LONG-RANGE GRAVITY-AIDED AUTONOMOUS UNDERWATER VEHICLE NAVIGATION

by

Franz Heubach

Submitted in partial fulfillment of the
requirements for the degree of Master of Applied Science

at

Dalhousie University
Halifax, Nova Scotia
March 2022

This thesis is dedicated to my friends and family. Thank you for all your support and encouragement. I love you.

Table of Contents

List of Tables

List of Figures

List of Abbreviations and Symbols Used

2D two-dimensional

3D three-dimensional

ANT AUV navigation testbed

API application programming interface

ARL Army Reasearch Laboratory

AUV autonomous underwater vehicle

CHS Canadian hydrographic service

CL cooperative localization

CLI command line interface

.csv comma separated value

DCAF DRDC collaborative autonomous framework

DDS data distribution service

DEM digital elevation model

DKF distributed Kalman filter

DPM distributed partical mapping

DRDC Defense Research and Development Canada

DVL Doppler velocity log

ECA ECA Group

EKF extended Kalman filter

FFT fast Fourier transform

FPF full plane fit

FTS first order Taylor series

GAN gravity aided navigation

GDAL geospatial data abstraction library

GEBCO General Bathymetric Chart of the Oceans

GEOS geometry engine open-source

GIS geographic information system

GNSS global navigation satellite systems

GPS global positioning system

HITL hardware-in-the-loop

ICCP iterative closest contour point

IGA island model genetic algorithm

IGRF International Geomagnetic Reference Field

IMU inertial measurement unit

INS inertial navigation system

ISL Intelligent Systems Laboratory

JPL Jet Propulsion Laboratory

KF Kalman filter

LBL long baseline

MEMS micro-electro-mechanical systems

MOR mid-ocean ridge

NASA National Aeronautics and Space Administration

NPF nine point fit

ROS Robot Operating System

RPS ROS parameter study

SBL short baseline

SLAM simultaneous localization and mapping

STL standard triangle language

TAN terrain aided navigation

TERCOM terrain contour matching

TOF time of flight

TSF two subgroup fit

UKF unscented Kalman filter

UNCLOS Canada's United Nations Convention on the Law of the Sea

USBL ultra-short baseline

UUV unmanned underwater vehicle

Background

$\mathbf{0}$ The N-dimensional vector with zeros for all elements

$\boldsymbol{\omega}_{\text{IMU}}$ The ground truth angular velocity vector of the vehicle in the imu-centered reference frame

$\boldsymbol{a}$ The vector representing gravitational acceleration

$\boldsymbol{a}_{\text{IMU}}$ The ground truth acceleration vector of the vehicle in the imu-centered reference frame

$\boldsymbol{H}_k$ The Jacobian of h relative to the state $\boldsymbol{X}$

$\boldsymbol{L}_{k-1}$ The state transition function Jacobian relative to the process noise

$\boldsymbol{M}_k$ The Jacobian of the measurement with respect to the measurement noise $\boldsymbol{v}_k$

$\boldsymbol{p}$ A second generic quaternion other than $\boldsymbol{q}$

$\boldsymbol{p}$ The vector to the point at which you are calculating the gravitational potential

$\boldsymbol{p}_i$ The vector to the position of the centre of mass of the i^{th}mass element

$\boldsymbol{q}$ A four element vector representing a quaternion

$\boldsymbol{q}_\epsilon$ The vector part of the quaternion when using $\boldsymbol{\epsilon}$ to represent the vector portion

$\boldsymbol{q}_{\mathcal{GL}}$ The quaternion representing the rotation from the global coordinate frame to the local coordinate frame.

$\boldsymbol{q}_{\mathcal{LG}}$ The quaternion representing the rotation from local coordinate frame to global coordinate frame.

$\boldsymbol{q}_v$ The vector part of the quaternion

$\boldsymbol{Q}_{k-1}$ The process noise covariance matrix

$\boldsymbol{R}_k$ The measurement noise covariance matrix in the measurement reference frame

$\boldsymbol{S}_k$ The measurement residual covariance

$\boldsymbol{U}_{k-1}$ The input vector from time set k to $k-1$

$\boldsymbol{w}$ The process noise

$\boldsymbol{x}$ The ground truth velocity of the DVL sensor in the DVL-centered reference frame

$\boldsymbol{x}_{\mathcal{G}}$ A three dimensional coordinate in the global reference frame

$\boldsymbol{x}_{\mathcal{L}}$ A three dimensional coordinate in the local reference frame

$\boldsymbol{x}_m$ The position vector of the gravity gradiometer measurement $[x, y]$ in the xy-plane

$\boldsymbol{Z}_k$ The measurement vector

δx_i The size of the i^{th}element in the x direction

δy_i The size of the i^{th}element in the y direction

δz_i The size of the i^{th}element in the z direction

$\dot{\boldsymbol{q}}$ The rate of change of the quaternion $\boldsymbol{q}$

$\dot{\boldsymbol{x}}_{\text{DVL}}$ The vehicle velocity in three dimensions as measured by the DVL

ϵ_x The first element of the vector portion of the quaternion

ϵ_y The second element of the vector portion of the quaternion

ϵ_z The third element of the vector portion of the quaternion

η	The scalar portion of the quaternion
$\hat{\boldsymbol{r}}$	The unit vector pointing from $\boldsymbol{r}$ to $\boldsymbol{r}_i$
$\mathcal{N}$	The N-dimensional Gaussian distribution
∇	The gradient operator
$\omega_{\mathcal{L}}$	The angular rate in three-dimensional space in the local reference frame
ρ	The density of the i^{th}element
Σ^2_{DVL}	The covariance matrix associated with the DVL velocity measurement
$\Sigma^2_{\boldsymbol{\omega}}$	The covariance matrix associated with gyroscopes in the imu-centered reference frame
$\Sigma^2_{\boldsymbol{a}}$	The covariance matrix associated with accelerometers in the imu-centered reference frame
θ_m	The gradiometer's gravity gradient direction measurement
$\tilde{\boldsymbol{v}}_k$	The estimated measurement residual, also known as the residual
A	The amplitude of a Perlin noise layer
a_z	The z component of $\boldsymbol{a}$
f	The spatial frequency of the Perlin noise layer
G	The universal gravitational constant, 6.674×10^{-11}
g	The gravity anomaly
h	The non-linear measurement function
i	The index of the mass element
i, j, k	The quaternion units

k The index of the Perlin noise layer

l The lacunarity of successive Perlin noise layers

m_i The mass of the i^{th}mass element

M_m The magnitude of gradiometer's gravity gradient measurement

n The number of individual mass elements

q_w The scalar part of the quaternion

r The magnitude of the different between $\boldsymbol{r}$ and $\boldsymbol{r}_i$

r The persistence of successive Perlin noise layers

r The vector going from point $\boldsymbol{p}$ to $\boldsymbol{p}_i$

t The variable that the Perlin noise interpolation function is parameterized with

V The gravitational potential

x The x component of the $\boldsymbol{r}$ vector

y The y component of the $\boldsymbol{r}$ vector

z The z component of the $\boldsymbol{r}$ vector

Inertial Navigation Modelling

$\boldsymbol{a}$ The acceleration of the vehicle during the time step T

$\boldsymbol{q}$ The INS estimate of the current vehicle quaternion orientation

$\boldsymbol{x}$ The INS estimate of the current vehicle position estimate in three-dimensional Cartesian space

$\boldsymbol{x}$ The INS vehicle positional estimate using the EKF

$\dot{\boldsymbol{x}}$ The INS estimate of the current vehicle velocity in three-dimensional Cartesian space

$\dot{\boldsymbol{x}}$ The velocity of the vehicle during the time step T

k The subscript denoting the current time step

$k+1$ The subscript denoting the next time step

T The time step used for the EKF equations

Particle Filter

$\Delta\theta_{ins}$ The change in the vehicle's yaw between prediction steps of the particle filter

Δd_{ins} The distance travelled by the vehicle's center of mass between prediction steps of the particle filter

ESS The effective sample size; a measure used to represent how many particles effectively represent the weighted sample distribution

σ_θ The standard deviation of all the particle heading estimates

σ_m The standard deviation of the gradiometer gravity gradient direction measurement

σ_x The standard deviation of all the particle x position estimates

σ_y The standard deviation of all the particle y position estimates

θ^{k+1} The particle filter's estimate of the vehicle yaw after the prediction step

θ^{k+1} The particle filter's estimate of the vehicle yaw before the prediction step

θ^{k-1} The previous inertial navigation system estimate of the vehicle's yaw

θ_i The *a priori* gravity gradient map's gradient direction at the i^{th} particle's location

θ_m The gradiometer's gravity gradient direction measurement

θ_{ins}^k The latest inertial navigation system estimate of the vehicle's yaw

f_θ The jitter multiplication factor for the heading estimate

f_x The jitter multiplication factor in the x coordinate direction

f_y The jitter multiplication factor in the y coordinate direction

i The particle index

j_θ The jitter added to the particle filter heading estimate

j_x The jitter in the x coordinate direction of the particle filter positional estimate

j_y The jitter in the y coordinate direction of the particle filter positional estimate

n The number of Gaussian distributions to use for the approximation of the likelihood function.

