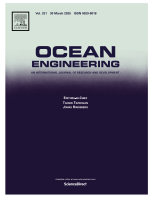
Ocean Engineering 340 (2025) 122345

Contents lists available at ScienceDirect

Ocean Engineering

journal homepage: www.elsevier.com/locate/oceaneng



Research paper

Integration of spatial database systems and sampling-based path planning for optimizing maritime navigation

Nikolai Lauvås *, Tor Arne Johansen , Jo Arve Alfredsen

Norwegian University of Science and Technology, Department of Engineering Cybernetics, O. S. Bragstads plass 2D, Trondheim, 7034, Trøndelag, Norway

ARTICLE INFO

Keywords:
Path planning
Maritime navigation
Autonomous vehicles
Spatial database
Electronic navigational charts

ABSTRACT

This paper addresses the challenge of representing configuration spaces for sampling-based path planning in maritime navigation scenarios by using data-efficient vector maps stored in spatial database systems. The proposed approach optimizes the performance of fundamental algorithm operations in large environments, such as collision checking, by using spatial indexing to efficiently reduce the number of geometries that need to be evaluated by computationally expensive spatial predicates. An implementation combining the SpatiaLite database system, standardized electronic navigational charts (ENC) map features, and the widely used Open Motion Planning Library (OMPL) demonstrates practical applicability. The implemented system provides collision-free paths for maritime navigation, includes a graphical user interface, and is incorporated to a system for autonomous surface vehicles. Simulations show that the implementation supports multiple planning algorithms in generating valid paths in four representative large-scale maritime environments: a cluttered archipelago, a river inlet, a peninsula, and a fjord transit.

1. Introduction

The pursuit of greater autonomy in robotic systems often depends on effective motion planning. These motion planners operate on the basis of spatial representations of the environment, aiming to identify valid paths from initial to desired states while adhering to a set of constraints.

Common algorithms for solving motion planning problems include graph-based search algorithms such as A* (Hart et al., 1968) and Dijkstra's (Dijkstra, 1959), as well as sampling-based algorithms like Rapidly-exploring Random Trees (RRT) (LaValle and Kuffner, 2001), Probabilistic Roadmaps (PRM) (Kavraki et al., 1996), and their path-optimizing extensions RRT* and PRM* (Karaman and Frazzoli, 2011). The choice of algorithm is intrinsically related to the data structure used to represent the environment, which in turn influences the storage requirements and performance. For instance, graph-based algorithms necessitate an *a priori* discretization, often achieved through an occupancy map for geographical planning. However, the resolution of such maps presents a trade-off between computational demands and loss of fidelity.

Sampling-based motion planners (SBMPs) differs by incrementally sampling states at planning time to construct data structures to represent the planning problem. They reach their planning goal through combining a state sampling function, a sampled state validity checker, and a local path planner that checks for constraint violations when connecting sampled states. Optimal SBMPs additionally minimize a cost function, such as path length or expended traversing effort, and can be extended to support multi-objective optimization where trade-offs between competing criteria must be balanced. Although SBMPs offer flexibility, they lack the *resolution-optimal* and *optimally efficient* theoretical guarantees provided by graph-based planners such as A*.

Vector representations (e.g., polygons) are well suited as spatial environments for SBMPs and, in contrast to rasterized representations, have the advantage of decoupling resolution from the represented area (i.e. a square can be represented with four points independently of the area it covers). However, managing complex environments of polygons can be challenging without efficient data structures and a proper management system. Spatial databases extend relational models to support spatial features, offering a promising solution to this challenge. They support indexing for efficient data retrieval and a query language to extract, analyze, and edit data, making them a convenient choice for handling complex geospatial environments. In addition, spatial databases are widely used in geographic information systems (GIS), which ensures the broad availability of existing software implementations.

1.1. Literature review

A variety of environment representations have been utilized in path planning, typically chosen based on the data's source and format. This review of the literature examines some common

* Corresponding author.
E-mail addresses: nikolai.lauvas@ntnu.no (N. Lauvås), tor.arne.johansen@ntnu.no (T.A. Johansen), jo.arve.alfredsen@ntnu.no (J.A. Alfredsen).

https://doi.org/10.1016/j.oceaneng.2025.122345
Received 27 January 2025; Received in revised form 2 June 2025; Accepted 29 July 2025
Available online 6 August 2025

representations, particularly emphasizing their application to maritime navigation and path planning. Table C.9 provides a concise overview of the advantages and disadvantages of the reviewed literature. For recent advances in SBMPs, the reader is referred to the comprehensive works of Gammell and Strub (2021) and Elbanhawi and Simic (2014).

Occupancy grids, as first popularized in Moravec and Elfes (1985), have seen significant research effort, in part due to the ease with which they can be updated from real-time sensor data. Each cell in the grid is given a value according to how likely it is to be occupied by an obstacle, encoding the uncertainty of the sensor data.

An alternative to uniform occupancy grids, quadtrees, as named in Finkel and Bentley (1974), represent grids of varying resolutions in tree structures where each internal node has four children, representing the four quadrants of its parent, with the entire represented area as the root node. Because each child node can also be divided into four or given an occupancy state, quadtrees can achieve a variable resolution by adjusting the depth of individual branches in the tree.

For maritime navigation, Shah and Gupta (2020) utilized quadtrees in conjunction with visibility graphs generated from polygons representing a maritime environment to increase the performance of the A* algorithm. This enabled them to successfully demonstrate long-distance planning in large maritime environments, with potential applicability to other polygonal planning domains.

Instead of creating a rasterization of the polygonal area, Candeloro et al. (2017) presented an approach in which a Voronoi diagram was created from the vertices of land obstacles in a maritime environment. The Yen-Dijkstra algorithm was then applied to the Voronoi edges to compute the shortest path on the roadmap, which was subsequently refined to produce the final desired trajectory.

In contrast to the solutions of Shah and Gupta (2020) and Candeloro et al. (2017), the solution proposed in this paper enables direct planning in large polygonal environments, without rasterization or other significant preprocessing required. This simplifies the reuse of data for multiple purposes, such as the target search planner presented in Lauvås and Alfredsen (2023).

For three-dimensional planning, Deeken et al. (2018) proposed a framework for semantic map representation based on an extended version of the PostGIS spatial database. In the setup, spatial operations from the database provide the robot with reasoning functionality and are used to generate an occupancy grid compatible with the standard bindings of the robot operating system (ROS) (Quigley et al., 2009) for path planning.

Another viable alternative to rasterizing the entire environment of a robotic vehicle is to store only the boundaries of obstacles. This concept is proposed by Dallolio et al. (2022), where heavily modified data from ENCs stored in an indexed SQLite database are used in anti-grounding and planning subsystems of an autonomous vehicle. The outlines of the ENC depth contours were stored as points that occurred at regular intervals, which foregoes the need for spatial extensions to the spatial database. This comes at the cost of a significant increase in storage capacity and less chart data available for other functionality.

Blindheim and Johansen (2022) presented a custom framework for the development of hydrographic information systems (HIS) that utilize data from ENCs, along with examples of relevant applications of the framework for maritime systems. Their main research question about how to improve access to hydrographic data is also partially answered by the ENC database presented in this paper, but instead of a Python-based API, this paper utilizes the already mature and established field of spatial database management systems to solve the problem.

Enevoldsen et al. (2022) presented a sampling-based approach to maritime path planning where the RRT* planning strategy, ENCs, and elliptical-like representations of COLREGs were combined. This provides a short-horizon planner that could be utilized as a navigational aid integrated in an ECDIS, or as a component in an autonomously navigating vehicle. Their approach for achieving uniformly distributed sampling by transforming the ENC data to a triangulated space is fundamentally different from the approach presented in this paper, which aims to achieve an optimized and simpler rectangular rejection sampling strategy on the ENC polygons.

1.2. Contribution

This paper bridges a research gap by leveraging spatial databases for environment representation in SBMPs, drawing on prior work in both domains. The key advantages of this approach include simplified vector data management, efficient data retrieval through R*-tree indexing, and the availability of algorithms for spatial operations that can be utilized in path-planning algorithms.

