UNIVERSITY OF TROMSØ UIT

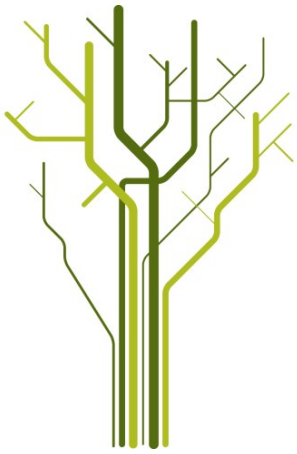
FACULTY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF DEPARTMENT OF COMPUTER SCIENCE

Feature Detector: a support system for tracking satellite detected dynamic and permanent features

Joakim Jacobsen

INF-3990 Master's Thesis in Computer Science

May 2013



Abstract

Even with today's technologies many tasks relies on humans to be completed correctly. Engineers must monitor steps in large chains of operations, and verify the results before the next process is allowed to continue. In many such systems, a lot of useful data passes by without ever been stored for efficient future usage. Even though some operations must be verified by an experienced human eye, many, if not all, could benefit from computed assistance.

When processing satellite imagery there are a lot of steps involved before there is a final product. This thesis examines one specific part of the process, how to determine if a feature observed is permanent or not. A self learning geographical information system implementation that can determine the state of specific features will be presented. The system is capable of filtering out permanent installations from vessel traffic in highly dense areas.

Further we'll see that with such a system at the core, other useful functionality can easily be extended on top of it. Such functionality could be tracking of vessels, oil spills or ice floes, the latter two which have been implemented.

With such a system at hand, the day to day tasks of engineers monitoring satellite observations can be made easier and less error prone. In addition such a historical view of the data can help with improving existing services as well as those still under development.

Acknowlegements

First and foremost I would like to thank everyone at KSAT, especially my supervisor Gudmundur Jøkulsson, for always being available and helping out with problems along the way. I have learned a lot about the world of geomatics which I was completely oblivious to.

I would also like to thank my fellows students, teachers and the administration at UiT.

Big thanks to my loving and supporting family, my girlfriend, and all you other guys!

Keep it lean and keep it clean!

Contents

List of Figures

List of Tables

List of Acronyms

GML Geographic Markup Language 6

KSAT Kongsberg Satellite Services 1

SAR Synthetic Aperture Radar 1

NPD Norwegian Petroleum Directorate vii

GIST Generalized Search Tree 14

GIS Geographic Information System 2

WGS World Geodetic System 21

CRS Coordinate Reference System 21

MBR Minimum Bounding Rectangle 21

DOM Document Object Model 12

Glossary

dynamic feature is a feature changing geometry over time, both shape and position 2

EO-service is a Operational monitoring service based on earth observation data 2

feature is an object or a natural phenomena 2, 14

moving feature is a feature changing spatial position over time 2

shapefile is a geospatial vector data format for geographic information system software 7

Chapter 1

Introduction

Even with today's technologies many task relies on humans to be completed correctly. Engineers must monitor steps in large chains of operations, and verify the results before the next process is allowed to continue. In many such systems, a lot of useful data passes by, without ever been stored for efficient usage in the future. Even though some operations must be verified by an experienced human eye, many, if not all, could benefit from computed assistance.

Kongsberg Satellite Services (KSAT) is a commercial Norwegian company, uniquely positioned to provide ground station and earth observation services for polar orbiting satellites. Among the services provided are offshore observation, mostly in the north, but still expanding. For the Norwegian coastline the Norwegian Petroleum Directorate (NPD) releases public data on all offshore installations. This extra information is useful when evaluating Synthetic Aperture Radar (SAR) imagery, but unfortunately such information is not available world wide.

When processing a SAR imagery there are a lot of steps involved before the customer receives the final product, depending on the service to be provided these procedures vary. When delivering reports on vessel traffic, one of the procedures in the process-chain is to distinguish all permanent objects observed from moving ones. As of today these tasks rely completely on human experience with no historical-data-support.

This thesis presents Feature Detector, a self learning Geographic Information System (GIS) that can digest a variety of data structures into a spatio-temporal database. The system can determine the state of features observed, which could be permanent offshore installations or moving vessels.

1.1 Problem Definition

The general objective is to develop a system for identification and tracking of moving features and dynamic features, detected by analysis of satellite imagery.

The features in question are primarily man-made maritime constructions such as vessels and off-shore installations, but also natural ocean phenomena such as ice floes and oil-slicks.

The goal is to create an information system to assist earth observation analysis procedures in KSAT maritime EO-services. To achieve this, the following sub-objectives are defined.

1.1.1 Moving features with static shape

1. Design a data storage model for moving spatial-temporal objects, geometric objects that change position over time.
2. The model shall support temporal topology relations such that a history of an object's movements over time are conserved (tracking).
3. The model shall support "ageing" strategy for detected objects – taking into consideration the history of observation events per area.
4. The storage model shall support spatial and temporal indexing for search and retrieval.

1.1.2 Feature matching when adding new detection data

1. Define rules for matching a new detection with existing objects, for example based on spatial and temporal distance, and object geometry.
2. Define rules for ageing and for objects to become obsolete (obsolete "old" objects).

3. Define rules for classification of objects as “permanent”, “static”, or “moving”.

4. Develop ingest functions for new detection datasets, applying, the defined rules.

5. Develop functions to add information about detected objects correlated with vessel traffic information (i.e. adding absolute object ID and temporal relations).

6. Develop functions to extract objects per area, time and classification(for example “static”).

1.2 Scope and Limitations

The thesis focuses on the design of the data model and the algorithm for detecting permanent installations. The design should be suitable for easy insertion and extraction of data related to the detection algorithm. The data model should keep and abstraction level to suit other types of data structures such as oil-spill collections and ice-floes collections. Ingest functions are implemented for the latter two to demonstrate the database model, the focus will not be on complex ingestion functionality for a variety of data types. The reason for this is the current state of the ice-floe data collections, which can be considered under construction.

1.3 Method and Approach

Three paradigms divide the discipline of computing [9] (i) Theory, (ii) Abstraction, and (iii) Design.

Theory is based on mathematics and follows four steps for developing a coherent, valid theory:

1. Characterize objects of study (definition)

2. Hypothesize possible relations among them (theorem)

3. Determine whether the relationships are true (proof)

4. Interpret results

The abstraction paradigm is an experimental scientific method, and is used to investigate a phenomenon based on the following four steps:

1. Form a hypothesis
2. Construct a model and make a prediction
3. Design an experiment and collect data
4. Analyse results

The design paradigm is founded in engineering and follows four steps to form the basis for constructing a system aimed at solving a problem:

1. State requirements
2. State specifications
3. Design and implement the system
4. Test the system

This thesis will be based on the design and abstraction paradigm. First, implement a system that supports the insertion and extraction of necessary data types. Secondly, use the system to implement a algorithm for ingesting data in a manner corresponding with the problems defined. The system will in turn be tested to see if the requirements are met.

1.4 Outline

Chapter 2 provides background information on data used and some related work.

Chapter 3 describes the design and implementation of Feature Detector, a data model and its matching algorithm.

Chapter 4 presents tests and results

Chapter 5 summarizes the work done.

Chapter 2

Background and Related Work

KSAT delivers earth observation services based on feature extraction and analysis of satellite imagery, primarily maritime services using SAR images over the ocean. The analysis performed for current operational services is typically feature extraction based on a single SAR image, assisted by external datasets such as met-ocean data, vessel traffic information, and off-shore installation maps. Observation services in general are evolving from ”feature detection” towards ”change detection”. This implies that in addition to alerting (oil-spill near land, ice approaching oil-rig) one uses repeated observation (new images over same area) to maintain information on changes of the observed feature over time. At KSAT there is current activity in a few of the service areas related to tracking objects and change detection over time:

- develop automated tracking of detected vessels between observations, and to separate vessels from semi-static off-shore installations.
- develop analysing methods for change detection for oil-spills between observations, in the time frame of a few observations (days).
- develop analysing methods for change detection for ice floes between several observations, ultimately collecting information about long-term trends in ice movement (seasons/years).

Given the nature of satellite based earth observation, where observations are merely irregular snapshots in time and space, the data model needed to build

feature tracking and change detection history requires some specialization as opposed to plain spatial data models or tracking systems based on regular observations and/or continuous sensor data. To facilitate the development we see the need for a specialized feature storage system, based on a data model to support the special temporal topology attributes needed to maintain the detection history of a given feature. The feature store will be based on a state-of-the-art spatial database system, specialized for the purpose of a data model for satellite detected dynamic features. In addition we see the need for special data ingestion functions, to perform simple matching/correlation of moving features upon ingestion, and for ingestion of more complex datasets representing the result of analysis of change in dynamic features, including complex temporal relations such as merging and splitting features.

2.1 Datasets Involved

2.1.1 Vessels and maritime installations

When a SAR image is processed the resulting product is referred to as a ”observation” or ”package”. Each package consists of a XML meta data file, and one or more Geographic Markup Language (GML) data files. The packages is structured as in figure 2.1.

The XML files contains information about the observation, such as package name, time of observation, which satellite was used and collection type(ship, oil etc.)

The GML files contains information on all the features the observation detected. Each feature has a set of attributes, some are; detection time, feature type, length, width, heading and position.

The dataset used spans over a period of just over two months, more precisely from January 1. through Mars 5. It contains a total of 473 observations and 144566 features.

2.1.1.1 Confidence Estimates

All features in the Vessel-GML files are given one of the confidence estimates, low, medium or high. The implementation is based on equation

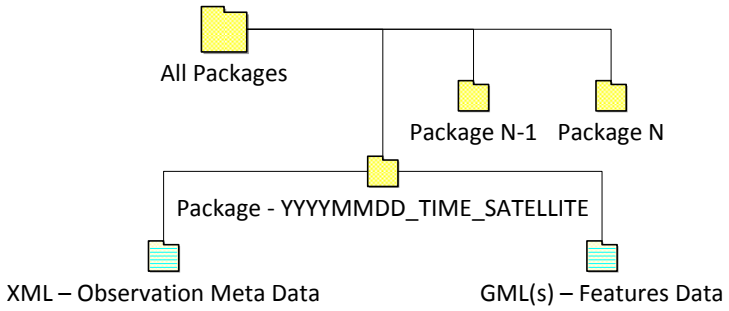


Figure 2.1: Package Folder Structure

14 in [8]. The confidence estimate says something about the certainty of presence of the feature reported to be observed. One of the methods applied in setting the confidence level is the number of actually observed features weighted against the expected number to observe in that area. If there are much fewer features observed then expected, they are given a higher confidence level.

2.1.2 Reference dataset from NPD

In order to verify the correctness of the implementation a reference dataset [2] from Norwegian Petroleum Directorate (NPD) is used. The dataset consists of shapefiles with corresponding meta data. Figure 2.3 displays an example map generated from NPD's website [3].

These datasets are never ingested in the database, but viewed in a graphical interface together with the processed data. This because the system should be self learning, based on a set of rules of what defines these permanent offshore installations.