N_p The sample size, also known as particles, used by the particle filter

w_i^k The particle weight of the i^{th} particle before the particle filter's update step

w_i^{k+1} The particle weight of the i^{th} particle after the particle filter's update step but before normalization

x^{k+1} The easting portion of the particle filter's positional estimate of the vehicle's center of mass in a projected coordinate system after the prediction step

x^k The easting portion of the particle filter's positional estimate of the vehicle's center of mass in a projected coordinate system before the prediction step

x_{ins}^k The latest easting portion of the inertial navigation system positional estimate of the vehicle's center of mass in a projected coordinate system

x_{ins}^{k-1} The previous easting portion of the inertial navigation system positional estimate of the vehicle's center of mass in a projected coordinate system

y^{k+1} The northing portion of the particle filter's positional estimate of the vehicle's center of mass in a projected coordinate system after the prediction step

y^{k} The northing portion of the particle filter's positional estimate of the vehicle's center of mass in a projected coordinate system before the prediction step

y_{ins}^{k} The latest northing portion of the inertial navigation system positional estimate of the vehicle's center of mass in a projected coordinate system

y_{ins}^{k-1} The previous northing portion of the inertial navigation system positional estimate of the vehicle's center of mass in a projected coordinate system

Gradiometer Model

$\angle\nabla g$ The planar direction or heading of the gravity gradient

$\mathbf{\Sigma}_g$ The covariance matrix of the gyroscope measurement defined in the body-centered reference frame

$\|\nabla g\|$ The planar magnitude of the gravity gradient

$\mathcal{N}$ The normal distribution

σ_θ The standard deviation of the Gaussian noise applied to the heading of the gravity gradient

σ_m The standard deviation of the Gaussian noise applied to the magnitude of the gravity gradient

g_z The vertical component of the gravity gradient

x The coordinate direction in line with the cardinal direction East

y The coordinate direction in line with the cardinal direction North

Terrain Generation

l The lacunarity or the ratio of frequencies of successive Perlin noise layers

n_{octave} The number of octaves required to guarantee a minimum detail size

p_o The period of the first Perlin noise layer or the base period

s The size of the minimum guaranteed detail within the generated terrain

Abstract

Autonomous underwater vehicles (AUV) are a mobile platform for underwater sensing, an environment relatively unexplored. Georeferencing measurements is difficult due to the challenge of AUV localization. The rapid attenuation of radio frequencies underwater restricts AUVs from using the global position system (GPS), the above-water solution to localization. Underwater localization relies on dead-reckoning, the integration of vehicle inertia measurements to arrive at a position estimate. However, the dead-reckoned position error is unbounded. This error can be bounded using a source of position feedback. Terrain aided navigation (TAN) — using georeferenced geophysical terrain maps can provide that feedback. TAN shows significant promise as a method for long-range, passive underwater AUV navigation, especially gravity-aided navigation (GAN). This thesis presents a TAN algorithm that uses a gravity gradiometer and gravity gradient maps to successfully limit dead-reckoning error by a factor of 25 over a 500 km long AUV mission, with a localization accuracy of 1 km. The TAN algorithm exploits the correlation between terrain and the gravity anomaly to use a global database of bathymetry maps (GEBCO) with 400 m resolution. The mission was simulated in the AUV navigation testbed (ANT), a collection of tooling developed during this thesis to accelerate research in TAN. Among the contributions made by the ANT, is a inertial navigation system (INS) that emulates the uncertainty characteristics of a commercial navigation grade INS (Kearfott Seanav) — to simulate dead-reckoning error growth. Parts of the ANT have been released to the research community as open-source, and are being used by researchers in the Intelligent Systems Laboratory (ISL) at Dalhousie University.

Acknowledgements

Thank you to my supervisor Dr. Mae Seto for your guidance, feedback, and critique from start to finish of this thesis.

Thank you to my supervisory committee Dr. Vincent Sieben and Dr. Robert Bauer for your comments, suggestions, and feedback.

Thank you to my lab colleagues for your feedback, support, and good company.

Thank you to the open-source community for all the amazing software that enabled this thesis.

Thank you to my funders who through their resources enabled the work in this thesis: Nova Scotia Graduate Studies (NSGS), Irving Shipbuilding Inc., the Killam Family Foundation (Killam Scholar), and Defence Research and Development Canada (DRDC).

Chapter 1

Introduction

The autonomous underwater vehicle (AUV) is a mobile under-water sensing platform. The on-board autonomy enables underwater missions that would otherwise require a larger vehicle for manned missions and removes the operating area restrictions imposed by a tethered vehicle [1]. This reduces the mission cost and enables longer duration missions for ocean sensor measurements / data collection [1]. This data is limited only by the sensors the AUV can support, which is limited by its onboard energy, and size. The lower cost of, and potential of AUV missions, and the leap in embedded computing technology, means sensors specifically built for the AUV platforms have proliferated in the last two decades [2]. When making measurements underwater, the measurements must be georeferenced to be of any use to the AUV mission.

Georeferencing data underwater is especially difficult because the global positioning system (GPS) is not available underwater. This is due to the rapid frequency-dependent attenuation of radio frequencies by salt water [3] [4]. The effective range of radio frequency communication under water is on the order of 10 m [4]. Reliance on the acoustic communication channel has important consequences, underwater communication channels are low-bandwidth, low baud-rate, and unreliable, therefore vehicle localization is difficult [3]. Localization refers to the vehicle's ability to georeference itself, and navigation refers to how a vehicle travels from one location to another. Over longer missions, localization becomes essential to navigation so the two words are often incorrectly used interchangeably or in similar contexts [5]. During shorter missions the vehicle localization can be performed using dead-reckoning [6].

Dead-reckoning uses the numerical integration of measurements of angular velocity and acceleration to obtain the vehicle position [7]. This integration action also integrates the measurement errors which leads to unbounded error growth with time [7]. Over shorter distances (10 km to 500 km) this can be mitigated with a better inertial navigation system (INS). An INS implements dead-reckoning [6]. However, over longer distances (>500 km) the error accumulated over time becomes unusable for navigation even with the most expensive INS [8, 6].

For example, underwater port-to-port navigation from Halifax, Nova Scotia to Southampton, England is approximately 4500 km. Long-range underwater navigation

is necessary when the surfacing for a GPS fix is undesirable or impracticable (e.g. under-ice [9], stealth, or deep-water operations). Using the Kearfott Seanav INS with a circular error probable of 0.5% of distance travelled [8], the accumulated error would be approximately 45 km (95% confidence interval, assuming Gaussian distributed). This is an unacceptable amount of accumulated error — larger than the width of the English channel (approximately 30 km). This justifies the research field of long-distance AUV navigation.

Dead-reckoned AUV navigation in general requires position feedback [10]. This can be in the form of acoustic beacons integrated on surface platforms that access GPS [11], other underwater vehicle(s) [12], or georeferenced landmarks [13]. These methods all require an active acoustic communication channel which can be intermittent [3] and could be interfered with (jammed). Stationary beacons need to be installed and georeferenced before the mission. They are obviously limited in operating range [13] and therefore of no value for long distance missions. Other methods require other vehicles (e.g. ships) and the ocean surface to be available which is not the case for under-ice navigation [14, 9]. An emerging method for AUV navigation is to use physical features of the terrain as position references [15, 16, 17].

Terrain aided navigation (TAN) refers to AUV localization and navigation using sensors to detect and recognize geophysical terrain features [6]. This includes the bathymetry sensed through sonars [10], the magnetic field with magnetometers and magnetic gradiometers [18, 19], and the gravitational field through the use of gravimeters and gravity gradiometers [20, 21]. These methods are in varying stages of technological readiness. Each method uses *a priori* georeferenced maps to localize the AUV [6]. There are various TAN algorithms that have been explored — some as simple as matching to the nearest map location [22, 23], other more complicated methods propagate state estimate distributions using a Kalman filter or particle filter [24, 25, 26, 27].

The goal of this thesis is to develop a TAN algorithm that limits the dead-reckoned error growth over long distances, is passive, resistant to interference, and uses readily-available information. Active sensors emit energy into the water, passive sensors do not [17]. Gravity aided navigation gravity aided navigation (GAN) was chosen as the

method of TAN that meets the requirements.

GAN is resistant to interference as it requires large amounts of mass to be moved to change the local gravity field. The gravity field is more persistent as it is less dynamic than the magnetic field [17]. The magnetic field is also not stable enough to navigate by near the poles whereas the gravity field would be. Additionally, the temporal aspects of the gravity field like tidal effects can be filtered somewhat with wave models. The work in this thesis is closely related to work by Pasnani and Seto [28]. Pasnani and Seto use the gravity anomaly field directly to perform TAN. This thesis presents a novel algorithm to perform underwater TAN through exploiting the correlation between the bathymetry and gravity anomaly.

The work presented in this thesis evaluates whether the global General Bathymetric Chart of the Oceans (GEBCO) [29] bathymetry database can be used to navigate a vehicle with a gravity gradiometer. The benefit of this versus the gravity anomaly maps used in [28] is the increased resolution (400 m [29] versus 2 km [17]) and better availability of maps. Both are global databases, the GEBCO database being of higher resolution. Additionally, the presented TAN provides a separate method for localization dependent on a different set of maps but using a similar sensor. This creates the option to fuse both methods with particle filter state estimate to increase the localization accuracy even further. This promising approach is left for future work.

Towards developing a low-risk testing method for the TAN algorithm presented in this thesis, the author developed tools and processes that will benefit others in AUV navigation as well as users of the ROS ecosystem for research, and reduce duplication of research tooling. The tools together make up what this thesis refers to as the AUV navigation testbed (ANT). Early work in this thesis was presented at the IEEE AUV 2020 virtual conference [30]. The ANT and environmental modelling techniques were open-sourced and presented at the IEEE Oceans 2021 conference in San Diego [31]. Tools that are part of the ANT are currently used by researchers in the Intelligent Systems Laboratory (ISL).