The primary contributions of this paper are:

- Demonstrating how spatial database systems can effectively supply the primitive procedures required for environment representation in SBMPs.
- Introducing a platform-agnostic framework that integrates spatial databases with SBMPs for enhanced flexibility.
- Delivering a validated C++ implementation combining SpatiaLite and the Open Motion Planning Library (OMPL) (Sucan et al., 2012), tailored for maritime navigation by operating on readily available electronic navigational charts (ENC) issued by national hydrographic offices.
- Extending the framework's utility by integrating it with the LSTS toolchain for autonomous vehicles (Pinto et al., 2013).

The approach is validated as a global path planner through 12000 simulated benchmarking runs across four representative maritime scenarios and ten SBMPs. While the current implementation excludes dynamic objects, it is suitable for decision-support applications or autonomous navigation in low-traffic maritime areas where vessel encounters are rare. Future work will focus on extending the framework to incorporate COLREGs compliance and dynamic obstacle handling, thereby broadening its applicability to more complex and congested maritime scenarios.

2. Underlying concepts

2.1. Spatial database management systems

The OpenGIS® Simple Features Access (SFA) described in Herring (2011a), also ratified in ISO 19125, defines a common object model for storing geometric objects such as points, curves, and surfaces within a specific reference system. The second part of the standard defines a structured query language (SQL) extension that facilitates storing, retrieving, and querying collections of features, which may include both geometric and non-spatial attributes (see Herring (2011b)). This includes definitions of relational operators between geometries as given in the dimensionally extended 9-intersection model (DE-9IM), such as operations for detecting when one geometry covers or intersects another.

Support for SFA in databases is typically achieved through extensions to existing non-spatial DBMSs, such as the PostGIS extension to PostgreSQL and the SpatiaLite extension to SQLite3. These extensions implement subsets of the SFA standard, converting a DBMS into a spatial database management system (SDBMS) or simply a spatial database.

A common application of spatial databases is the storage and management of data for a GIS. The open-source Geospatial Data Abstraction Library (GDAL), as documented in Warmerdam and Rouault (2021), along with its OGR Simple Features Library, is widely used in this context and supports multiple SFA formats. The `ogr2ogr` utility, utilized in this paper to convert between simple features storage formats, is one of the many tools provided by GDAL. The open-source QGIS Geographic Information System (QGIS Development Team, 2021) is also heavily reliant on GDAL, and was utilized to create all maps in this paper.

2.2. Sampling-based motion planning

SBMPs function by iteratively sampling points within a representation of the configuration space. These samples are used to construct internal data structures, wherein a local path planner assesses the feasibility of connecting one point to another. Following the notation introduced by Karaman and Frazzoli (2011), the planning problem for SBMPs can be formally defined in d dimensions as follows: Given the configuration space $\chi = (0,1)^d$, the obstacle subset $\chi_{obs} \in \chi$ and the obstacle free subset $\chi_{free} = closure(\chi \backslash \chi_{obs})$, the objective is to find a function $\sigma[0,1] \rightarrow \mathbb{R}^d$ that defines a feasible path from $\sigma(0) = x_{init}$ to $\sigma(1) = x_{goal}$ subject to the constraint $\sigma(\tau) \notin \chi_{obs}, \forall \tau \in [0,1]$. The dimension of χ is defined as $d = 2$ in this paper, according to the target use case of ASV path planning.

Beyond producing a feasible path, introducing a quality measure for comparing multiple feasible paths allows for optimization according to a specified metric. Given the set of all potential paths Σ and the subset of feasible paths $\Sigma_{free} \subset \Sigma$, a quality measure $c : \Sigma \rightarrow \mathbb{R}$ provides a single numeric value to be optimized, with the aim of approaching the optimal feasible path $\sigma^* \in \Sigma_{free}$. The subset of asymptotically optimal SBMPs is designed to ensure that the probability of discovering σ^* approaches one asymptotically as the computational effort approaches infinity.

To facilitate planning decisions within the configuration space χ, SBMPs employs a set of primitive procedures which, according to the notation in Karaman and Frazzoli (2011), can be summarized as follows:

- `Sample`$_i(\omega)$ returns the i-th sample from a sequence ω of independent and identically distributed set of points in χ.
- `SampleFree`$_i(\omega)$ returns the i-th sample from a sequence ω of independent and identically distributed set of points in χ_{free}.
- `CollisionFree`(x, x') returns `True` if the line segment between x and x' does not pass through χ_{obs}.
- `Cost`(x, x') returns the cost of extending a path containing x with a line segment to x'.

A comprehensive review of SBMPs, including a more extensive catalog of the primitive procedures are provided by Elbanhawi and Simic (2014). For the rapidly advancing field of asymptotically optimal SBMPs, Gammell and Strub (2021) provides a recent review.

2.3. Electronic navigational charts

The use of navigational charts dates back centuries, offering static spatial information such as land areas, expected depths, and various objects of interest to mariners. In recent years, the charts have been digitized, following the standards set by the International Hydrographic Organization (IHO) in the S-57 format, "Transfer Standard for Digital Hydrographic Data" (International Hydrographic Organization, 2000a). The real-world entities represented by the S-57 format are categorized into a finite number of types according to an ontology defined in the IHO Object Catalogue (International Hydrographic Organization, 2000b). Each object is defined using both metadata and spatial features, with spatial features represented in a format like SFA that includes geometries such as points, lines, and polygons.

The S-57 standard encompasses not only spatial data but also metadata, including the covered area, intended use, and a confidence rating of its accuracy for each chart. To maintain the relevance of each chart, national hydrographic offices issue updates throughout the chart's lifetime. These updates must be applied to the original S-57 chart to ensure that it consistently provides accurate and pertinent spatial information.

While the IHO is developing a new standard known as S-100 to replace the existing S-57 standard, formalization and widespread distribution of this new standard are still pending. In the interim, S-57 remains the primary standard for digital hydrographic data.

3. Integration of spatial databases in planning algorithms

This section outlines how a spatial database can be used to store the configuration space of a planning problem and provides a mapping to the primitive procedures required by an SBMP algorithm. The basic concept is summarized in Fig. 1.

3.1. Primitive procedures provided by a spatial database

The characteristic sampling of SBMPs through the primitive procedure `SampleFree`$_i(\omega)$ is typically implemented using rejection sampling. Assuming the existence of a `Sample`$_i(\omega)$ procedure to generate random samples from χ, the task of the database reduces to ascertaining that the samples lie within χ_{free}. This involves converting the sample ω into a geometric representation based on the shape and size of the entity being planned for, with a point geometry as the simplest form. Given a single table containing all obstacle features for χ_{obs}, the spatial predicate `ST_Intersects` can be used to only return obstacles that collide with the sample geometry. If no intersecting obstacle geometries are returned, it is returned by the `SampleFree`$_i(\omega)$ procedure.

A similar approach can be applied to the `CollisionFree`(x, x') procedure, but instead of using the sample geometry in the `ST_Intersects` spatial predicate, a geometry representing the transition between states is introduced. The simplest option is a straight line in a Linestring geometry.

