Defence Research and Development Canada
Recherche et développement pour la défense Canada

An Interpolation System for Position Report Databases

Marie-Odette St-Hilaire
Dan Radulescu
Tim Hammond
Eric Lefebvre
OODA Technologies Inc.

Prepared by:

OODA Technologies Inc.
4891 Av. Grosvenor, Montreal Qc, H3W 2M2

Project Manager: Anthony W. Isenor
Contract Number: Call-up 10, No. 4500862723 to W7707-115137
Contract Scientific Authority: Tim Hammond

Defence R&D Canada – Atlantic

Contract Report
DRDC Atlantic CR 2013-137
October 2013

An Interpolation System for Position Report Databases

Marie-Odette St-Hilaire
Dan Radulescu
Tim Hammond
Eric Lefebvre
OODA Technologies Inc.

Prepared by:

OODA Technologies Inc.
4891 Av. Grosvenor, Montreal Qc, H3W 2M2

Project Manager: Anthony W. Isenor
Contract Number: Call-up 10, No. 4500862723 to W7707-115137
Contract Scientific Authority: Tim Hammond

Defence R&D Canada – Atlantic

Contract Report

DRDC Atlantic CR 2013-137

October 2013

Abstract

The aim of this work is to increase the capabilities of position report databases, like Defence Research and Development Canada (DRDC)'s Maritime Situational Awareness Research Infrastructure (MSARI) or the Global Position Warehouse (GPW). For this purpose, a position report is a record of identity, location and time that is reported by a particular vehicle, usually a ship, in order to indicate its own position at a particular moment. The Automatic Identification System (AIS) provides most of the position reports in the databases cited above, but Vessel Monitoring Systems (VMS) and Long Range Identification and Tracking (LRIT) are also prominent sources. Position report databases already allow for a wide range of queries based on characteristics of the reports themselves. DRDC Atlantic is interested in providing the ability to construct and manage *interpolation queries* that rely on inference about what happened between position reports. The goal of this contract was to construct fundamental infrastructure, data structures and algorithms that are required for such query-building. The long term goal of this work is to make it easy for operators with little database training to construct *interpolation queries*. In support of these goals, an Interpolation System was developed to convert AIS position reports into *tracks*, which are then stored in an Interpolation Database. These *tracks* are so constructed as to avoid crossing land obstacles that are defined using polygons. This obstacle avoidance capability is the major novelty of the work.

Résumé

Les travaux visent à rehausser les fonctions des bases de données des comptes-rendus de position, comme linfrastructure de recherche en connaissance de la situation maritime (MSARI) ou lentrepôt de données du système mondial de localisation (GPW) de Recherche et développement pour la défense Canada (RDDC). Aux fins de ces travaux, on entend par compte-rendu de position tout enregistrement de lidentité, de lemplacement et de la date et lheure transmis par un véhicule particulier (le plus souvent un navire) afin dindiquer son emplacement à un instant précis. Le Système d'identification automatique (SIA) fournit lessentiel des comptes-rendus de position versés dans les bases de données mentionnées ci-dessus, doublé dautres sources comme le Système de surveillance des navires (SSN) et le programme didentification et de suivi à distance des navires (LRIT). Les bases de données des comptes-rendus de position prennent déjà en charge bien des types dinterrogations, en fonction des caractéristiques de ces comptes-rendus. RDDC Atlantique cherche à permettre de créer et gérer des requêtes par interpolation, qui seraient fondées sur des inférences sur ce qui sest passé entre les comptes-rendus de position. Le contrat visait à établir linfrastructure, les structures de données et les algorithmes de base nécessaires pour créer de telles requêtes. À long terme, les travaux visent à permettre même à des utilisateurs sans formation poussée sur les bases de données de créer des requêtes par interpolation. Cest pourquoi nous avons développé un système d'interpolation qui convertit les comptes-rendus de po-

sition en « pistes », puis les enregistre dans un base de données dinterpolation. Ces pistes sont créées de façon à contourner des obstacles (îles et masses continentales) définis par polygones. Cette fonction de contournement est la principale innovation des travaux décrits.

Executive summary

An Interpolation System for Position Report Databases

Marie-Odette St-Hilaire, Dan Radulescu, Tim Hammond, Eric Lefebvre; DRDC Atlantic CR 2013-137; Defence R&D Canada – Atlantic; October 2013.

Background: DRDC Atlantic is interested in improving the capabilities of position report databases, like its own Maritime Situational Awareness Research Infrastructure (MSARI). For this purpose, a position report is a record of identity, location and time that is reported by a particular vehicle, usually a ship in this context, in order to indicate its own position at a particular moment. The Automatic Identification System (AIS) provides most of the position reports in the MSARI database, but that database is designed to hold other types, like Vessel Monitoring Systems (VMS) or Long Range Identification and Tracking (LRIT) reports as well. DRDC believes the desired improvement will come from providing the ability to construct and manage *interpolation queries*. These queries, which are named and characterized here for the first time, rely on inference about what happened between position reports. They allow the user to account for vessel motion in continuous time, not just when there are reports. The goal of this paper is to describe some fundamental infrastructure, data structures and algorithms that are required for *interpolation query* building.

Results: A software application known as the Interpolation System (IS) was constructed under contract by OODA Technologies. The IS interpolates between position reports according to a collection of rules called the Interpolation Method (IM) and stores results in an Interpolation Database (IDB). The IM rules expect the IS to be able to find obstacle-avoiding paths. In other words, when the straight line connecting two position reports runs over land, the IS is expected to be able to find the shortest connecting route that avoids the land. Naturally, this requirement assumes that the IS is also capable of recognizing collisions with land in the first place. The prototype IS produced in this contract had all these capabilities and the IDB it supports was designed with a view to future work. In particular, the IDB was designed to support an Interpolation Query Application (IQA), still to be created, which would facilitate the construction of *interpolation queries*. The multithreaded design of the IS allowed for the processing over 16 million position reports in less than 40 minutes.

Significance: The operations centres charged with Maritime Situational Awareness (MSA) may be interested in this work because it suggests they can extract more value from the position report databases, like the Global Position Warehouse (GPW). By employing the sort of pre-processing demonstrated here on real AIS data with the IS, they will be able to answer a whole host of *interpolation queries* that were never thought to be possible before. For example, they would be able to determine how long a ship spent in a particular area

or its closest point of approach to a sensitive area. The processing performance of the IS is promising because it makes interpolation that considers obstacle avoidance a realistic option for the typical AIS data throughput experienced at DRDC (over 16 million reports per day). The IS also allows for a distributed hardware architecture, which is an advantage in a context where maritime data increase yearly.

Future Plans: After further refinement of the IS, the intent is to build an Interpolation Query Application (IQA) to construct and manage interpolation queries based on the *tracks* produced by the IS. This system would facilitate interpolation queries of position report databases and ultimately make it easy for operators with little database training to construct such queries.

Sommaire

An Interpolation System for Position Report Databases

Marie-Odette St-Hilaire, Dan Radulescu, Tim Hammond, Eric Lefebvre ; DRDC Atlantic CR 2013-137 ; R & D pour la défense Canada – Atlantique ; octobre 2013

Contexte : DRDC Atlantique cherche à rehausser les fonctions des bases de données des comptes-rendus de position, comme sa propre infrastructure de recherche en connaissance de la situation maritime (MSARI). Aux fins de ces travaux, on entend par compte-rendu de position tout enregistrement de lidentité, de lemplacement et de la date et lheure transmis par un véhicule précis (dans le présent contexte, le plus souvent un navire) afin dindiquer son emplacement à un instant précis. Même si le Système d'identification automatique (SIA) fournit lessentiel des comptes-rendus de position versés dans cette base de données, elle peut aussi prendre en charge dautres données, comme celles du Système de surveillance des navires (SSN) et du Programme didentification et de suivi à distance des navires (LRIT). RDDC estime que les améliorations voulues découleront de la possibilité de créer et de gérer des *requêtes par interpolation*. Ces requêtes, ici désignées et décrites pour la première fois, sont fondées sur des inférences sur ce qui sest passé entre les comptes-rendus de position. Lutilisateur peut, à laide de ces requêtes, tenir compte des déplacements des navires en continu plutôt qu'uniquement leur emplacement à lenvoi de comptes-rendus de position. Le présent document vise à décrire linfrastructure, les structures de données et les algorithmes de base nécessaires pour créer des *requêtes par interpolation*.

Resultats : Une application nommée le Système dinterpolation (SI) a été développée par OODA Technologies en vertu dun contrat. Le SI fait linterpolation entre les comptes-rendus de position à laide dun ensemble de règles nommée Méthode dinterpolation (MI) puis en enregistre les résultats dans une base de donnée dinterpolation (BDI). Selon les règles de la MI, le SI devrait reconstruire des « pistes » qui évitent les obstacles. En dautres termes, si la droite qui relie les emplacements de deux comptes-rendus de position traverse la terre ferme, le SI devrait pouvoir trouver le chemin le plus court qui relie ces deux points en contournant cet obstacle. Cette exigence présuppose, évidemment, que le SI peut reconnaître ces collisions avec la terre. Le prototype de SI créé dans le cadre de ce contrat intégrait toutes ces fonctions, et la BDI quil prend en charge a été conçue en tenant compte des autres travaux prévus. La BDI, en particulier, a été conçue de façon à prendre en charge une application de requêtes par interpolation, lARI, qui reste à développer ; elle vise à simplifier la création de requêtes par interpolation. Grâce à son architecture multifil, le SI a pu traiter plus de 16 millions de comptes-rendus de position en moins de 40 minutes.

Portée : Les centres des opérations responsables de la connaissance de la situation maritime (CSM) et la Garde côtière canadienne seront vraisemblablement intéressés par ces tra-

vaux, car ils laissent penser quon peut extraire plus de renseignements des bases de données des comptes-rendus de position, comme lentrepôt de données du système mondial de localisation (GPW), que ce qu'elles contiennent. Par des techniques comme le prétraitement démontré ici par le SI, mais appliqué sur des données réelles du SIA, ils pourront utiliser leurs bases de données existantes pour répondre à une vaste gamme de questions quon croyait impossibles à répondre. Quelles questions ? Par exemple, combien de temps un navire a passé dans une région donnée, ou encore à quel point il a été le plus près dune zone dintérêt. Les performances du SI en traitement des données sont prometteuses, car on peut réellement envisager de créer des interpolations qui tiennent compte de la terre ferme à partir de SIA avec un rendement semblable à celui observé à RDDC, cest-à-dire plus de 16 millions de comptes-rendus par jour. Le SI prend aussi en charge une architecture matérielle distribuée, ce qui constitue un avantage dans un contexte où les données maritimes augmentent chaque année.

Perspectives : Après avoir amélioré le SI, nous voulons développer une application de requêtes par interpolation (ARI), qui permettra de créer et de gérer les requêtes fondées sur les « pistes » reconstruites par le SI. Ce système viserait à simplifier la création de requêtes par interpolation afin d'interroger les bases de données de comptes-rendus de position et, au bout du compte, de permettre même à des utilisateurs sans formation poussée sur les bases de données de créer des requêtes par interpolation.

Table of contents

List of figures

List of tables

This page intentionally left blank.

1 Introduction

This paper suggests a strategy for extracting more value from position report databases, like the Global Position Warehouse (GPW), which stores position reports as well as other types of data. For this purpose, a position report is a record of identity, location and time that is reported by a particular vehicle, usually a ship in this context, in order to indicate its own position at a particular moment. The Automatic Identification System (AIS) is currently the most prominent source of position reports in Maritime Situational Awareness (MSA), but other sources, like Vessel Monitoring System (VMS) or Long Range Identification and Tracking (LRIT) are also in widespread use.

Position report databases, like DRDC's Maritime Situational Awareness Research Infrastructure (MSARI), already allow for a wide range of queries based on characteristics of the AIS position reports. DRDC Atlantic would like to expand on these capabilities by providing the ability to construct and manage *interpolation queries* that rely on inference about what happened between position reports. Given a polygonal area A on the surface of the ocean, examples of *interpolation queries* include the following:

1. Provide a list of ships by name that crossed A yesterday.
2. Provide the names and positions of all the ships that were in A at a specific time t.
3. How many ships traversed A yesterday?
4. How long did a specific ship s spend in A last week?

Some may think that the *interpolation queries* above are existing capabilities of position report databases. We concede that existing databases can go part of the way towards answering these questions because they can typically search for all the position reports in A. If the spacing between position reports is small relative to the size of A, then this will indeed provide an acceptable approximation to the desired answer. In general, however, the spacing between position reports is highly variable. With the Automatic Identification System (AIS) data in MSARI, for example, the time between position reports varies from seconds to days and even longer. Thus, it would not be surprising if the spatial separation between position reports was large relative to A. In that case, many ships could traverse A without making a single position report inside that region, and thus the approximation above would be very misleading. To answer the question properly, we must make some form of inference about what happened between reports.

The queries above all depend in some way on the *tracks* of the ships in question. In this paper, the term *track* represents the trajectory of the ship: it provides an account of vessel position in continuous time. Thus, the aim of interpolation is to convert position reports, of the sort provided by AIS, into *tracks*. The idea is that then these *tracks* can be used to answer the questions above. The conversion from reports to tracks requires some level of modeling.

The simplest model for how ships might move between position reports is, of course, a straight line. When the reports are not widely separated, ships may indeed follow lines of contant bearing, or rhumb lines, between waypoints. On a spherical globe, however, the shortest distance between two points is determined by the great circle route between them. Still greater realism is introduced by regarding the earth as an oblate spheroid and computing geodesic arcs from this model. Most comercial vessels are steered along such routes, at least approximately, by sailing between computer-generated waypoints. Ships must also avoid obstacles, especially the land. Thus, a rhumb line connection between two position reports will not typically provide the best prediction of vessel trajectory, when that line runs over the land. The current work is predicated on the hypothesis that it is possible to recognize when lines run over land. Moreover, when lines do so, the shortest land-avoiding route should provide a better interpolation than does the straight line. If these hypothese are true, then interpolation should be done with land-avoidance in mind. That is exactly what this project tried to do.

According to the Statement of Work (SOW) underlying this work [1], reports were to be turned in *tracks* according to rules known collectively as the Interpolation Method (IM). These rules consider successive AIS reports from the same ship. They are defined in terms of two temporal parameters: t_{min}, t_{max} (both in minutes) and a speed v_{max} in knots. v_{max} is taken to be a speed that the ship cannot exceed. The rules are as follows:

1. Whenever the temporal separation (in minutes) between two successive AIS position reports from a given ship is less than or equal to t_{min}, the IM assumes that the ship sailed straight between those reports at a constant speed v_c
2. v_c is calculated as the spatial separation divided by the temporal separation (expressed in knots).
3. If the calculated speed v_c exceeds v_{max}, then the ship status is *indeterminate* over the time interval.
4. Whenever the temporal separation (in minutes) between two successive position reports from a given ship is greater than or equal to t_{max}, the IM considers the ship status to be *indeterminate* over the time interval.
5. If the temporal separation is between t_{min} and t_{max}, the ship is taken to have sailed at constant speed v_c by the shortest route that avoids land.
6. If, however, that constant speed v_c along the shortest land-avoiding route exceeds v_{max}, then the ship status is *indeterminate* over the time interval.

Thus, in applying the IM rules, every pair of position reports in the AIS database will either be connected by a *track*, or the ship's status will be considered *indeterminate* over the respective time interval. In other words, *interpolation queries* will have to be designed to deal with both possibilities.

The goal of this project was to construct some fundamental infrastructure, data structures and algorithms that would be useful to the construction of *interpolation queries*. The long term goal of this work is to make it easy for operators with little database training to construct them.

To fulfill this goal, an IS was developed to convert all the AIS position reports in a database like MSARI into *tracks* according to the rules of the IM. These *tracks* are then stored in a separate database called the IDB.

The initial objectives of this work also included the development of an Interpolation Query Application (IQA) to construct and manage *interpolation queries* based on *tracks* constructed by the IS. A small prototype of the IQA was built and demonstrated but requirements evolved during the project to focus on the IS. These requirements include the implementation of an obstacle avoidance algorithm and the development of optimization strategies to keep pace with the MSARI's high data inflow (16 million reports per day). In other words, the IS had to be able to process AIS reports significantly faster than they were coming into MSARI.

This document summarizes all the technical activities and achievements of the Call-Up 10 against contract W7707-115137, with the exception of the IQA prototype.

It includes:
- A high level description of the IS architecture, including main components description, information flow and multithreading strategies (Section 2).
- A description of the IS including details of all decision making algorithms. These main components are: interaction with MSARI (Section 3), interpolation (Section 4), obstacle avoidance (Section 5) and route generation (Section 6).
- A description of the IDB and its data model (Section 7).
- A description of performance experimentation and results (Section 8).