The work in this thesis has three main contributions. Firstly, a TAN algorithm that is passive, resistant to interference, limits dead-reckoning error growth, and uses

readily-available information. Secondly, an INS model implementation that has error growth properties of current state-of-the-art INS and can be used for simulating AUV dead-reckoning. Thirdly, the development of a set of tools (the ANT, and ROS parameter study (RPS)) to enable and ease underwater AUV research, especially in the field of TAN.

The proposed TAN algorithm uses existing global bathymetry maps from GEBCO to perform gravity gradient aided localization of the AUV to provide a position error bound on dead-reckoning (details in sections 4.4.1 and 5.6). The algorithm exploits the correlation between the terrain and the gravity anomaly to derive the gravity gradient. Using the gravity gradient heading for navigation makes the algorithm more robust to the uncertainties that arise when using the gravity gradient. The particle filter provides a position estimate limited by the accuracy of the *a priori* maps provided to the vehicle. The particle filter algorithm successfully limits the error growth of the dead-reckoned position estimate to 1 km — limited by the resolution of the gravity gradient map input (the 400 m resolution bathymetry from GEBCO was converted to a resolution of 1 km during derivation of the gravity anomaly). Over a distance of 500 km the median particle filter position error was approximately 1 km. Compared with the median dead-reckoning error of 25 km, the particle filter reduces position error by a factor of 25 (details in section 5.6). To simulate dead-reckoning an INS model was needed.

An INS uses accelerometers and gyroscopes to measure the vehicle acceleration and angular velocity, respectively. These are integrated twice and once, respectively, to obtain the vehicle position and orientation. Numerically integrating inertial measurements to estimate position and orientation is referred to as dead-reckoning [32]. The Gazebo physics simulator does not include an implementation of an INS model, only an inertial measurement unit (IMU). There are also no libraries that implement an INS model that could be adapted for use within the Gazebo physics simulator. Therefore, This thesis documents the creation and testing of an INS model that has error properties similar to current state-of-the-art INSs [8]. The position estimate of the simulated INS was used as the comparison for the implemented TAN algorithm, and is part of the collection of tools developed during the work in this thesis referred

to as the ANT.

The ANT was created to accelerate the development and low-risk testing of TAN in an underwater environment. It uses the Gazebo simulator, the ROS, and the unmanned underwater vehicle (UUV) simulator [33]. Gazebo is an open-source physics-based simulator for robotics. ROS is a robotics tool chain that provides middleware for writing modular, component-based robotics software architecture, and a communications layer for data sharing between processes. The UUV simulator is a software package that uses ROS and the Gazebo simulator, to provide a 3D vehicle URDF-model, a vehicle dynamics model, and a vehicle controller for a torpedo-shaped AUV (the ECA Group (ECA) A9) that shares kinematic similarities with the type of vehicle that could be used for on-vehicle testing of the TAN algorithm developed during the work in this thesis.

The collection of tools that comprise the ANT, include the ANT command line interface (CLI). A CLI is a text based interface that executes program commands from the command line. The ANT CLI is written in Python, a popular language, and open-sourced to the community on GitHub (the most popular code-sharing platform). The work in the ANT CLI was published as part of the OCEANS 2022 conference in San Diego where the work was presented [31]. The ANT CLI contains functionality to visualize Perlin noise (a coherent type of noise often used to generate artificial terrain), create artificial terrain with various parameters under the users control, augment existing terrain with noise as a way to introduce missing detail into lower-resolution raster data, use bathymetry and optionally density to derive an approximation of the local gravity anomaly using the correlation between the terrain and the gravity anomaly, and facilitate a pipeline from a bathymetry raster to a 3D solid mesh collision and visual model within the Gazebo simulator.

The terrain generation technique is useful in TAN research to explore how terrain properties affect the performance of an algorithm (details in section 4.1.3). This promotes a deeper understanding of the algorithm limitations before on-vehicle implementation, and in-water testing. The terrain augmentation technique contributes a way for researchers to take existing lower-resolution measurements (which are more readily available, e.g. GEBCO) and add realistic detail, with noise properties under

the user's control, while maintaining the integrity of the original measurements (details in section 4.1.4). The bathymetry can then be transformed into the expected 3D model format for visual and collision awareness within the Gazebo physics simulator.

The ANT CLI supports a pipeline to transform a raster into a model within the Gazebo simulator using the Blender tool [34]. Blender is a widely used open-source 3D modelling and animation software tool and is the recommended way to create 3D models for the Gazebo environment. The pipeline makes use of existing open-source Python libraries that support reading raster data [35], the Blender Python application programming interface (API), and matplotlib a Python plotting library [36]. The pipeline reads the raster data, imports it into Blender, turns it into a 3D solid triangular mesh, uses matplotlib to generate a colour map and overlays it onto the mesh as a visual, exports the files to a Gazebo model directory, and generates the necessary configuration files for the models use within Gazebo. This pipeline accelerates a process to complete in a few minutes when it can otherwise take a day.

This thesis will cover the methods, implementations, results, and consequences of these contributions. The thesis begins by introducing state estimation in section 2.1. State estimate is a broad topic that covers methods which estimate the current state of a system based on uncertain input. In this case the AUV pose when it is transiting or taking measurements under water. The standard localization method on most AUVs is to use a form of dead-reckoning (section 2.2) [2]. The implementation of AUVs dead-reckoning is often based on an inertial navigation system (INS). However, dead-reckoning has no error bound on the position estimate error [7]. Therefore, some method of position feedback needs to be employed to bound the error growth.

Georeferenced acoustic transponders can be used for position feedback. This is usually in the form of an existing acoustic transponder network and works in a manner similar to GPS [37, 3]. Each transponders uses the time a message takes to travel i.e. time-of-flight, to range the vehicle from the known transponder position. Related research, benefits, and drawbacks of acoustic transponder localization is introduced in section 2.3. Transponders can also be used integrated to other mobile platforms, or other AUVs [12]. The other vehicle's known position can then be used to cooperatively localize the AUV. This is introduced in section 2.4. Other forms of localization can

use *a priori* information about the environment (maps) to assist state estimation methods [28].

Research within mapping related to AUV navigation and localization is introduced in section 2.5. Maps summarize, record, and georeference the information associated with a specific geophysical region [38]. The terrain types are: the ocean floor (bathymetry), the Earth's magnetic field (geomagnetic), and Earth's gravitational field (gravity). The properties and research related to these terrain types are introduced in sections 2.6.1 to 2.6.3, respectively. Mapping the environment and using the same maps to localize the vehicle is referred to as simultaneous localization and mapping (SLAM).

Research related to SLAM, and the reasons to use SLAM, are covered in section 2.7. SLAM is appropriate when revisiting a known area more, which enables loop closure [39]. For long distance transits however, SLAM loop closures are less useful and can only be used on the return trip; if there is one. Therefore, TAN aided navigation using *a priori* maps is introduced in the next section.

TAN compares a georeferenced *a priori* map to AUV measurements to localize the vehicle. TAN methods using the different geophysical terrain types first introduced in section 2.6, are covered in section 2.8. Research related to navigation using acoustic sensors and the ocean floor (bathymetry) is introduced in section 2.8.1, using the geomagnetic field is covered in section 2.8.2, and using the gravity field is introduced in section 2.8.3. Then, the background section covers some of the mathematical concepts used extensively within this thesis, starting with reference frames.

Reference frames are introduced in section 2.9, and quaternion conventions are introduced in section 2.11. Quaternions have three main conventions [40], each can be mixed and matched with each other, making it important to be clear about which convention is used. Both quaternions and reference frames are featured heavily in the implementation of the INS. The INS was implemented using the EKF framework, the de-facto standard [32, 41]. The EKF is a linearization of a non-linear system's state space to use the Kalman filter (KF) framework [32]. Given small enough steps most non-linear systems can be approximated using linear propagation and correction,

justifying the EKF wide-spread use [32]. The structure of an EKF is introduced in detail in section 2.10. The EKF based INS is used as input to the TAN algorithm together with the environment sensor models.

The environment is modelled using existing terrain from a global database of ocean floor bathymetry, the GEBCO database (details in section 2.12). Perlin noise [42], introduced in section 2.12.1, is a method to generate custom terrain or augment existing terrain with higher detail. The environment for GAN needs to include a measure of the gravity anomaly. The correlation between the gravity anomaly and the terrain can be used to derive the gravity gradient [43]. This is first introduced in section 2.12.2.

To conclude the background section a summary of the motivation for the work in this thesis (section 2.13) is provided. The background introduces AUV navigation and localization, covers the concepts needed for the work in this thesis, and summarizes the thesis work motivation. Next, the methods used throughout this thesis are introduced.

The methodology section (chapter 3) covers the problem and solution flow for the work in this thesis. Multiple decision-tree diagrams pose the problems and challenges encountered during this thesis, and the solutions and implementations as a result of them. These diagrams show the conceptual flow for the work presented within this thesis. The reader is then introduced to ROS and the Gazebo simulator, which are fundamental to the implementation of the work in this thesis, in section 3.1. These tools are the basis of the ANT, a collection of tools developed during the work in this thesis to acelerate research processes within TAN. The ANT is introduced in section 3.2. The ANT is used for low-risk simulation based testing and is a prerequisite for integration testing, and in water testing.

Section 3.4 discusses the work completed towards vehicle integration and in-water testing of the TAN algorithm on the IVER 3, a torpedo shaped AUV. The methodology section concludes with the methods used for designing the TAN algorithm in section 3.3, and studies that are going to be used to test the performance of the TAN in comparison to dead-reckoning (INS) in section 3.3.1. This leads to the implementation of the work in this thesis.