The SFA DE-9IM model also offers alternative spatial predicates that could be used in the `SampleFree`$_i(\omega)$ and `CollisionFree`(x, x') procedures, such as the `ST_Within` spatial predicate. The choice of a predicate

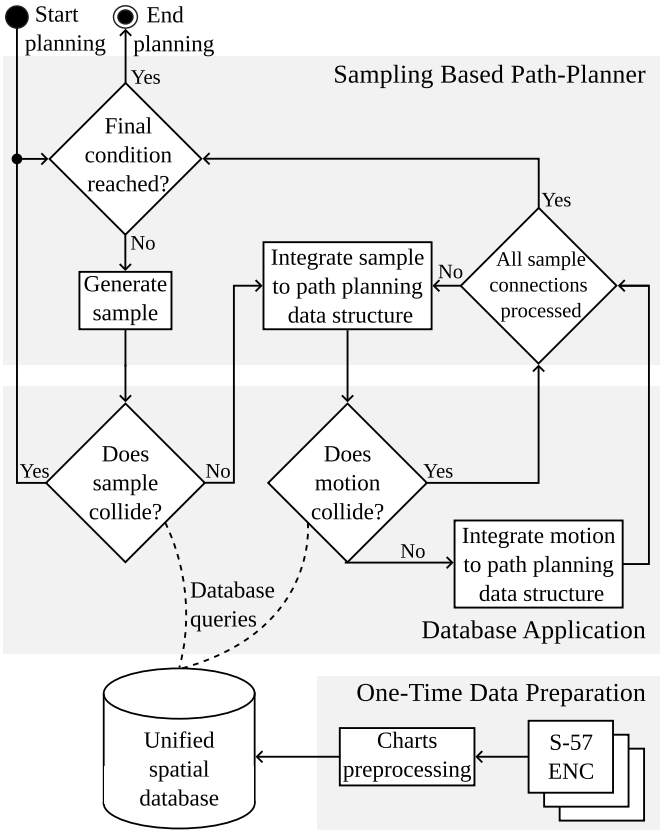


Fig. 1. Overview of the proposed platform-agnostic concept that integrates a spatial database with an SBMP. During execution, the SBMP utilizes efficient queries provided by the database application to access the environment representation stored in the spatial database and check samples and motion between samples for collisions. Prior to execution, the S-57 dataset is preprocessed once to generate the data stored in the unified spatial database.

should be taken according to the specific optimizations available in the utilized spatial database management system.

3.2. Preparing the database

To simplify further database operations, all available datasets that are relevant to the motion planner's configuration space are collected, classified, and combined into tables representing χ_{obs} and χ_{free}. For spatial features where accuracy ratings are available, such as in the S-57 standard, the `ST_Buffer`(geometry, distance, segments) operation can be used to enlarge obstacles with a surrounding buffer to guarantee that the produced path is safe. While `ST_Buffer` is not defined in the SFA, it is commonly implemented in spatial databases. This step may be omitted if accuracy ratings are not available.

3.3. Database optimizations

The efficiency of the `SampleFree` and `CollisionFree` procedures is vital to reducing the runtime of the SBMP algorithms. During the development of this concept, four approaches to optimize querying performance were identified:

- Reducing the number of intersection checks by using a spatial index to filter out irrelevant geometries.
- Terminating queries once an intersection occurs to avoid additional `ST_Intersects` operations.
- Utilizing persistent database statements to avoid recompilations of SQL to bytecode. Instead, the locations of points and lines can be changed using the parameter binding mechanism.
- Unifying multiple datasets, obstacles, and obstacle-free objects in two tables, one for χ_{obs} and another for χ_{free} tables, to reduce the number of queries needed.

3.4. Application interface

The application interface provides functionality that simplifies defining planning problems for the motion planning algorithm. If the user is human, a graphical user interface can be used, while if the path planner is to be integrated into a robotic system or utilized by another algorithm, a simple text-based interface suffices.

On completion, the SBMP algorithm returns a series of states from its internal data structures representing the planned path. These states must subsequently be converted to an appropriate format according to their intended application. For visualization purposes, a Linestring geometry allows the path to be drawn within a GIS system. In a robotic system, transforming to the system's motion primitives is more appropriate.

4. Example implementation

This section presents a practical implementation of maritime path planning using spatial features derived from S-57 ENCs. The features are stored in a SpatiaLite database, which provides a portable format that can be accessed through a database application. Instead of implementing the SBMPs directly, an integration of OMPL by Kavraki Labs (Sucan et al., 2012) was opted for to enable experimentation with multiple state-of-the-art planning algorithms. Finally, the setup was integrated into the open-source LSTS toolchain for robotic vehicles (Pinto et al., 2013). This integration includes a GUI plugin for the Neptus control and command center (vehicle operator GUI), as well as a custom database application integrated into the C++-based DUNE robotic middleware, which runs on-vehicle.

4.1. Chart preprocessing

The preprocessing stage is implemented as a Bash-script, provided in the Supplementary Materials, which integrates detailed comments and SQL queries to clearly delineate each step. This script executes four key operations, summarized as follows:

4.1.1. Collecting charts and chart updates

The charts used in this paper were provided by the Norwegian Hydrographic Service as multiple chart files scattered across a folder hierarchy. The first step in the script collects all S-57 related files found in the folder hierarchy and stores them in a single folder. This enables GDAL to apply update files to each chart, which happens automatically if they are stored in the same folder.

4.1.2. Convert charts to spatiaLite database

The next step converts the spatial features in the S-57 format to the SpatiaLite format by running the ogr2ogr tool in GDAL (Warmerdam and Rouault, 2021) on each ENC, and writing the results to a single portable database. This allows single-file access to larger and more detailed datasets than the S-57 format allows.

The S-57 format classifies ENCs according to their intended use and detail levels, and by including charts from only the same class, overlapping geometries are avoided. Throughout this paper, charts classified for approach (INTU = 4) are utilized, as it provides the highest detail level which covers all of the Norwegian costal areas.

4.1.3. Create valid SpatiaLite tables

The S-57 format allows multiple types of geometries within a single ontological category, while the spatial columns in SpatiaLite can only contain a single geometry. After GDAL has merged all the charts into a single SpatiaLite database, each table, representing the ontological categories of the ENCs, is further divided into separate tables according to geometry. The R*-tree spatial indexing can then be enabled to speed up queries.

4.1.4. Preprocessing for SQL queries

The tables *navigable* and *innavigable* are then created to represent χ_{obs} and χ_{free}. Navigable areas are defined according to the ontological category for depth areas (DEPARE), and further restricted according to depth values that are greater than a specified level (DRVAL1 > min depth). Innavigable areas are based primarily on the ontological category for land area (LNDARE), along with several other obstacles such as pontoons, buoys, and awash rocks.

A buffer zone around all objects in the *innavigable* table is introduced, with the S-57 defined category zone of confidence (CATZOC) levels as a guideline for its size. To ensure that this does not cause χ_{obs} and χ_{free} to overlap, corresponding steps are taken to remove areas from the *navigable* table.

An optional final step to further optimize distance cost calculations involves converting the database tables from the EPSG4326 coordinates used in the S-57 format to a suitable UTM projection, such as EPSG32632 (UTM32N). The conversion replaces the more computationally expensive distance calculations of latitude and longitude pairs with simpler Euclidean distance calculations.

4.2. Electronic navigational chart database

To verify the presented approach, a spatial database was created from S-57 ENCs covering parts of Norway's Trøndelag region, as shown in Fig. 2. Following the preprocessing steps described above, buffers of 4 m surrounding point and Linestring features were selected, as well as 5 m buffers surrounding land features. It should be noted that the selected buffer sizes are generally too small to guarantee safe navigation according to the CATZOC definitions of the charts and should therefore only be applied for simulation purposes. This process produced a 225 MB database file, which incorporates spatial indexing and encompasses a navigable region of 14347.8 km^2 as well as an innavigable region of 34470.7 km^2.

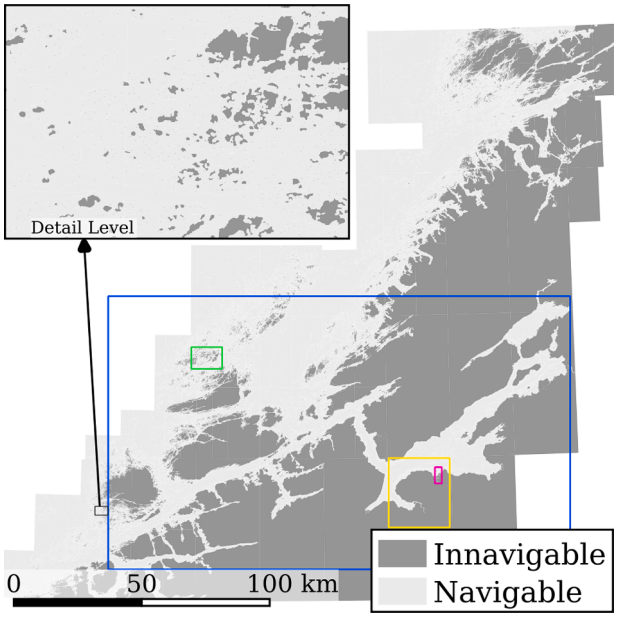


Fig. 2. Overview of the 225 MB spatial database extracted from S-57 datasets covering the Trøndelag region of Norway (S-57 detail level: DSID_INTU = 4). Green, blue, magenta, yellow : Sampling areas for planning scenarios 1-4 respectively. Red: location of the enlarged section, indicative of the level of spatial detail contained in the database.