2.2 Related Work

2.2.0.1 Spatial Temporal Data

Due to the requirements specified in section 1.1 the systems model must support spatial-temporal relations. The spatio-temporal aspect of this thesis is the problem of tracking features. The problems related to spatial temporal data in [10] differ in terms of objects shapes since they work with static

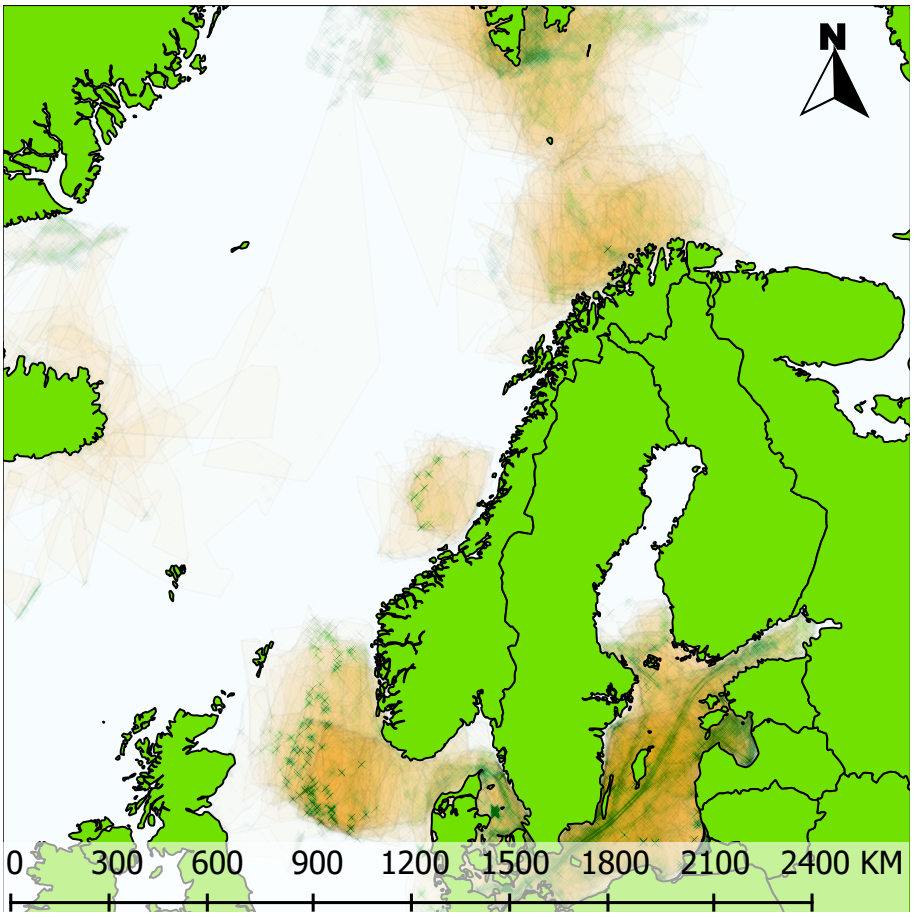


Figure 2.2: Overview of all observation bounding boxes and the detected features(x). The darker areas contains the most features.

shape objects, such as a parcel, but it takes a similar approach for tracking objects over time.

2.2.0.2 Object Detection

Most of the research related to feature detection and shape recognition are done in the fields of video tracking [11] and image processing. This thesis relies on data with only a single coordinate for a given time. And the time intervals are much longer then in videos.

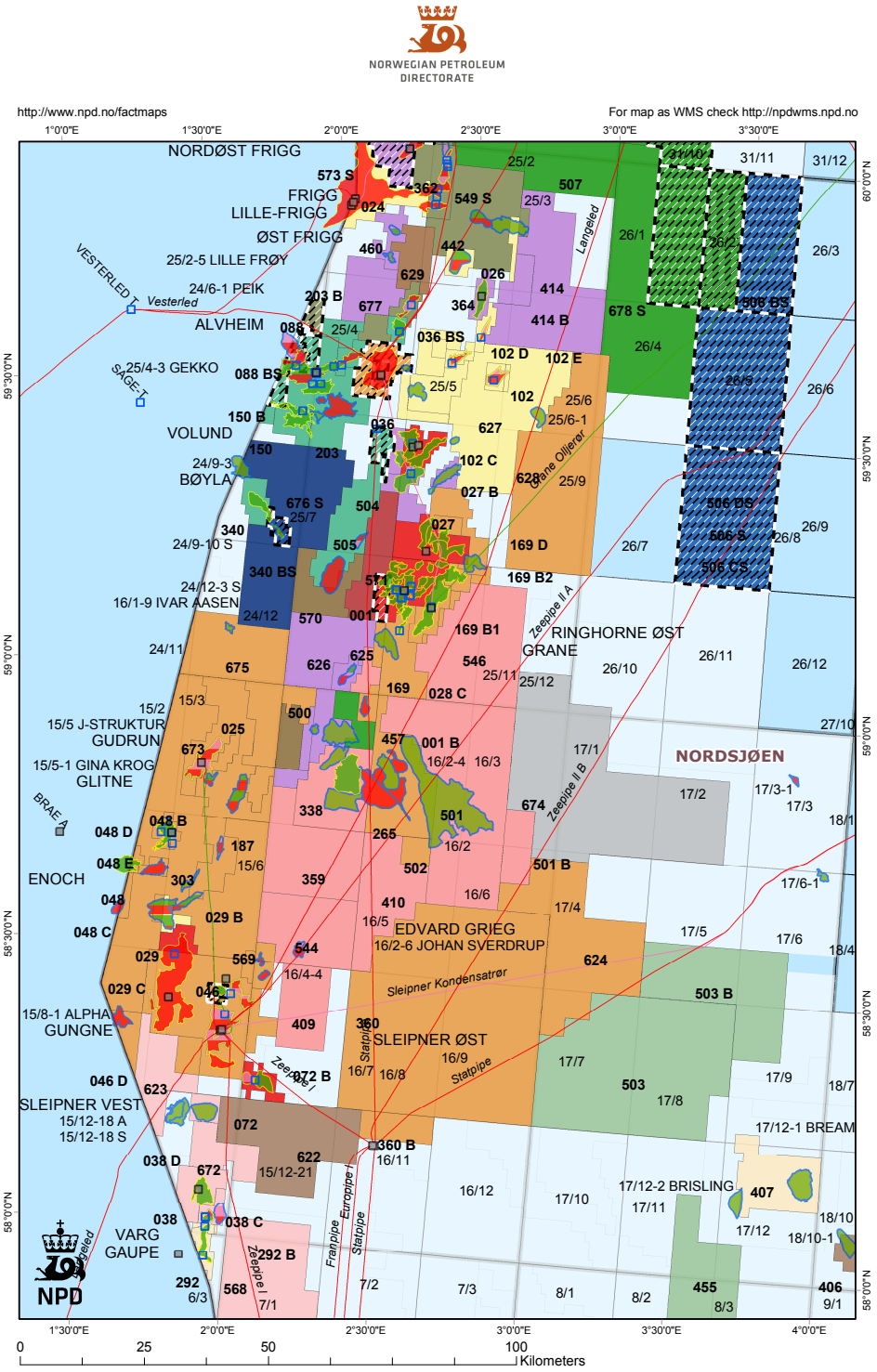


Figure 2.3: Reference dataset from NPD [3]

Chapter 3

Design and Implementation

3.1 Architecture

This chapter describes Feature Detector, a parser and processing system for detecting permanent offshore installations, and classifying moving features, based on satellite imagery.

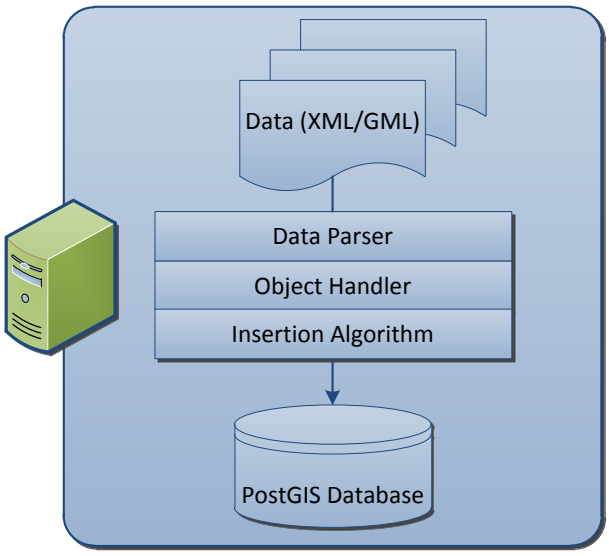


Figure 3.1: System Architecture

The implementation consists of three larger components. From the top; A data parsers, extracting data from XML and GML files. A object handler to get structured data from the parser. The algorithm for inserting and updating data. And at last the underlying database model for storage and queries. The architecture aims to be suitable for extension, which could be parsers supporting different data structures, and insertion of those.

First, a overview of the system design will be presented. Then a jump down to the bottom of the stack to look at the database model in two phases, the initial ideas and design of the system, then the final model solving issues discovered in the first design proposed. At last the insertion algorithm will be presented.

3.1.1 Program Design

Figure 3.2 shows the data flow of the system. The system usage in practice would be to listen for new files in a folder. As of now there are few established data structures that are suitable for automatic parsing and database insertion, the main focus is therefore on packages structured as described in section 2.1.1. This structure also applies for the oil-spill dataset. For now the program is a script that executes and parses a folder for packages. This parser collects the absolute path to all packages and their content, the list of packages is then sorted by date.

A object instance is initiated for every package, the initial input of .XML and .GML files is then passed through a Document Object Model (DOM) parser. The DOM parser fetches all wanted data into memory as a structured package instance. Each in-memory package is a observation, and each observation has a container of all its corresponding features.

At this point the input files are parsed and ready for processing. Next step is the insertion part, this is where the system communicates with the database. There are no structured class mapper between the database and insertion algorithm, the reason for this is that most of the database output back to the program are sets of few variables to reduce the cost of queries. No need to bloat memory with fully contained class instances when only using one or two of the attributes. All communication goes through psycopg2 [6] using standard postgresql queries.

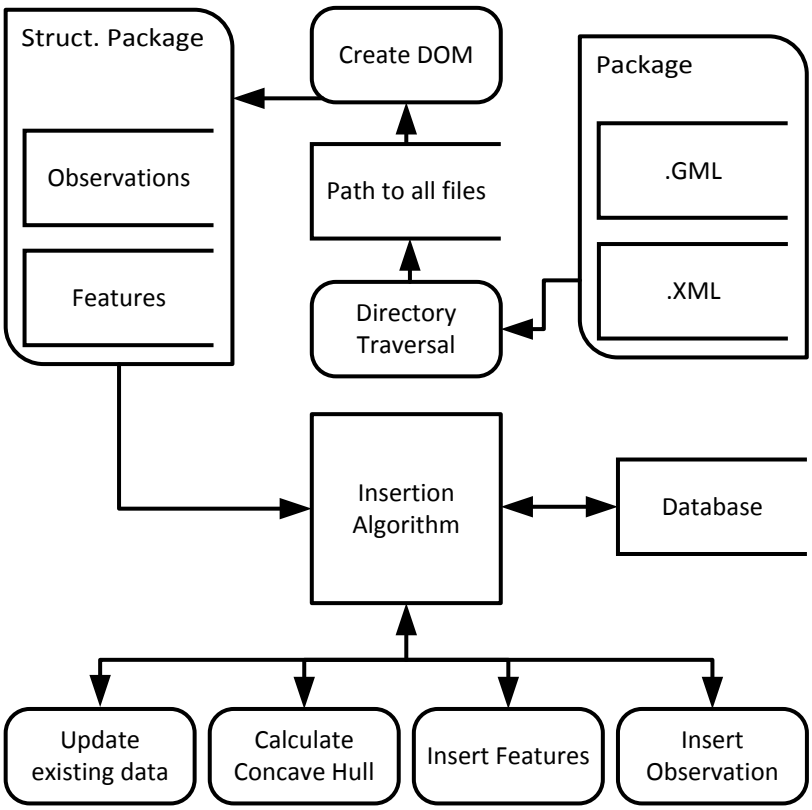


Figure 3.2: Data Flow Diagram

The insertion algorithm goes through four phases in one cycle: i) *Insert Observation* creates a new row in the database for the current observation, ii) *Insert Features* processes all features in the current observation, iii) *Calculate Concave Hull* is a recursive function that queries the database to generate a optimal bounding box for the observation, iv) *Update existing data* queries the database for objects that wasn't touched by the algorithm for this cycle, and updates them. The insertion algorithm will be explained in detail in section 3.3.

3.2 Database Model

There are several database systems that support spatial data types and functionality[1][2]. This system uses PostgreSQL with the PostGIS spatial extension. PostGIS is open source and implements the most common GIS functionally needed. First we will have a look at the "intuitive" model design, before digging deeper into the final model. This will hopefully give a better understanding of the design choices made along the way. Keep in mind that a primary goal is to classify permanent features, while maintaining an abstract design for further extension.

3.2.1 Initial Model

Considering the objectives and requirements for the data storage model outlined in subsection 1.1.1, a initial design was implemented 3.3. These requirements can be summarized as follows.

- Features must know it's position at a given time, which would be it's detection time.
- The model must support some sort of tracking functionality, i.e where was feature f, at time t.
- Every feature has a age, that says something about the amount of times it has or hasn't been observed (see Figure 3.5), relative to time for a given area.
- For spatial indexes, Generalized Search Tree (GIST) is used.

This model is able to store the most necessary data needed to identify and object. It keeps track of features age, for example features that has been observed many times at the same position are given a higher age and could be considered permanent, while features with age under a certain value are considered moving. High age equals high certainty of the claim made.