The experiment section (chapter 4) covers the implementation details for the ANT thoroughly, this includes the break down of the state equations for the EKF-based INS model in section 4.1.1, and the creation of the Gazebo world model used for the simulation environment in section 4.1.2. The application of Perlin noise to create terrain, to the user's specifications, as a way of testing TAN algorithms is explained in section 4.1.3. Perlin noise can also be used to add detail to lower resolution terrain as a way to enhance the realism of a simulation environment, this process is discussed in section 4.1.4.

Once the missing density field is generated, and the bathymetry has sufficient detail the gravity gradient is derived using the correlation between the terrain and the gravity anomaly [44]. The details of this conversion are covered in section 4.1.5. The gravity gradient is used to simulate the gravity gradiometer. The gravity gradiometer model is discussed in section 4.1.6. The gravity gradiometer model is used as position feedback for the TAN algorithm. The model used for particle filter based TAN algorithm is discussed in detail in section 4.4. The DRDC collaborative autonomous framework (DCAF) project works towards vehicle integration and in-water testing for the TAN algorithm.

Parts of DCAF that the author directly developed are discussed in detail in section 4.5. This includes the bounce behaviour, and vehicle behaviour arbitration. The bounce behaviour confines the vehicle to a virtual boundary surrounding a predefined operating region. The arbiter manages the actions requested by behaviours within DCAF and selects which action gets executed on the vehicle based on priority and action request time. Before implementation on the vehicle the TAN is tested within the ANT.

The ANT provides a CLI that accelerates testing of TAN related algorithms. The CLI provides the terrain generation, terrain augmentation, gravity anomaly derivation, pipeline from raster to Gazebo world model, and peripheral functionality. The ANT CLI is discussed in section 4.2. Additionally, a tool to enable parameter studies with ROS and Gazebo simulator was also developed, the ROS parameter study CLI, discussed in detail in section 4.3. The results of the tests for the ANT and the TAN algorithm are shown in chapter 5.

The results and discussion section (chapter 5) shows the INS model has the error characteristics of current state-of-the-art INS models, therefore, verifying its use to develop dead-reckoning navigation methods. The results of the INS model results are discussed in section 5.1. The verification of the tools developed as part of the ANT, including the Gazebo world models, terrain generation, terrain augmentation, and gravity map derivation are shown and discussed in sections 5.2 to 5.5, respectively. Results of testing the TAN algorithm performance when subjected to gravity gradiometer heading noise and density uncertainty, due to the constant density assumption, are presented and discussed in sections 5.6.1 and 5.6.2, respectively. The results show the particle filter algorithm successfully limits error growth, and localizes the vehicle to within a 1 km (limited by map resolution used for the simulation studies).

The thesis concludes with a reiteration of the contributions of this thesis, a summary of the implication of the results (chapter 6), and the author's recommendations for follow-on future work (section 6.1).

Chapter 2

Background

This chapter covers the background needed for the thesis work. This includes research in the field of AUV localization and navigation to provide context and motivation for the presented work. It also introduces and explains mathematical concepts used in later sections.

2.1 State Estimation

For an AUV to localize itself and effectively navigate, some form of state estimation is required as the AUV position is not directly observable or measurable. The state estimation here concerns the vehicle pose: the vehicle position and attitude.

The KF, and variations there of, are arguably the de facto standard in state estimation [32, 41] (the KF is covered in detail in section 2.10). However, non-parametric state estimation techniques like the particle filter and its variants have become increasingly common [45, 26, 46, 27, 20]. For a summary of state estimation techniques related to terrain aided navigation see Table 1 in Carreno et al. [10]. State estimation in the context of AUV localization and navigation is also discussed in detail in work by Paul et al. [3].

For example, a Kalman filter error state formulation is used to integrate an IMU, with a Doppler velocity log (DVL), acoustic transponders like an long baseline (LBL), pressure sensor (to measure depth), and altitude measurements for a complete AUV navigation solution [47]. A Doppler velocity log (DVL) measures acoustic Doppler effect along four directions [47], used for over-ground velocity feedback. A long baseline (LBL) system measures round-trip travel times between a vehicle transceiver and four remote transponders [47], used for vehicle position feedback relative to the LBL system.

TAN is highly non-linear, due to the non-linear nature of geophysical terrain [25]; therefore, the unscented Kalman filter (UKF) has been used (e.g. [25], [48]). The UKF uses sigma points to transform the Gaussian distribution through a non-linear function, rather than linearization of the non-linear function (EKF), and does not require the often difficult task of computing the Jacobian matrices [49]. The UKF is

generally more accurate than the EKF, and the sigma point-based probability distribution transform is not limited to Gaussian (unimodal) distributions [49]. However, the UKF is more conceptually complex, computational expensive, and often may not perform better [41]. For this reason the UKF was not pursued for use within this thesis.

Sampling-based techniques like the particle filter, also referred to as sequential Monte Carlo, condensation, and natural selection are able to represent non-linear, multi-hypotheses, and un-parameterized probability distributions [50]. They are, therefore, ideal for state estimation where the state transition or measurement model are non-linear, as is the case for TAN [16], due to the non-linear nature of geophysical terrain.

Particle filter based state estimation, as applied to TAN is heavily relied on by numerous research works [51, 27, 16, 28], making them attractive for research in TAN. Effort has gone into making particle filters less likely to prematurely converge in featureless environments by slowing convergence in uninformative terrain [51], and more efficient in representing the underlying bathymetry estimates of particles [27]. A TAN solution for gliders also makes use of the particle filter [15]. This demonstrates the versatility, and active use of particle filter based algorithms in the field of TAN.

Often the particle filter based TAN relies on the existing dead-reckoning solution that comes standard on the AUV [2]. The dead-reckoning algorithm, implemented in the INS, provides the high fidelity vehicle state estimation over short distances [32], complemented by the lower frequency position feedback updates associated with TAN [17].

2.2 Dead Reckoning

Dead-reckoning uses the previous position (or prior), the velocity vector, and the time step to estimate the current position [3]. This velocity can be directly measured (for e.g. with a Doppler velocity log) or estimated by integrating the acceleration vector over time. The integration of acceleration with respect to time estimate position is referred to as inertial navigation and is the basis of most AUV navigation systems [2].

Military grade INS cost $ 1 million and have a drift performance of 1 nm/day (1 nautical mile is approximately 1.85 kms), navigation grade INS are on the order of $ 100K and can limit the drift to 1 nm/hour, and a micro-electro-mechanical systems based INS costing thousands of dollars have > 10 nm/hour drift [18].

The INS can be integrated with the DVL. The DVL uses Doppler shift of acoustic measurements to get a direct measure of the three-dimensional (3D) relative velocity vector over ground [3]. The velocity is relative to the surface that the four or more beams are reflected by [3]. This is usually the vehicle velocity relative to the ocean floor. The DVL has a nominal standard deviation of $0.3\,\mathrm{cm\,s^{-1}}$ to $0.8\,\mathrm{cm\,s^{-1}}$ and cost from $20K to $80K [3]. This direct measure of the velocity puts a bound on the INS velocity estimate's error growth and therefore, increases localization and navigation performance.

The DVL is heavily relied upon in situations where a direct measure of the relative velocity can improve the navigation accuracy of an INS. This is true for most underwater navigation scenarios unless the altitude of the vehicle is beyond the range of the DVL or stealth is required, as the DVL emits acoustic signals. The DVL can also be used to give object relative velocity to map rotating and translating icebergs [52]. Additionally, the DVL is often used in hull inspections to give an error bounded estimate of the vehicle's velocity relative to the ship's hull [53].

A tightly coupled approach using the measurement of each DVL beam makes the navigation solution robust to DVL beam drop outs. This avoids situations where receiving the returns from three beams is equivalent to total drop out causing a loss in information that could be used for state estimation [47].

The DVL is a sensor that actively adds energy to the water in the form of acoustic energy from the vehicle, this is not desirable in situations where the vehicle must remain undetected. There are other (passive) localization methods that use acoustic communication where the vehicle only listens. This is similar to GPS [37]. Acoustic transponders can be configured to work this way.

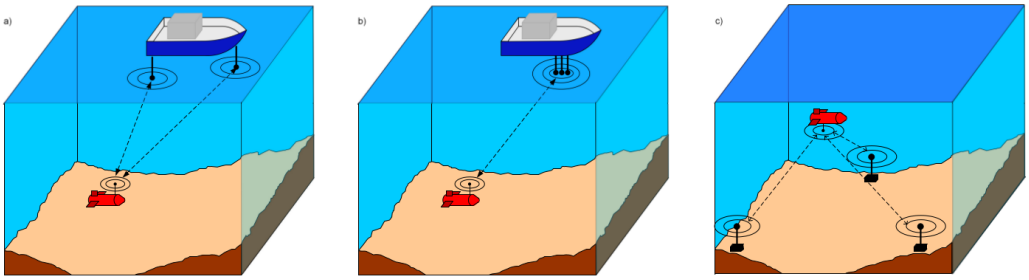

Figure (2.1) (a) Short baseline (SBL) (b) Ultra-short baseline (USBL) (c) Long baseline (LBL). Figure 1.2 from Paull et al. [54]

2.3 Acoustic Transponder Navigation

Ultra-short baseline (USBL), short baseline (SBL), and LBL are types of acoustic transponder and beacon configurations, as shown in Figure 2.1 [54]. The general concept is that the vehicle uses the acoustic transponders, the time of flight (TOF) of acoustic pings, or the phase difference between received signals to determine its location relative to the georeferenced transponders [54].