4.3. Integrating the database application and OMPL

Integration of the database application with OMPL (Sucan et al., 2012) utilizes the SQLite3 C/C++ interface, which upon initialization loads the SpatiaLite extension that enables spatial queries. Starting with the `SampleFree` procedure, OMPL provides a standardized interface for a state validity checker through a C++-lambda function that can be set while configuring the planner. To determine if the sample is in a navigable area, the database application uses a custom lambda function to extract coordinates from OMPL's internal state representation and execute a precompiled SQL query.

The interface for the `CollisionFree` procedure in the OMPL environment is provided through the `MotionValidator` class object, which is designed as a superclass to be inherited from by the user defined procedure. By default, OMPL provides a collision checker that checks discrete states at regular intervals between the two states that are to be connected by the planner, as shown in Fig. 3(a). While this approach provides a simple solution by reusing the same state validity checker as the `SampleFree` procedure, it introduces a sampling resolution that is unnecessary for a vector chart environment. From Fig. 3(a), it is also apparent that a sufficiently small obstacle can "hide" between the checked point geometries, causing a collision to be missed, i.e. an obstacle defined in a line geometry. Instead, the superior option is to check for collisions with a line or polygonal geometry that covers the straight path between two states, as shown in Fig. 3(b). This approach may also be more computationally efficient as it can be implemented in a single database query.

4.4. Database queries

The database queries for the sample validity and motion collision checks are given in SQL query 1 and SQL query 2, both adhering to the database optimizations mentioned above. The queries follow a similar structure and return what can be considered as Boolean results, where no data means no intersecting geometries, and any data signifies an intersection (collision). The actual data are irrelevant and can be set to any value, such as the `SELECT 1` used in the queries.

Table 1
Input parameters for planning problem definition in the Neptus GUI.

Name	Description
Planning Timeout	The maximum time before terminating planning.
Planner	The SBMP algorithm used.
Initial Position	Defined in the EPSG4326 coordinate system.
Goal Position	Defined in the EPSG4326 coordinate system.
Sampling Area	A rectangular planning area.
Speed	The desired speed along the generated path.
Speed Units	An enumerated value (m/s, %, RPM etc.).

SQL Query 1: Sample validity check query. The coordinates `(?1, ?2)` represents the sample to check for validity.

```
1 SELECT 1 FROM navigable WHERE ROWID IN (
2   SELECT ROWID FROM SpatialIndex
3   WHERE f_table_name = 'navigable' AND
4     search_frame = BuildMbr(?1,?2,?1,?2)
5 ) AND ST_Intersects(
6   MakePoint(?1,?2, 32632),
7   navigable_geometry
8 ) LIMIT 1;
```

SQL Query 2: Motion validity query. The coordinates `(?1, ?2)` and `(?3, ?4)` represent the two samples that define the motion to be checked for validity

```
1 SELECT 1 FROM innavigable WHERE ROWID IN (
2     SELECT ROWID FROM SpatialIndex
3     WHERE f_table_name = 'innavigable' AND
4       search_frame = BuildMbr(?1,?2,?3,?4)
5 ) AND ST_Intersects(
6   MakeLine(
7     MakePoint(?1,?2, 32632),
8     MakePoint(?3,?4, 32632)
9   ),
10  innavigable_geometry
11 ) LIMIT 1;
```

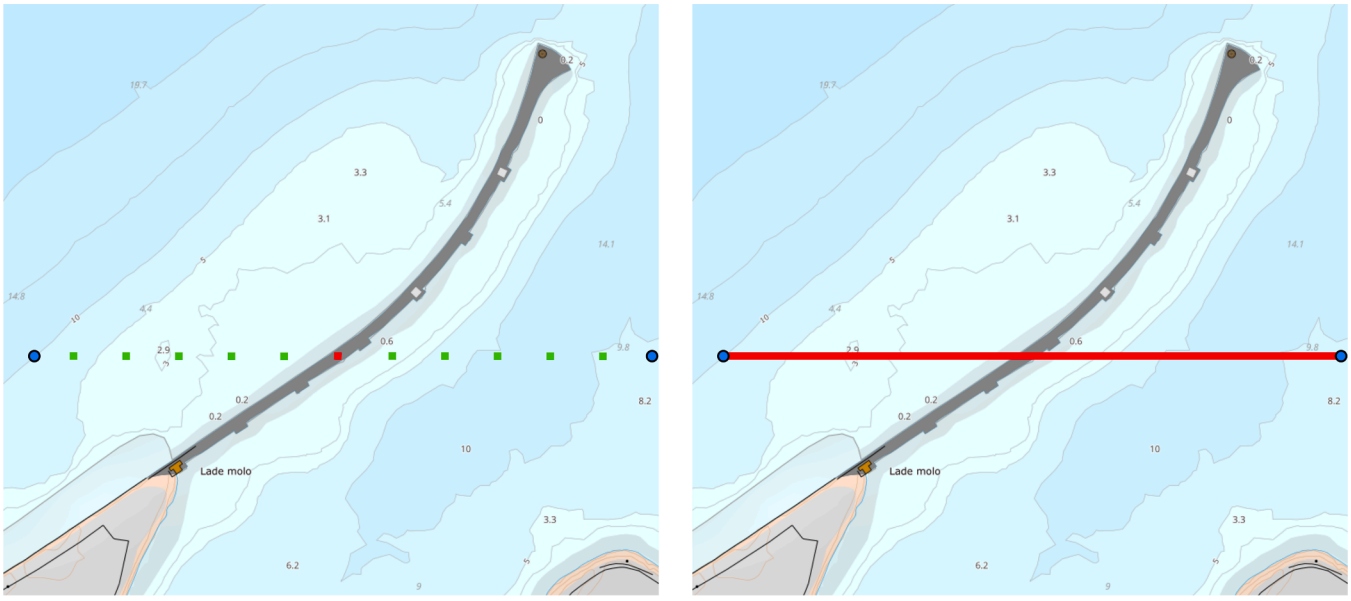
The order of execution is important for both queries, and the SQLite query optimizer executes the subquery of lines 2-4 first. This uses a minimum bounding rectangle, specified by the `BuildMbr` function, to select only geometries that are in the same area as the geometry representing a sample or motion. Subsequently, the two `WHERE` clauses are applied, with the indexed `ROWID` taking precedence, while the `ST_Intersects` spatial predicate of line 5 is applied afterward. This query structure is suggested by the Spatialite documentation to fully profit from the spatial index (SpatiaLite Development Team, 2025). The last line of both queries ensures that the query ends at the first detected obstacle to avoid any further computation.

The question marks in the queries are SQLite's syntax for variable binding in precompiled queries and represent EPSG32632 Northing and Easting coordinates in this context.

4.5. Application interface

The motion planner was integrated with the LSTS toolchain to facilitate its utilization in unmanned maritime vehicle systems. The custom graphical user interface (GUI), allows users to define a planning problem by providing the information outlined in Table 1. This problem definition is converted to the toolchain's communication format[1] and transmitted to the DUNE middleware, where the OMPL integration is applied to generate a safe path. Upon completion of the planning process, the generated path is translated into the system's `GoTo` motion primitive and transmitted to the vehicle's guidance, navigation, and control system

[1] The inter-module communication protocol (IMC) (Martins et al., 2009)



(a) Multiple point geometries (Default OMPL)

(b) One LineString geometry (Proposed solution)

Fig. 3. Different collision checking methods. Each geometry is checked for collisions using separate database queries. Blue points: samples to be connected, red point: collision detected, green points: collision free. Red line: LineString geometry. (Basemap: Kartverket).

to initiate mission execution. The Supplementary Materials provide a demonstration of how to run the implemented planner from the Neptus GUI, covering problem definition and result visualization.

