleets. Journal of Field Robotics 36, 333–354.

Xu, H., Hinostroza, M., Soares, C.G., 2018. Estimation of hydrodynamic coefficients of a nonlinear manoeuvring mathematical model with free-running ship model tests. International Journal of Maritime Engineering 160.

Xue, L., Lekkas, A.M., 2020. Comparison of ai planning frameworks for underwater intervention drones, in: Global Oceans 2020: Singapore – U.S. Gulf Coast, pp. 1–9.

Zeabuz, 2023. Redefining waterborne mobility. https://www.zeabuz.com/ , (accessed: 08.10.2023).

Appendix

A. PDDL Domain for autonomous surface vehicles

This appendix presents the PDDL 2.1 domain for marine autonomous surface ships used in the numerical simulations A, B and C presented in section 4. In the first two simulations, the domain used is slightly different than in third simulation, as it does not consider of ship fuel condition: *(at-start (fuel ship))*, for the actions: *durative - action move-ship*, *durative - action (un)load-cont* and *durative - action repair-ship*.

```
(define (domain mission-planning-mass)
(:requirements :typing :durative-actions)
  (:types
     ship - vehicle
     port - location
     cont team_rep team_refuel - subject
     route)
  (:predicates
    (at ?physical_obj1 - subject ?location1
    - location)
    (available ?vehicle1 - vehicle)
    (busy ?vehicle1 - vehicle)
    (loaded ?subject1 - subject ?vehicle1 - vehicle)
    (connects ?route1 - route ?location1 - location
    ?location2 - location)
    (in_port ?location1 - location ?port1 - port)
    (route_available ?route1 - route)
    (no_rep ?vehicle1 - vehicle)
    (repaired ?vehicle1 - vehicle)
    (no_fuel ?vehicle1 - vehicle)
    (fuel ?vehicle1 - vehicle)
    )

(:functions
           (distance ?O - location ?L - location)
           (route-length ?O - route)
            (speed ?V - vehicle)
            )
  (:durative-action move-ship
       :parameters ( ?V - vehicle ?O - location ?Port -
       port ?L - location ?Port1 - port ?R - route)
       :duration (= ?duration
                 (/ (route-length ?R) (speed ?V)))
       :condition (and
                       (at start (at ?V ?O))
                       (at start (in_port ?O ?Port))
                       (at start (in_port ?L ?Port1))
                       (at start (fuel ?V))
                       (at start (connects
                   ?R ?Port ?Port1))
       )
       :effect (and
                  (at start (not (at ?V ?O)))
                  (at end (at ?V ?L))
        ) )
  (:durative-action move-fueling-ship
       :parameters ( ?V - vehicle ?O - location ?Port -
       port ?L - location ?Port1 - port ?R - route)
       :duration (= ?duration
                 (/ (route-length ?R) (speed ?V)))
```

```
      :condition (and
                      (at start (at ?V ?O))
                      (at start (in_port ?O ?Port))
                      (at start (in_port ?L ?Port1))
                      (at start (no_fuel ?V))
                      (at start (connects
                   ?R ?Port ?Port1))
      )
      :effect (and
                (at start (not (at ?V ?O)))
                (at end (at ?V ?L))
       ) )
(:durative-action load-cont
      :parameters ( ?V - ship ?L - port ?P - cont)
      :duration (= ?duration 60)
      :condition (and
           (over all (at ?V ?L))
           (at start (at ?P ?L))
           (at start (fuel ?V))
           (at start (available ?V))
      )
      :effect (and
           (at start (not (available ?V)))
           (at start (busy ?V))
           (at start (not (at ?P ?L)))
           (at end (loaded ?P ?V))
       ) )
 (:durative-action unload-cont
      :parameters ( ?P - cont ?L - port ?V - ship )
      :duration (= ?duration 50)
      :condition (and
           (over all (at ?V ?L))
           (at start (loaded ?P ?V))
           (at start (fuel ?V))
           (at start (busy ?V))
      )
      :effect (and
           (at start (not (loaded ?P ?V)))
           (at end (at ?P ?L))
           (at start (not (busy ?V)))
           (at end (available ?V))
       ) )
(:durative-action repair-ship
      :parameters ( ?V - ship ?L - port ?K - team_rep)
      :duration (= ?duration 60)
      :condition (and
           (at start (at ?V ?L))
           (at start (at ?K ?L))
           (at start (fuel ?V))
           (at start (no_rep ?V))
      )
      :effect (and
           (at start (not (no_rep ?V)))
           (at start (repaired ?V))
       ) )
(:durative-action refuel-ship
      :parameters ( ?V - ship ?L - port ?Y - team_refuel)
      :duration (= ?duration 80)
      :condition (and
           (at start (at ?V ?L))
           (at start (at ?Y ?L))
           (at start (no_fuel ?V))
      )
      :effect (and
           (at start (not (no_rep ?V)))
           (at start (fuel ?V))
       ) )
```

CRediT authorship contribution statement

M.A. Hinostroza: Conceptualization, Investigation, Software and Writing - original draft preparation. **A.M. Lekkas:** Conceptualization, Methodology, Funding acquisition and Writing - review.