LBL beacons are distributed through the mission area and use triangulation to localize the vehicle [55]. The vehicle uses the phase difference of the USBL (transponders spaced $< 10\,\text{cm}$) to determine the heading to the USBL, and TOF to determine the distance. The SBL uses transponders on either end of the ships hull to triangulate the vehicle position [54]. LBL is more accurate over longer distances but requires stationary georeferenced beacons on the ocean floor. This is not required by the USBL and the SBL. [54].

Acoustic communication has inherent lag, due to the low speed of sound under water. This lag is on the order of $2\,\text{s}$ for an operating area of $350\,\text{m}$ by $350\,\text{m}$ [47]. Therefore, the state estimation must account for significant time delay in consecutive received signals.

A long distance acoustic transponder array was used for a 12-day under-ice mission in the Canadian High Arctic [9]. This mission is considered the longest under-ice mission to date and its purpose was to collect bathymetric survey data for Canada's Canada's United Nations Convention on the Law of the Sea (UNCLOS) application.

However, the acoustic array was used to home the vehicle over a long-distance towards a moving target, not for vehicle localization [9].

Related work by Di Qiu et al. attempted to use acoustic sources of opportunity for navigation [56]. Their work explored the use of a training phase where the vehicle detects acoustic sources within the area and geotags their location. The database of acoustic sources can then later be used for navigation. This method can be used in conjunction with TAN for applications like geomagnetic aided navigation to improve localization and navigation performance [57]. This method for navigation requires many active acoustic sources, and is most useful in areas with underwater activities [56].

A general discussion and an in depth review of the literature surrounding acoustic transponder localization and navigation can be found in Paull et al. [54] and Paull et al. [3].

Acoustic transponder localization can complement TAN or be used as ground truth localization to test TAN solutions [47]. Passive localization is achievable using acoustic transponders if the AUV has its time precisely synchronized with the transponders time. Then one-way travel time can be used to range the vehicle from the base station without the vehicle having to transmit anything. However, TAN has the benefit of not requiring stationary beacons (LBL) or other vehicles with SBL or USBL to localize the AUV. This is especially relevant for longer distance navigation ($> 1000\,\text{km}$), as many LBL based transponders would be required to provide an acceptable localization solution given LBL beacon spacing is on the order of 30 km to 50 km [58]. For these reasons acoustic transponder localization is not realistic for long distance AUV navigation. A mitigation strategy includes placing these beacons on mobile platforms (other AUVs) to perform cooperative localization.

2.4 Cooperative Localization

Cooperative localization (CL) uses several vehicles to aid with localization; for example, surface vehicles or other underwater vehicles. CL relies on the basic idea that vehicles have location uncertainty and sharing that information with other vehicles

forms a better localization estimate [3].

Underwater, this information sharing is done over the acoustic communication channel, which adds the challenges of acoustic communication to CL. CL is an expansive field that covers homogeneous and heterogeneous teams of AUVs with any assortment of vehicles, including surface vehicles. It is covered in more detail in work by Paul et al. in Chapter 11 of the Encyclopedia of Robotics [54]

Recent exploratory work focused on the use of CL to improve the performance of TAN. Distributed Kalman filters (DKFs) were suggested as a way to use a formation of AUVs to increase the localization accuracy of geomagnetic aided navigation [18]. The AUV formation could be used to extend the baseline of the magnetometer, giving a more accurate vector measurements of the magnetic field using multiple flux gate magnetometers [18]. This was proposed as better method to detect unexploded ordnances; however, experimental results implemented directly on an AUV formation were not completed [18]. Djapic et al. did present successful representative experiments as precursors to future experimentation. These experiments included swimmer formations detecting mines and two magnetometers detecting the direction of a passing car at a distance of 15 m [18].

Cooperative localization requires active communication between vehicles. This cannot be achieved passively, and therefore, does not meet the passivity requirement of the TAN algorithm. Next, improvements in mapping related to TAN will be discussed. Georeferenced maps are a prerequisite to TAN, unless performing SLAM (introduced in section 2.7).

2.5 Mapping

A map summarizes and records the measurements about a specific phenomena in an area. Maps in the context of navigation relate a measurement to a spatial position (georeferencing). There has been effort to sum different terrain maps in a region to form an integrated visual representation.

Bodus-Olkowska et al. show that the fusion of bathymetric data, sonar images, and magnetic anomaly contour data can serve as an integrated visualization summary

of the sea-floor [59]. The multi-beam echo sounder bathymetric measurements provide the depth information with precise positioning, the side-scan sonar images show the dimensions and locations of objects on the seafloor, and marine magnetometer measurements provide visualization of the ferromagnetic characteristics of features below the sea floor [59].

Sparse measurements have been interpolated using the Kriging algorithm to create prior field maps that can be used as navigational aids [60]. This method, in conjunction with the iterative iterative closest contour point (ICCP) matching algorithm and an interpolated gravitation field map, successfully limited the AUV growth in localization error [60].

Barkby et al. demonstrated the use of a Gaussian process map as the bathymetry representation during particle filter SLAM. The map representation allowed loop closure without direct re-visitation of previously covered areas [27].

It is useful to determine if navigation within an area is likely to give good results. This is referred to as navigability analysis [61]. There are numerous works on navigability analysis of gravity maps and their analysis is beyond the scope of this thesis, some related references can be found in Liu et al [61].

This section provided a brief overview of the research directly related to mapping in the context of TAN. The research aims to extract as much useful information from the available measurements, or fuse it with other data to expand the amount of usable measurements. Maps usable for TAN contain information about the different geophysical terrain types, bathymetry, magnetic, and gravity.

2.6 Geophysical Terrain Types

Each subsection here describes a geophysical terrain type that can be used for TAN. The properties of the terrain type and the sensors used to measure the terrain type are described.

2.6.1 Bathymetry

Bathymetry refers to the topography of the ocean floor. Sonar imaging is used to extract information from the bathymetry. The different types of sonar imaging are side-scan, multi-beam, forward looking, mechanical scanning and imaging, and synthetic aperture [54]. These are shown in Figure 2.2. Side-scan sonar pro-

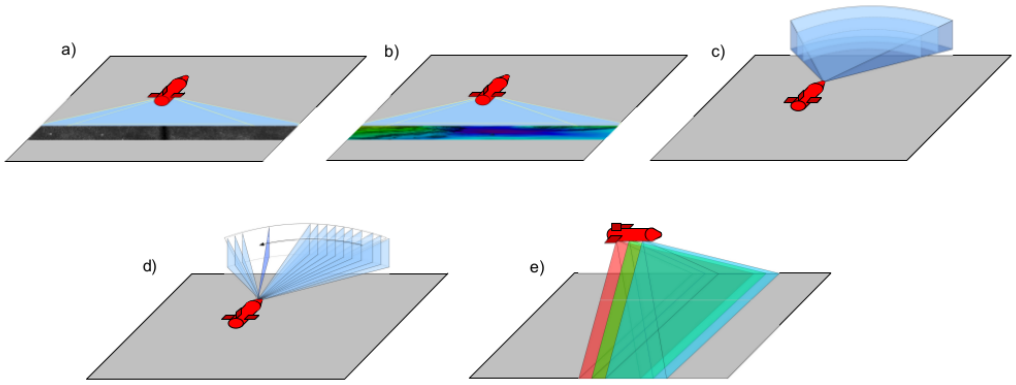

Figure (2.2) Sonar sensor swaths (a) Side scan (b) Multi-beam (c) Forward looking (d) Mechanical scanning and imaging (d) Synthetic aperture. Figure from Chapter 11 of the Encyclopedia of Robotics written by Paull et al. [54].

duces two-dimensional (2D) mosaicked images of the ocean floor beneath the vehicle. Bathymetry extraction from side-scan sonar data was demonstrated in Mackenzie et al. [62]. Multi-beam sonars uses the returns of an array of acoustic signals and specifically arranged transducers, from which a bathymetric map can be constructed [54]. The forward-looking sonar's primary purpose is mapping features directly in front of the vehicle [3]. This is useful for omni-directional remotely operated vehicles (ROV) that can approach these features at low speeds (e.g. for man-made underwater structure inspection) [3]. Unlike the multi-beam sonar, the mechanical scanning and imaging sonar uses a single beam. It takes time to scan the single beam which causes the mosaicking of these images through time to be more complex [3]. Finally, synthetic aperture sonar uses the displacement of its host vehicle to create a virtual array, making the resolution independent from the target and sensor, at the cost of complex image processing and tight vehicle motion tolerance [3]. A more comprehensive overview can be found in Chapter 11 of the Encyclopedia of Robotics written by Paull et al.[54].

2.6.2 Geomagnetic Field

Earth's total magnetic field is produced by the movement of molten iron within the Earth's core. The magnetic field varies from 25 000 nT to 70 000 nT [59].

Underwater magnetic field measurements are usually collected with surface vehicles [58], as AUV localization is poorer than global navigation satellite systems (GNSS) localization for georeferencing the collected measurements.

The total magnetic field measured by a magnetometer on-board a mobile vehicle is compromised by the main Earth field, the vehicle induced field, temporal variations, and the crustal field [63]. A magnetometer measures the total magnetic field; however, the crustal field is desired for navigation. The main Earth field is modelled by the International Geomagnetic Reference Field (IGRF) [64]; therefore, it is the easiest to remove from a sensor measurement [65]. Vehicle magnetic field compensation techniques can remove the vehicle's own magnetic field from the measurement to within sub nT accuracy [65].

Temporal variations in the measured magnetic field, often caused by space weather, can be removed using frequency filtering [63]. The filtering works as long as the frequency of the temporal variations is significantly lower than the variation due to the change in vehicle position, so it may not work as well for slow moving boats and submarines [63]. Daily temporal variations in the Earth's magnetic field can be on the order of 50 nT [58].