The model supports indexed geometries, together with time related data and a object relational table(tracks) it's possible to monitor objects

[1] `http://www.gaia-gis.it/gaia-sins/`
[2] `http://www.terralib.org/`

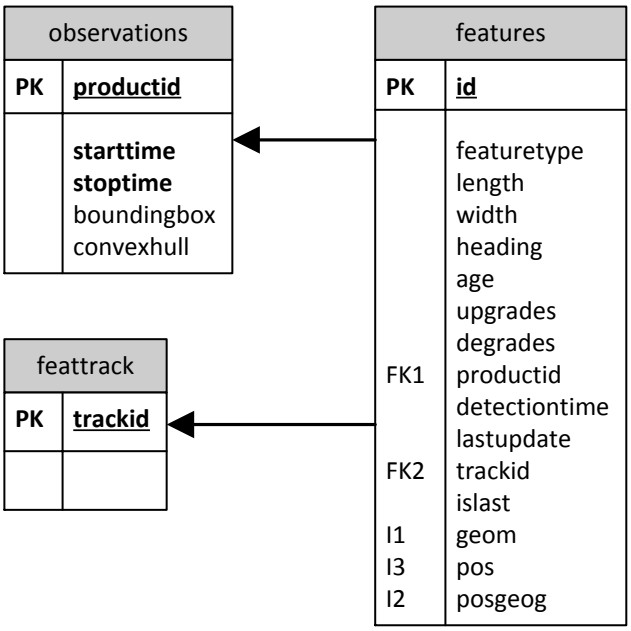


Figure 3.3: The First Database Model

movement and shape over time.

The initial thoughts on ageing and matching was to look for existing features within a given radius, if a match was found, the new feature was never added to the database only the age for the already existing feature was simply increased. The best case scenario then for determining the location of a permanent object would be to look for areas with closely clustered, highly aged features. One could then state that, somewhere among these features, probably the center coordinate of them, there is a permanent installations of some kind. This problem is illustrated in figure 3.4. In addition such an approach would lead to complex SQL-queries when retrieving information, nor would the model maintain the abstraction level wanted for tracking. This because the fact that new matching features was never inserted to the database.

For the second approach, the aspects of tracking was more carefully considered. All features are initialized with *islast* to **true** as default. When searching for matches, only features tagged with *islast* as **true**, are compared. This because, when a match is found and a new track is created,

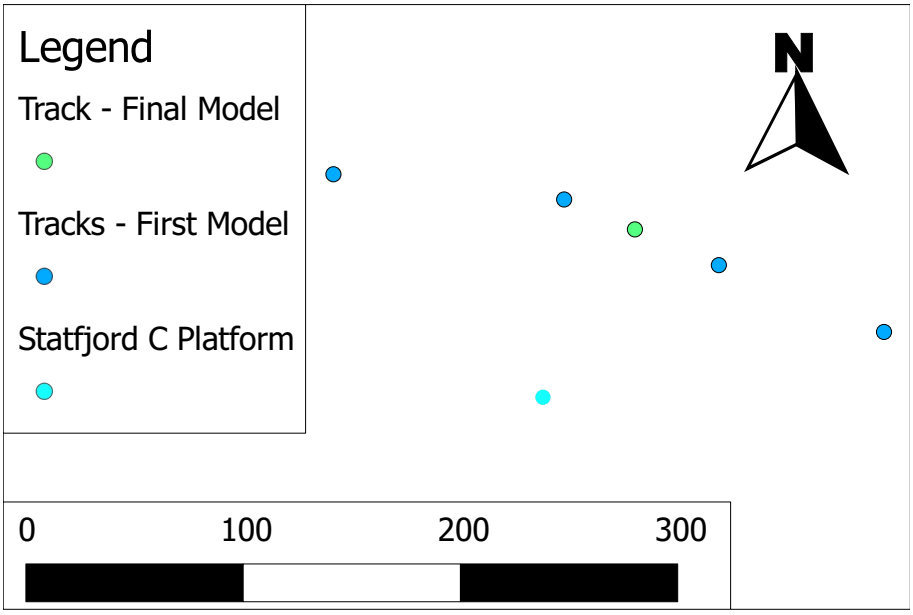


Figure 3.4: Illustration of the clustering problem that occurs with the first model and algorithm where several permanent tracks are formed, instead of one as it should be.

the already existing feature is tagged with *islast* to **false**. A track would then consist of features where all except the last one has it *islast* tag set to **false**. This solution would solve the tracking abstraction that the previous one didn't maintain.

The second approach comes with a side effect that contradicts a key aspect, the possibility to retrieve permanent features. What happens when matching is only done against the last feature in a track, is that the track stretches out from it initial position. The last feature in the track could be a long way from the source. This isn't wrong, because a track will obviously move, but we wont get a reliable results such as in the first approach, when querying for permanent features.

Imagine we do the matching with a radius of R meters, and detected a matching feature f_1 at position (x_1, y_1) and a track is created, containing feature f_1 and f_2. Let's say that f_2's position (x_2, y_2) was R meters east of (x_1, y_1), then after N matches in that track, the last in track could at worst be $N * R$ meters east of the initial position (x_1, y_1).

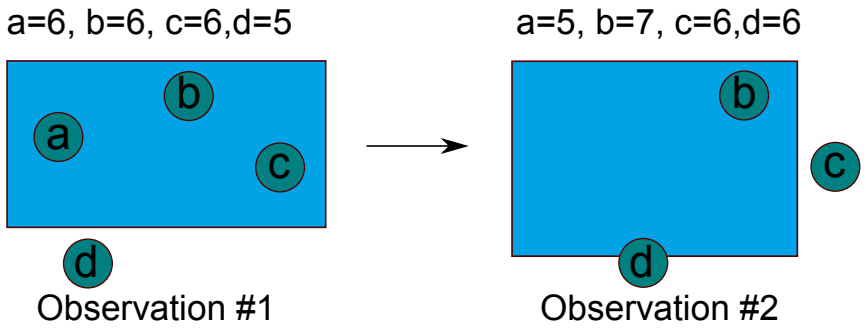


Figure 3.5: The initial ageing mechanism illustrated. A,b,c,d are existing features in the database. The darker squares are observation bounding boxes. All features with a matching position from the new observation increases their age. In observation 2 there is no matching feature at the position of a, and it's age decreases.

Let's summarize. By only increasing the age of a initial feature, and then discarding the matching ones we get a fairly precise but hard to use permanent detection, as well as loosing the possibility of tracking. If we compare matches to the last feature in a track, we never discard information, but as we have seen, the permanent detection isn't reliable.

3.2.2 Final Model

After evaluated the first two approaches of tracking and detection of permanent installations, a new model was designed. The new model sort of combines the two previous ones. It can take advantage of the wast amount of data from observations to determine the location of permanent features without discarding them. And still use the same model to track moving dynamic features.

The idea is to store the information obtained by matches in the track table. By combining the experiences made from previous testing, an improved track table, that serves both as it intuitive usage implies, and a tool for detecting permanent features is introduced.

The final model introduces seven columns to the track table, *lontot* and *lattot* is the total longitude and latitude of features in that track. Together with a total number of features in the track *count*, the *menapos* is updated for every new feature: $(\mathtt{lontot} + \mathtt{lattot})/\mathtt{count} = \mathtt{meanpos}$.

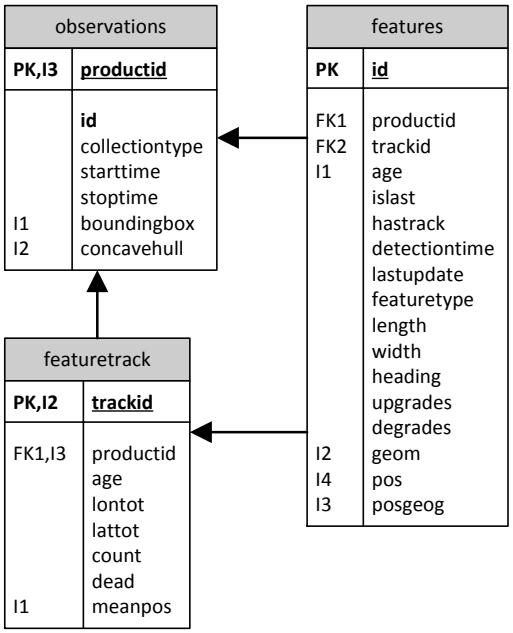


Figure 3.6: The Final Database Model

Meanpos, meaning mean position of all features in the track, is the key here. Instead of only doing matching against features, the algorithm looks for matching tracks first, then features. In addition to previous mentioned attributes of the new tracking table, there is age. Since a track (in terms a of tool to determine what are permanent) is a entity generated by features ”passing by”, the age attribute is inherited from the last feature in the track. Together with the count column, holding the number features that has ”contributed” to the track's state, there are now two parameters to evaluate the confidence of a permanent installation.

In section 3.2.1 a brief description of the mechanics in the algorithm related to the database model is given. We have seen how the model design have evolved based on lessons learned from the first design, and a final model has been presented. In the next chapter a detailed description of the insertion algorithm will be presented.

3.3 Insertion Algorithm

Object recognition are much easier for the human eye then it is for software. We can relatively easily follow an object, frame by frame, while it changes shape and rotates, and confidently state it's the same. Today most object recognition or feature detection software rely on low interval image captures, and objects with distinctive shape and color.

For this thesis neither of the two criteria are fulfilled. An object is only identifiable by it's longitude/latitude at a given time. These observation, or snapshots in time, are collected at most twice a day for the same area. In addition, the SAR can only guarantee a correctness within a diameter of 500 meters. That means, given a object O at a position x,y, O could really be at any position within a radius of 250 meters. With this in mind, and a improved tracking model at hand, we will see how to improve matching of permanent features. Remember that this algorithm is designed for the insertion of data containing vessel detection information, not oil-slicks and ice floes.

3.3.1 Key Features

A brief description of PostGIS functionality [5] applied in the algorithm.

DWithin returns true if the geometries are within the specified distance of one another. For geometry units are in those of spatial reference and for geography units are in meters and measurement is defaulted to use spheroid=true (measure around spheroid), for faster check, use spheroid=false to measure along sphere.

Distance for geometry type returns the 2-dimensional Cartesian minimum distance (based on spatial ref) between two geometries in projected units. For geography type defaults to return spheroidal minimum distance between two geographies in meters.

Overlaps returns TRUE if the Geometries share space, are of the same dimension, but are not completely contained by each other.