5. Simulated results

The SpatiaLite integration to OMPL was assessed through a simulation study that looked at number of successful runs within a predetermined planning time, the average length of the produced solutions and the average planning time needed before a solution was provided. The study included four planning scenarios derived from selected regions of the ENC database (Fig. 2). Ten optimization planners were selected based on their availability in OMPL and the results of a preliminary series of tests. Detailed descriptions of each planner are given in their respective papers: FMT* (Janson et al., 2015), kBIT* (Gammell et al., 2015), kABIT* (Strub and Gammell, 2020b), RRT* (Karaman and Frazzoli, 2011), RRT# (Otte and Frazzoli, 2015), AIT* (Strub and Gammell, 2020a), InformedRRT* (Gammell et al., 2018), TRRT (Jaillet et al., 2010), LBTRR (Salzman and Halperin, 2016), and RRTXstatic (Otte and Frazzoli, 2015).

5.1. Simulated planning scenarios

The four simulated scenarios were chosen from the charts database to represent a diverse range of maritime environments. The selected areas and planning scenarios can be characterized as follows:

1. **Archipelago:** The Froan archipelago consists of extensive shallows and sounds among hundreds of smaller and larger islands, representing a severely cluttered navigational environment (Fig. 4).
2. **Fjord/Coast transit:** A long-distance planning scenario that starts at the inner parts of the Trondheim Fjord and includes a transit from the fjord toward the coast (Fig. 5).
3. **Harbor/River inlet:** The planning area includes part of the meandering Nidelven river that passes through Trondheim harbor and features narrow passages, jetties, floating docks and buoys (Fig. 6).

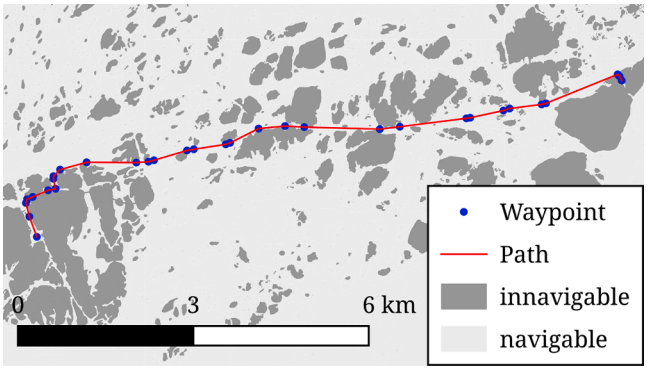


Fig. 4. Planning scenario 1: Archipelago (Navigable area 84.6 km^2, innavigable area 32.6 km^2, valid sample ratio 72.20 %).

4. **Shoreline:** The path around the Byneset peninsula represents a longer transit than the previous scenarios but contains considerably more navigable space (Fig. 7).

The exact coordinates for the initial and goal positions of each scenario are given in Table A.6, and the extent of the sampled areas of each scenario is stated in Table A.5.

5.2. Benchmark setup

The simulations were executed on a Dell Precision Mobile Workstation 5550 with an Intel® Core™ i7-10875H and 32GB memory running on the 64-bit Ubuntu 20.04 Linux distribution.

All four planning problems were benchmarked using ten optimizing planners from OMPL, with the maximum planning run time purposely set relatively low in order to reveal performance differences between the planning algorithms. For each planner, 300 runs were executed per planning problem, for a total of 12,000 runs. The benchmarks were facilitated by the OMPL benchmarking tools (Moll et al., 2015).

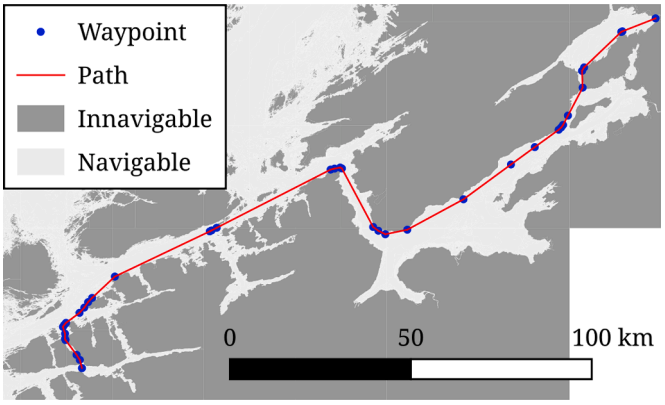


Fig. 5. Planning scenario 2: Fjord/Coast transit (Navigable area 6683.6 km^2, innavigable area 21017.6 km^2, valid sample ratio 24.13 %).

Table 2
Number of successful solutions found after 300 runs per planner.

Planning scenario	1	2	3	4	
Planning time[s]	180	90	60	20	Total [%]
FMT*	276	299	299	300	97.83
kBIT*	238	300	298	300	94.67
kABIT*	195	298	290	298	90.08
RRT*	138	0	42	281	38.42
RRT#	47	0	13	241	25.08
AIT*	24	37	257	300	51.5
InformedRRT*	153	0	58	278	40.75
TRRT	91	164	158	285	58.17
LBTRRT	194	256	201	298	79.08
RRTXstatic	27	0	12	244	23.36

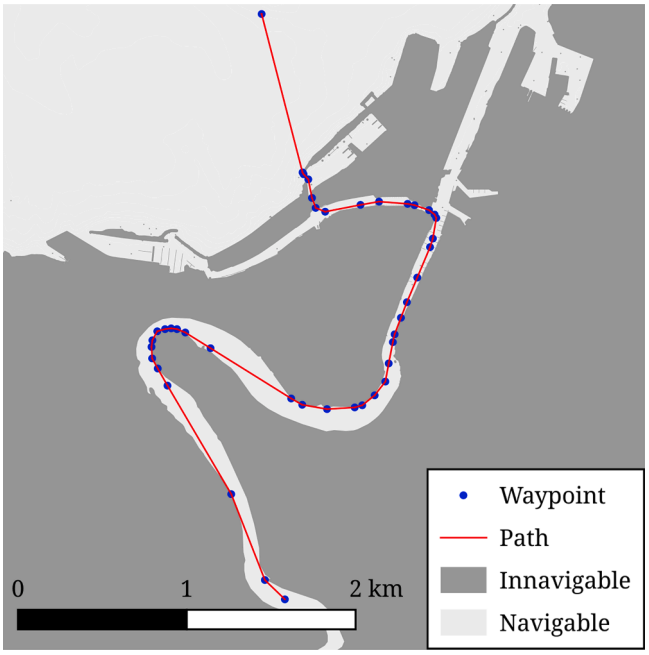


Fig. 6. Planning scenario 3: Harbor/River inlet (Navigable area 8.4 km^2, innavigable area 17.7 km^2, valid sample ratio 32.00 %).

5.3. Benchmark results

Path-planners can be evaluated according to several performance metrics, and this paper considers three different metrics: the number of successful planning runs (Table 2), the average solution length for each planner (Table 3), and the average time before delivering a valid solution (Table 4). **Note:** The results in Tables 3 and 4 were calculated using only the successful planning runs.

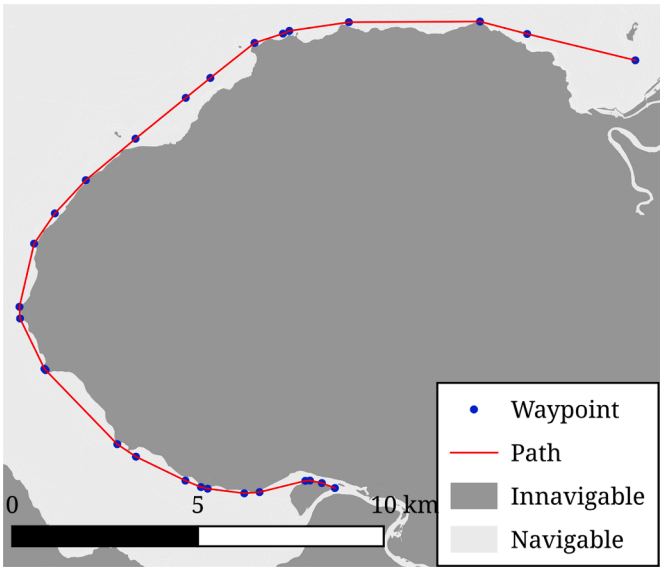


Fig. 7. Planning scenario 4: Shoreline (Navigable area 201.9 km^2, innavigable area 888.4 km^2, valid sample ratio 18.5 %).