Among the types of magnetometers are the flux-gate and Caesium-vapor or optically-pumped magnetometer, the latter being orders of magnitude more accurate, to within a sub nT accuracy [66]. The flux-gate magnetometer measures the vector gradient of the magnetic field as well as the magnitude, whereas the Caesium-vapour magnetometer only measures the magnitude.

The magnetic field decays as $1/r^3$, where r is the distance from the magnetic source [63]. An increase in altitude acts as a low pass filter to the frequencies contained in the magnetic field, removing the higher frequency components [63]. For this reason, magnetic field maps of the crustal field sampled at lower altitudes contain more higher

frequencies. This allows maps from lower altitudes to be filtered for higher altitudes [63]. The converse is much more challenging but has been done for small altitude changes [57] and is referred to as downward continuation.

Distortions in the magnetic field created by newly introduced ferromagnetic objects have been used to locate underwater meteorites using magnetic field surveys [67].

2.6.3 Gravitational Field

Gravitational acceleration is inversely proportional to the square of the distance to the mass contributing to the gravity field. When measuring the gravitational field with a gravimeter, the measurement consists of the magnitude of the gravity field, the vehicle induced acceleration, the thermal drift, the tidal effect, and the perceived gravitational effect (Eötvös effect) [68]. Gravimeters measure the magnitude of the gravity field, whereas gravity gradiometers measure the gradient of the magnetic field. Gravity gradiometers have the benefit of being insensitive to vehicle-induced acceleration, thermal drift, and the Eötvös effect [68]. The signal of interest is the gravity anomaly, the difference between the geoid gravity and the normal gravity [44]. The geoid gravity acceleration is calculated perpendicular to the surface of the geoid. The normal gravity acceleration is the gravity calculated with the reference ellipsoid. Gravity anomalies on the Earth's surface range from approximately -50 mGal to 50 mGal [69], where 1 mGal is $10^{-5}\, m/s^2$.

The tides occur due to the gravitational pull of the Moon. The cyclic movement of large fluid masses has significant effects on the gravitational field. The Eötvös effect occurs due to the change in centrifugal acceleration induced by the change in vehicle angular velocity around Earth's axis. The effect is induced by velocity of the vehicle in the direction or anti-direction of the Earth's spin [70]. These effects are important when trying to isolate the spatial information from the gravity field.

The tidal effect can be removed using tidal models; the vehicle induced acceleration can be removed by taking accurate measurements of the vehicle depth; thermal drift is managed by actively keeping the sensor temperature constant; and the perceived gravitational effect can be removed using accurate measurements of the vehicle's direction of travel relative to the rotation of the Earth [68].

When performing surveys of marine gravitational fields it is important to minimize the effect of vehicle-induced accelerations on measurements [71]. AUVs have the advantage of being underwater, i.e. unaffected by higher frequency waves, making them ideal for marine gravity surveys [71].

Gravimeters are often harder to integrate into smaller AUVss [72]. Recent work integrated a gravimeter into a 50 cm diameter and 70 cm height titanium pressure vessel [68] — where a small AUV has a cylindrical diameter of 14 cm [73]. This is one reason for the lack of their wide spread use. The development of a custom gravimeter to detect sub ocean floor mineral deposits successfully met a resolution requirement of $0.1\,mGal$ ($1 \times 10^{-5}\,\mathrm{m\,s^{-2}}$), and was designed specifically to be integrated in an AUV [68].

Ishihara et al. attempted to use the gravity field produced by high-density volumes to detect sub ocean floor mineral deposits [74]. However, they were unsuccessful in detecting the suspected gravity anomalies [74] at an AUV 40 m altitude, using the custom gravimeter developed by Shinohara et al. [68].

Due to their relationship with bathymetry, gravity field maps can be used to plan shipboard surveys, as ocean structures that are greater than 10 km to 15 km appear in maps with an accuracy of $4 - 7$mGal [75]. These maps can also be used for undersea volcano discovery, petroleum exploration, and tectonic plate research [75]. AUV-based gravity surveys were suggested to enable exploration of small-scale (1 m to 1000 m) features in the oceanic crust [76]. AUV-based gravity surveys would allow measurements at depth, closer to the small-scale features [76]. Being closer to the mass source that produces the gravity anomaly is beneficial because the acceleration due to gravity attenuates proportional to the square root of the distance to the source mass [44]. This is especially desirable around the mid-ocean ridge (MOR), a hot-spot for critical chemical and biological processes [76].

The quality of digital elevation model (DEM) derived gravity field maps was shown to increase from an error of 23% to an error of 5% when density maps where included rather than making the constant density assumption [77]. However, density maps are still uncommon.

Finally, there has been exploration into using the gravitational field for obstacle avoidance. Yan et al. presented an obstacle detection method based on the change in the gravity gradient that successfully detected objects with a mass greater than 1×10^9 kg from 600 m away.

These geophysical terrain types were all considered when developing the TAN algorithm. Bathymetry requires the use of active sensors, and therefore does not meet the passivity requirement. The magnetic field has significant temporal variation [58], is not persistent, and requires filtering to remove the vehicle's affect on the magnetic field[63]. The gravity field is persistent [44], and gravity gradiometer have the benefit of being unaffected by the vehicle-induced acceleration, thermal drift, and the Eötvös effect [68].

When maps of the geophysical terrain types do not exist, there is the possibility of mapping and using those maps at the same time to localize. This is simultaneous localization and mapping (SLAM). SLAM is an extension of TAN that removes the requirement for *a priori* maps. It is outside the scope of work within this thesis, however, it is briefly reviewed to acknowledge the importance of this extensive field.

2.7 Simultaneous Localization and Mapping

Another field with some attention is SLAM. Its main attractive feature is the ability to place a vehicle in an environment without prior maps and have the vehicle localize itself within the map that it incrementally builds [53]. If the vehicle successfully recognizes an area that it has previously visited, it can set the drifted position equal to the previous position and propagate the change backward to decrease the error of all previous measurements [53]. This recognition of an already visited area and subsequent propagation of uncertainty removal is referred to as loop closure [78].

There have been recent developments in the area of SLAM in the context of TAN. Begum et al. proposes the idea of using island model genetic algorithm (IGA) based SLAM [79]. The work aims to use the property of natural selection to select the most probable map for the loop closing, while keeping multiple hypotheses for the robot's possible position. It was experimentally tested within a small structured office

environment, were the requirements of speed and memory efficiency are less than in a longer mission in an ocean vehicle with limited computation power.

Extensive experimental work was presented by Barkby et al. using particle filter based SLAM and a Gaussian process map model in an underwater environment. The map model for the bathymetry was shown to allow loop closure without significant terrain overlap [27]. Barkby et al. also presented the incorporation of low-resolution bathymetric maps into SLAM as the prior belief [80]. This also improved the TAN performance.

The likelihood of a SLAM solution succeeding for a gravitational field environment is explored by Pasnani et al. [17]. Their work found that the landmark-to-landmark variability was a better representation of the potential for SLAM success. This measure can be used to assess the feasibility of a navigation solution before investing resources [17].

Next, the different methods of TAN, using the geophysical terrain types introduced in section 2.6, are discussed. The latest research developments in each area are explored and discussed, leading to the conclusion that gravity aided navigation will be used for the implementation of the TAN algorithm.

2.8 Terrain Aided Navigation

In the context of vehicle localization, TAN uses the measurement of features that contain location-specific information to localize the vehicle. However, terrain is often used interchangeably with ocean floor bathymetry.

One of the most difficult environments to navigate within is the under-ice environment. Under-ice environments are low-contrast and have repetitive ice textures. The presence of the ice which makes it difficult or impossible for surface ship aiding [54] so navigational aids like acoustic transponder arrays [54] cannot be easily deployed. Therefore, there is research that specifically deals with challenges faced in under-ice conditions (e.g. [9], [52], [55]). The method for TAN chosen aims to not limit the possibility of navigating in such areas.

There are three general types of geophysical terrain types that contain location specific information within a measurement: bathymetry, the geomagnetic field, and gravitational field, introduced in section 2.6.

2.8.1 Bathymetry Aided Navigation

Bathymetry based navigation is the most established, as sonars are mature and a single measurement is information rich relative to a single gravitational or geomagnetic field measurement. Research has aimed to make using bathymetry measurements as a navigational aid more robust and accurate.

A novel way to store bathymetric particle filter based grid maps was introduced by Barkby et al. [78]. The method uses the redundancy and overlap in the bathymetric map stored by each particle to build a particle ancestry-based tree data structure. This algorithm reduces the algorithmic complexity of the distributed partical mapping (DPM) to linear complexity. The work shows that particle filter based SLAM using bathymetric measurements is not only feasible but robust through extensive experimentation. [78].

Work by Decktor and Rock shows that SLAM performance can be significantly improved if the information content of the bathymetry is taken into consideration when weighting measurement updates [51]. The experimental results showed the method prevented overconfidence of the state estimate and maintained nominal convergence rates in areas with featured bathymetry. These considerations considerably improve the robustness of the particle filter based state estimate in featureless bathymetry [51].

Marine glider vehicles have also received attention as a platform for TAN [15]. Claus and Bachmeyer show that a particle filter based algorithm is successful in limiting the growth of position error from 5.5 km to 90 m [15]. Glider vehicles especially benefit from TAN and other navigational aids, as dead-reckoning drifts significantly due to their unique method of vehicle actuation.