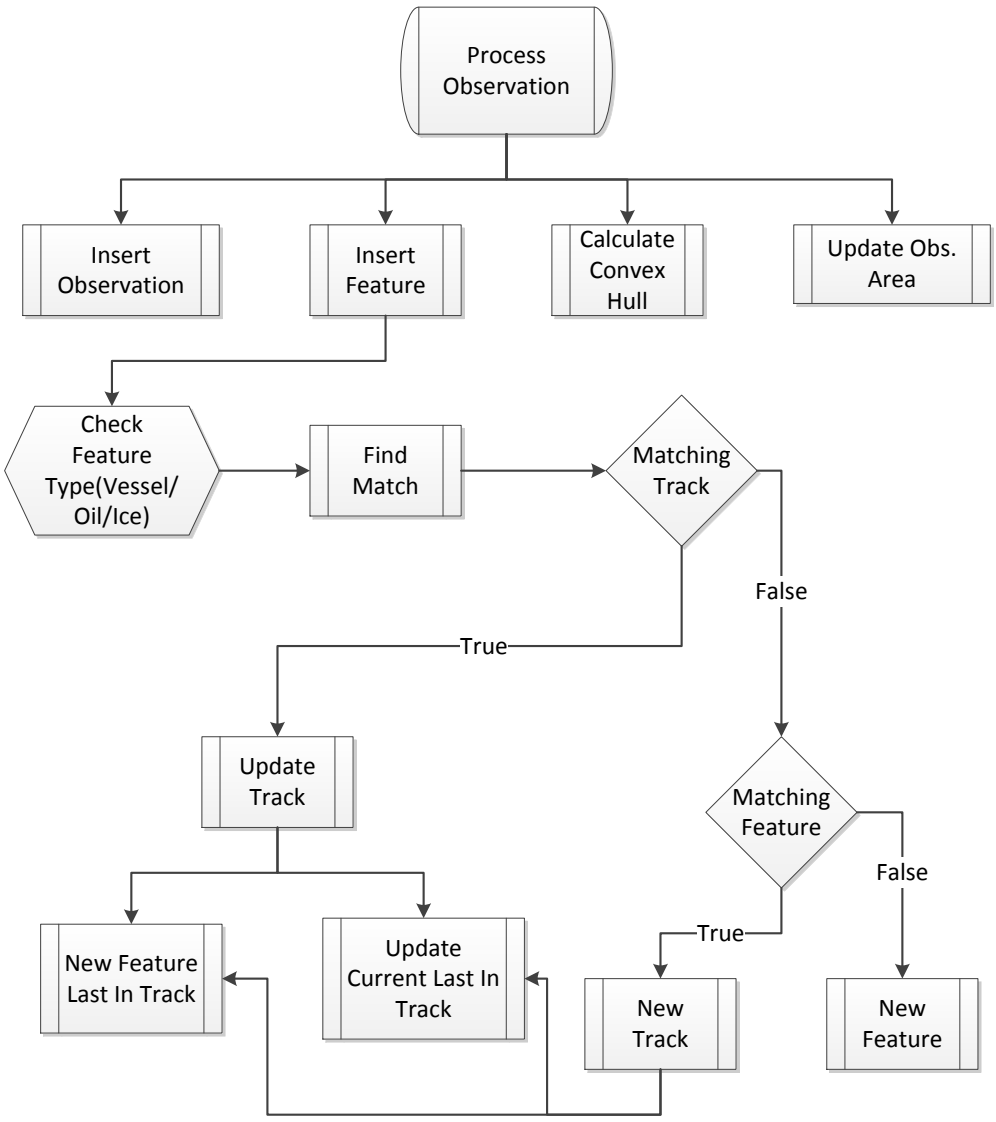


Figure 3.7: Process Observation

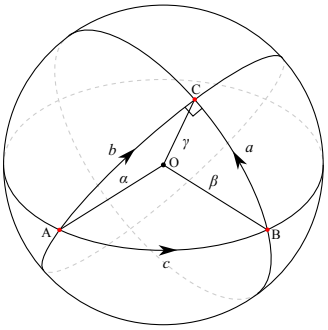


Figure 3.8: Spherical Triangle

3.3.1.1 Coordinates

Coordiantes are a key concept in every Geographic Information System (GIS) database. In PostGIS there are two different data types for storing coordinates, namely ”geometry” and ”geography”. Geography type features are always stored in World Geodetic System (WGS), latest revision, WGS84. Measurements based on geography features will be in meters instead of Coordinate Reference System (CRS) units and PostGIS will use geodetic calculations instead of planar geometry. Doing calculations with the geography type is more expensive, but gives a much more accurate result.

3.3.1.2 Bounding Boxes

Minimum Bounding Rectangle (MBR) A MBR is the simplest form of a bounding box. It creates a square with the minimum and maximum (x, y) values of the coordinates within as it's corners.

Convex Hull takes a MBR and tries to shrink it. It can be compared to putting a rubber band around the collection of points.

Concave Hull produces the smallest polygon of the three bounding boxes. In PostGIS the degree of shrinkage can be adjusted [1].

Given a observation a MBR and Convex Hull is generated before it is passed though the insertion algorithm because it cost less to do it at this stage. There are on the fly functionality integrated in PostGIS to generate this bounding boxes, but its more efficient to pre generate them and use indexes. The one exceptions is the Convex Hull, which is generated after a observation has processes all its features. The reasons for generating the

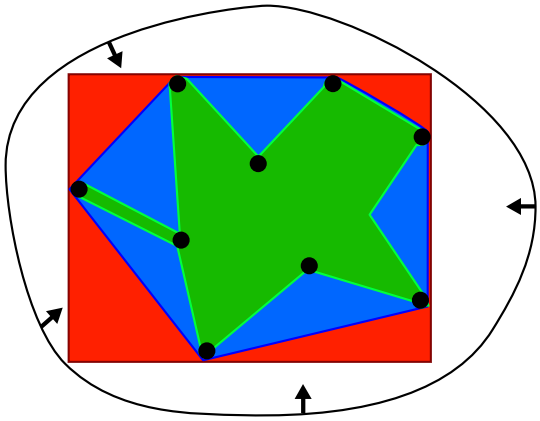


Figure 3.9: Bounding Boxes. Red: MBR, Blue:Convex Hull, Green:Concave Hull.

Concave Hull at this stage, is simply because it is a PostGIS aggregate function, so it needs a set of points(feature coordinates). Some observations contains to few features to create a Convex Hull, which are considered unreliable, and are therefore discarded for testing purposes.

3.3.2 Basic Runtime

Figure 3.7 shows the flow of the insertion algorithm. At the top "Process Observation" receives a package(an observation and all it's features) and creates a timestamp: *current observation time*. Note that *current observation time* is the local time of when the package is processed, not when the SAR scan was performed. Let's say this is the first feature in the first observation, so there will be no matching track or feature. The new feature is simply added to the database, and tagged with another timestamp: *lastupdate*=current time. The algorithm iterates all features in the same fashion. When there are no more features it queries the database to get a concave hull bounding box, or a different one if the concave hull bounding box isn't available. The current observation area is then updated: All features within the current observation's bounding box, with a *lastupdate* timestamp set before the *current observation time*, meaning all previously added features that wasn't updated in this cycle are degraded.

3.3.3 Searching for matching features

Traditionally when referring to a track in the context of GIS, we assume it to be something like a GPS track that keeps the history of a hikers route

or where you have driven the last week with your car. This is also true for this model, but in addition, well take advantage of the information we have to determine where there possibly are permanent objects.

We have seen how to handle new features having no existing features matching it's position. But what happens if a feature finds one or more existing features within a given radius of itself? A new track is created the first time an existing feature is matched. If the query returns several matches, the closest one is chosen. The new track is given a *Trackid* equal to the first feature in it. A mean position of the track is calculated based on the position of it's members. This is done by storing the total latitude and longitude, and number of members. This mean position i used when querying for matching tracks.

The order of matching is tracks first, then single features. Matching is either or, it wont make sense for a new inserted feature to create a new track, and also join an existing one. So why this particular order of the two tests? Let's consider the opposite one, a new feature f_1 is inserted for processing and finds a feature f_0 and they form a new track t_1. The process finishes and we evaluate the results. It turns out that within our matching radius R where track t_1 was created, there already where a track t_0. Now there are two tracks, namely t_1 and t_0, where there really only should be one.

3.3.4 Ageing and Classification

The life of a feature is decided by the age variable. If age goes below zero, the feature is treated as dead. A dead feature is in reality either a moving feature, most likely a vessel, or it can be a false observation by the satellite. The higher the age, the more certain the stated classification is.

There are several operation and parameters in the algorithm that influences the age-weighting. First there is an initial age, that is the same for all features. Secondly the confidence estimates(low,medium and high) can increase the starting age by one two or three. The initial age, and confidence estimate together decides the starting age of a feature. Some observation consists of images processed in multiple polarization channels, namely horizontal and vertical transmit and receive. Denoting the transmit and receive polarizations by a pair of symbols, a radar system using H and

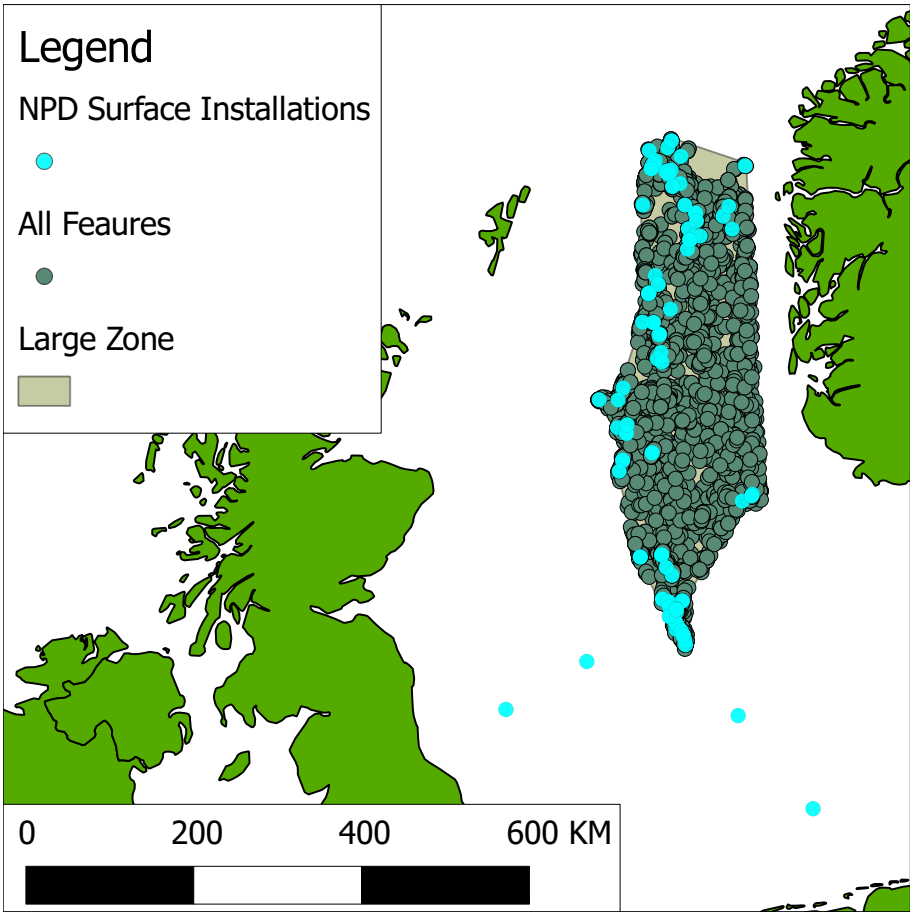


Figure 3.10: Area with NPD installations and all unprocessed features that fall within that area.

V linear polarizations can thus have the following channels:

- HH - for horizontal transmit and horizontal receive
- VV - for vertical transmit and vertical receive
- HV - for horizontal transmit and vertical receive
- VH - for vertical transmit and horizontal receive

When a features is processed for insertion, the algorithm checks if the current observation already have processed another feature at the same position. Note that this is done BEFORE any interaction with the database is initiated. This is done with a simple (key,value) store check, if there

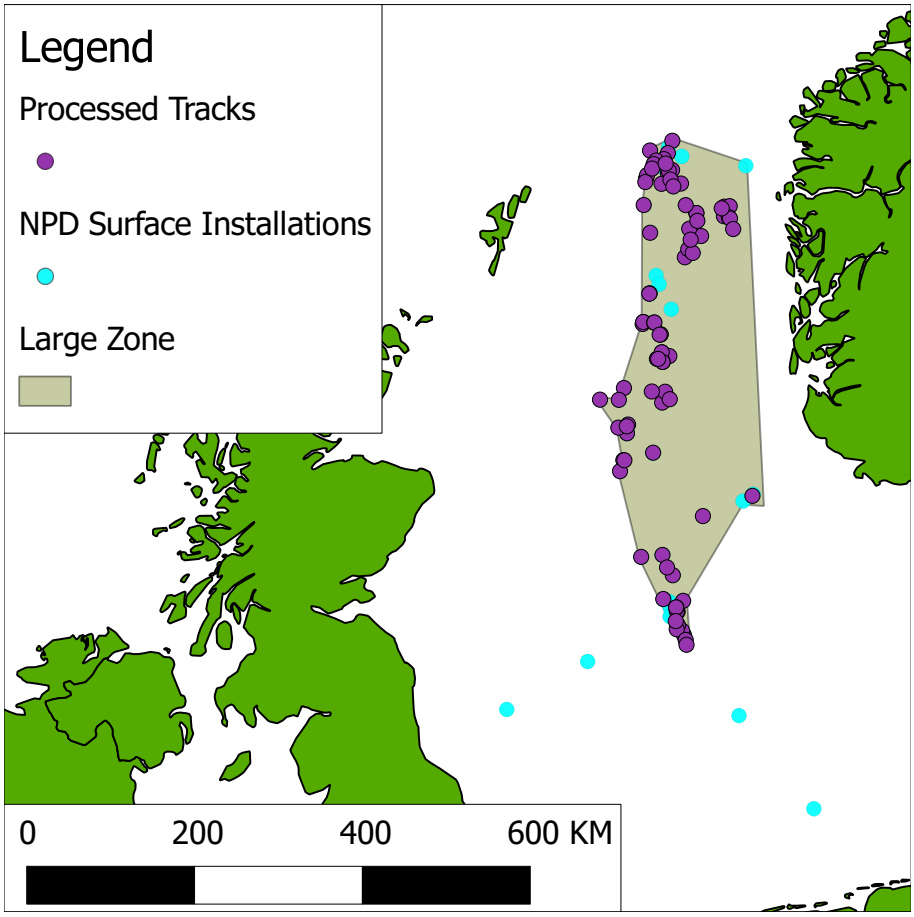


Figure 3.11: Results of processing features with the insertion algorithm. Remaining tracks with more then 5 contributors are here displayed, moving features are degraded to age blow zero.

is a match, that feature has been observed using a different polarization channel. For each extra polarization channel, the confidence level of the current feature is added to the starting age(initial age + confidence level) of the feature that first was inserted in that cycle.