We have also inserted a final section:
- A view toward the future and suggested enhancements (Section 9);

and several annexes:
- Initial stakeholders' requirements (Annex A)
- A description of the grid-based approach used to speed up detection of whether reports are on land(Annex B).
- The installation steps for a Windows environment (Annex C).
- Guidelines for running the IS and the shapefile printer (Annex D).

2 High Level Design

This section presents the high level description of the IS architecture.

2.1 Components

This section identifies the software units that make up the IS. Each software unit groups functionalities from the same process. Software units, with the exception of the shapefile printer, are linked to requirements from the original statement of work (namely I.1, I.2, F.1, F.2 and IM.1 to IM.9, see Annexe A).

The software units are presented in Figure 1 and the following sections briefly describe them.

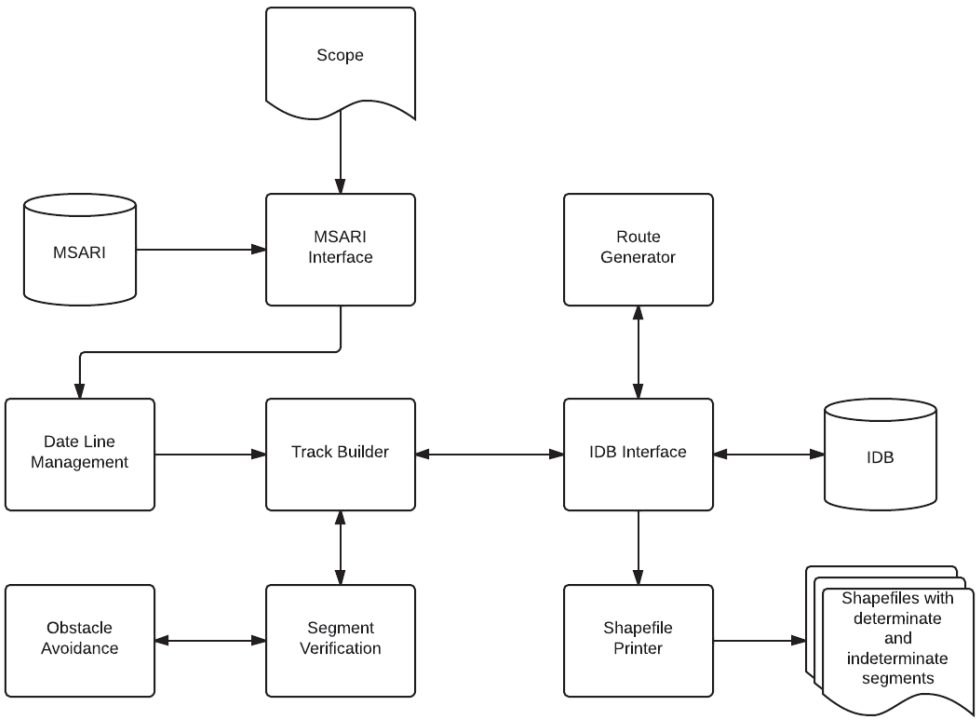

Figure 1: Main functional components of the IS. The route generator and shapefile printer and are not invoked during the interpolation by the IS. The route generator pre-processes charts (shapefiles) to facilitate shortest path construction, and the shapefile printer allows a user to view the *tracks* produced by the IS as a shapefile.

2.1.1 MSARI interface

The MSARI interface is responsible for retrieving AIS data from MSARI satisfying the following conditions:

1. non-null position,
2. within the scope (time and space),
3. position not out-of-range (in [-90,90],[-180,180]),
4. vessel type,
5. valid Maritime Mobile Service Identity (MMSI) (not MMSI = 0 or 999 for instance),
6. not on land,
7. unique timestamp for each MMSI and
8. more than one report for each MMSI.

The implementation relies on Java Database Connectivity (JDBC) and involves a temporary Structured Query Language (SQL) table and an SQL function to speed up the operation. The functionalities of this component are detailed in Section 3.

2.1.2 Track Builder

The Track Builder creates *tracks* from the AIS position reports. If a *track* already exists for a particular ship in the IDB, the Track Builder refreshes the *track* with new reports from MSARI. The Track Builder is described in Section 4.1.

This component is implemented in Java and relies on the Java Topology Suite (JTS) for the geometric representation of *tracks*, segments and reports.

2.1.3 Date Line Management

This component makes sure that reports lying on opposite sides of the date line are connected by the shortest route, instead of by a segment traversing the globe. It is necessary because the JTS, on which much of the IS was based, represents ship positions on a two dimensional map of the earth instead of representing them on a globe. Since the dateline traverses the Arctic Ocean as well as the Pacific, this presents a practical as well as just a theoretical problem. See Section 4.2 for the details. This component was developed also with Java and the JTS.

2.1.4 Segment Verification

A segment is here understood to be a straight line (in two dimensions) connecting two points. The segment verification software unit identifies *indeterminate* segments according to the IM and makes sure that all segments are geo-feasible (i.e. speed is feasible and there is no collision with land). The decision making process is presented in Section 4.3.

The implementation relies on the JTS but also requires coast cells stored in the IDB to speed up collision detection.

2.1.5 IDB and IDB Interface

The IDB interface is responsible for interacting with the IDB in the following ways:
- Fetching all tracks corresponding to an input MMSI list.
- Storing and updating tracks.
- Storing and fetching sailing routes (represented as a graph).
- Fetching coast cells that intersect with the scope's bounding box, augmented with a 2 degree buffer.

The implementation relies on JDBC and Hibernate Spatial [1].

The IDB is a GIS-enabled database hosted on a PostgreSQL Database Management System (DBMS) with PostGIS extensions. Its data model is described in Section 7.

2.1.6 Route Generator

This component is responsible for converting a given map into a graph (a collection of vertices and edges) that is useful in computing obstacle-avoiding paths. In this paper, a map consists of a number of polygons that delineate the boundary between land and water. The vertices of the coastline polygons become the vertices of the graph. The route generator then adds an edge to the graph between a pair of vertices whenever it is possible to sail straight between them without crossing land. Once every pair of vertices has been considered, graph construction is complete. Details can be found in Section 6. Graph construction only has to be performed once for a given map, so it is really a precursor to the interpolation process, not part of it.

This component is also implemented with Java and uses JTS and GeoTools [2].

2.1.7 Shapefile Printer

The shapefile printer was developed to visualize interpolation results. Interpolation can be performed without this component, and it can be used without the other IS components,

1. http://www.hibernatespatial.org/
2. http://www.geotools.org/

as long as there are *tracks* in the IDB. It creates three shapefiles (and associated metadata files):

– `Land.shp` : Polygons representing the world map used for the collision detection and obstacle avoidance.
– `DeterminateTracks.shp` : Lines representing all determinate segments contained in the IDB.
– `IndeterminateTracks.shp` : Lines representing all indeterminate segments contained in the IDB.

To visualize the IDB content, the user has to launch the Shapefile Printer application and load the three shapefiles with a Geographic Information System (GIS) application such as Quantum GIS [3]. See more details on execution in Annex D.

2.2 Information Flow

Interpolation starts when the user defines the scope and launches the IS application and finishes when the tracks are stored or updated in the database. The flow of information in and out of the IS is represented in Figure 2 and detailed below:

1. The scope is defined by editing the `ScopeConfig.csv` file. A bounding box and a time interval must be defined with the following format:
   `wLong;eLong;sLat;nLat;startTimestamp;endTimestamp`
   See Sections 3 and D.1.1 for details on the scope definition.
2. Once the application is launched, several types of information are retrieved from databases:
   (a) MSARI Database (DB): position reports corresponding to the scope
   (b) IDB: The graph of sailing routes (a graph is a collection of vertices and edges) produced by the Route Generator (see Section 6 for details on routes).
   (c) IDB: Coast cells lying in the bounding box of the scope. This bounding box is augmented with a buffer of 2 degrees on each side of the scope limits (see Annex B for details about coast cells).
3. The position reports data are converted to the IS's internal report data model. Duplicated position reports and MMSI numbers (these numbers are used in AIS messages to identify individual ships) with only one position report are filtered out (see also Section 3).
4. A hash map that links MMSI numbers to their corresponding time-ordered reports is built and stored internally.
5. The IDB is queried to get all *tracks* having the same MMSIs contained in the MMSI-reports hash map.

3. `http://www.qgis.org/`

6. These *tracks* are converted from their representation in the IDB to the IS's internal *track* java class representation.
7. The *tracks* are then stored in an internal IS datastructure that facilitates searching for *tracks* by MMSI number.
8. For each element of the MMSI-reports hash map:
   (a) A *track* is instantiated with the reports. If there is already a *track* associated with that MMSI in the IDB, the *track* is augmented with the new reports.
   (b) The *track* is inspected to see if any of the new reports cross the date line. If there is such a crossing, artifact reports are created to avoid issues caused by representing the world map in two dimensions (see Section 4.2 for details).
   (c) The interpolation method initially proposes linking consecutive reports with a straight line segment. All *track* objects within the IS are composed of such segments (in other words, a *track* is a piecewise linear manifold). The segments of which a *track* is composed can be *indeterminate* or they can cross the date line, cases that are handled seamlessly by the java code for the *track* class. Methods of this class can compute the ship's course and speed along a segment or get the ship's position at any given time. Segments can also be represented as a JTS Linestring.
   (d) Each proposed segment is verified following the rules of the IM, which are described in Section 4.3. Only segments linking new reports (which will be part of the MMSI-reports hash map) are verified. The coast cells are used at this step.
   (e) If the proposed segment has duration (temporal length) between t_{min} and t_{max} and crosses land, it is replaced by the shortest path connecting the two reports together (unless that shortest path must have been traversed at speeds in excess of v_{max}, in which case the segment is *indeterminate*). This step is referred as obstacle avoidance. The obstacle avoiding path will aways be composed of straight line segments. The graph computed by the Route Generator is used at this step. See Section 5 for details on that process.
9. Once a *track* is built from consecuting line segments of verified geo-feasibility, it is mapped to the IDB data model and stored or updated in IDB.

2.3 Multithreading

The IS application makes heavy use of multithreading, in recognition of the fact that *tracks* are independent of one another. Ten *tracks* can be built in parallel while sixteen other threads take care of storing the data in the database.

The parallel approach not only speeds up the entire process but is essential when dealing with large amounts of data. If *tracks* are not stored away as they are being created but kept

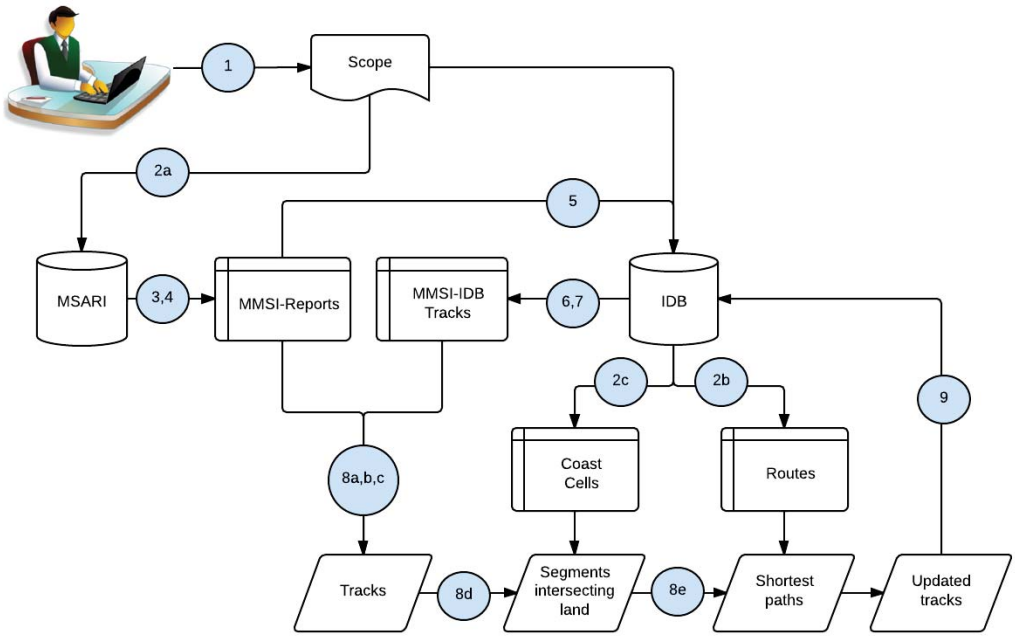

Figure 2: Flow of information in and out the IS. This schema does not include the route generation nor the shapefile printing, as they are not necessary to the interpolation. The identifiers within blue circles point to corresponding entries in the list of section 2.2 above.

in memory until there are no more *tracks* to process, there is a danger of running out of memory and crashing the application. This complication occurred early in development at a time when the entire data set of *tracks* was first created then transferred to the database for storage. Without the concurrent setup, it was impossible to handle an entire day of MSARI reports. A flow diagram is presented in Figure 3.

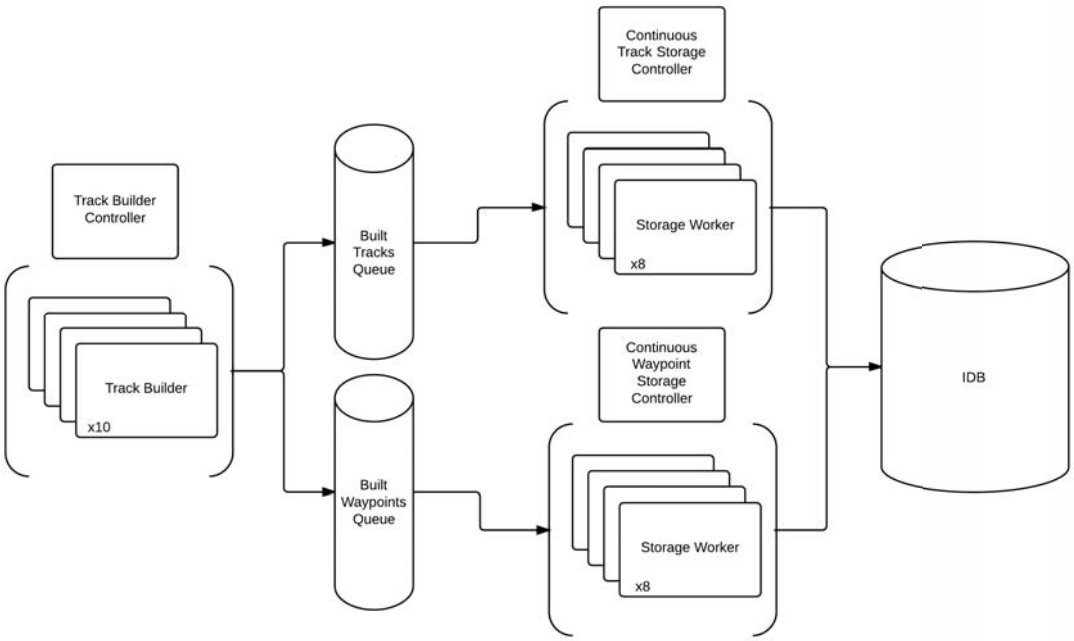

Figure 3: The figure shows the flow of *tracks* as they get created and are sent to storage. 10 Track Builders work concurrently to build *tracks* and waypoints. 8 worker threads for each object type then gather the data and store it into the database, as the data becomes available. Each worker thread processes a batch of 5 000 *tracks* or waypoints respectively. Note that although waypoints are intrinsically linked to their respective *track* objects, they can be stored asynchronously because there is no foreign key linking them, and the IDs that link them are already generated at the time of storage (see Section 7 for details on the interpolation data model.

3 MSARI Interaction

Reports are fetched automatically from MSARI based on the scope parameters. After one year of continuous storage, MSARI may contain over 6 Terabytes (TB) of MSA data (mainly AIS). Operating on such large amounts of data slows report fetching. To counter this effect, reports are retrieved in two steps, using a temporary table:

1. A temporary table is created containing only reports that are:
   (a) AIS,
   (b) with a non-null position,
   (c) within the scope (time and space),
   (d) position not out-of-range (in [-90,90],[-180,180]),
   (e) of type VESSEL (i.e. not AIRCRAFT, BASE STATION, UNKNOWN, AID-TO-NAVIGATION),
   (f) with a valid MMSI (not MMSI = 0 or 999 for instance).
2. The temporary table is queried to get only reports that are not on land. These reports are ordered by MMSI and timestamp.

Note that, if there are reports inside and outside the scope with the same MMSI, reports outside are ignored.