2.8.2 Geomagnetic Aided Navigation

As distance from the source increases, created magnetic fields decay faster than radio frequency signals, making geomagnetic aided navigation harder to jam [63]. This makes geomagnetic navigation attractive as a back up navigation system for aerial vehicles that rely heavily on GNSS [63]. Unlike navigation that uses acoustic sensors or communications, measuring the magnetic field does not emit signals; therefore, it can be used for missions that require stealth.

There are also situations were the magnetic field contains more features than bathymetry, making it a better candidate for localization [57]. This depends on the mission environment. There has been work to apply familiar state estimation techniques and navigation techniques to geomagnetic aided navigation.

Additionally, Mu et al. proposed adaptive EKF that uses stochastic linearization to solve the problem of linearization error by determining the linearization region, size, and uncertainty. The work aims to minimize the magnetic field linearization error to decrease the potential localization error of the EKF-based geomagnetic aided navigation [19].

There has been research specifically focused on the affordability and size of the AUV platform [57]. Quintas et al. showed that geomagnetic aided navigation can be applied on smaller vehicles [57]. This shows that research in underwater geomagnetic aided navigation is potentially less cost-prohibitive for smaller research institutions.

2.8.3 Gravity Aided Navigation

Gravity aided navigation (GAN) has seen significant attention due to it being a stealth navigation methodology, being unjammable, and able to derive gravity field maps from widely available and high resolution DEM [43]. Recent developments have applied known state estimation techniques, and matching algorithms to GAN.

Wang et al. used a notable GAN solution in the South China Sea [72]. The algorithm was not implemented on-board an AUV because the gravimeter was too large. Therefore, the measurements were collected by a surface ship which can be georeferenced against the ship's GNSS position as a form of ground truth [72]. The

position drift was successfully limited to 3 nm. The dead-reckoning drift during the trail was 200 nm [72]. ‘

Han et al. discuses the use of classic matching algorithms for GAN, and states that ICCP matching for GAN relies on small inertial navigation error to avoid divergence and mismatching [81]. The original terrain contour matching (TERCOM) algorithm has high computational complexity, therefore, Han et al. suggest a TERCOM-based algorithm based on the shortest path algorithm to achieve constant time complexity [81].

There are numerous examples of GAN used in the literature to supply the position error bound on the otherwise unbounded INS error. Liu et al. applied both EKF and UKF based state estimation with gravity measurement based updates and effectively limited the position error [24]. There are papers as early as 1990 that implement GAN [82]. Jircitano et al. from Bell Aerospace discuss the merits of using gravity gradient as a method for gravity aided navigation. Parameter studies using synthetic terrain are performed to show the effect of terrain quality on the performance of the gravity gradiometer navigation system [82]. Jircitano et al. mention bathymetry can be used interchangeably with the gravity anomaly information when the gravity anomaly information is not available. However, there is no mention of actually performing and testing the gravity gradiometer navigation system using the bathymetry [82]. My thesis aims to explore this work.

There has been other work adapting EKF state estimation to GAN. One important aspect is the linearization error that comes with the EKF. Xiong et al. presented a neural network approach to linearization of the gravity map within the context of EKF state estimation [83]. The method was compared to other linearization methods like nine point fit (NPF), first order Taylor series (FTS), full plane fit (FPF) technique, and two subgroup fit (TSF). This work shows that using neural network linearization effectively reduces the linearization error of the EKF in simulation, and has significant benefits for highly non-linear gravitational field measurements [83].

Finally, small improvements to the unscented Kalman filter specifically for gravity-aided navigation have been proposed [25], making the UKF relevant to GAN.

Bathymetry aided navigation requires the use of active sensors, and therefore does not meet the passivity requirement, it was discussed to be complete in the coverage of TAN. The magnetic field has significant temporal variation [58], is not persistent, and requires complex filtering to remove the vehicle's effect on the magnetic field[63]. This makes magnetic aided navigation less attractive than the alternative, GAN. The gravity field is persistent [44], and effectively unjammable [43]. If a gravity gradiometer is used to perform GAN, then measurements are unaffected by the vehicle induced acceleration, thermal drift, and the Eötvös effect [68]. Using the gravity gradient facilitates the correlation between bathymetry and the local gravity anomaly [43]. This would make the requirement of *a priori* maps less restrictive, by using large readily available maps from global databases of bathymetry like General Bathymetric Chart of the Oceans (GEBCO) [29].

This concludes the review of the literature in field of AUV navigation. The choice of using GAN with a gravity gradiometer has been justified. The following sections cover details needed for some of the mathematical concepts used in the thesis. This starts with a section on reference frames (section 2.9), a detailed explanation of the EKF (section 2.10), and quaternion conventions (section 2.11). The two latter sections cover the background needed for the implementation of AUV dead-reckoning, later discussed in section 4.1.1. Next, is a section on environment modelling (section 2.12). This section covers the background for the implementation of some of the tools created as part of the ANT. This includes Perlin noise in section 2.12.1 and the derivation of the local gravity anomaly maps using the correlation between the bathymetry and the gravity anomaly in section 2.12.2. The background chapter concludes with a reiteration of the thesis motivations and the choices as informed by the literature.

2.9 Reference Frames

This report uses a right-handed Cartesian coordinate system, where counter-clockwise rotations around an axis are defined as positive. This is consistent with the convention in the Gazebo simulator. The vehicle has a body-centered reference frame. The x, y, and z coordinate axis of the body-centered reference frame are shown in Fig. 2.3 as the red, blue, and green axis, respectively. This convention is used by the ROS

visualization tool **rviz**, and is kept consistent in this report. This body-centered

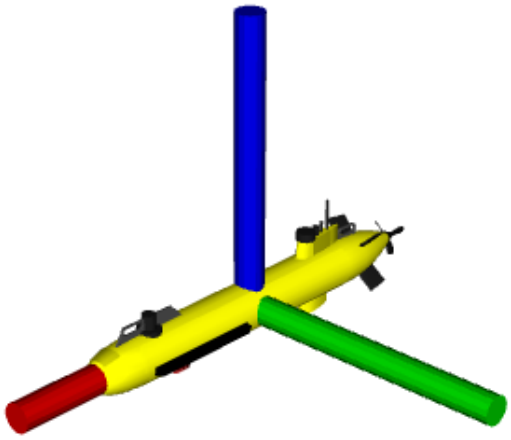

Figure (2.3) The body-centered reference frame x, y, and z axis shown by the red, blue, and green axis, respectively.

reference frame is denoted by the subscript $\mathcal{L}$, for local reference frame. The state estimate of the EKF is performed in the global reference frame, in this case the world frame within the Gazebo simulation environment. The world frame is technically a Cartesian plane approximation of the Earth's surface evaluated at a specific origin. Using an Earth-fixed reference frame was not required. The global reference frame is denoted by a subscript $\mathcal{G}$, for global.

The DVL has its own reference frame that is rotated by $+\pi$ around the y axis. A visual of the DVL reference frame is shown in Fig. 2.4.

On a vehicle, sensors are limited by size of the vehicle and the power requirement

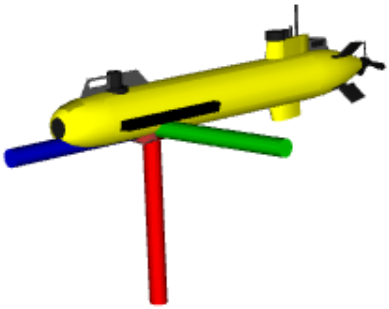

Figure (2.4) The DVL reference frame in which the DVL measurements are recorded and published.

of the sensors. In the simulated vehicle the on-board sensors include an inertial measurement unit (IMU), an altimeter, a depth sensor and a DVL. The IMU includes a 3-axis gyroscope and a 3-axis accelerometer. The gyroscope measures the rotation rates around the axes in the body-centred reference frame $\boldsymbol{\omega}_{\mathcal{L}}$. The accelerometer measures the accelerations along the axis of the body-centered reference frame $\boldsymbol{a}_{\mathcal{L}}$. With integration, the vehicle position, velocity, and attitudes can be estimated.

The system state can be formulated to not include vehicle dynamics if the inertial measurements can be measured at a high enough frequency. Instead of using control inputs as the inputs to the model, the effect of the control inputs on the vehicle's inertia can be measured directly with high frequency IMU measurements. This allows the IMU to be used as a replacement for a vehicle dynamic model [84] [40] [37, pp. 121–122, 361–365]. The IMU sensor model is defined by eq. (2.1),

$$\begin{aligned} \boldsymbol{a}_{\text{IMU}} &= \boldsymbol{a} + \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{a}}^2) \\ \boldsymbol{\omega}_{\text{IMU}} &= \boldsymbol{\omega} + \mathcal{N}(\mathbf{0}, \Sigma_{\boldsymbol{\omega}}^2) \end{aligned} \tag{2.1}$$

where $\boldsymbol{a}$ is the ground truth acceleration vector of the vehicle in the IMU-centred reference frame, $\boldsymbol{\omega}$ is the ground truth angular velocity vector in the IMU-centered reference frame, $\Sigma_{\boldsymbol{a}}^2$ is the accelerometer covariance matrix in the IMU-centred reference frame, and $\Sigma_{\boldsymbol{\omega}}^2$ is the gyroscope covariance matrix in the imu-centered reference frame. The goal of the INS model was black-box modelling of the Kearfott INS. IMU sensor drift was not needed to emulate the position error drift properties of the INS, therefore, it was not included in the IMU model. A full detailed implementation of an INS was not necessary, and outside the scope of this thesis.