Table 3
Average solution length [m] over 300 runs per planner (standard deviations included in Table B.7).

Planning scenario	1	2	3	4
Planning time[s]	180	90	60	20
FMT*	13334	250906	7691	31987
kBIT*	11728	222591	7075	31254
kABIT*	11845	223584	6964	31249
RRT*	11663	N/A	7443	31225
RRT#	11837	N/A	7457	31389
AIT*	12151	229557	7372	31254
InformedRRT*	11626	N/A	7448	31216
TRRT	17659	323605	9091	41001
LBTRRT	14477	261929	7926	38746
RRTXstatic	11813	N/A	7399	31408
Avg[m]	12813	252029	7587	33073

Table 4
Average time to first solution [s] over 300 runs per planner (standard deviations included in Table B.8).

Planning scenario	1	2	3	4
Planning time [s]	180	90	60	20
FMT*	82.9	24.3	16.3	3.9
kBIT*	75.6	20.4	15.0	2.1
kABIT*	75.3	27.0	15.5	2.6
RRT*	120.2	N/A	47.8	6.9
RRT#	103.6	N/A	50.4	7.6
AIT*	105.7	48.6	27.4	1.6
InformedRRT*	113.0	N/A	48.6	6.3
TRRT	39.5	30.5	25.3	3.1
LBTRRT	68.7	43.1	34.1	3.5
RRTXstatic	111.2	N/A	43.4	7.6

6. Discussion

The sample paths for the benchmarked scenarios shown in Figs. 4–7 were found by the *k*-nearest version of the batch-informed trees (kBIT*) algorithm (Gammell et al., 2015). These paths demonstrate that the

SpatiaLite integration with OMPL can provide feasible and safe paths in a maritime environment represented by commonly available ENCs.

Due to the inherent randomness of SBMPs, each planning run yields different results, necessitating a statistical performance evaluation. Their ability to consistently provide valid solutions within the allotted time was therefore empirically demonstrated through the success rate across 300 runs per scenario (Table 2). The results reveal that the FMT* and kBIT* algorithms provides the highest success rates in the four planning scenarios, with FMT* only significantly outperforming kBIT* in the complex archipelago scenario (Fig. 4). The challenging nature of this scenario can be attributed to a dense distribution of valid sample regions combined with a high likelihood of collision, as the environment contains numerous small obstacles that frequently trigger the motion validity checker. Further analysis and comparisons of the average distance of collision-checked motion segments may explain why planners like InformedRRT* perform above average in the archipelago scenario while failing in the fjord/coast transit scenario.

Considering the quality of the solution, Table 3 shows that on average the kBIT* algorithm finds the shortest path lengths, only slightly surpassed by the kABIT variant in scenarios three and four. The poorer performance of FMT* can be attributed to its one-pass nature, which terminates planning after the first feasible solution is found. In contrast, anytime planners like kBIT* continue refining the solution after an initial path is found, and will consequently increase the solution with regards to the optimizing heuristic with improved database throughput.

The shoreline scenario (Fig. 7) turns out to be the simplest planning problem presented, as evident from the success rates shown in Table 2 and the planning times in Table 4, despite being the second largest area and having the lowest probability of drawing valid samples. This outcome can be explained by the lack of clutter between the start and end points of the planning task, which allows a larger proportion of samples to be connected in the internal search data structures.

Several techniques are being explored to achieve faster convergence towards feasible and optimal paths for SBMPs. *Lazy collision checking* (Bohlin and Kavraki, 2000) involves delaying the computationally expensive `CollisionFree` procedure until a locally optimal path between samples has been found. This is increasingly beneficial as the cost or amount of collision checks increase, such as the high rejection rates in the cluttered archipelago example or when the spatial databases are large. The three planners that exhibit the highest success rates all incorporate lazy collision checking, highlighting its benefit when integrated with a spatial database to represent the planner's configuration space (Table 2).

Beyond parameter tuning, alternatives to the uniform sampling technique with rejection sampling employed in the presented solution may provide performance improvements in specific scenarios. The harbor/river inlet scenario could, for instance, benefit from using the bridge test (Hsu et al., 2003) or other informed sampling techniques to bias it toward narrow passages. A performance comparison to the triangulated method of Enevoldsen et al. (2022) would also be interesting for further work.

A benefit of utilizing the polygonal representation of the environment is that it allows for sampling from a continuous set of states in the configuration space without requiring explicit definition of all considered states. This provides an infinite number of states that are implicitly represented by the spatial region, providing a dense sampling domain. It also allows for discovering paths through narrow passages in the configuration space that may be missed with fixed-resolution sampling.

The benchmarks of motion planners evaluate not only the algorithms *per se* but also their implementations and any intermediate data structures. Libraries like OMPL mitigate some of these effects by providing a unified framework for implementing and testing motion planning algorithms. Performance differences may also arise from the specific configurations and parameters of each algorithm. Although fine-tuning the parameters according to the properties of the problem domain and environment could influence effectiveness, it was considered outside the scope of this work.

The integration of the path planner and spatial database into the LSTS toolchain and DUNE enables it to supply safe paths for unmanned vehicles based on this platform (Pinto et al., 2013). As an example, a variation of the kBIT* algorithm has been deployed and tested on a system of ASVs that performs robotic search and tracking of underwater acoustic transmitters (Lauvås et al., 2022) (see Supplementary Materials for videos demonstrating its use). The solution supports the system by generating collision-free full-coverage cooperative search paths with a greedy planner (Lauvås and Alfredsen, 2023), and is also an essential component of the vehicle formation controller that optimizes vehicle geometry during the localization and tracking of acoustic transmitters. In the latter case, each vehicle executes the planner at 5 s intervals to maintain safe navigation toward its last assigned position (see video in the supplementary materials for a single vehicle example). For this planning scenario, the sampling area and distance between initial and goal states are significantly smaller than the presented scenarios (below 2000 m), and consequently have shorter planning times than any shown in Table 4.

Although the presented solution is limited to static obstacles, the SDBMS enables dynamic environment representation by facilitating the continuous addition, modification, and removal of obstacle geometries. This capability, similar to the elliptical COLREGS representations employed in Enevoldsen et al. (2022), could accommodate other vehicles and uncharted objects derived from sources such as AIS and onboard sensors. Additionally, this would require that the planner is executed at regular intervals to account for the changing environment, which may not be viable in long-range planning scenarios. A common approach to solve such uses is to have two planning stages, where the first step is a global planner with static obstacles, and the second stage is a local planner with static and dynamic obstacles. The first step establishes a long-horizon path once, while the vehicle follows the paths of a second stage, which accounts for dynamic obstacles by running at regular intervals with start and destination points selected from the long-horizon plan.

Since the vehicle is considered a single-point geometry in the current case, the generated paths impose no limitation on movement. This may be acceptable for small and highly maneuverable vehicles, but other approaches will have to be implemented for larger ships. Dubins paths (Fossen et al., 2015) and real vehicle geometries can be introduced to accommodate larger and less maneuverable vehicles, which is supported by several planners in OMPL.

Over larger distances, the tide and sea state along the path may change. While long-term accurate prediction of sea state remains challenging, tidal levels are generally predictable and can be accounted for when generating the path database by adjusting the buffer area surrounding obstacles or through a more restrictive minimum depth. The chart database maintains the depth range of the navigable polygons to support this feature as future extensions to the presented implementation.

The R*-tree spatial index organizes geometries into a hierarchical structure composed of axis-aligned minimum bounding rectangles (MBRs), with the individual geometries as leaf nodes. A notable consequence of this design is that collision checks, which use an MBR where the collision-checking geometry forms the diagonal, produce rectangles of increasing size as the direction of the collision-checking geometry diverges from the axes. In an area of nearly homogeneous complexity, a larger MBR will therefore encompass more geometries and cause reduced efficiency. If there is prior knowledge about the predominant direction of planning and collision checks, such as the general azimuth of the planning area, it may be advantageous to rotate the axes of the R*-tree to align with these anticipated directions, thereby enhancing the efficiency of the spatial index.