The scope's spatial and temporal components are logically linked by an *AND* (see SQL query below). Therefore, reports corresponding to the scope have a time stamp included in the time interval and are located inside the scope's bounding box. Also, the time interval can't include future dates, i.e. the upper time limit must be equal or smaller than the actual date-time. This restriction is imposed by the DBMS. The spatial component of the scope was not designed to overlap the dateline. For instance, if the longitude limits are set to $[170, -160]$, the MSARI DBMS will convert these to $[-160, 170]$ (swapping the eastern and western limits so that the former is less than the latter). To work around this restriction, it is necessary to use two scopes, processing reports in two batches.

The SQL query for the first operation is:

```
SELECT r.report_timestamp AS report_timestamp, r.mmsi AS mmsi,
r.position_geom AS position_geom INTO TEMPORARY subset FROM reports r
WHERE r.position_geom IS NOT NULL
AND (r.data_quality & (2|8|256) = 0)
AND ST_Intersects(position_geom,
ST_MakeEnvelope(:wLong, :sLat, :eLong, :nLat, 4326))
AND r.report_timestamp BETWEEN startTime AND endTime
```

```
AND r.data_type_id = 1
AND r.entity_type_id = 1;
```

and the SQL query for the second operation is

```
SELECT s.report_timestamp, s.mmsi, ST_Y(s.position_geom),
ST_X(s.position_geom)
FROM subset s WHERE NOT isOnLand(s.position_geom)
ORDER BY s.mmsi, s.report_timestamp;
```

The SQL function `isOnLand` is part of MSARI. See Annex B for details on this function.

Once reports are retrieved from MSARI, two additional filtering steps are performed:

1. For a given MMSI, reports with duplicate time stamps are filtered out. In other words, if two reports share the same MMSI and timestamp, one is ignored.
2. Only *track*s with more than one report are interpolated.

Note that reports are fetched regardless of the source. The consequence is that the resulting *tracks* can be made of reports produced by different sources, for example coastal AIS mixed with space-based AIS. In other words, we perform fusion at the source level by simply aggregating AIS reports with the same MMSI. The main advantage for that is that we don't end up with overlapping *tracks* that would have required track-level fusion during post-processing.

4 Interpolation

The interpolation method takes reports identified by a unique MMSI and produces a *track*, which is then stored in the IDB. Interpolation is performed by the following software components (see Figure 1): *Date Line Management*, *Track Builder*, *Segment Verification* and *Obstacle Avoidance* . These components are further described in this section, with the exception of the obstacle avoidance, which is presented in Section 5.

4.1 Track Builder

As mentioned above, the IS represents a *track* as a collection of line segments that are connected together in succession. A segment can be: *indeterminate* or *determinate*. An *indeterminate* segment is either not geo-feasible (speed must exceed v_{max}), connects two reports separated by a time gap greater than t_{max}, or intersects land in such a way that the shortest obstacle-avoiding route is not geo-feasible. The *track* java class provides the methods needed to compute the position, speed and course of the ship at any point in time. The speed and course over a given segment are based on the Vincenty's formula, which computes the great circle distance between two positions [2].

A segment is a straight line connecting two *waypoints*, where a *waypoint* is composed of a position in space and a timestamp. *Waypoints* are constructed primarily from AIS reports, but may also be created to either:

1. allow for date line crossing,
2. avoid obstacles.

In both cases, the *waypoints* are identified accordingly in the IDB.

Waypoints and segments are illustrated in Figure 4, where the following notation is introduced:

– The time gap between *waypoints* A and B is ΔT_{AB} or simply ΔT when *waypoints* are not specified.
– The great circle distance between *waypoints* A and B as D_{AB} or just D.
– The speed and course computed over the segment is V_{AB} and C_{AB} or V and C respectively, where $V_{AB} = D_{AB}/\Delta T_{AB}$

Once reports are fetched, filtered, grouped by MMSI and ordered in time, they are interpolated following the method illustrated in Figure 5. The first step is to instantiate a *track* object and attach the time-ordered reports to it. Secondly, a check is performed on every consecutive pair of reports to establish whether reports cross the date line. When there is such a crossing, interpolated reports are created using the method of Section 4.2. Then segments are created to link each report and verified according to the rules of the IM (see section 4.3). If there is a collision between a segment and land, the obstacle avoidance

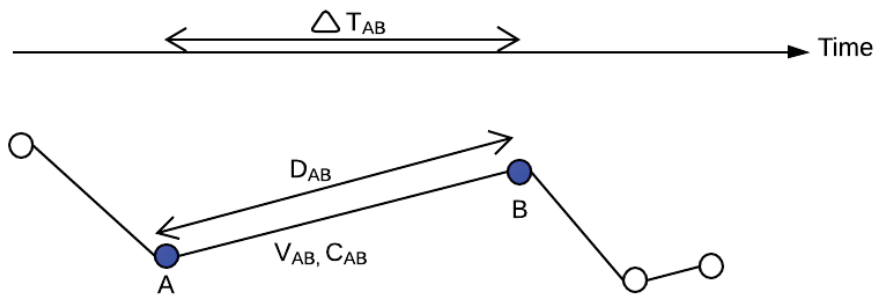

Figure 4: Interpolation model and notation.

algorithm is used to replace this segment with the shortest path that avoids land. The replacement path is always made up of straight line segments and is traversed at constant speed. The resulting *track* is then stored in the IDB.

4.2 Date line management

The IS was designed to use the JTS library for all the topological operations that might arise in constructing an *interpolation query*. These operations include the following Boolean methods that might be applied to a segment (with reference to various spatial objects, like lines and polygons): *is crossing*, *is within*, and *touches*. The *LineString* class from JTS was used to represent segments. Since JTS assumes that *LineString*s are two dimensional objects, there is no way to build a segment that crosses the date line. For instance, when JTS's *LineString* constructor is provided with the points $A = (lat_A, lon_A) = (32, -179)$ and $B = (lat_B, lon_B) = (34, 179)$, the line created will link these two points instead of linking $(32, -179)$ to $(33, -180)$ and $(33, 180)$ to $(34, 179)$ as it would appear on a sphere. The upper part of Figure 6 illustrates this example.

To get around this problem, an algorithm was implemented to manage date line crossing. This algorithm is used in both Interpolation and Route Generation.

In the case of interpolation, each pair of reports is inspected and additional dummy *way-points* are possibly added following the algorithm illustrated in Figure 7 and described in the following steps:

1. Each pair of reports (report A and B) are analyzed to decide if the path crosses the date line. It does if all of the following conditions hold:
   (a) Longitude sign of report A and report B are different (e.g. -179 degrees and 179 degrees).

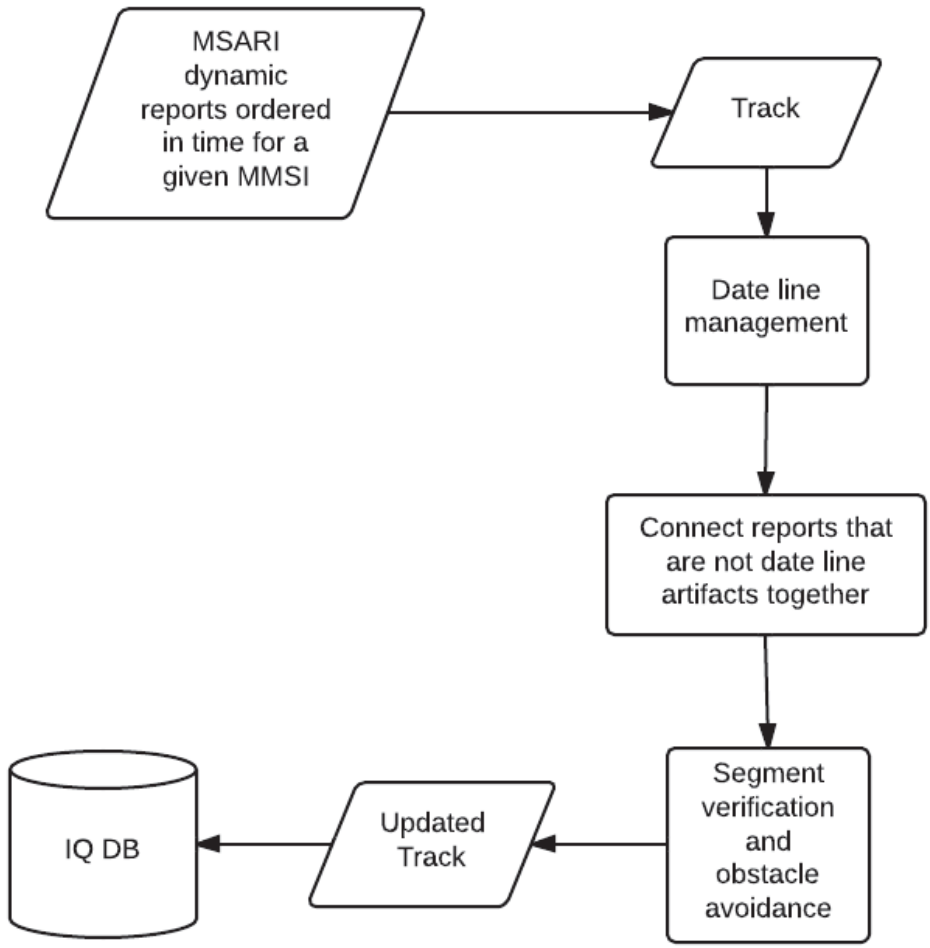

Figure 5: High level description of the interpolation method

(b) Cartesian distance is greater or equal to 180 degrees.

(c) Circular distance is less or equal to $v_{max}\Delta T_{AB}$.

(d) Both reports are not on the date line. But it is possible to have one report on it.

2. When the segment does cross the date line, we create dummy *waypoints* C and D on the East and West date lines. These dummy *waypoints* are used to force date line crossing and avoid having segments that nearly circumscribe the globe (as illustrated in Figure 6). If one report is on the date line, we create only C or D.

3. Assign timestamps t_C and t_D to *waypoints* C and D respectively as follows:
   – IF $t_A < t_B$ (track direction is East to West) THEN $t_D = t_A + \Delta t_{AD}$ AND $t_C = t_D + 1 milliseconds(\text{ms})$
   – IF $t_A > t_B$ (track direction is West to East) THEN $t_C = t_B + \Delta t_{BC}$ AND $t_D = t_C + 1\text{ms}$
   – Where $\Delta t_{AD} = \Delta t_{AB} D_{AD}/(D_{AD} + D_{CB})$ and $\Delta t_{BC} = \Delta t_{AB} D_{CB}/(D_{AD} + D_{CB})$

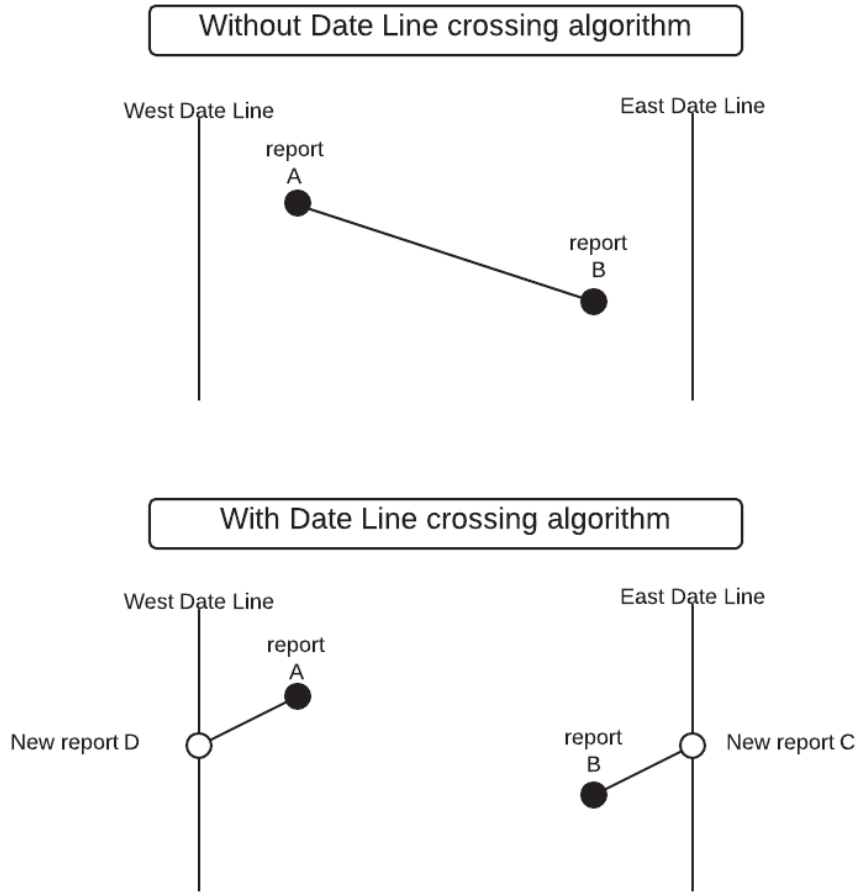

Figure 6: Date line crossing management. The upper illustration represents how segments are built using only JTS *LineString* functionality. The second illustration shows how it is managed by the IS.

4. Identify dummy *waypoints* C and D as interpolations for date line crossing. At the segment creation step (see the box labelled 'Connect reports that are not date line artifacts together' in Figure 5), no segment is built between reports identified as date line artifacts. See the bottom part of Figure 6 for an example of segments produced across the date line.

4.3 Segment verification

Once the track is created, each segment is verified following the approach described in Figure 8 (corresponding to the *Segment Verification* box in Figure 5). Segment verification involves the following steps:

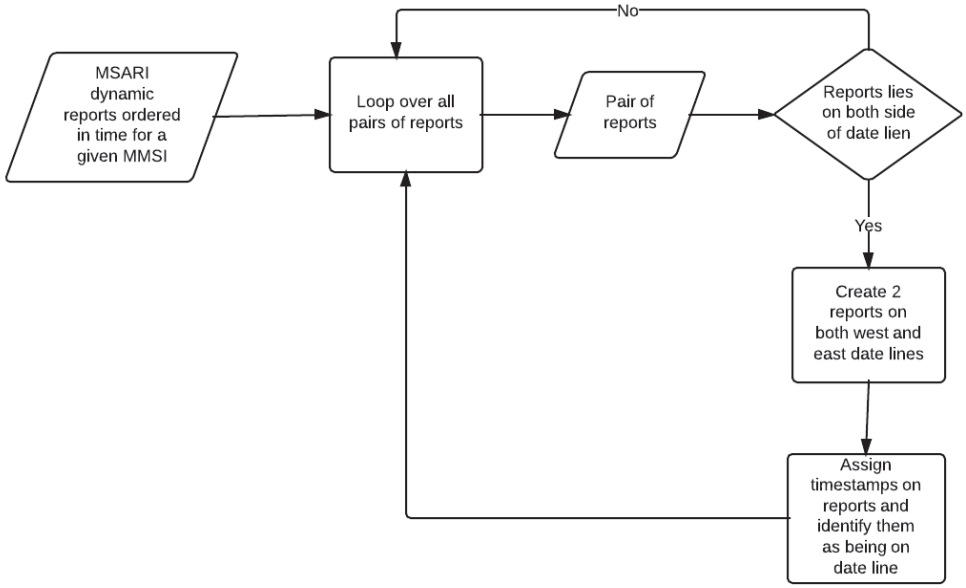

Figure 7: Date line crossing management algorithm involved in interpolation (see Figure 5) and in the Route Generation.

1. Verify if the segment's time gap is longer than or equal to t_{max}. If so, the status of the ship is considered *indeterminate* over the time interval. t_{max} was taken to be 6 hours for all types of ships. Based on 24 hours worth of AIS data from MSARI (see Section 8.1.1), about only 0.3% of all time gaps exceed 6 hours.
2. Verify if the segment's speed exceeds v_{max}. If it does, the status of the ship is considered *indeterminate* over the time segment. v_{max} was taken to be 20 knots. Note that this value could be adapted to the ship type or even the individual ship for greater precision.
3. Verify if the segment's time gap is shorter than or equal to t_{min}. If so, the IM assumes that the ship sailed straight between the waypoints. In other words, the segment is determinate and the IM assumes that it does not collide with land. Based on the 24 hours MSARI sub set, t_{min} was fixed to 5 minutes, the median interval between reports.
4. The next steps aim to decide whether or not the segment crosses land, in which case we have to perform obstacle avoidance. It is important to understand that both the collision verification and the obstacle avoidance are time and memory consuming operations. It pays to take steps that constrain these problems to a more manageable size.
5. The principal constraint is based on the *track envelope*, which is a bounding box that is just big enough to contain all the AIS reports that are associated with a given

track. The first step is find the set FCC of coastal cells[4] that are contained, at least partially, within the *track envelope* (see Section B for details about the grid based approach and the concept of coastal cells). FCC stands for *focus coastal cells*. If FCC is empty, the ship must have sailed in open water and the collision verification step can be skipped. We can limit the collision verification to just the coastal cells in FCC, instead of the complete set of 6240 coastal cells. Note that finding the set FCC is an operation that need only be performed once for the *track*. See Figure 9a for an illustration of *focus coastal cells*.