Given an example run with a initial age of 10. A feature has a confidence estimate set to medium, giving it a starting age of 12($10 + 2 = 12$). Next that feature is injected to the algorithm, which later in the cycle finds a feature with a confidence level of 3 at the exact same coordinate, giving our initial feature a new starting age of 15. This ensures that objects detected with heigh confidence is given more weight, and isn't treated as a moving object that easily.

Another rule worth noting when it comes to ageing is how the age is passed along the features within a track. When a new track is formed, or a new feature is added to a track, we want the age of the track to increase, making the possibly of a permanent installation stronger. The new feature inherits the current last in tracks age, its given an extra age point for the match, and the new features starting age subtracting the initial age. Example when new feature is added to track with a initial age of 10: current last in track has an age of 24, and starting age of the new feature is 15. When the new feature becomes the new last in track and inherits the age of the previous one, it age will be: $24 + 1 + (15 - 10) = 30$.

3.3.4.1 Degrading and Bounding Boxes

The last step in the insertion cycle is the degrading process. Given the observation bounding box, all existing features in that bounding box with an earlier timestamp is fetched from the database. While processing the features in the observation an average confidence level based on the features confidence level is given the observation(1,2,or 3). All fetched features age are then degraded with the observation confidence level before the update is committed.

As mentioned there are different possibilities when it comes to bounding boxes. A problem discovered when using MBR bounding boxes was that in certain problem areas, many features where degraded too often. The reason for this were the area the MBR spans would often be to big relative to the features detected in that area. Since the MBR is generated from the minimum and maximum values of it's containing points coordinates, there are large areas where the observation haven't detected anything. The assumption made from this discovery is that, in such an area where nothing is observed, already existing objects within that area shouldn't be degraded either. To avoid such areas smaller bounding boxes are generated when possible.

3.4 Extending the System

The intended usage of the system requires it to be suitable for extension. The model should be versatile so that the process of coupling new functionality doesn't require too much change and time implementing. In

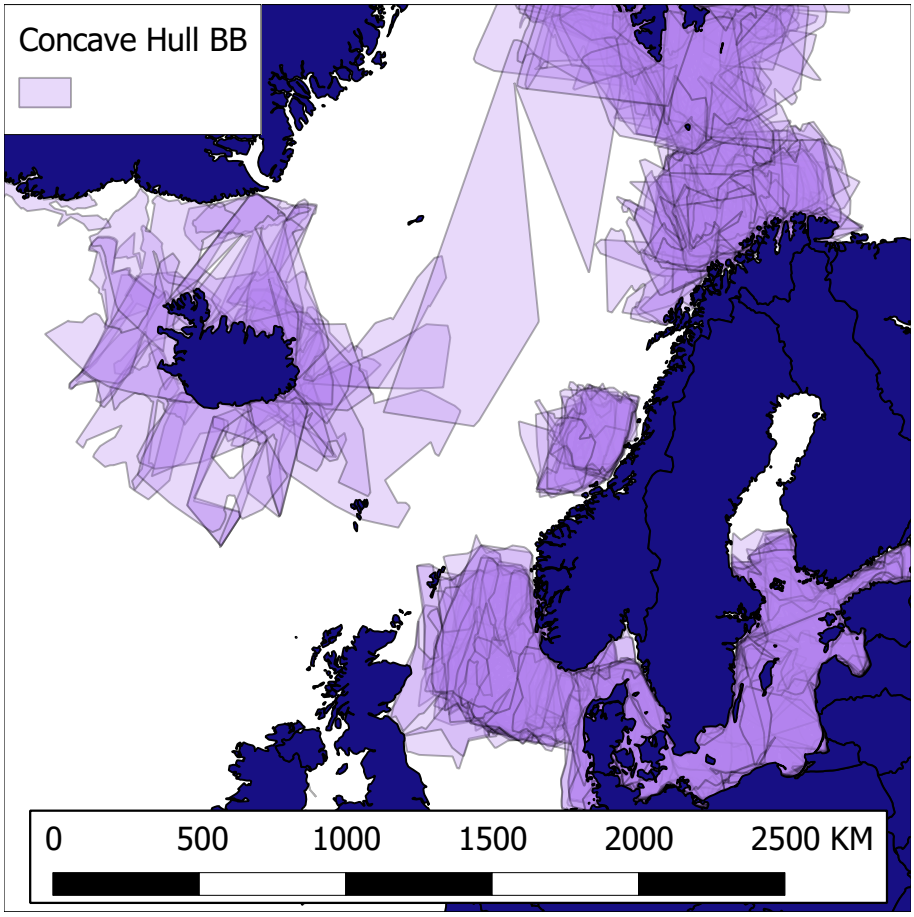


Figure 3.12: Bounding Boxes of type Concave Hull.

time the system could serve as a complete overview of observation done for several services. Included in the work done the support of two new ingestion features is implemented besides the vessel-collection format, namely oil-slicks and ice floes detections.

3.4.1 Oil Spills

The structure of the oil-spill collections are structured in the same way as the vessel data collection. XML-files with meta data about the observation, and GML-files with the collection members. The same file parsers as applied to the vessel data isalso used here. When inserting oil-spill into the database there is no need for the insertion algorithm, since the usage is only for storage purposes. The system supports ingestion of both oil and vessel collection at the same time, meaning their raw-data can reside in the same root folder

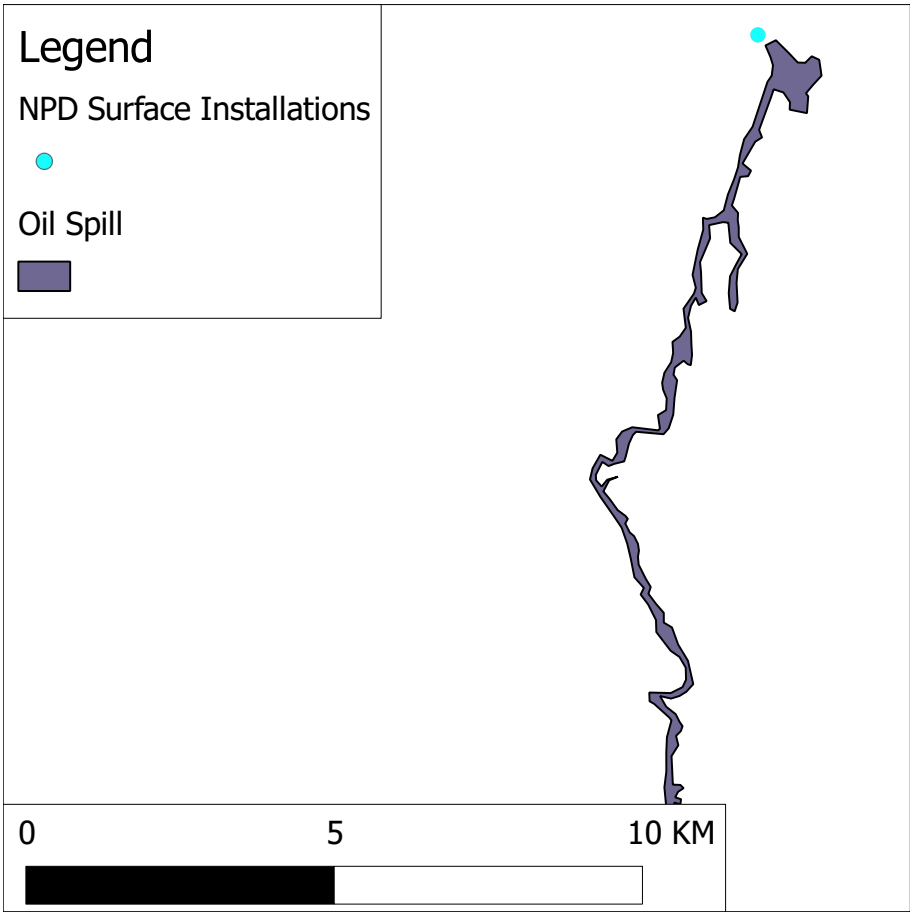


Figure 3.13: Oil Spill

and the system will process it without any problem.

3.4.2 Ice Tracking

The third ingestion functionality implemented is the support for ice-tracking. The ice tracking data comes in form of shapefiles. For the time being the structure of this data isn't complete since the project is under development at KSAT. Therefore only a simple ingest function is implemented to demonstrate the models flexibility.

For parsing the shapefiles a third party parsers is used[1]. The shapfiles contains information on the ice floes detected and possible relations. These relations are between pairs of ice-floes, so when insertion them into the

[1] `https://code.google.com/p/pyshp/`

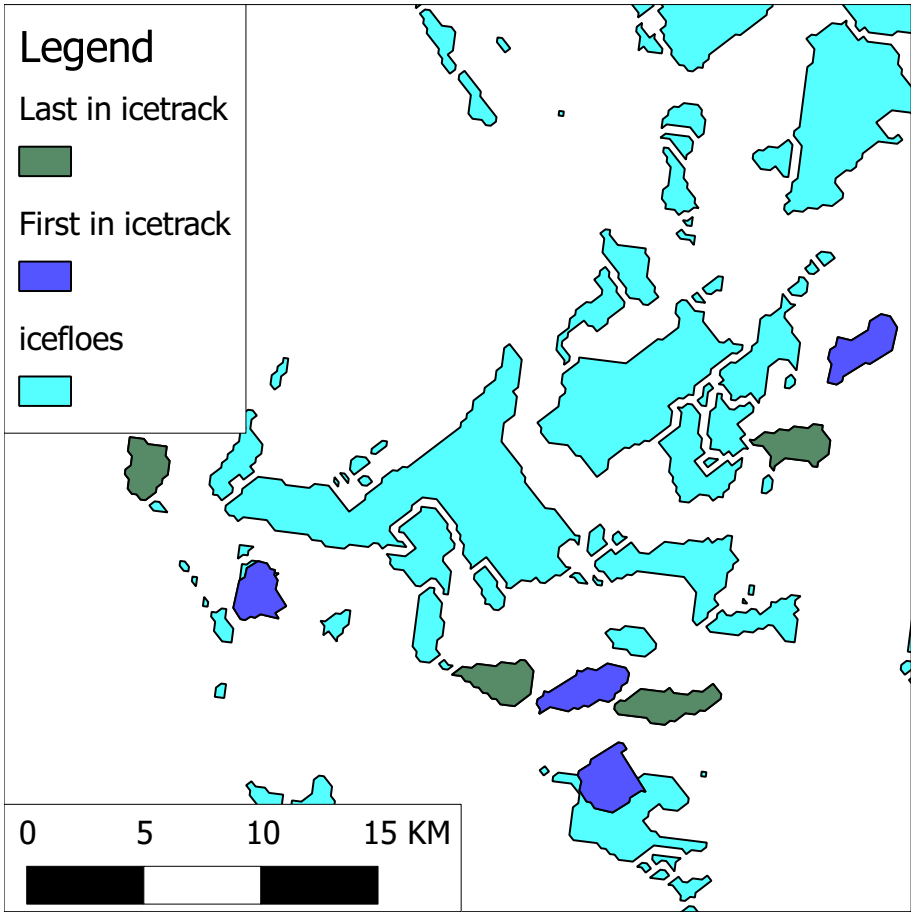


Figure 3.14: Ice-floe tracking

database tracks are generated. One minor change is made to the model for supporting ice-tracking as a result of the structure of the current shapefiles, and that is a new column *UUID* which is a floes identifier. The first detected ice floe becomes first in track, and the second becomes the last. All ice floes in linked to a track with a trackid that equals the uuid of the first in track.

Chapter 4

Experiments

This chapter presents the experience made from experiments performed with Feature Detectors insertion algorithm.

4.1 Experimental Setup

Development and testing is done in the same environment, a Lenovo T61 equipped with Intel Core 2 Duo T7700 running at 2.4GHz and 3GB of RAM.

The computer was running 32bit Windows 7, with PostgreSQL 9.0 and PostGIS 2.0.1. All software is implemented using python 2.7, with psycopg2 [6] as mapper to PostGIS.

For review the results in a graphical interface Quantum GIS 1.8.0-Lisboa is used, together with with the "TM WORLD BORDERS 0.3" dataset from [7] and the NPD dataset [2].

4.2 System Parameters

- The radius a match falls within. Track or feature.
- A length and height buffer the matched *feature* must be within. This affects the ratio of track creations.

- Type of bounding box: MBR, Convex Hull and Concave Hull. This determines to what extent a observation is allowed to degraded untouched objects for that cycle.
- General age increase and decrease rules. How high of a age makes a track permanent or how low makes it dead.
- In addition there are ageing rules based on the confidence estimate of the SAR detection.
- In total here are four ageing parameters; Initial age(default: 5), High confidence(default: 3), Medium confidence(default: 2), Low confidence(default: 1).