Another strategy for optimizing the performance of collision checks is to decompose large polygons. This reduces the size of the MBRs used

by the spatial index, enabling it to filter out irrelevant geometries more efficiently before the computationally expensive `ST_Intersects` predicate is applied. Further investigation is needed to evaluate the real-world performance improvements from polygonal decomposition and rotating the R*-tree axes to determine whether the additional effort is justified. Neither of these optimizations was implemented in the proof-of-concept presented in this paper.

7. Conclusion

This paper has demonstrated the feasibility and benefits of using spatial databases to represent the environment in sampling-based path planning algorithms (SBMPs), utilizing spatial features from electronic navigational charts to develop an effective maritime path planner. The work makes four key contributions to the field: First, it establishes that spatial database systems effectively provide the geometric query and collision detection procedures required for environment representation in SBMPs, offering benefits such as simplified vector data management and efficient spatial operations through R*-tree indexing. Second, it introduces a platform-agnostic framework that enhances flexibility in motion planning applications across different domains. Third, it delivers and validates a practical C++ implementation combining SpatiaLite and OMPL that processes navigational features (including land areas, depth areas, and isolated dangers) from standard electronic navigational charts issued by national hydrographic offices. Fourth, it extends the utility of the framework by successfully integrating it with the LSTS toolchain for autonomous vehicles, demonstrating practical applicability.

The approach was validated through 12,000 benchmark runs across four representative maritime scenarios. The results demonstrate that the system successfully generates safe paths, with batch informed tree and fast marching tree algorithms achieving the best overall performance. The implementation relies exclusively on open source tools, which enable free replication and modification, with the presented implementation available in the Supplementary Materials.

CRediT authorship contribution statement

Nikolai Lauvås: Writing – original draft, Validation, Project administration, Investigation; **Tor Arne Johansen:** Writing – review & editing; **Jo Arve Alfredsen:** Writing – review & editing, Supervision.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

The electronic navigational charts used in this work were provided by the Norwegian Mapping Authority, Hydrographic Service. They are reproduced under license No. 30/072020/1. The maps included in this paper were created using the free and open-source QGIS software (QGIS Development Team, 2021).

Supplementary material

Supplementary material associated with this article can be found in the online version at 10.1016/j.oceaneng.2025.122345

Appendix A. Exact Coordinates for Benchmark Runs

Table A.5
Search boundaries (UTM32N) for the benchmark runs.

	Easting		Northing	
#	Min	Max	Min	Max
1	473221	485284	7076546	7085058
2	440732	621256	6998055	7104997
3	568399	571101	7031678	7038044
4	550328	574171	7014449	7041627

Table A.6
Start and goal locations (UTM32N) for the benchmark runs.

	Easting		Northing	
#	Start	Goal	Start	Goal
1	483567	473557	7082426	7079827
2	620781	466425	7100264	7003109
3	569142	569354	7035964	7032506
4	569142	561314	7035964	7024321

Appendix B. Standard Deviations for Benchmark Runs

Table B.7
Standard deviation for Tab. 3 (average solution length [m] over 300 runs per planner).

Planning Scenario	1	2	3	4
FMT*	544.9	22301.5	157.9	380.5
kBIT*	136.0	539.9	449.0	57.8
kABIT*	181.1	4817.8	474.1	70.4
RRT*	72.1	N/A	39.5	87.8
RRT#	143.6	N/A	29.5	149.5
AIT*	243.2	7552.3	267.8	70.8
InformedRRT*	68.0	N/A	34.5	74.9
TRRT	2349.8	50039.3	572.7	2637.5
LBTRRT	618.0	10158.0	522.5	1681.6
RRTXstatic	115.3	N/A	253.7	138.9

Table B.8
Standard deviation for Tab. 4 (average time to first solution [s] over 300 runs per planner).

Planning Scenario	1	2	3	4
FMT*	43.1	12.8	9.3	1.9
kBIT*	41.2	10.4	9.1	2.3
kABIT*	43.5	14.9	10.2	3.0
RRT*	39.6	N/A	9.2	5.1
RRT#	42.2	N/A	8.3	5.5
AIT*	36.4	22.3	13.1	2.1
InformedRRT*	38.4	N/A	8.5	4.8
TRRT	40.3	34.4	13.3	3.8
LBTRRT	50.1	17.9	11.5	3.2
RRTXstatic	36.8	N/A	8.1	5.2

Appendix C. Literature Review Table

Table C.9 provides a concise overview of the advantages and disadvantages of the papers included in the literature of this paper.

Table C.9
Literature review overview

Method	Advantages	Disadvantages
Occupancy Grids (Moravec and Elfes, 1985)	- Intuitive representation - Encodes sensor uncertainty effectively - Easily updated with real-time sensor data	- Uniform resolution may oversimplify complex environments - Rapidly increasing computational an storage cost required to represent large or detailed environments
Quadtrees (Finkel and Bentley, 1974)	- Variable resolution adapts to environment complexity - Suitable for large maritime environments	- Complex tree structure increases computational complexity (e.g for updates) - Requires preprocessing for polygonal data
Quadtrees with Visibility Graph (Shah and Gupta, 2020)	- Visibility graphs improves A* performance - Applicable to any polygonal environment	- Additional preprocessing overhead - Requires maintaining both quadtree and visibility graph (e.g for updates)
ENC based Voronoi Diagrams (Candeloro et al., 2017)	- Generates paths from obstacle vertices - Yen-Dijkstra optimizes shortest paths - Computationally efficient	- Limited to polygonal environments - Requires post-processing for path refinement
Semantic Map (PostGIS) (Deeken et al., 2018)	- Supports 3D planning with spatial reasoning - Compatible with ROS for robotics - Flexible for semantic data integration	- Indirect path-planning through generated occupancy grid - Computationally intensive for real-time use
Rasterized ENC Boundaries (SQLite) (Dallolio et al., 2022)	- Avoids full rasterization - High performance with indexed database - Suitable for anti-grounding	- Higher storage requirements than vector format - Reduced chart data availability for other uses
Custom Hydrographic Information System (Blindheim and Johansen, 2022)	- Tailored for maritime ENC data access - Supports diverse hydrographic applications	- Custom framework may lack portability - Less wider used than standard spatial databases
RRT* with ENC Sampling (Enevoldsen et al., 2022)	- Integrates COLREGs for maritime compliance - Uniform sampling in triangulated space - Short-horizon planning for ECDIS	- Complex transformation of ENC data - Limited to short-horizon planning
Direct Polygonal Planning on ENC features in spatial database (Proposed in this paper)	- No rasterization or preprocessing of polygons required - Utilizes efficient spatial indexing - Facilitates data reuse for other purposes (e.g., target search) - Utilize widely available open-source components	- Spatial indexes adds data redundancy - Spatial database adds complexity - Current implementation lacks support for dynamic obstacles and COLREGs compliance - Sampling-based path planners provide no real-time guarantees

References

Blindheim, S., Johansen, T.A., 2022. Electronic navigational charts for visualization, simulation, and autonomous ship control. IEEE Access 10, 3716–3737. https://doi.org/10.1109/ACCESS.2021.3139767

Bohlin, R., Kavraki, L.E., 2000. Path planning using lazy PRM. In: Proceedings 2000 IEEE International Conference on Robotics and Automation. Vol. 1, pp. 521–528. https://doi.org/10.1109/ROBOT.2000.844107

Candeloro, M., Lekkas, A.M., Sørensen, A.J., 2017. A Voronoi-diagram-based dynamic path-planning system for underactuated marine vessels. Control Eng. Pract. 61, 41–54. https://doi.org/10.1016/j.conengprac.2017.01.007

Dallolio, A., Bergh, T.K., De La Torre, P.R., Overaas, H., Johansen, T.A., et al., 2022. ENC-based anti-grounding and anti-collision system for a wave-propelled USV. In: OCEANS 2022 - Chennai. IEEE, Chennai, India. https://doi.org/10.1109/OCEANSChennai45887.2022.9775262