6. Determine the subset C of coastal cells in FCC that are touched by the segment. This is currently accomplished by checking all of the members of FCC in turn. If C is empty, we can assume that the ship sailed straight between waypoints. Figure 9b shows an example of intersecting coastal cells.
7. Verify if the segment crosses land in any of the coastal cells in C (see Figure 9c for an example). If so, the straight line trajectory between the two reports is judged to be inadequate. In that case, the segment has to be replaced by a collection of segments making up the shortest land-avoiding route R between the reports. See Section 5 for details about obstacle avoidance.
8. Verify if the route R can be travelled without exceeding v_{max}. In essence, check that $\sum_{i \leq N} D_{A_i B_i} \leq v_{max} \Delta T$, where N is the number of segments making up the route. If not, it means that the computed route is not geo-feasible and the initial segment is identified as *indeterminate*.
9. The resulting track is stored in IDB.

4. A coastal cell is a square region on a map that contains both land and water.

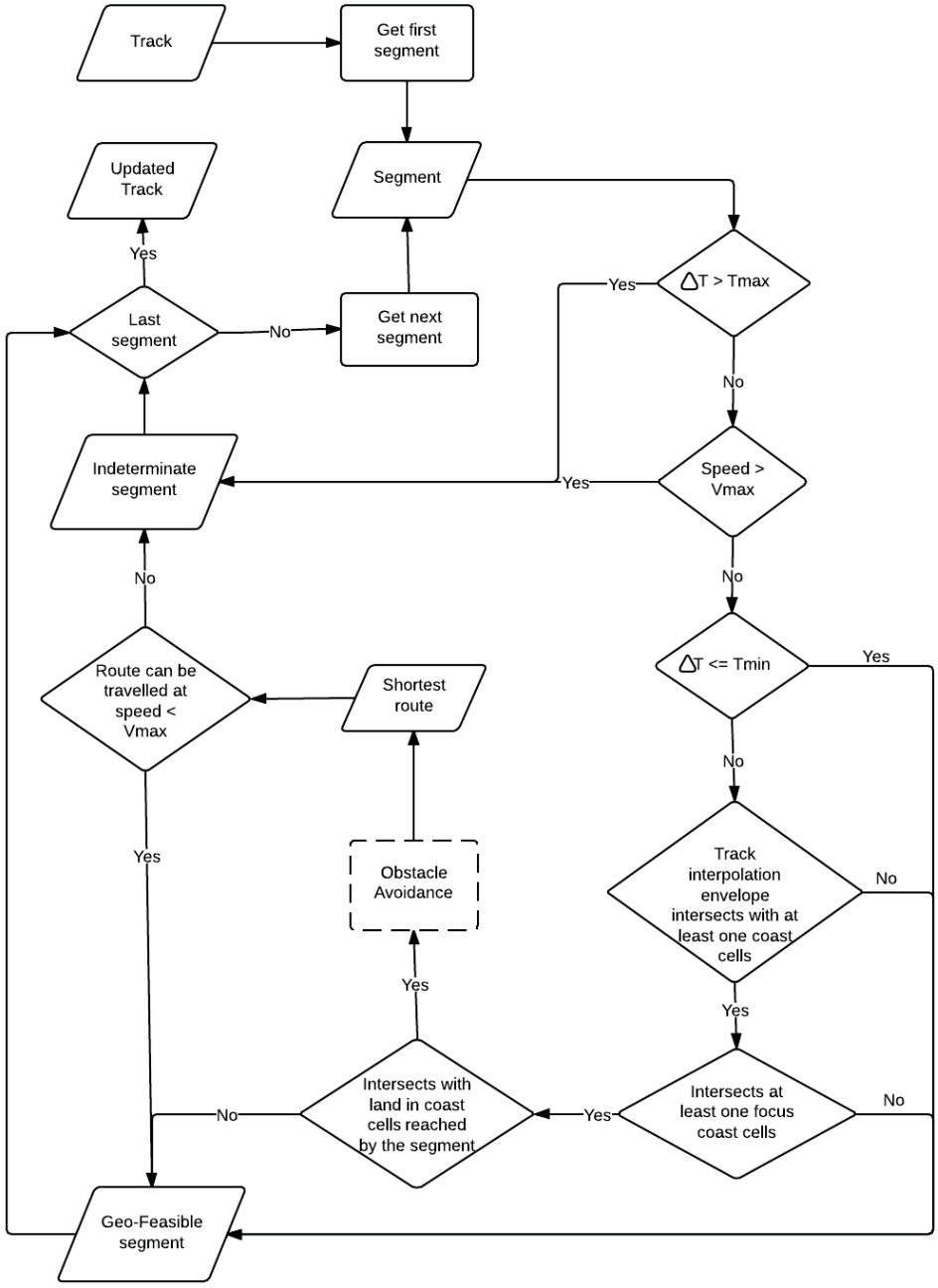

Figure 8: Segment verification method. The dashed box is detailed in Section 5.

(a) The *focus coastal cells*, in yellow, are the coastal cells intersecting with the track envelope. The track envelope is the dashed box.

(b) The track intersects the red *focus coastal cells*.

(c) The red segment intersects land in the red *focus coastal cell*.

Figure 9: Identification of segments crossing land.

5 Obstacle Avoidance

The IM used throughout this project generally creates a straight line between a ship's broadcasted reports to represent the vessel's path. In some cases, however, the straight line representation can end up intersecting with land. For example, if a ship travels along a curved coastline, and the vessel's reports are widely separated, the resulting track might well cut through the coast. In these cases, the obstacle avoidance algorithm handles the re-routing of track segments that cross land by finding the shortest land-avoiding path between the start and destination points. These land-avoiding paths tend to hug coastlines.

The obstacle avoidance algorithm is based on an algorithm provided by Tim Hammond.

The algorithm was divided into two stages: a preprocessing stage, and a dynamic stage. The preprocessing stage is the most expensive piece of code in terms of execution time. Its purpose is to alleviate the computation time of the dynamic stage by converting the map to a graph (G) (i.e., a collection of vertices and edges) containing viable routes between coastline vertices, which can be readily applied to the shortest path algorithm. This process is detailed in the next section.

The dynamic stage of the algorithm is triggered by the IS at runtime, when a segment of a vessel's path is found to cross land. The goal at this point is to find the shortest alternate path that does not leave the water.

As a first step, an area of uncertainty is established around the segment's start and destination points. Given two position reports A and B (separated in time by ΔT_{AB}) from a ship whose maximum speed is v_{max}, all possible paths connecting A to B must be contained within an ellipse. This ellipse has foci at A and B and is defined as the locus of points whose total distance from both A and B is equal to $v_{max}\Delta T_{AB}$. The AOU was taken to be the smallest rectangle that bounds that ellipse, see Figure 10 . The reason a rectangular AOU was used instead of an elliptical one is because ellipses are not supported by the JTS. That is not to say, however, that a custom *ellipse* java class could not be develped to provide a tighter AOU in future work.

Once the AOU has been constructed, the graph (G) is searched and all the vertices that fall within the AOU are added to a temporary graph (T). Any edge between these verties is also added to the temporary graph. T is essentially a subgraph of G that is entirely contained within the AOU. Figure 11 shows an example of an AOU and a loaded preprocessed path.

The final step before searching for the shortest path is to add the start and destination waypoints (A and B) to the graph (T) and connect these points to all the other accessible vertices already in T. For a vertex in T to be accessible from A, the segment connecting the vertex to A must not cross land. Similarly, for a vertex to be accessible from B, the segment connecting the vertex to B must not cross land.

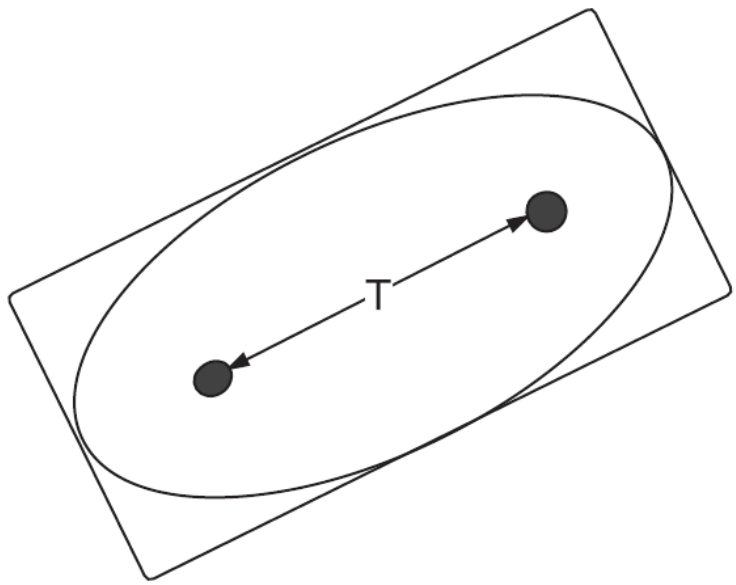

Figure 10: This figure indicates two position reports with black blobs. It shows the AOU for these reports as a rectangle that bounds an ellipse. That ellipse in turn represents the area within which the ship must have traveled between the two reports, assuming the vessel did not exceed the assumed maximum speed v_{max}.

At the end of this step, the shortest path connecting A to B must be composed of edges from T. Figure 12 shows a zoomed in section of the Figure 11 with start and destination points added along with all the usable paths.

The final step is to apply the shortest path algorithm, which uses an implementation of Dijkstra's algorithm provided by GeoTools. The only requirement to make the algorithm work is to provide a weight for each edge in the graph T. The weight assigned to each edge is given by its great-circle length. Figure 13 shows the shortest path found for the example of the previous two figures.

If the new path respects all the restrictions previously established, it is marked as determined and new interpolated reports are created for the IDB. These reports are marked as *interpolated* because they aren't real reports sent out by the vessel. As such, they don't contain a MSARI ID. The timestamp of interpolated reports is determined by assuming the ship travels at constant speed between its start and destination points. The timestamp value is therefore proportional to the distance travelled along the total path length.

The interpolation process as a whole is illustrated in Figure 14.

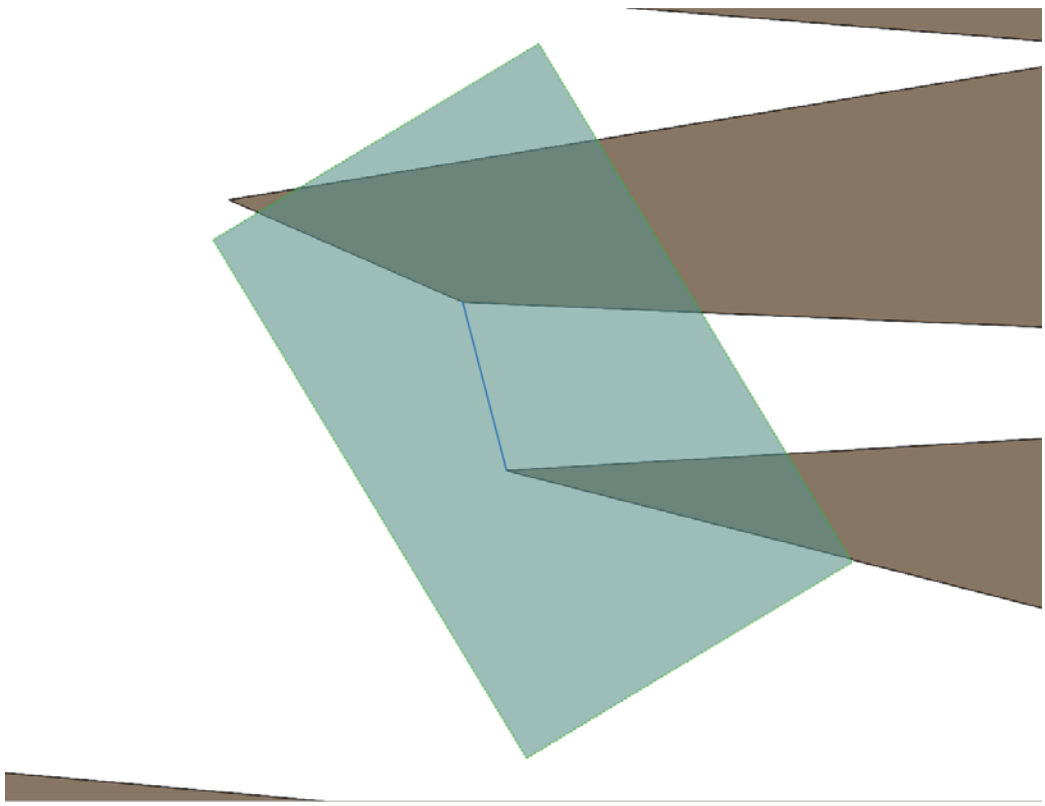

Figure 11: This figure shows an AOU as a semi-transparent rectangle along with the only preprocessed path (in blue) that is fully contained by the AOU. Other land edges (land is shown in brown) may traverse the AOU, but they are not fully contained within it and are therefore not part of the subgraph (T) used in obstacle avoidance.

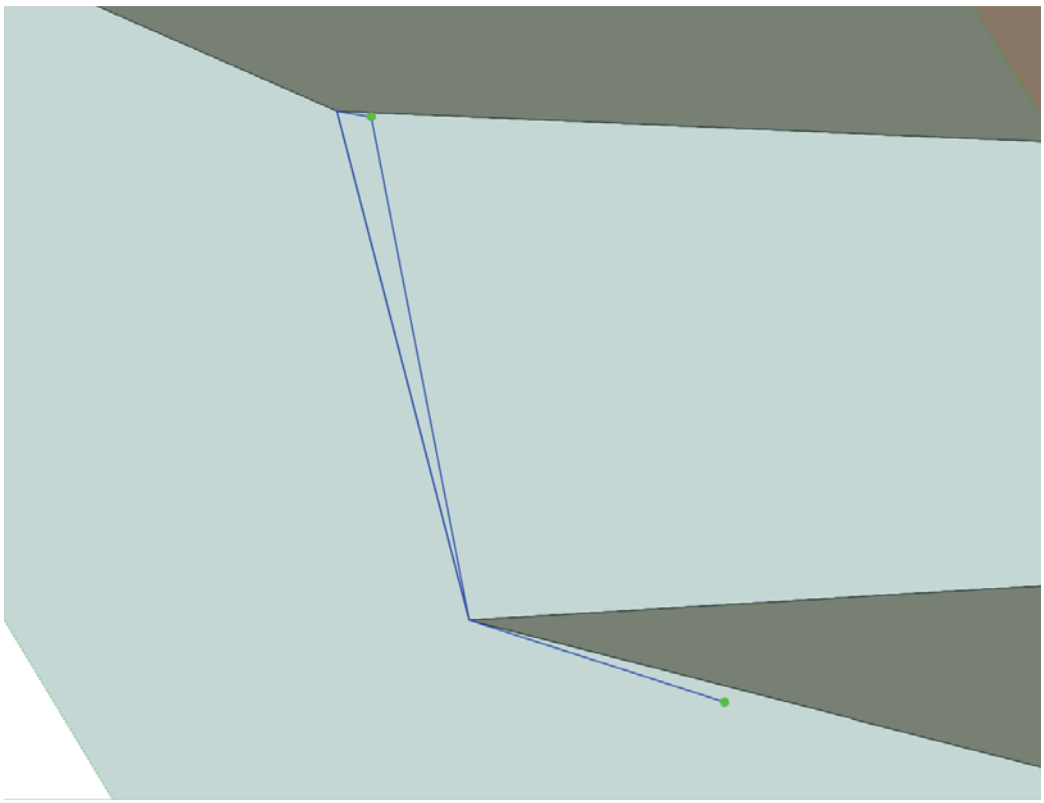

Figure 12: Water is shown here in pale blue, while land is indicated in greenish-brown. The start (bottom) and destination points (top) are indicated with bright green circles along with all the valid paths (in blue) connecting these points to land vertices.