When testing the two first parameters listed will be tweaked to compare the influence they have on the results.

4.3 NPD Reference data and Test Zones

The NPD reference dataset used for testing and verification of Feature Detector contains all installations on the Norwegian continental shelf. This includes all surface, sub sea, onshore and offshore installations. For performing tests all non surface and offshore installations are filtered out. The remaining subset is divided into two zones, a large and a small one. The large zone consists of 131 permanent installations and the small zone has 13 permanent installations. Among these installations there are huge oil platforms and smaller loading wells.

The KSAT dataset containing features detected from SAR imagery span a much wider area then the zones used for testing. So for evaluation, only features within the test zones are taken into count. Among the 473 observations and 144566 features, 6141 features lies within the large zone and 665 in the small one. A number of test using different matching criteria are run, and the results will be evaluated.

In addition to the test zones, other areas where information on permanent installations can be obtained will also be used to check the correctness of the Feature Detectors algorithm. There are many observations in areas with offshore wind farms [4] at the Danish coast, two of those will be given a closer look, namely Rødsand I and Rødsand II.

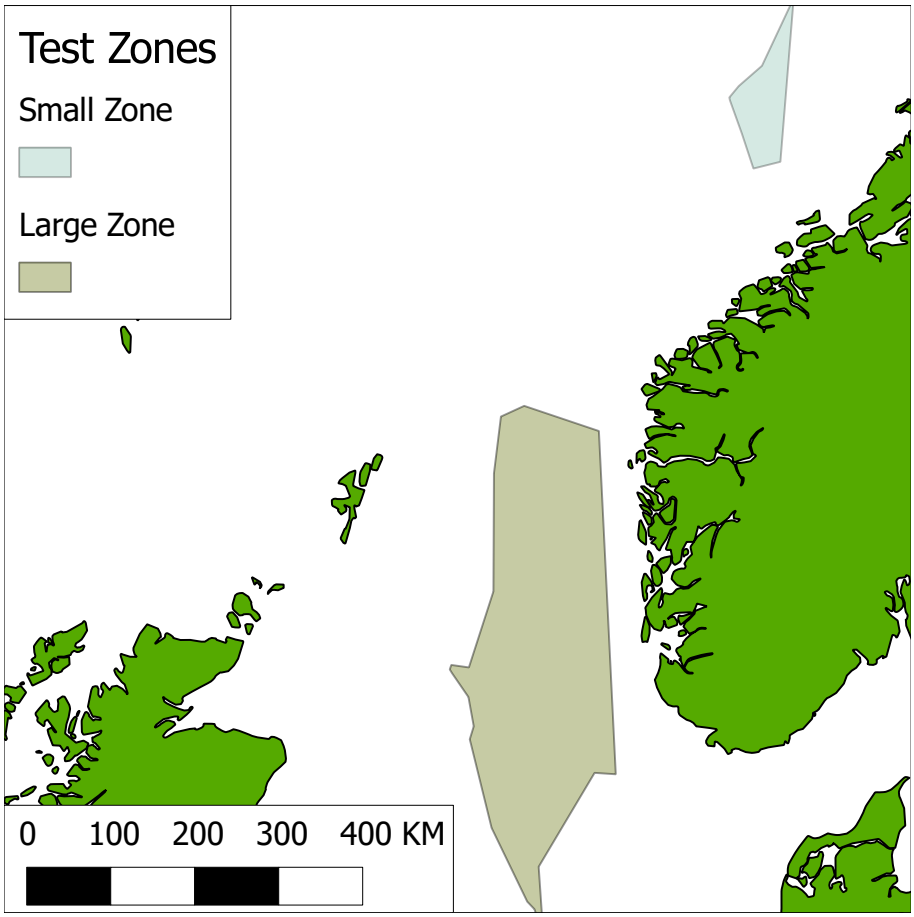


Figure 4.1: Test Zones

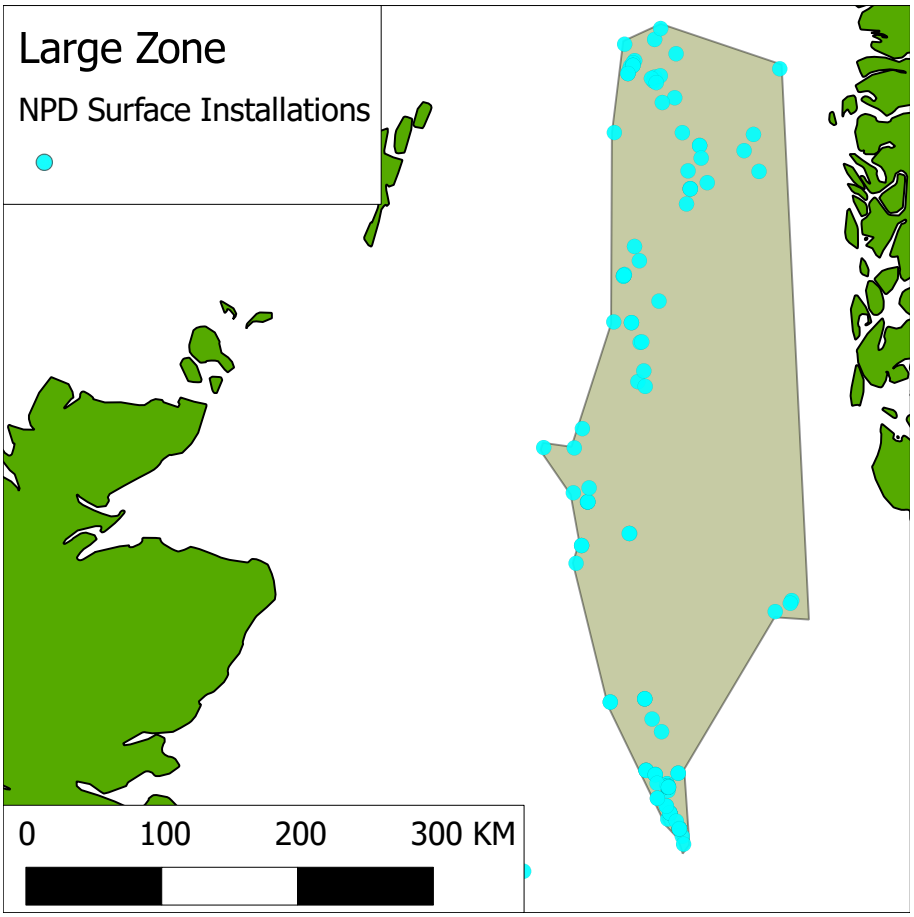


Figure 4.2: Large Zone

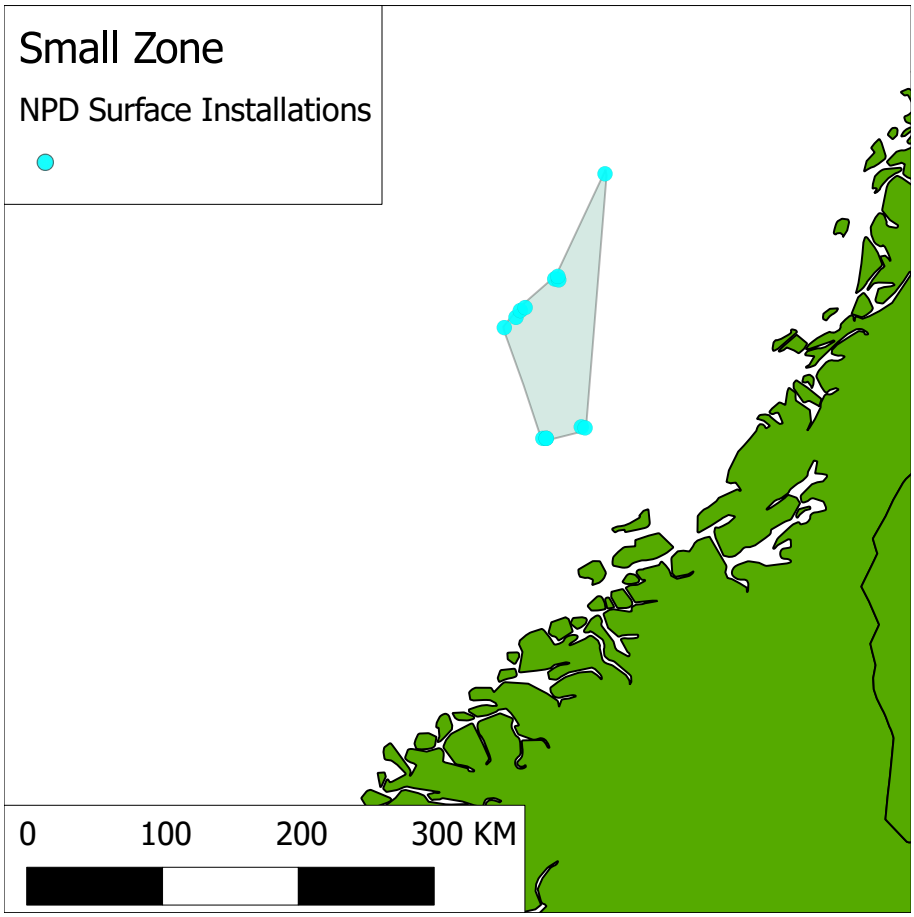


Figure 4.3: Small Zone

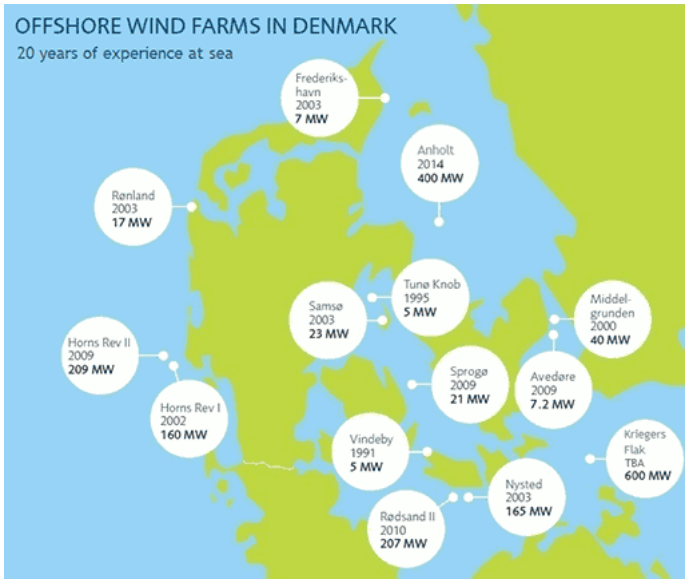


Figure 4.4: Offshore Wind Farms in Denmark

Figure 4.5: The Rødsand II/Nysted offshore wind farms in Denmark.

Offshore Installation	Features Observed
Murchison A	7
Snorre A	6
Snorre B	7
Visund	7
Gjøa	4

Table 4.1: In some area's there are few features observed. This must be considered when evaluating the precision of the system. These are installations in the large zone with less then ten features near(350 meters) the installation.

Field	Installation(s)
Albuskjell	F/BS/FL
Økofisk	Vest
Edda	Edda/FL
Yme	B/STL
Odin	Odin
Frigg	Nordøst
Frøy	Frøy
Total	11

Table 4.2: Near some installations there are no features observed in a radius of 350 meters. This can be caused be either few observations, or they are hard to detected with radar scans.

4.4 Error prone Areas and Installations

When locating permanent features some areas are harder to work with then others. Permanent installations with few surrounding objects the system can deal with rather easily, but where there are clustered features over a small area it gets harder. Some installations have "stealth"-capabilities making them hard to observe with SAR. Other issues can be areas where installations have few(see table 4.1) or none(see table 4.2) observed features nearby. Because of the lack of the data in such areas, the confidence of the NPD-dataset is adjusted.