Deeken, H., Wiemann, T., Hertzberg, J., 2018. Grounding semantic maps in spatial databases. Robot. Autonom. Syst. 105, 146–165. https://doi.org/10.1016/j.robot.2018.03.011

Dijkstra, E.W., 1959. A note on two problems in connexion with graphs. Numer. Math. 1 (1), 269–271. https://doi.org/10.1007/BF01386390

Elbanhawi, M., Simic, M., 2014. Sampling-based robot motion planning: a review. In: IEEE Access. Vol. 2, pp. 56–77. https://doi.org/10.1109/ACCESS.2014.2302442

Enevoldsen, T.T., Blanke, M., Galeazzi, R., et al., 2022. Sampling-based collision and grounding avoidance for marine crafts. Ocean Eng. 261. https://doi.org/10.1016/j.oceaneng.2022.112078

Finkel, R.A., Bentley, J.L., 1974. Quad trees a data structure for retrieval on composite keys. Acta Inform. 4 (1). https://doi.org/10.1007/BF00288933

Fossen, T.I., Pettersen, K.Y., Galeazzi, R., et al., 2015. Line-of-sight path following for dubins paths with adaptive sideslip compensation of drift forces. In: IEEE Transactions on Control Systems Technology. Vol. 23, pp. 820–827. https://doi.org/10.1109/TCST.2014.2338354

Gammell, J.D., Barfoot, T.D., Srinivasa, S.S., et al., 2018. Informed sampling for asymptotically optimal path planning. IEEE Trans. Robot. 34 (4), 966–984. https://doi.org/10.1109/TRO.2018.2830331

Gammell, J.D., Srinivasa, S.S., Barfoot, T.D., et al., 2015. Batch informed trees (BIT*): sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In: 2015 IEEE International Conference on Robotics and Automation (ICRA), pp. 3067–3074. https://doi.org/10.1109/ICRA.2015.7139620

Gammell, J.D., Strub, M.P., 2021. Asymptotically optimal sampling-based motion planning methods. Ann. Rev. Control Robot. Auton. Syst. 4 (Volume 4, 2021), 295–318. https://doi.org/10.1146/annurev-control-061920-093753

Hart, P.E., Nilsson, N.J., Raphael, B., 1968. A formal basis for the heuristic determination of minimum cost paths. IEEE Trans. Syst. Sci. Cybern. 4 (2), 100–107. https://doi.org/10.1109/TSSC.1968.300136

Herring, J.R. (Ed.), 2011a. Simple Feature Access - Part 1: Common Architecture. Inc., Open Geospatial Consortium. 1.2.1 edition.

Herring, J.R. (Ed.), 2011b. Simple Feature Access - Part 2: SQL Option. 1.2.1 ed., Inc., Open Geospatial Consortium.

Hsu, D., Jiang, T., Reif, J., Sun, Z., 2003. The bridge test for sampling narrow passages with probabilistic roadmap planners. In: 2003 IEEE International Conference on Robotics and Automation (Cat. No. 03CH37422). Vol. 3, pp. 4420–4426. https://doi.org/10.1109/ROBOT.2003.1242285

International Hydrographic Organization, 2000a. IHO transfer standard for digital hydrographic data.

International Hydrographic Organization, 2000b. S-57 Appendix A IHO object catalogue.

Jaillet, L., Cortés, J., Siméon, T., et al., 2010. Sampling-based path planning on configuration-space costmaps. IEEE Trans. Robot. 26 (4), 635–646. https://doi.org/10.1109/TRO.2010.2049527

Janson, L., Schmerling, E., Clark, A., Pavone, M., et al., 2015. Fast marching tree: a fast marching sampling-based method for optimal motion planning in many dimensions. Int. J. Robot. Res. 34 (7), 883–921. https://doi.org/10.1177/0278364915577958

Karaman, S., Frazzoli, E., 2011. Sampling-based algorithms for optimal motion planning. Int. J. Robot. Res. 30 (7), 846–894. https://doi.org/10.1177/0278364911406761

Kavraki, L.E., Svestka, P., Latombe, J., Overmars, M.H., et al., 1996. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. In: IEEE Transactions on Robotics and Automation. Vol. 12, pp. 566–580. https://doi.org/10.1109/70.508439

Lauvås, N., Alfredsen, J.A., 2023. A configurable greedy planner for collaborative robotic search in acoustic fish telemetry surveys. In: OCEANS 2023 - Limerick. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/OCEANSLimerick52467.2023.10244511

Lauvås, N., Urke, H.A., Alfredsen, J.A., 2022. Design and validation of a system of autonomous fish tracking vehicles. In: OCEANS 2022 - Hampton Roads. Institute of Electrical and Electronics Engineers (IEEE). https://doi.org/10.1109/OCEANS47191.2022.9977384

LaValle, S.M., Kuffner, J.J., 2001. Randomized kinodynamic planning. Int. J. Robot. Res. 20 (5), 378–400. https://doi.org/10.1177/02783640122067453

Martins, R., Dias, P.S., Marques, E.R.B., Pinto, J., Sousa, J.B., Pereira, F.L., 2009. IMC: a communication protocol for networked vehicles and sensors. In: OCEANS 2009-EUROPE. https://doi.org/10.1109/OCEANSE.2009.5278245

Moll, M., Şucan, I.A., Kavraki, L.E., 2015. Benchmarking motion planning algorithms: an extensible infrastructure for analysis and visualization. IEEE Robot. Autom. Mag. 22 (3), 96–102. https://doi.org/10.1109/MRA.2015.2448276

Moravec, H., Elfes, A., 1985. High resolution maps from wide angle sonar. In: 1985 IEEE International Conference on Robotics and Automation Proceedings. Vol. 2, pp. 116–121. https://doi.org/10.1109/ROBOT.1985.1087316

Otte, M., Frazzoli, E., 2015. Rrtx: real-time motion planning/replanning for environments with unpredictable obstacles. In: Algorithmic Foundations of Robotics XI. Springer International Publishing, Cham, pp. 461–478. https://doi.org/10.1007/978-3-319-16595-0_27

Pinto, J., Dias, P.S., Martins, R., Fortuna, J., Marques, E.R.B., de Sousa, J.B., 2013. The LSTS toolchain for networked vehicle systems. 2013 MTS/IEEE OCEANS - Bergen. https://doi.org/10.1109/OCEANS-Bergen.2013.6608148

QGIS Development Team, 2021. QGIS Geographic Information System. Open Source Geospatial Foundation. Accessed: 2025-01-22. http://qgis.org

Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A., 2009. ROS: an open-source robot operating system. ICRA Workshop on Open Source Robotics. 3 (3.2).

Salzman, O., Halperin, D., 2016. Asymptotically near-optimal RRT for fast, high-quality motion planning. IEEE Trans. Robot. 32 (3), 473–483. https://doi.org/10.1109/TRO.2016.2539377

Shah, B.C., Gupta, S.K., 2020. Long-distance path planning for unmanned surface vehicles in complex marine environment. IEEE J. Ocean. Eng. 45 (3), 813–830. https://doi.org/10.1109/JOE.2019.2909508

SpatiaLite Development Team, 2025. SpatiaLite Documentation. Accessed: 2025-01-22. https://www.gaia-gis.it/fossil/libspatialite/index

Strub, M.P., Gammell, J.D., 2020a. Adaptively informed trees (AIT*): fast asymptotically optimal path planning through adaptive heuristics. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 3191–3198. https://doi.org/10.1109/ICRA40945.2020.9197338

Strub, M.P., Gammell, J.D., 2020b. Advanced BIT* (ABIT*): sampling-based planning with advanced graph-search techniques. In: 2020 IEEE International Conference on Robotics and Automation (ICRA), pp. 130–136. https://doi.org/10.1109/ICRA40945.2020.9196580

Sucan, I., Moll, M., Kavraki, E.E., 2012. The open motion planning library. Robot. Autom. Mag. 19, 72–82. https://doi.org/10.1109/MRA.2012.2205651

Warmerdam, F., Rouault, E., 2021. GDAL documentation. Accessed: 2025-01-22. https://gdal.org/gdal.pdf