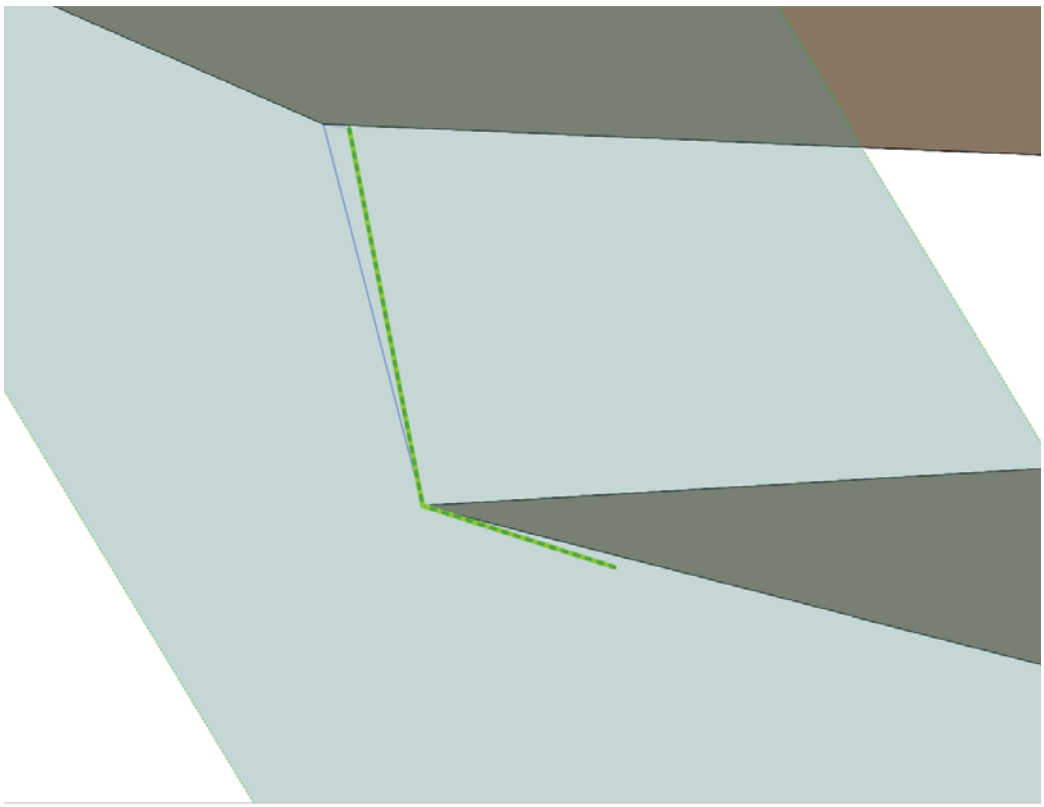

Figure 13: This figure shows the correct shortest path found by Dijkstra's algorithm (in dotted green).

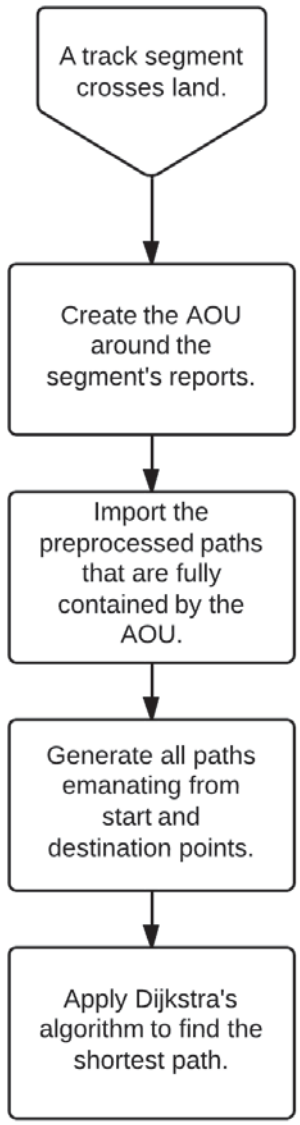

Figure 14: This figure sums up all the major steps involved in the obstacle avoidance algorithm.

6 Route Generation

The goal of preprocessing routes is to alleviate the amount of work required at runtime when applying the obstacle avoidance algorithm. This procedure constructs the graph G used by the obstacle avoidance algorithm beforehand, so that when it is needed, it is more readily available. It is also at this preprocessing stage that the map resolution is chosen. The graph G is intrinsically linked to the chosen map resolution.

The ship routes along coastlines won't change with time because the land is assumed to be static, which is why this process can be extracted from the rest of the IS application and completed beforehand.

More specifically, the algorithm starts out with a detailed world map containing GIS polygons representing landmasses. The map is then simplified using the Douglas-Peucker algorithm, which is available in the JTS library. It's important to note that, when simplifying, this algorithm doesn't preserve topology. It is therefore possible that a simplified map will have overlapping polygons and islands that are completely removed by the algorithm. Every simplification algorithm has its drawbacks. Of those available with JTS, this one was deemed the most reliable. See Figure 15 for an example of the impact of simplification on land representation and thus on obstacle avoidance.

Once the map is simplified, every coastal polygon vertex is added to a graph. The final stage of graph construction attempts to connect every vertex with every other one, as long as:
– the two vertices are within a distance $D_{max} = v_{max}t_{max}$ from each other.
– the segment connecting the two points doesn't cross land.
Segments crossing the dateline are also connected, using the techniques of Section 4.2.

The resulting graph G is used in obstacle avoidance, as outlined in Section 5. The graph is stored in the IDB. The required sections of the graph can then be fetched by the obstacle avoidance algorithm as needed. Figure 16 shows an example of a processed graph.

The preprocessing of routes is very slow. The simplified map and graph delivered to DRDC took three days to generate. By comparison, the routes on the most detailed map available are estimated to take over ten thousand years to generate.

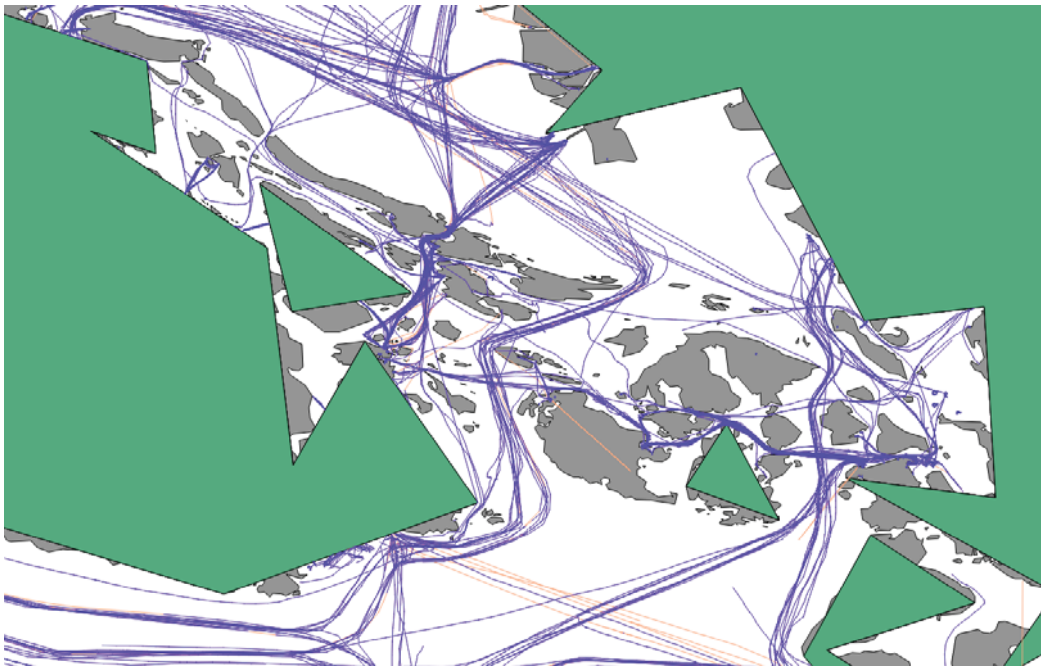

Figure 15: Interpolation results for 24 hours of AIS data in the Strait of Georgia, performed with a map of 43 832 vertices (in green). The blue segments are *determinate* while the orange are *indeterminate*. The high resolution map is illustrated in gray. We can see that some segments that were accepted as *determinate* using the low resolution map do, in fact, cross land on the higher resolution one.

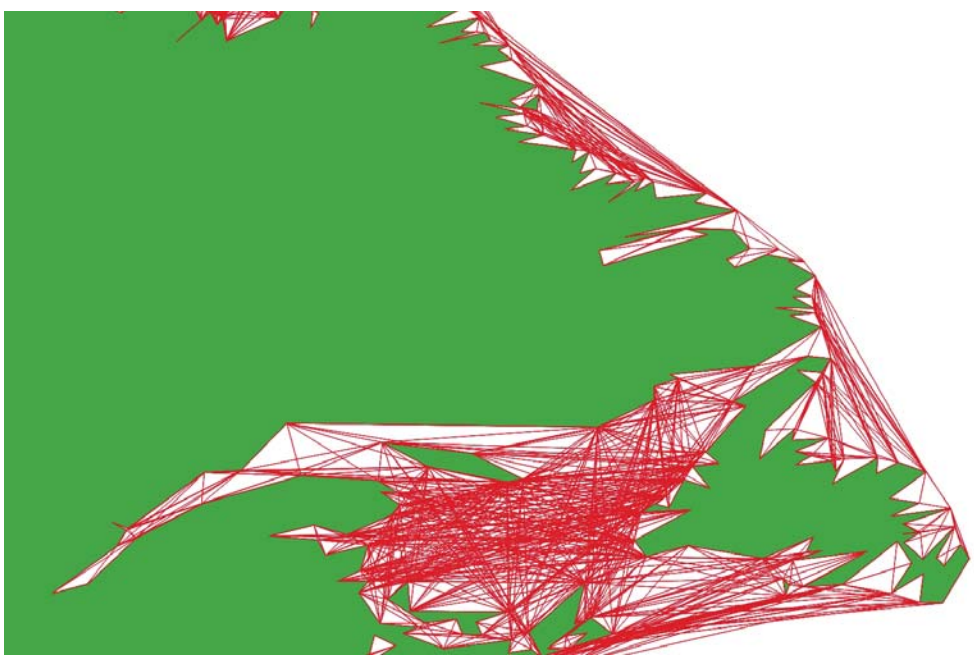

Figure 16: This figure shows an example of preprocessed paths along the East coast of Canada.

7 Interpolation Database Description

The Interpolation Database contains the interpolation results and all the data required to interpolate between the original position reports. This section describes the IDB content.

The physical data model of the IDB is illustrated in Figure 17. The data model has two main components:

1. tables to store interpolation results, described in Section 7.1, and
2. tables to store the data required to perform interpolation, described in Section 7.2.

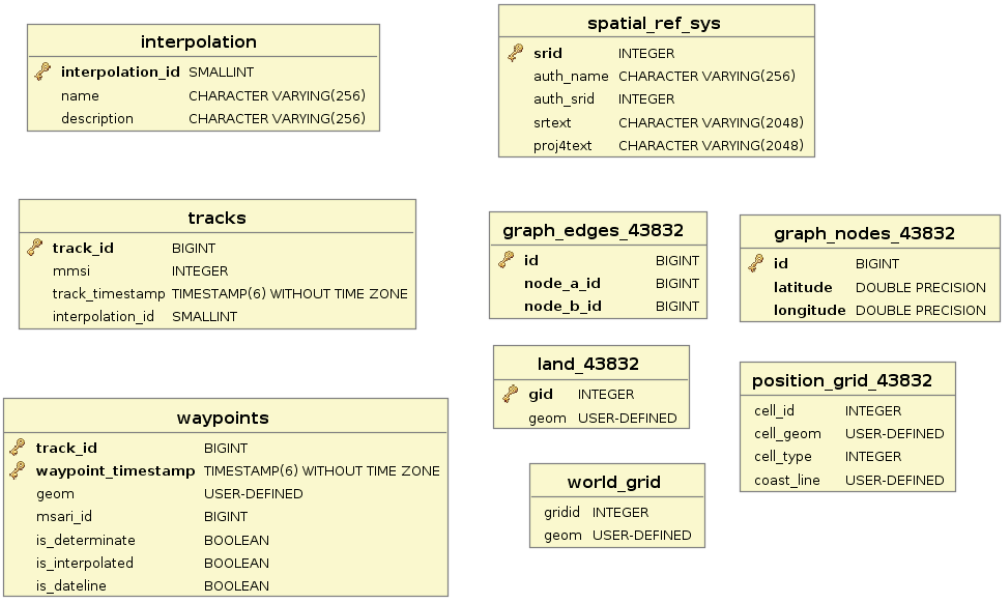

Figure 17: Interpolation database model.

7.1 Interpolation results storing

The conceptual model for interpolation results is illustrated in Figure 18. This simple model stores only the minimal information required to rebuild *tracks*. Each *track* is represented by a MMSI, a creation time stamp and a list of *waypoints*. These *waypoints* are geo-referenced, have a time stamp and can be linked to MSARI via the `MSARI ID` , which is MSARI's report identifier. There are three flags describing which type of *waypoint* and segment it is. If the segment starting with a given *waypoint* is *indeterminate*, the `Is Segment Determinate` flag is false. If the *waypoint* is an artifact created to allow the segment to cross the date line, the flag `Is DateLine` is set to true, and if the *waypoint* was created as part of obstacle avoidance, the flag `Is Interpolated` is set to true. Note that, if `Is DateLine` or `Is Interpolated` is true, then the `MSARI ID` is *null*. Also, the model does

not include a segment entity because it would have created data duplication (a segment can be uniquely defined by its two *waypoints*). The `Interpolation` entity is metadata that is used to identify which interpolation method was used to build the *track*.

The physical representation of the conceptual model illustrated in 18 is shown in Figure 17, which shows the following tables: `tracks`, `waypoints` and `interpolation`.

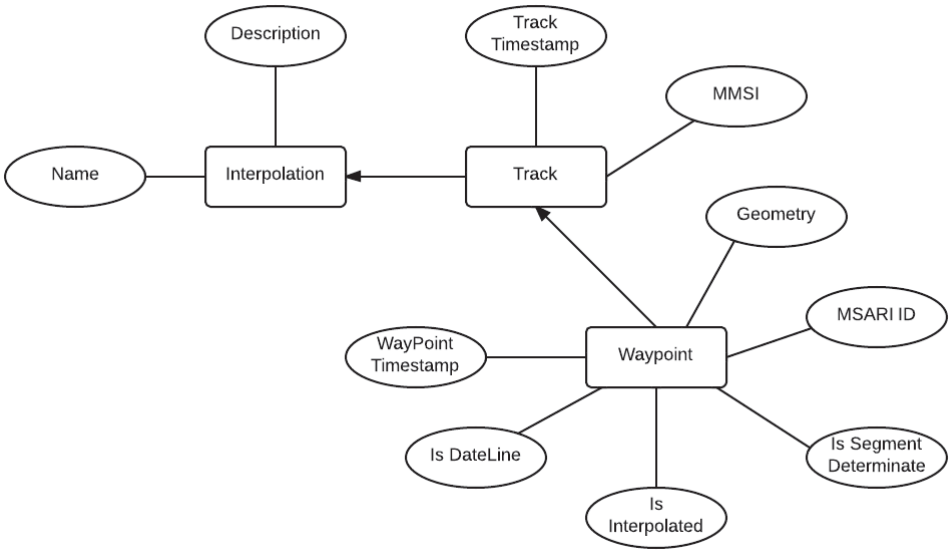

Figure 18: Conceptual model for interpolation results.

Note that foreign keys between `tracks` and `waypoints` and `interpolation` are not implemented because it would considerably slow the storing and updating procedure. The integrity is managed by the application by providing the track ID (`track_id` for `tracks` and `waypoints`) instead of having the DBMS managing it.

7.2 Data required for interpolation

Tables `graph_edges` and `graph_nodes` contain the graph G that is used in obstacle avoidance. This graph is created from the map stored in the table `land`. The data in these tables is produced as described in Section 6. Note that the graph tables are not GIS enabled, but `land` is.

Table `position_grid` is a customized grid covering the globe, where each cell is identified as covering the land (based on `land` table), the sea or both (coast cells). See Annex B for details.

8 Performance

At the time of writing, MSARI was taking in about 16 000 000 AIS reports (all types) in a 24 hour period and this is likely to remain constant for a while. Ultimately, the IS would process MSARI's data on a daily basis and thus has to be able to keep pace with MSARI's inflow.

Part of this project was to estimate how long it would take to process 16 000 000 reports end to end, and identify where the bottle necks are. The goal was to be able to process a 24 hour data set in less than one day.

To reach that goal, four main strategies were applied:

1. Algorithm improvement: diverse algorithms used in the interpolation, such as determining if a segment crosses land and obstacle avoidance, were iteratively refined for speed and memory usage. The algorithms presented in this document are the result of several optimization iterations.
2. Scaling: different map resolutions and parameter values such as t_{min} and t_{max} were tested. Map resolution greatly impacts route generation and the obstacle avoidance. An analysis of the impacts is presented in Section 8.2 and 8.3.
3. Database and SQL query design: Queries to fetch reports from MSARI were optimized based on MSARI's future capacity (see Section 3) and the IDB was designed for massive storing (see Section 7).
4. Architecture exploitation: The interpolation and the track storing are multithreaded (see Section 2.3). Such exploitation of the computer's architecture allowed us not only to speed up the process greatly, but also to decrease memory use. Furthermore, the Java Virtual Machine (JVM) parameters, particularly those related to memory usage and garbage collection, were configured to improve performance. These settings are presented in Section 8.4.