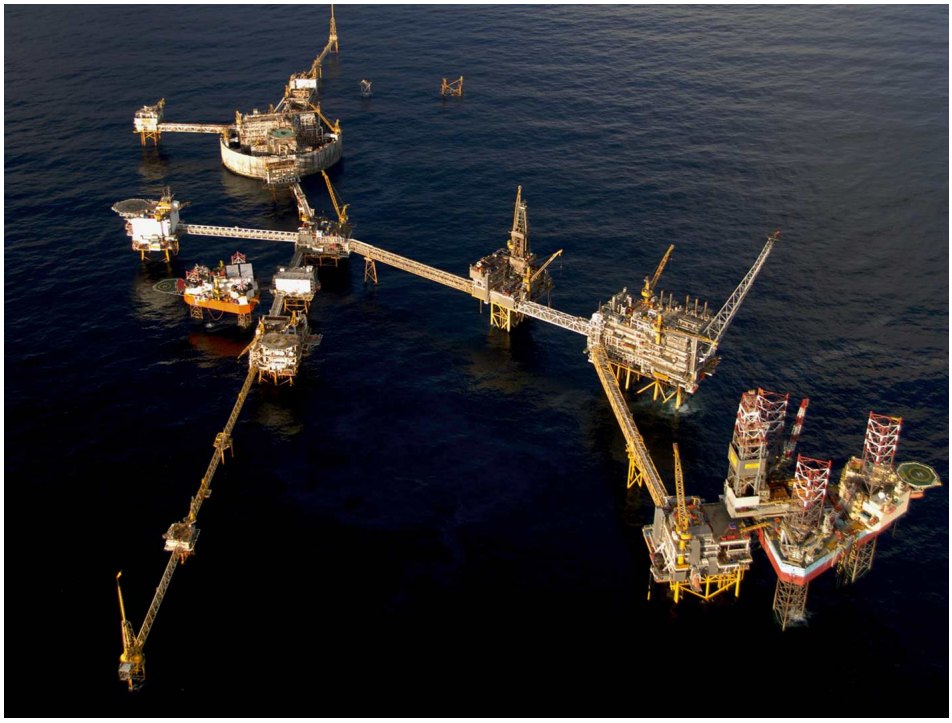


Figure 4.6: Ekofisk

4.4.1 Example - The Ekofisk Field

One such field is the Ekofisk field, that consists of many surface installations close together. Since the algorithm rely on many observation within a given radius, it gets hard to separate the smaller installations from the each other. Figure 4.8 and figure 4.9 illustrates this problem, where tracks are created but deleted as no new feature is found at the same position. Here there can are room for improvement in the algorithm, the solution could be to create regions where different ageing rules apply.

4.4.2 Loading Wells and Boats

In some parts of the test zones there are loading wells and loading boats stationed. Some of these installations can be hard to detected, but can be relieved by dense vessel traffic near by. The loading boats stationed above subsurface installations aren't listed as surface installation in the NPD data set, but they are detected by the system. These feature types are together with the complicated areas considered when giving the NPD dataset confidence estimates.

Figure 4.7: Ekofisk

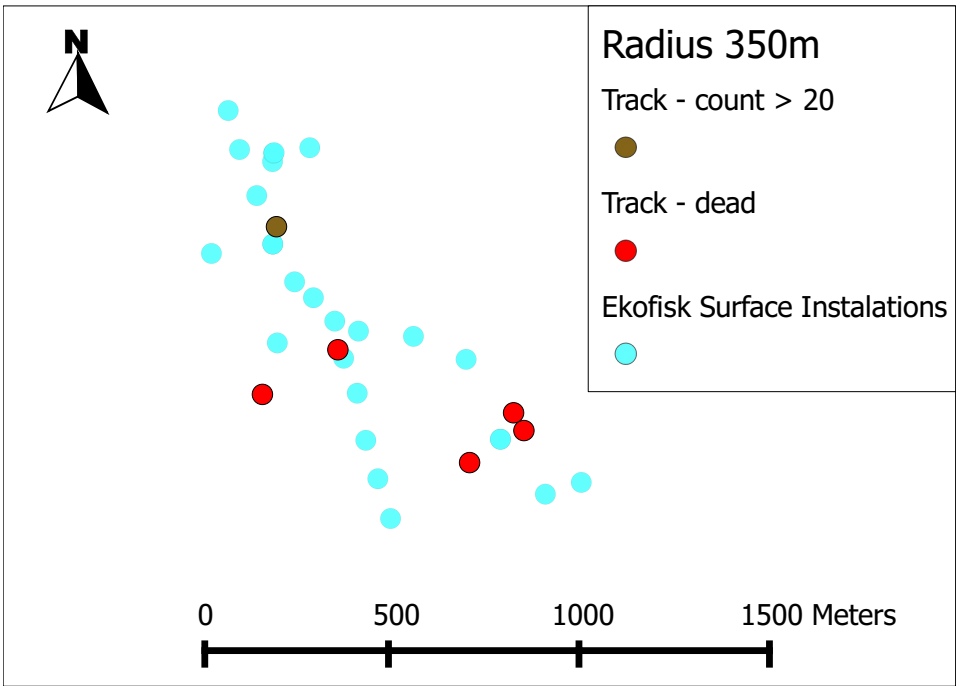


Figure 4.8: Results compared to the Ekofisk field. Permanent and dead tracks after processing all observations with a radius buffer of 350 meters.

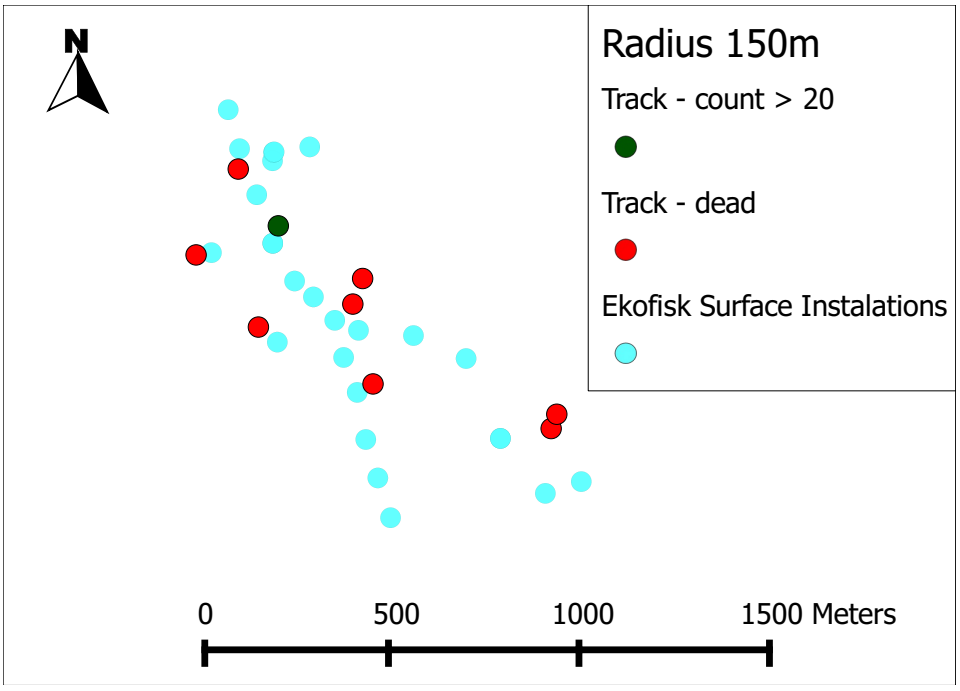


Figure 4.9: Results compared to the Ekofisk field. Permanent and dead tracks after processing all observations with a radius buffer of 350 meters

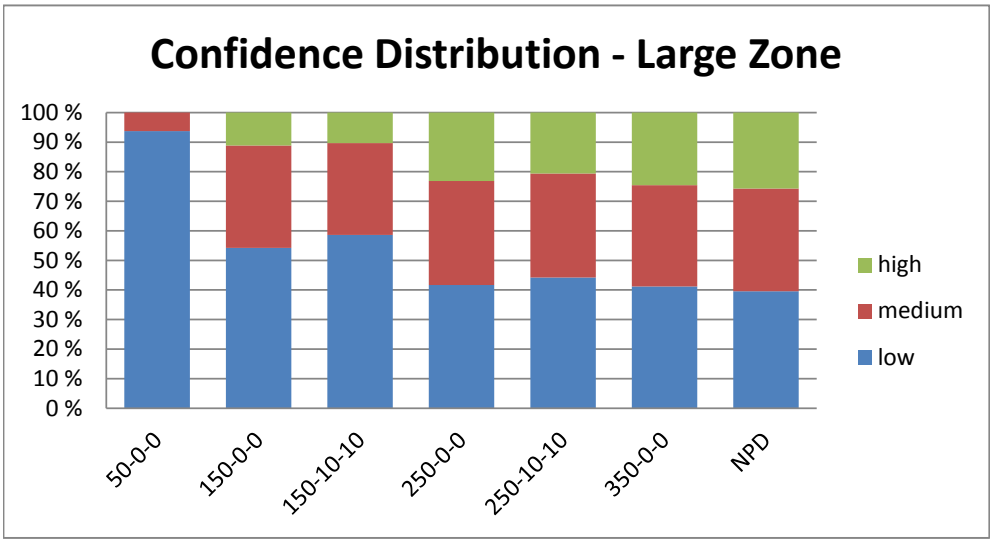


Figure 4.10: Distribution of confidence level among permanent installations.

4.5 Results

This sections evaluates the results from the experiments done. For testing the system capability of detecting permanent installations, the dataset is processed using four different radius values; 50, 150, 250 and 350 meters. For test runs with a radius of 150 and 250 meters the length and width parameters are also adjusted, here with a buffer of 10 meters. The length and width parameter only affects the matching of features, and not existing tracks. Figure 4.12 and figure 4.13 shows the results of the test runs. Here the confidence of tracks(permanent installations) are defined as either low, medium or high. The confidence level is decided by number of features that has contributed to the track.

- Count above 5 equals low
- Count above 10 equals medium
- Count above 20 equals high

Note that the total number of permanent installations in a zone is given by the amount of low confidence detections. Medium and high rated installations are subset of the low rated ones, see figure 4.10. The NPD

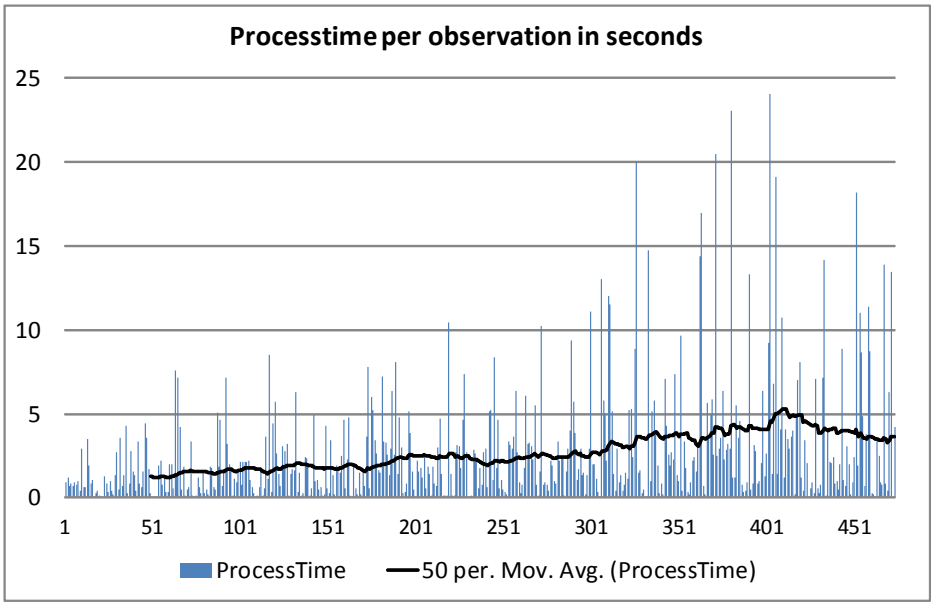


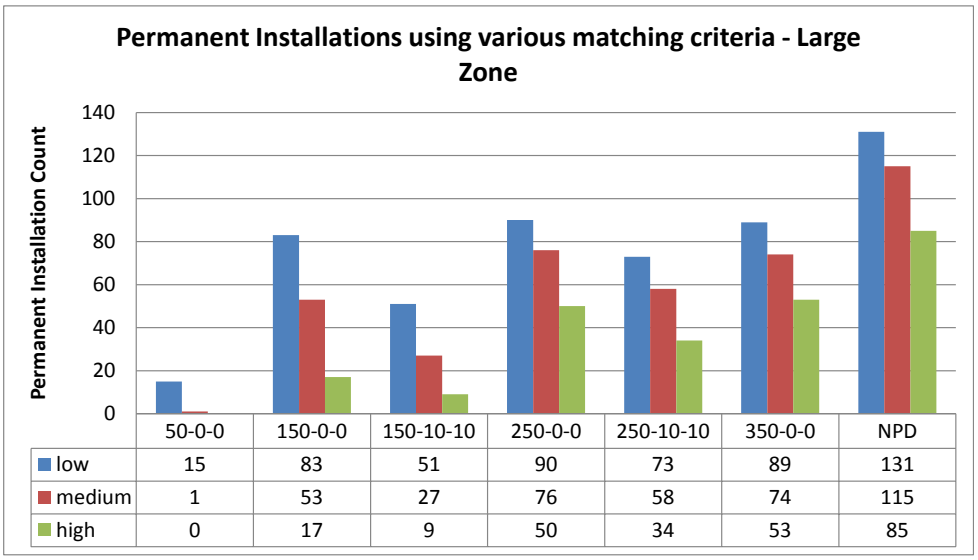
Figure 4.11: Process Observation times in seconds. These times are way below the time limit given to the system as a step in a chain of processes, which is in the order of minutes.

surface installations are also given a confidence level, the reason for this is that some installation are hard for the satellite to detect, and other installations doesn't have enough observation data to be included in the evaluation. Read more about this in section 4.4 where sources of error are discussed.