This section presents the performance testing setup and results.

8.1 Experimental description

Two main experimentations were set up to test the IS. The first one aimed to test performance aspects related to a large data volume. This experiment is described below and results are discussed in Sections 8.2, 8.3 and 8.4. The second one focuses on the quality of the interpolation results. It includes only artificial reports, created to test specific limit cases. The data set and the results are described in Section 8.5.

A local clone of the MSARI DB was filled with AIS data from exactEarth and the Maritime Safety and Security Information System (MSSIS), collected from 2013-02 to 2013-05-14 02:44:53, totaling 13 374 074 reports (11 169 171 from MSSIS and 2 204 903 from

exactEarth). In this data set, 13 368 153 reports have a time stamp inside the time interval 2013-05-13 00:00:00 to 2013-05-14 02:44:53, i.e. less than 0.02% of the reports were outside this time interval.

The interpolation was performed with the scope [-180,180],[-90,90] and 2013-05-13 00:00:00 to 2013-05-14 02:44:53 (therefore, we attempted to process more than 24 hours worth of data).

The first step is to filter reports from MSARI, as described in Section 3. The number of filtered reports varies based on the map resolution used (see Table 1).

As the resolution gets coarser:

1. the coast line also gets coarser and some bays are filled in, resulting in a loss of reports (reports that are actually at sea are incorrectly judged to come from the land);
2. islands disappear and some coast lines are retracted, resulting in a gain of reports (reports actually on land are judged to originate at sea).

It seems that the overall tendency is towards an increase in the number of reports judged to originate from the sea, as the map resolution gets coarser. Regardless of the resolution used, more than half of the reports were judged to come from land, but the original report numbers do include some AIS messages that are not actually position reports (they include, for example, static and voyage related messages). This high proportion is probably best explained by the fact that many AIS reports originate in harbours that are necessarily close to the border between land and sea, places where correct discrimination is particularly hard. We suspect that the error rate in distinguishing land from sea is high at all the resolutions used in Table 1 because it seems unlikely that more than half of all AIS reports in the dataset really came from land.

Map vertices	**Number of Sea reports**	**Proportion of Reports at Sea**	**Number of Distinct Ships**
7 716	5 851 799	43.8%	52 312
30 144	5 894 232	44.1%	54 288
38 842	5 781 415	43.3%	52 312
43 832	5 739 469	42.9%	53 679
1 643 285	5 607 744	41.9%	54 788

Table 1: The number of dynamic reports on sea fetched from MSARI, when the distinction between land and sea was made using different map resolutions. The resolution is quantified by the number of vertices in the map.

8.1.1 Distribution of time gaps

The interpolation algorithm is sensitive to t_{max} and t_{min}. So, in order to gain a better understanding of the typical time gap distribution in MSARI, a statistical analysis was performed

on the 24 hour data set. The 5 847 504 time gaps in the data set were loaded and analysed with R [5].

The descriptive statistics summary of the time gaps (all results are in seconds) is presented in Table 2.

We can remove outliers from the data set, where outliers are defined by:
– time gap > min(Q3 + 1.5 IQR, max(time gaps)) = 360 s + 1.5*300 s = 810 s
– time gap < max(Q1 - 1.5 IQR, min(time gaps)) = min(time gaps) = 1 s
where $Q1$ is the first quartile, $Q3$ is the third quartile, and IQR = Q3 - Q1. The resulting descriptive statistics summary is also presented in Table 2.

From that Table, we can see that only about 50% of all time gaps are less than 5 minutes (which is the selected t_{min}). Moreover, it was computed that 0.3% of all time gaps exceed t_{max} = 6 hours.

	All data set	Outliers removed
Minimum	1.0	1.0
1st Quartile (Q1)	60.0	50.0
Median	307.0	305.0
Mean	508.5	251.3
3rd Quartile (Q3)	360.0	358.0
Maximum	91090.0	810.0

Table 2: Descriptive statistics summary (in seconds) of the time gaps involved in the 24 hour data set.

8.2 Processing time

Since the interpolation and the track storing operations are now multithreaded, it is no longer possible to compute exactly how long each operation takes for an entire run. However, before the system was made multithreaded, some profiling was performed, allowing us to identify which operations are the most time consuming. These results are presented in Table 3 for a very coarse map (7 716 vertices).

With the multithreaded implementation, it was possible to monitor run time for only two operations: Fetching Reports from MSARI and Interpolation with Storing. The latter operation is a combination of the Modelling Tracks, Segment Verification, Obstacle Avoidance and Storing Tracks operations from Table 3. Results are presented in Figure 19. We can see that the time required to process the data set is roughly independent of the map resolution. This result is counter-intuitive. It was expected that the time spent interpolating and storing tracks would increase with the map resolution in a non-linear way. The more vertices the

5. http://www.r-project.org/

Operation	Details	Proportion
Fetching reports from MSARI	Filtering and fetching reports	14.3%
Modeling tracks	Connecting reports together, Computing speed and course on segments, Duplicate report removal	12.4%
Segment verification	Comparing duration to t_{min} and t_{max} Check speed and identifying *indeterminate* segments Verifying intersection with coastal cells Verifying intersection with land in coastal cells	0.0 % 0.5% 0.5%
Obstacle avoidance	Creating shortest obstacle-avoiding paths	29.4%
Storing tracks	Mapping *tracks* to IDB model and storing them in the database	42.9%

Table 3: Time spent for each operation as a proportion of total run time. Results were computed before multithreaded implementation, for the coarse map (7 716 vertices) and the 24 hour data set.

map contains, the more complex is the graph G and so the number of operations required in performing obstacle avoidance increases. As a hypothesis to explain the observed results, we propose that the number of segments judged to require obstacle avoidance (by the rules of the IM) must have been decreasing with map resolution. Thus, although each individual obstacle avoidance call took longer with increased map resolution, the reduction in the number of these calls could have compensated enough to maintain a roughly constant run time. These results are promising in the sense that they indicate a relatively low sensitivity of overall run time to map resolution.

It is also worth mentioning that the MSARI clone used in the experiment above contains only 24 hours worth of data. The actual MSARI installed at DRDC-Atlantic contains more than a year of data. Therefore, the time spent to fetch reports may increase as MSARI increases in size. On the other hand, the machine used in the experiment to host MSARI is not the same as the one at DRDC-Atlantic, the latter being more capable.

Finally, the same data set was processed with t_{min} of 4, 5 and 6 minutes. The time difference in total run time was found to be negligible (less than 30 seconds). It was decided to keep t_{min} at 5 minutes, because many land crossings were mis-identified in increasing t_{min} from 5 minutes to 6, as will be shown below. Again, it is possible that the run-time impact of raising t_{min} may become more significant with still higher resolution maps.

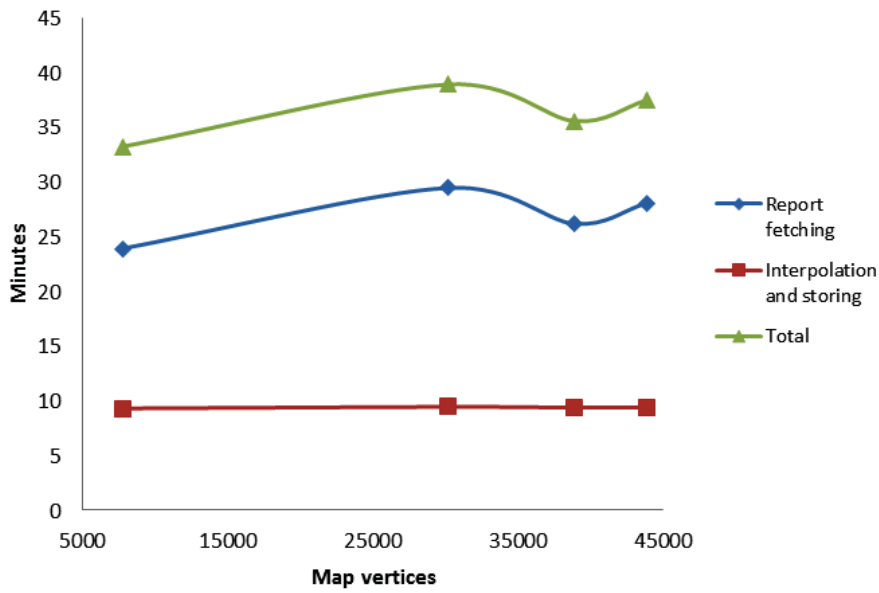

Figure 19: Time in minutes spent processing the entire 24 hour data set. The map resolution is quantified by the number of vertices.

8.3 Processing efforts

This section provides an overview of how frequently the segment verification (see Section 4.3) and the obstacle avoidance (see Section 5) processes were used in processing the 24 hour data set. This analysis was performed with the 43 832 vertex map.

Of the 53 679 unique ships, 3 712 had only 1 report, leaving 49 965 *tracks* requiring interpolation.

Figure 20 illustrates the number of segments that were processed at each step of the Segment Verification and Obstacle Avoidance processes. It also shows the proportion of segments remaining at each step. Although these results are for only one map resolution, the same experiment with different map resolution produced very similar results.

The experiment was also repeated with t_{min} = 4 minutes and t_{min} = 6 minutes. As expected, fewer segments reach the obstacle avoidance step as we increase t_{min}. Results are displayed in Table 4.

With t_{min} = 6 minutes, 27% of the initial segments have a time gap lower than t_{min}, which is about half the number obtained with t_{min} = 5 minutes. These results are consistent with the time gap distributions presented in Section 8.1.1.

An increase of 1 minute in the t_{min} threshold (from 5 to 6 minutes) makes the application miss about 33% of the land collisions. This indicates that the IS is sensitive to that thresh-

old. However, this sensitivity is not linear, since the number of land collisions missed in going from $t_{min} = 4$ to 5 minutes is low.

t_{min} (minutes)	Number of segments crossing land	Proportion of segments crossing land	Segments created by obstacle avoidance
4	2 202	0.039%	3 537
5	2 096	0.037%	3 353
6	1 393	0.029%	1 681

Table 4: Collision detection for different values of t_{min}, all produced with the 43 832 vertices map.

8.4 Memory requirements

It was impossible to run the 24 hour experiment with 8 Gigabyte (GB) of Random-Access Memory (RAM) using a sequential process. That is why the application was made multithreaded (see Section 2.3 for more details). The memory usage of the final multithreaded application was not profiled because the performance was found to be satisfactory (see sections above).

The experiments with the 24 hour dataset were performed with the following Java parameters. We suggest using the same parameters to ensure good performance. Note that 8 GB of RAM are required.

```
-server
-Xmn128m -Xms512m -Xmx8g
-XX:PermSize=128m
-XX:MaxPermSize=256m
-XX:+UseParallelGC
-XX:+AggressiveHeap
-XX:SoftRefLRUPolicyMSPerMB=36000
-Djava.awt.headless=true
-XX:+HeapDumpOnOutOfMemoryError
-XX:HeapDumpPath=/tmp
```

8.5 Quality testing

A second data set was used to test the quality of the interpolation output. This data set includes a limited number of artificial reports (created specifically for the experiment) for 26 use cases representing limit cases. These cases are not detailed here, but can be found in the `Text.java` class, located in the `ca.drdc.iqa.main` package of the IS source code. Basically, the following situations and combinations thereof were tested:

1. Reports on land,
2. ship crossing the date line,
3. ship sailing on the date line,
4. ship sailing near the date line without crossing it,
5. obstacle avoidance for ship crossing the date line,
6. ship sailing at speed exceeding v_{max},
7. all time gaps exceeding t_{max},
8. obstacle avoidance when no routes (no graph G) are available,
9. ship sailing in the Canadian Arctic archipelago (presented in Figure 21).

The IS succeeded at all tests, with the exception of tests involving reports on land. For the moment, the IS can only process reports that are located on water because obstacle avoidance cannot work when reports come from land. The MSARI interaction component was implemented to filter out any reports on land. However, if the IS is used with another AIS database, one should be aware that on land reports have to be filtered out before being processed. Note, however, that the IS will not stop running if it is provided with on land reports. It will identify the segments connecting such reports with others as *indeterminate*.

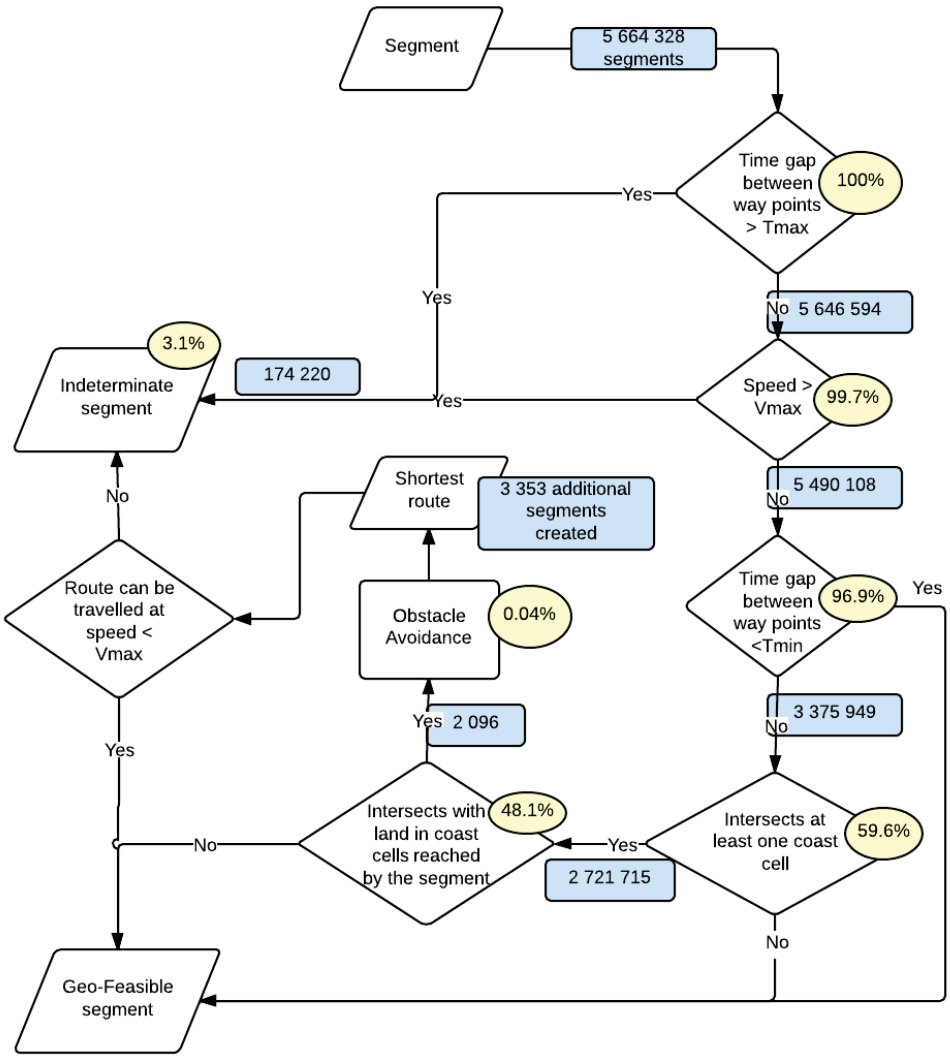

Figure 20: Number of segments and their proportion at each step of the Segment Verification and Obstacle Avoidance processes. This experiment was performed with the 43 832 vertex map, t_{min} = 5 minutes, t_{max} = 6 hours, v_{max} = 20 knots.

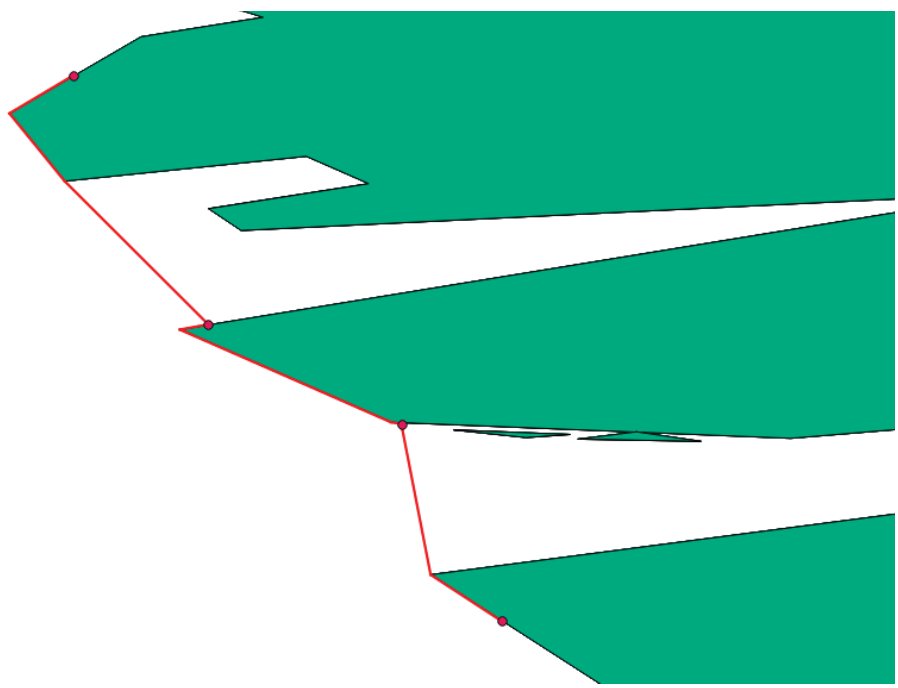

Figure 21: Results for the unit test involving a ship sailing along the shores of Victoria Island. The red dots are the ship's contacts and the red line is the *track* as computed by the IS. This case was challenging because each segment connecting the reports was intersecting with land. Therefore, obstacle avoidance was used to find the shortest path between each report. The map had 43 832 vertices.

9 Way Ahead

This concluding section suggests improvements that would enhance the IS. It also points out problems that will persist and may affect the system.

- The report fetching operation could be parallelized to speed up the process. This operation is the only one in the IS that was not designed to be concurrent. The scope query could be split into several queries segregated by the available MMSI within a scope. Each thread would essentially be operating on a single track from the onset of the application.
- Currently, the scope query is not designed in a way that allows it to overlap with the dateline. This restriction should be removed.
- The alternate route produced by the obstacle avoidance actually touches the land. In reality, a ship sails along the shore without touching it. The obstacle avoidance algorithm could be modified to include a buffer to the land (zones around the land's polygons). So the same algorithm would be used, but instead of considering the vertices of the land, it would use the vertices of the land's buffer.
- Currently, the interpolation algorithm produces a lot of potentially insignificant segments along tracks. Applying a statistics fitting on the report contacts to reduce the number of segments produced, for instance, a genetic algorithm based approach, as presented in [3] and [4], produces in average 10 times less segments than connecting subsequent reports together as it is done currently. The consequences are limited, because a ship usually sails in straight lines following way points. The advantages would be reduced memory requirements for the IDB, reduced fetching time for existing tracks and less processing for the collision avoidance. Conversely, the model would take more time to produce. A trade-off analysis would have to perform to see if advantages prevail over disadvantages.
- Many AIS reports are sent by moored ships, and AIS transponders behave differently when ships are moored, generally transmitting less often. The transponders are also likely to be turned off when ships are moored. For these reasons, it seems important to investigate the capability to recognize time intervals where the ship is stationary. If these intervals can be recognized successfully, then the rules of the IM can be changed for them, giving a more realistic interpolation. This would improve processing speed, because it is not necessary to check for obstacle avoidance in a moored ship.
- The most significant problem with the IS in its current form is that it can only interpolate between reports that are judged to be at sea. Currently, more than half of all AIS reports are judged to come from land (see Table 1). We do not believe the percentages reported in that table to be an accurate reflection of the true rate at which AIS reports originate on land. They are an artifact of the low map resolution employed in distinguishing land from sea. It would be instructive to get a more reliable measure of this percentage. To be useful, interpolation must be possible on a high proportion of the original AIS dataset. To achieve this, it may be necessary to implement map compression algorithms that reliably shrink the land area, as the number of coastal vertices is reduced. In this way, coarser

maps would filter out less and less AIS data. In addition, the IM could be adjusted to just interpolate linearly (skipping obstacle avoidance) when one or both reports in a segment are close to the boundary between land and sea.

References

[1] Hammond, T. (2013), Call-Up for Standing Offer W7707-115137: Design and prototype a system for managing AIS database queries that require interpolation between position reports Defence R&D Canada – Atlantic.

[2] (2013), Great-circle distance (online), Wikipedia, http://en.wikipedia.org/wiki/Great-circle_distance (Access Date: Sep 2013).

[3] M.-O., St-Hilaire, E., Lefebvre, and C., Helleur (2008), Track Modeling for Maritime Surveillance, *Proceeding of the 11th International Conference on Information Fusion, Cologne, Germany*.

[4] Lefebvre, Eric (2008), Application Development for Multi-Sensor Integration within a Common Operating Environment (MUSIC): Cycle 3, Technical Report DRDC Ottawa CR 2008-103.

Annex A: Stakeholder Requirements

This section presents the initial requirements for the Interpolation Query system (hereafter named the system) as extracted from the SOW [1].

The requirements are divided in three categories:

1. Interface
2. Functional
3. Implementation

The interface requirements are the requirements that apply to the input and output the system shall support. The functional requirements are the requirements applying to the processing of the information. Although several design solutions may exist to respond to the interface and functional requirements, the SOW has implementation requirements that have been further elicited during the meetings with the Scientific Authority. While the interface and functional requirements are high level and should remain unchanged during the course of this project, the implementation requirements may evolve and will reflect the system design choices. When available, justifications or references will be included as annotations for the implementation requirements (i.e. the results of research work).

A.1 Interface Requirements

A.1.1 Input Requirements

I.1 The system shall access AIS database provided by the user.

I.2 The system shall provide a means to define the scope by:

(a) temporal components,

(b) spatial components, and

I.3 The system shall provide a means to capture temporal reference objects.

I.4 The system shall provide a means to capture spatial temporal objects.

I.5 The system shall provide a means to clip tracks based on reference objects.

I.6 The system shall provide a means to select tracks based on reference objects.

I.7 The system shall provide a means to capture output variables.

A.1.2 Output Requirements

O.1 The system shall provide a means to output filtered information from the *Interpolation Database*.

O.2 The system shall output the information using the user defined output variables.

A.2 Functional Requirements

F.1 The system shall process AIS contact reports to produce ship tracks (a ship track includes ship characteristics and ship trajectory as function of time.)

F.2 The system shall store the ship tracks in a dedicated *Interpolation Database*.

F.3 The system shall provide a means to retrieve the information from the *Interpolation Database*.

F.4 The system shall have the capability to filter the *Interpolation Database* based on:

(a) temporal components,

(b) spatial components,

(c) ship characteristics, and

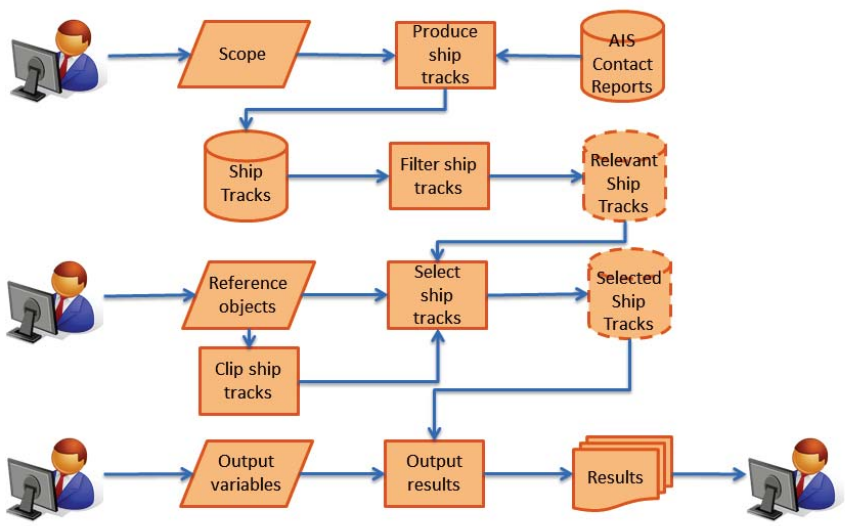

Figure A.1: Functional Flow Block Diagram (to be revised).

A.3 Implementation Requirements

A.3.1 The Interpolation Method

IM.1 The Interpolation Method shall define the assumptions about what happened between AIS position reports with two temporal parameters: t_{min}, t_{max} both in minutes and a speed v_{max} in knots.

IM.2 The Interpolation Method shall treat the speed v_{max} as a speed that the ship cannot exceed.

IM.3 When the ship type is unknown, the Interpolation Method shall define a default value for v_{max}.

IM.4 Whenever the temporal separation (in minutes) between two successive AIS position reports from a given ship is less than or equal to t_{min}, the Interpolation Method shall assume that the ship sailed straight between those reports at a constant speed v_c

IM.5 The Interpolation Method shall calculate the speed v_c as follow: the spatial separation divided by the temporal separation (expressed in knots).

IM.6 If the calculated speed v_c exceeds v_{max}, then the Interpolation Method shall consider the ship status as indeterminate over the time interval.

IM.7 Whenever the temporal separation (in minutes) between two successive position reports from a given ship is greater than or equal to t_{max}, the Interpolation Method shall consider the ship status as indeterminate over the time interval.

IM.8 If the temporal separation is between t_{min} and t_{max}, the Interpolation Method shall consider the ship sailing at speed v_c by the shortest possible route.

IM.9 The Interpolation Method shall establish the shortest possible route by avoiding sailing over land.

A.3.2 Temporal Reference Objects

TO.1 The system shall instantiate temporal reference objects based on user input. Temporal reference objects have a parameter "Name" and are used in the interpolation query.

TO.2 The system shall be able to instantiate temporal reference objects as: time intervals, points in time, or collections of either.

TO.3 The system shall be able to use the temporal reference objects to clip the relevant tracks down to either sub-tracks or waypoints ([time, position] ordered pairs) or collections thereof.

A.3.3 Spatial Reference Objects

SO.1 The system shall instantiate spatial reference objects based on user input. Spatial reference objects have a parameter "Name" and are used in the interpolation query.

SO.2 The system shall be able to use the temporal reference objects to clip the relevant tracks down to a polygon. The clipping would then yield the portion of the track inside the polygon.

SO.3 The system shall be able to use the spatial reference objects to select (i.e., include or exclude) individual relevant ships for further consideration in the query.

SO.4 The system shall be able to instantiate three types of spatial reference objects:

(a) Polygons are 2D shapes that are not necessarily convex.

i. Polygons let the system clip the relevant tracks, either into sub-tracks that are inside the polygon (with appropriate action on the boundary) or into sub-tracks that are not inside of it. Such clipping would occur when the user is interested in features of the sub track (e.g., average speed inside a polygon or total time spent inside).

ii. Polygons also let the software select which of the relevant ships will be given further consideration in the query, by including or excluding those whose relevant tracks go inside the polygon.

(b) Poly-lines are connected line segments that might cross one another.

i. These let the software clip a given track into one or more waypoints that are defined by the times and places at which the track crossed the poly-line.

ii. Poly-lines also let the software select which of the relevant ships will be given further consideration in the query, by including or excluding those whose relevant tracks cross the poly-line.

(c) Points are just (latitude, longitude) ordered pairs, assumed to lie at sea level

i. Points do not allow any clipping

A.3.4 Track Clipping Rules

TC.1 When a track is clipped, the system shall replace the relevant track by its clipped version in further query processing.

TC.2 The system shall establish clipping rule based on reference objects selected by the user.

TC.3 The system shall be able to establish clipping rule with intersection or union of reference objects as input by the user. For example, the system could clip by the intersection of polygon A with temporal interval I, which would return those portions of the relevant tracks that were in A during I. Or the system could clip by the union of A with I to get regions of the track that are either in A or occur during I.

Annex B: Grid Approach

One of the most expensive operations in the interpolation process is verifying if a track segment crosses an obstacle, even with a very coarse shore resolution. That is why this verification is only done when there are not enough evidence that the ship sailed straight between two points (see Section 4.3 for the rules). The high price comes from the intersection operations that are repeated all along the coast lines over the world.

A grid-based approach was proposed to speed-up that verification. The coast is discretized into regions, to reduce the spatial area where the obstacle avoidance check is performed. We identified a grid of square cells of one degree by one degree covering the globe and identified which cells contain coastline boundaries in a pre-processing operation. These coastal cells are loaded in memory at interpolation time (see Figure B.2). Instead of looking blindly if a given segment intersects with land, we first look if the segment intersects a coastal cell and if it does, we look for coast intersection only within that cell.

The same approach is used for the SQL function `isOnLand` , an optimized SQL function that is part of MSARI. It returns true, if a given position is inside the land polygon defines as part of MSARI. It uses the table `position grid` , a customized grid covering the globe, where each cell is identified as covering the land (based on `land` table), the sea or both (coastal cells). This table was generated from the table `world grid` , which contains a grid of 1 degree square cells covering the globe. This function is at least 10 times faster than the intersection operation offered by PostGIS. See Figure B.1 for a visual representation of the `position grid` table generated with a high resolution map (1 643 285 vertices). It is that same table `position grid` that is loaded in memory at interpolation time to perform the segment verification.

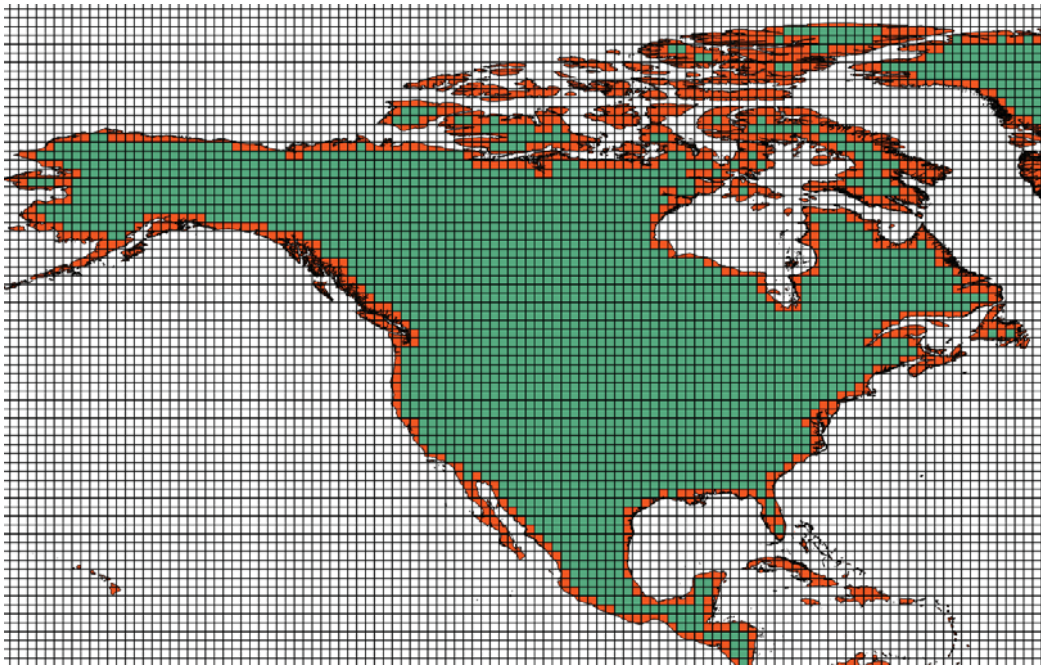

Figure B.1: White cells are identified as being in water, green cells as being completely in land and orange cells are coast cells.

Figure B.2: Coast cells for East of Canada with a high resolution map.

Annex C: Installation

This section details the installation of the IS on Windows 7 with 64 bits.

C.1 Java Installation

This section details the installation of Java, version `jdk1.6.0_45`, with install file `jdk-6u45-windows-x64.exe`.

1. Download `jdk-6u45-windows-x64.exe` from `http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html#jdk-6u45-oth-JPR`. Although it is free, some registration is needed before downloading.
2. Double click on `jdk-6u45-windows-x64.exe` to start installation with all default settings.
3. Set `Path` and `JAVA_HOME` environment variables:
   (a) Go to *Control Panel* > *System and Security* > *System* > *Advanced system settings* > *Advanced* tab then click on *Environment Variables*.
   (b) Select *Path* in *System variable* section and click *Edit*.
   (c) Add `;C:\ProgramFiles\Java\jdk1.6.0_45\bin` at the end then click *OK*.
   (d) Click *New...* in *User variables* for *yourusername* section to create `JAVA_HOME` variable.
   (e) Enter `JAVA_HOME` in *Variable name* and `C:\ProgramFiles\Java\jdk1.6.0_45` in *Variable value* then click *OK* and *OK* in the *Environment Variables* and *System Properties* windows.