4.5.1 Test Zone Results

The best results are achieved with a radius between 250-350 meters. Considering the fault margin of 500 meters, which is a little bit strict, this radius-range groups detected features nicely near the actual position of the permanent installations. This distance the varies for every track, but is in the range of 10-200 meters which is acceptable considering the resolution of SAR imagery.

The large zone consists of many hard-to-observe installations as discussed earlier. If we only consider the NPG-high rated installations vs the low-rated results produced by feature detector the hit-rate is rather good.



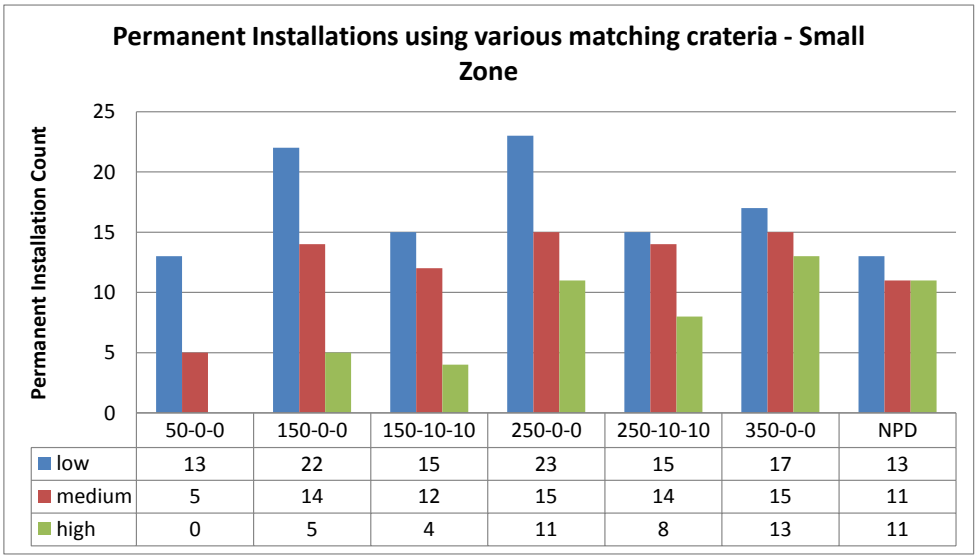
	50-0-0	150-0-0	150-10-10	250-0-0	250-10-10	350-0-0	NPD
low	15	83	51	90	73	89	131
medium	1	53	27	76	58	74	115
high	0	17	9	50	34	53	85

Figure 4.12: The Large Zone's Permanent Installations detected.

Given enough data over time the currently low-rated features fron the algorithm will become high-rated.It would be worse if the outcome was opposite and too many permanent tracks where produced by the system.

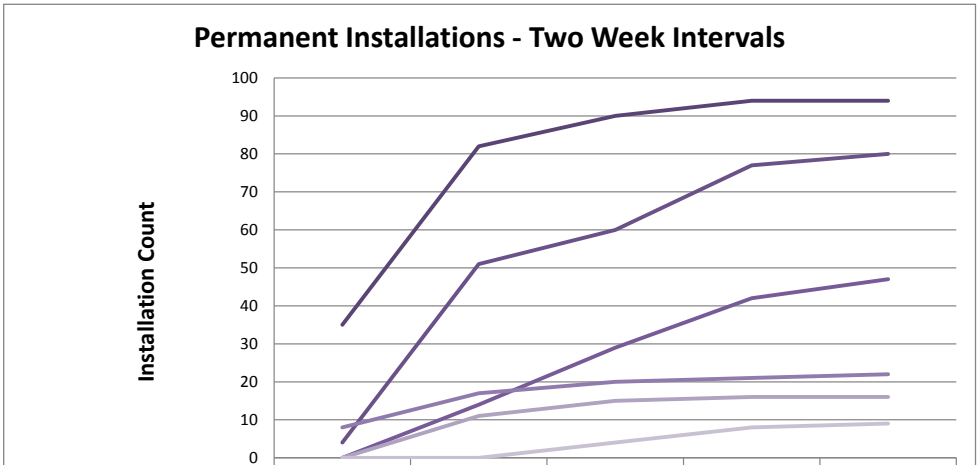
In the small zone the 250-350 radius range is nearly spot on. One interesting difference compared to the results from the large zone, is that it actually detects more features then the NPD dataset list as surface installations. One can then argue then argue that the algorithm doesn't degrade enough and creates too many tracks. After given the "overload"-features in question a closer look, it turns out that these are actually loading wells with stationary vessels, so it's a reasonable result after all.

Figure 4.14 shows the generation of new tracks as new packages are processed. After 7-8 weeks of data processing new tracks are seldom created, but the existing ones are given higher confidence. This gives more strength to the claim that low-rated tracks will eventually match the reference set.



	50-0-0	150-0-0	150-10-10	250-0-0	250-10-10	350-0-0	NPD
low	13	22	15	23	15	17	13
medium	5	14	12	15	14	15	11
high	0	5	4	11	8	13	11

Figure 4.13: The Small Zone's Permanent Installations.



	1-2	3-4	5-6	7-8	9
large zone low	35	82	90	94	94
large zone medium	4	51	60	77	80
large zone high	0	14	29	42	47
small zone low	8	17	20	21	22
small zone medium	0	11	15	16	16
small zone high	0	0	4	8	9

Figure 4.14: The creation of tracks over time.

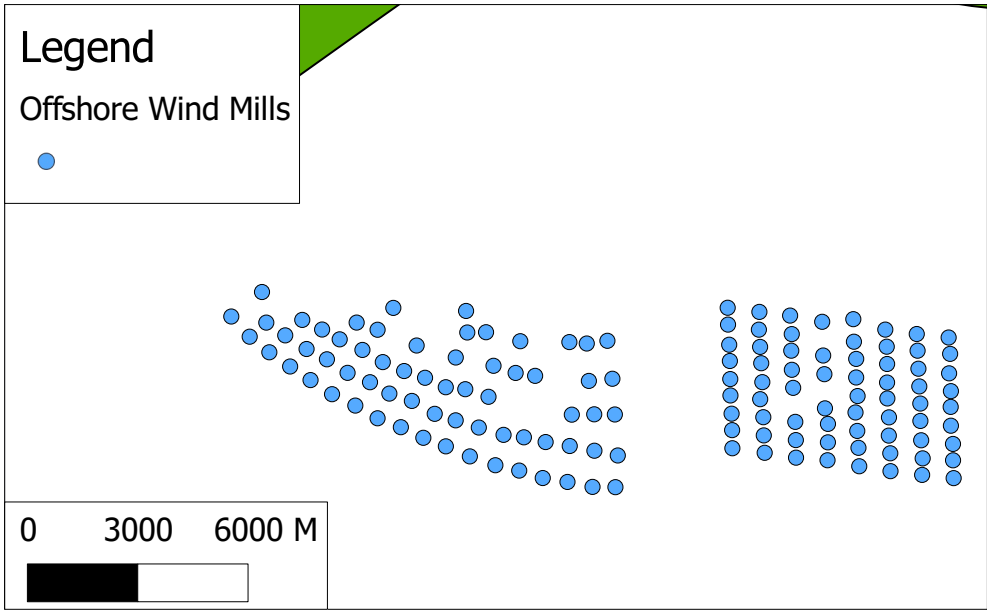


Figure 4.15: Wind farm detection. To the left the Roedsand II farm, and to the right Nysted.

Offshore Installation	Features Observed
Rødsand I	69
Rødsand II	66

Table 4.3: Windmills detected at the RødsandI/RødsandII wind farms. Here a matching radius of 350 meters are used.

4.5.2 Results - Offshore Wind Farms

Rødsand I consists of a total of 72 turbines[1] while Rødsand II has a total of 90 turbines[2]. Table 4.3 shows the results of the insertion algorithm.

[1] http://www.dongenergy.com/Nysted/EN/About_the_park/Introduction/Pages/Introduction.aspx

[2] http://www.eon.dk/Om-EON/Om-energi/Om-Rodsand-2/

Chapter 5

Conclusion

The thesis has presented the design and implementation of Feature Detector, a system for processing, storing and analysing satellite imagery to support and improve the quality of services provided by Kongsberg Satellite Services.

5.1 Summary

Feature Detector consists of as PostGIS database, and system for ingesting and processing different data structures implemented in Python.

The database model has proven to be versatile as it implement's support for different data collection types. The database supports efficient insertion and query-based retrieval for relevant data.

The insertion algorithm can distinguish between moving and static objects given enough data over time.

5.2 Discussion

The initial problem was to develop a information system to support earth observation analysis procedures. Such a system has been implemented, tested and tuned. Further we will discuss the requirements set in section 1.1 in specifics.

5.2.1 First Requirements

- A spatial-temporal database has been implemented.
- The model supports tracking of objects over time.
- A ageing algorithm has been implemented and it's parameters can easily be adjusted.
- The database implements indexes on the most commonly used columns.

5.2.2 Second Requirements

- Rules for matching features has been defined, and are easily adjustable parameters for further improvements.
- Rules for obsolete old objects are defined as age below zero.
- Confidence in terms of permanent objects can be analysed using the age and count of features in a track
- No matching against vessel correlated data is implemented, but there are no constraints on such an extension.
- The database model design makes it easy to retrieve per area information using queries, either from custom designed polygons or existing observation polygons.

5.2.3 Additional Functionality

- Ingestion of oil spill collection data has been implemented.
- Ingestion of ice floes and correlated tracking has been implemented.

5.3 Future Work

The model and algorithm designed and implemented in this thesis makes way for many possibilities. First and foremost the system is a simple to use but resourceful platform for keeping historical data of different types. It can be used for evaluation of already existing services as well as a

springboard for new services.

More specific improvements can be made by fine tuning the feature detector algorithm. As of today it produces more then good results but it can always get better.

A very useful and much worth tool is the ice tracking functionality of the system. As the progress of the research develops in that field, the system will start to adapt the new data structures and over time become a rich inventory of historical data. To see the shape and movement trends of ice floes after several years of data mining is valuable information in the oil industry.

Correlation of AIS data and moving features in the database is also an interesting usage of the system. This can help products related to vessel detection.

Bibliography

[1] Concave hull, May 2013. http://www.bostongis.com/postgis_concavehull.snippet.

[2] Norwegian petrolium department datasets, 2013. http://factpages.npd.no/ReportServer?/FactPages/geography/geography_all&rs:Command=Render&rc:Toolbar=false&rc:Parameters=f&IpAddress=1&CultureCode=en.

[3] Norwegian petrolium department map, 2013. http://www.npd.no/kart.

[4] Offshore wind farms in denmark, May 2013. http://www.southbaltic-offshore.eu/regions-denmark.html.

[5] Postgis api, 2013. http://postgis.refractions.net/docs/.

[6] psycopg, 2013. http://initd.org/psycopg/.

[7] thematicmapping, May 2013. http://thematicmapping.org/downloads/world_borders.php.

[8] Richard B OLSEN. Confidence estimates for ship detection. Technical report, Norwegian Defence Research Establishment, 2005.

[9] DAVID GRIES MICHAEL C. MULDER ALLEN TUCKER A. JOE TURNER PETER J. DENNING (CHAIRMAN), DOUGLAS E. COMER and PAUL R. YOUNG. Computing is a discipline. *ACM Education Board*, january 1989.

[10] P.J.M van Oosterom and C.H.J Lemmen. Spatial data management on a very large castral database. *Computers, Enviornment and Urban Systems*, 2001.

[11] Seungmin Rho Yunyoung Nam and Jong Hyuk Park. Intelligent video surveillance system: 3-tier context-aware surveillance system with metadata. *Springer Science*, 2012.


UNIVERSITY
OF TROMSØ