C.2 Installation of PostgreSQL 9.2, pgAdmin III, Stack Builder and PostGIS

This section describes how to install:
– PostgreSQL version 9.2.4
– pgAdmin III version 1.16.1
– PostGIS version: 2.0.3

1. Download `postgresql-9.2.4-1-windows-x64.exe` from `http://www.enterprisedb.com/products-services-training/pgdownload#windows`.
2. Double click on `postgresql-9.2.4-1-windows-x64.exe` to start installation with all default settings.
3. Enter and note password for postgres superuser. Password: `yourpssw`.

4. After PostgreSQL is installed, follow instructions to install Stack Builder (this will help you install new software). Launch it when installed.
5. At Stack Builder start up select PostgreSQL 9.2 on port 5432. If your network makes use of proxies, click on the button *Proxy Servers* to setup the coordinates. Then click *Next*.
6. In page *Please select the applications you would like to install* select *Categories > Spatial Extensions > PostGIS 2.0 for PostgreSQL 9.2 (64bit) v2.0.3* then click *Next*.
7. Follow these instructions to complete the installation:
   (a) Click *Next*.
   (b) Click on *I agree to accept the GNU license*.
   (c) Click *Next*.
   (d) Click *Next*.
   (e) Enter the PostgreSQL password you chose before (`yourpssw`) and then click *Next*.
   (f) If a pop-up window asks you to set up the *GAL_DATA* environment, click *Yes*.
   (g) Click *Close*.
   (h) Click *Finish*.

C.3 PostgreSQL configuration

PostgreSQL has to be configured to get an improved performance. The following describes the steps for configuration.

1. Browse to `C:\ProgramFiles \PostgreSQL \9.2\data` .
2. Rename file `pg_hba.conf` to `pg_hba.conf.orig`
3. Copy file `pg_hba.conf` from `CDroot: \Code\ISS` in `C:\ProgramFiles \PostgreSQL \9.2\data` .
4. Open the file with Notepad and edit line
   `host all all 192.168.48.0/24 md5`
   to match your network setup.
5. Rename file `postgresql.conf` to `postgresql.conf.orig` .
6. Copy file `postgresql.conf` from `CDroot: \Code\ISS`
   in `C:\ProgramFiles \PostgreSQL \9.2\data` .
7. Restart the PostgreSQL service either by using the pgAdmin III tool (*right-click*+Stop Service followed by *right-click*+Start Service) or manually by third clicking on *Computer* (on the left side of an Explorer window) and select *Manage* and then, select *Services* (on the left side), on the right side, third click on PostgreSQL and select *restart*.

C.4 Database configuration

This step details the required database configuration.

1. Create the Interpolation database from its template:
   (a) Start pgAdmin III and connect your server (*right-click*+Connect and enter your password).
   (b) *Right-click* on *Databases* and select *New Database...* to create the empty `i_db` database.
   (c) In the *New DataBase...* window enter in *Properties* tab:
      i. *Name*: `i_db`.
      ii. *Owner*: postgres.
      iii. Click *OK* to exit.
   (d) *Right-click* on `i_db` then select *Restore...*
      i. *Format*: Custom or tar.
      ii. *Filename*: browse to `CDroot: \Code \ISS \i_db_template.backup` .
      iii. *Rolename*: postgres.
      iv. Click *Restore*.
      v. Wait to see the end of the restore process: *Process returned exit code 1* and then, click *Cancel* to exit.
2. Modification in MSARI database: In order to fetch filtered data from MSARI, a custom SQL function *isOnLand* is required (with relies on its custom table `position _grid`). This function filters out reports on land. It thus requires the same map that used by the IS for obstacle avoidance.
   (a) Start pgAdmin III and connect to `msari` database server (*right-click*+Connect). If the `msari` database server is not visible, select *File>Add server*, create a name (`msari`), enter the Internet Protocol (IP) (or the name if your DNS server can resolve it) of the server, enter the PostgreSQL password to access that server, and click *OK*.
   (b) Add table `position _grid _43832` [6]: *Right-click* on `msari` then select *Restore...*
      i. *Format* : Custom or tar.
      ii. *Filename* : browse to `CDroot: \Code \ISS \position _grid _43832.backup` .
      iii. *Rolename* : postgres.
      iv. Click *Restore*.
      v. Wait to see the end of the restore process: *Process returned exit code 1* and then, click *Cancel* to exit.

6. "_43832" refers to the number of vertices in the map used to create that table.

(c) Add `isonland_43832` function:

i. Click on `msari` database to select it.

ii. Open query tool: menu *Tools* > *Query tool* or *Ctrl-E*.

iii. From there, click on *File* > *Open* and browse the file `CDroot:\Code\ISS\isonland_43832.sql`.

iv. Execute query: *Query* > *Execute* or *F5*.

v. Close the query window to exit.

Annex D: Execution

D.1 Execution of IS application

To run the application, follow these steps:

1. Copy folder `ISSapp` from `DVDroot: \Code \ISS` to your computer (any location will work).
2. Configure `DbConfig.csv` file (see Section D.1.2 for details).
3. Configure `ScopeConfig.csv` file (see Section D.1.1 for details).
4. Double click on `ISS_run.bat` to start the application.
5. You should see these messages:

```
------ Starting ISS application!!! ------
------ Application running...           ------
------ End of ISS application!          ------
Press any key to continue . . .
```

6. When application is finished just press any key to exit.
7. Repeat steps 2 to 6 for other simulations.

Two files are generated when running the application:
– `iss.log` : Log file of the application displaying application steps, warning and error.
– `stats.txt` : Statistics of the last application run.

D.1.1 Scope configuration

To run the IS, you must configure the scope by editing the `ScopeConfig.csv` file.

This file is used to configure the application running spatial and temporal limits. It has to be configured every time a new simulation scope is required, before starting the application.

The scope's definition is done by editing the second line of the file. The following describes the parameters that must be defined:
– wLong: West longitude limit;
– eLong: East longitude limit;
– sLat: South latitude limit;
– nLat: North latitude limit;
– startTimestamp: Start time of the simulation formatted as yyyy-MM-dd HH:mm:ss [7];
– endTimestamp: End time of the simulation formatted as yyyy-MM-dd HH:mm:ss.

Note that no parameter can be left empty. Note also that you can't use future dates, i.e. the *endTimestamp* must be equal or smaller than the present date-time. Also, the scope spatial

7. y: Year ; M: Month ; d: Day ; H: Hour ; m: Minute ; s: Second.

and temporal components are linked by an *AND* condition, meaning that each report must satisfy both conditions.

Make sure the file is located in the application folder.

D.1.2 Database configuration for the IS

To run the IS, you must provide the MSARI and Interpolation database parameters. This is done by editing the `DbConfig.csv` file. This file has to be configured before starting the application. It has to be modified each time you change either the database location or the name. If you always keep the same database configuration, you will edit this file only once.

Line 2 is for the Interpolation database and line 3 is for the MSARI database.

- DbName: Database name in the server
- ServerIP: Server IP address
- UserName: Username to access database
- Password: Password to access the database (link to UserName)

D.2 Execution of ShapeFileWriter application

The shapefile writer is an additional feature developed for testing. Although it was not part of requirements, it is delivered as part of call-up 10 work products.

It creates shapefiles with all tracks stored in the Interpolation database. This application was not optimized, so it is not memory efficient. As a consequence, you can't use the application with too much data. For instance, it can't process 24 hours of worldwide AIS data. But it can easily print all tracks generated for 24 hours of AIS data in Canadian waters. To overcome that problem, one strategy would be to truncate the IDB after some simulations to reach a size that the ShapeFileWriter is able to handle.

The following describes the steps to run the application:

1. Copy folder ShapeFilePrinterapp from `DVDroot:\Code\ShapeFilePrinter` to your computer (any location will work).
2. Configure `DbConfig.csv` file as described in Section D.2.1.
3. Double click on `ShapeFilePrinter_run.bat` to start the application.
4. You should see these messages:

```
------ Starting ShapeFilePrinter application!!! ------
------ Application running...   ------
log4j:WARN No appenders could be found for logger (org.jboss.logging).
log4j:WARN Please initialize the log4j system properly.
------ Shape files generated!  ------
```

```
Press any key to continue . . .
```

5. When application is finished just press any key to exit
6. Repeat steps 3 to 5 for other simulations.

Shapefiles and a log file should be generated :

- `shapefileprinter.log` : Log file of the application showing application steps , warning and error.
- `Land.shp` : Shapefile with the map's polygons at the resolution used to perform obstacle avoidance.
- `DeterminateTracks.shp` : Shapefile containing the lines representing all determinate segments in the Interpolation database.
- `IndeterminateTracks.shp` : Shapefile containing the lines representing all indeterminate segments in the Interpolation database.

D.2.1 Database configuration for ShapeFileWriter

To run the ShapeFileWriter, you must provide the Interpolation database parameters. This is done by editing the `DbConfig.csv` file. This file has to be configured before starting the application. It has to be modified each time you change the database location or name. If you always keep the same database configuration, you will edit this file only once.

- DbName: Database name in the server
- ServerIP: Server IP address
- UserName: Username to access database
- Password: Password to access the database (link to UserName)
- landTableName: Name of the table *land* in the interpolation database. If no additional maps are created, that should be `land_43832` .

List of symbols/abbreviations/acronyms/initialisms

AIS	Automatic Identification System	1
AOU	Area Of Uncertainty	x
DB	Database	7
DBMS	Database Management System	6
DRDC	Defence Research and Development Canada	i
GB	Gigabyte	36
GIS	Geographic Information System	7
GPW	Global Position Warehouse	1
IDB	Interpolation Database	vii
IM	Interpolation Method	2
IP	Internet Protocol	51
IQA	Interpolation Query Application	3
IS	Interpolation System	ix
JDBC	Java Database Connectivity	5
JTS	Java Topology Suite	5
JVM	Java Virtual Machine	31
LRIT	Long Range Identification and Tracking	1
MMSI	Maritime Mobile Service Identity	5
ms	milliseconds	15
MSA	Maritime Situational Awareness	1
MSARI	Maritime Situational Awareness Research Infrastructure	1
MSSIS	Maritime Safety and Security Information System	31
RAM	Random-Access Memory	36
SOW	Statement of Work	2
SQL	Structured Query Language	5
TB	Terabytes	11
VMS	Vessel Monitoring System	1

DOCUMENT CONTROL DATA

(Security classification of title, body of abstract and indexing annotation must be entered when document is classified)

1. ORIGINATOR (The name and address of the organization preparing the document. Organizations for whom the document was prepared, e.g. Centre sponsoring a contractor's report, or tasking agency, are entered in section 8.) OODA Technologies Inc. 4891 Av. Grosvenor, Montreal Qc, H3W 2M2	2. SECURITY CLASSIFICATION UNCLASSIFIED (NON-CONTROLLED GOODS) DMC A REVIEW GCEC: April 2011

3. TITLE (The complete document title as indicated on the title page. Its classification should be indicated by the appropriate abbreviation (S, C or U) in parentheses after the title.)

An Interpolation System for Position Report Databases

4. AUTHORS (Last name, followed by initials – ranks, titles, etc. not to be used.)

St-Hilaire, M.-O.; Radulescu, D.; Hammond, T.; Lefebvre, E.

5. DATE OF PUBLICATION (Month and year of publication of document.)	6a. NO. OF PAGES (Total containing information. Include Annexes, Appendices, etc.)	6b. NO. OF REFS (Total cited in document.)
October 2013	78	4

7. DESCRIPTIVE NOTES (The category of the document, e.g. technical report, technical note or memorandum. If appropriate, enter the type of report, e.g. interim, progress, summary, annual or final. Give the inclusive dates when a specific reporting period is covered.)

Contract Report

8. SPONSORING ACTIVITY (The name of the department project office or laboratory sponsoring the research and development – include address.)

Defence R&D Canada – Atlantic
P.O. Box 1012, Dartmouth, Nova Scotia, Canada B2Y 3Z7

9a. PROJECT NO. (The applicable research and development project number under which the document was written. Please specify whether project or grant.)	9b. GRANT OR CONTRACT NO. (If appropriate, the applicable number under which the document was written.)
11ho;11jo;06eo	Call-up 10, No. 4500862723 to W7707-115137
10a. ORIGINATOR'S DOCUMENT NUMBER (The official document number by which the document is identified by the originating activity. This number must be unique to this document.)	**10b. OTHER DOCUMENT NO(s). (Any other numbers which may be assigned this document either by the originator or by the sponsor.)**
DRDC Atlantic CR 2013-137	

11. DOCUMENT AVAILABILITY (Any limitations on further dissemination of the document, other than those imposed by security classification.)

(X) Unlimited distribution
( ) Defence departments and defence contractors; further distribution only as approved
( ) Defence departments and Canadian defence contractors; further distribution only as approved
( ) Government departments and agencies; further distribution only as approved
( ) Defence departments; further distribution only as approved
( ) Other (please specify):

12. DOCUMENT ANNOUNCEMENT (Any limitation to the bibliographic announcement of this document. This will normally correspond to the Document Availability (11). However, where further distribution (beyond the audience specified in (11)) is possible, a wider announcement audience may be selected.)

Unlimited

13. ABSTRACT (A brief and factual summary of the document. It may also appear elsewhere in the body of the document itself. It is highly desirable that the abstract of classified documents be unclassified. Each paragraph of the abstract shall begin with an indication of the security classification of the information in the paragraph (unless the document itself is unclassified) represented as (S), (C), (R), or (U). It is not necessary to include here abstracts in both official languages unless the text is bilingual.)

The aim of this work is to increase the capabilities of position report databases, like DRDC's Maritime Situational Awareness Research Infrastructure (MSARI) or the Global Position Warehouse (GPW). For this purpose, a position report is a record of identity, location and time that is reported by a particular vehicle, usually a ship, in order to indicate its own position at a particular moment. The Automatic Identification System (AIS) provides most of the position reports in the databases cited above, but Vessel Monitoring Systems (VMS) and Long Range Identification and Tracking (LRIT) are also prominent sources. Position report databases already allow for a wide range of queries based on characteristics of the reports themselves. DRDC Atlantic is interested in providing the ability to construct and manage *interpolation queries* that rely on inference about what happened between position reports. The goal of this contract was to construct fundamental infrastructure, data structures and algorithms that are required for such query-building. The long term goal of this work is to make it easy for operators with little database training to construct *interpolation queries*. In support of these goals, an Interpolation System was developed to con-vert AIS position reports into *tracks*, which are then stored in an Interpolation Database. These *tracks* are so constructed as to avoid crossing land obstacles that are defined using polygons. This obstacle avoidance capability is the major novelty of the work.

Les travaux visent à rehausser les fonctions des bases de données des comptes-rendus de position, comme l'infrastructure de recherche en connaissance de la situation maritime (MSARI) ou l'entrepôt de données du système mondial de localisation (GPW) de Recherche et développement pour la défense Canada (RDDC). Aux fins de ces travaux, on entend par compte-rendu de position tout enregistrement de l'identité, de l'emplacement et de la date et l'heure transmis par un véhicule particulier (le plus souvent un navire) afin d'indiquer son emplacement à un instant précis. Le Système d'identification automatique (SIA) fournit l'essentiel des comptes-rendus de position versés dans les bases de données mentionnées ci-dessus, doublé d'autres sources comme le Système de surveillance des navires (SSN) et le programme d'identification et de suivi à distance des navires (LRIT). Les bases de données des comptes-rendus de position prennent déjà en charge bien des types d'interrogations, en fonction des caractéristiques de ces comptes-rendus. RDDC Atlantique cherche à permettre de créer et gérer des requêtes par interpolation, qui seraient fondées sur des inférences sur ce qui s'est passé entre les comptes-rendus de position. Le contrat visait à établir l'infrastructure, les structures de données et les algorithmes de base nécessaires pour créer de telles requêtes. À long terme, les travaux visent à permettre même à des utilisateurs sans formation poussée sur les bases de données de créer des requêtes par interpolation. C'est pourquoi nous avons développé un système d'interpolation qui convertit les comptes-rendus de position en ≪ pistes ≫, puis les enregistre dans une base de données d'interpolation. Ces pistes sont créées de façon à contourner des obstacles (îles et masses continentales) définis par polygones. Cette fonction de contournement est la principale innovation des travaux décrits.

14. KEYWORDS, DESCRIPTORS or IDENTIFIERS (Technically meaningful terms or short phrases that characterize a document and could be helpful in cataloguing the document. They should be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location may also be included. If possible keywords should be selected from a published thesaurus. e.g. Thesaurus of Engineering and Scientific Terms (TEST) and that thesaurus identified. If it is not possible to select indexing terms which are Unclassified, the classification of each should be indicated as with the title.)

interpolation
AIS
maritime situation awareness
database

Defence R&D Canada

Canada's Leader in Defence
and National Security
Science and Technology

R & D pour la défense Canada

Chef de file au Canada en matière
de science et de technologie pour
la défense et la sécurité nationale

DEFENCE DÉFENSE

www.drdc-rddc.gc.ca