The DVL measures the speed-over-ground directly in the DVL-centered reference frame relative to the world frame. This is used to bound the error of the position estimate from the integrated accelerometer measurements. The DVL is a Gazebo plugin as part of the UUV Simulator package used [33]. The DVL is model is defined by eq. (2.2),

$$\dot{\boldsymbol{x}}_{\text{DVL}} = \dot{\boldsymbol{x}}_{\text{DVL}} + \mathcal{N}(\mathbf{0}, \Sigma_{\text{DVL}}^2) \tag{2.2}$$

where $\dot{\boldsymbol{x}}_{\text{DVL}}$ is the ground truth velocity vector of the DVL sensor (in the DVL-centered reference frame), and Σ^2_{DVL} is the covariance matrix associated with the DVL measurement.

A gravity gradiometer will be used as input to the gravity gradient-aided localization. The gradiometer sensor model is a simplified model that uses a gravity gradient map derived from the original bathymetry and an artificially generated density field.

$$\begin{bmatrix} \theta(\boldsymbol{x})_m \\ M(\boldsymbol{x})_m \end{bmatrix} = \begin{bmatrix} \arctan\left(\frac{\partial g(\boldsymbol{x})}{\partial x}, \frac{\partial g(\boldsymbol{x})}{\partial y}\right) \\ \sqrt{\left(\frac{\partial g(\boldsymbol{x})}{\partial x}\right)^2 + \left(\frac{\partial g(\boldsymbol{x})}{\partial y}\right)^2} \end{bmatrix} \tag{2.3}$$

where $g(\boldsymbol{x})$ is the gravity anomaly at the gradiometer location in the global reference frame, θ_m is the gravity gradient heading in xy-plane, M_m is the gravity gradient magnitude in the xy-plane.

These reference frame conventions will be used when introducing and implementing EKF-based dead-reckoning.

2.10 Extended Kalman Filter (EKF)

The integrated navigation system on-board the simulated vehicle uses an EKF for state estimation. The EKF is an algorithmic framework for recursive state estimation using a predict-update structure and is covered extensively in numerous references. The references used are Budiyono's Principles of GNSS, Inertial, and Multi-sensor Integrated Navigation Systems [37] and Ruiter's book Spacecraft Dynamics and Control: An introduction [85]. Also used was a report by the Army Reasearch Laboratory (ARL) and as a supplemental resource on strap-down inertial navigation using quaternion based EKF [86].

As an extension of the Kalman Filter for non-linear state estimation, the EKF relies on the assumption of white noise (Gaussian distributed, zero-mean). The EKF is a Taylor series approximation of the optimal Kalman filter and, therefore, does not provide optimal results, only an approximation [85].

The EKF uses a prediction of the state $\boldsymbol{X}_{k|k-1}$ using $\boldsymbol{X}_{k-1}$ and $\boldsymbol{U}_{k-1}$, where $\boldsymbol{X}_{k|k-1}$

is the prediction of the current step given the previous step, k is the current time step, $k-1$ is the previous time step, and $\boldsymbol{U}$ is the input vector from time step k to $k-1$. It also predicts the covariance matrix $\boldsymbol{P}_{k|k-1}$ given the previous covariance matrix $\boldsymbol{P}_{k-1}$. The prediction step of the EKF is given by eqs. (2.4) and (2.5),

$$\boldsymbol{X}_{k|k-1} = f(\boldsymbol{X}_{k-1|k-1}, \boldsymbol{U}_{k-1}) \tag{2.4}$$

$$\boldsymbol{P}_{k|k-1} = \boldsymbol{F}_{k-1}\boldsymbol{P}_{k-1|k-1}\boldsymbol{F}_{k-1}^{T} + \boldsymbol{L}_{k-1}\boldsymbol{Q}_{k-1}\boldsymbol{L}_{k-1}^{T} \tag{2.5}$$

where the matrices $\boldsymbol{F}_{k-1}$ is the state transition function Jacobian relative to the state defined by eq. (2.6), $\boldsymbol{L}_{k-1}$ is the state transition function Jacobian relative to the process noise defined by eq. (2.7), and $\boldsymbol{Q}_{k-1}$ is the process noise covariance matrix.

$$\boldsymbol{F}_{k-1} = \left.\frac{\partial f}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}_{k-1|k-1}, \boldsymbol{U}_{k-1}} \tag{2.6}$$

$$\boldsymbol{L}_{k-1} = \left.\frac{\partial f}{\partial \boldsymbol{w}}\right|_{\boldsymbol{X}_{k-1|k-1}, \boldsymbol{U}_{k-1}} \tag{2.7}$$

The $\boldsymbol{w} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{Q}_k)$ in eq. (2.6) is the process noise. If the additive process noise assumption holds then $\boldsymbol{L}_{k-1}$ becomes the identity matrix. In the case here, this assumption is violated; this is explained in section 4.1.1.

The update stage of the EKF uses a measurement $\boldsymbol{Z}_k$ and the predicted state $\boldsymbol{X}_{k|k-1}$ to determine the approximately optimal resultant state estimate by fusing the measurement information and uncertainty with the current state estimate and uncertainty. A more detailed discussion on error sources will be covered in section 4.1.1. The updated state is determined using eqs. (2.8) to (2.12):

$$\tilde{\boldsymbol{v}}_k = \boldsymbol{Z}_k - h(\boldsymbol{X}_{k|k-1}) \tag{2.8}$$

$$\boldsymbol{S}_k = \boldsymbol{H}_k\boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^{T} + \boldsymbol{M}_k\boldsymbol{R}_k\boldsymbol{M}_k^{T} \tag{2.9}$$

$$\boldsymbol{K}_k = \boldsymbol{P}_{k|k-1}\boldsymbol{H}_k^T\boldsymbol{S}_k^{-1} \tag{2.10}$$

$$\boldsymbol{X}_{k|k} = \boldsymbol{X}_{k|k-1} + \boldsymbol{K}_k\tilde{\boldsymbol{v}}_k \tag{2.11}$$

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1}(\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)^T + \boldsymbol{K}_k\boldsymbol{M}_k\boldsymbol{R}_k\boldsymbol{M}_k^T\boldsymbol{K}_k^T. \tag{2.12}$$

Equation (2.8) gives the estimated measurement residual $\tilde{\boldsymbol{v}}_k$ or innovation, where h is the non-linear measurement function. Equation (2.9) gives the measurement residual covariance $\boldsymbol{S}_k$ where $\boldsymbol{H}_k$ is the Jacobian of h relative to the state $\boldsymbol{X}$ (eq. (2.13)),

$$\boldsymbol{H}_k = \left.\frac{\partial h}{\partial \boldsymbol{X}}\right|_{\boldsymbol{X}_{k|k-1}} \tag{2.13}$$

and $\boldsymbol{M}_k$ is defined as the Jacobian of the measurement with respect to the measurement noise $\boldsymbol{v}_k \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{R}_k)$,

$$\boldsymbol{M}_k = \left.\frac{\partial h}{\partial \boldsymbol{v}}\right|_{\boldsymbol{X}_{k|k-1}}. \tag{2.14}$$

If the additive measurement noise assumption holds then $\boldsymbol{M}_k$ becomes the identity matrix. Equation (2.10) gives the near-optimal gain $\boldsymbol{K}_k$ for the filter update. The state update is given by eq. (2.11). Finally, the covariance update is defined by eq. (2.12) [87, pp. 55–57], referred to as the Joseph form of the covariance update equation [88]. The shorter covariance update equation given by,

$$\boldsymbol{P}_{k|k} = (\boldsymbol{I} - \boldsymbol{K}_k\boldsymbol{H}_k)\boldsymbol{P}_{k|k-1} \tag{2.15}$$

assumes that the optimal gain is used. There were issues with the symmetry of the covariance matrix update due to double point limited arithmetic precision. Equation (2.12), or the Joseph form guarantees the resulting covariance update to return a symmetric matrix, as it works for all gains [88].

This section introduced the general form of an EKF, the implementation of the EKF-based INS in section 4.1.1 makes heavy use of quaternion notation. The next section covers the notation and the convention used in this thesis.

Table (2.1) The two major conventions for quaternions. Adapted with minor modifications from [40]

Quaternion Type	**Hamilton**	**JPL**
Components order	$(q_w, \boldsymbol{q}_v)$	$(\boldsymbol{q}_v, q_w)$
Algebra Handness	$ij = k$ Right-handed	$ij = -k$ Left-handed
Function	Passive	Passive
Right-to-left product means Default notation, $\boldsymbol{q}$ Default Operation	Local-to-Global $\boldsymbol{q} \triangleq \boldsymbol{q}_{\mathcal{GL}}$ $\boldsymbol{x}_{\mathcal{G}} = \boldsymbol{q} \otimes \boldsymbol{x}_{\mathcal{L}} \otimes \boldsymbol{q}^*$	Global-to-Local $\boldsymbol{q} \triangleq \boldsymbol{q}_{\mathcal{LG}}$ $\boldsymbol{x}_{\mathcal{L}} = \boldsymbol{q} \otimes \boldsymbol{x}_{\mathcal{G}} \otimes \boldsymbol{q}^*$

2.11 Quaternion Conventions

The orientations are represented with unit quaternions. This avoids singularities and requires only polynomial multiplications to perform rotations and inverse rotations. There are many conventions found within the literature; Solà found seven common quaternion conventions and states that the convention used within a piece of literature is often not stated [40]. Table 2.1 summarizes the two major quaternion conventions: the Hamilton, and the Jet Propulsion Laboratory (JPL).

Equation (2.16) shows the convention used for the implementation of the EKF,

$$\boldsymbol{q} = \begin{bmatrix} \epsilon_x \\ \epsilon_y \\ \epsilon_z \\ \eta \end{bmatrix} \tag{2.16}$$

where $\boldsymbol{\epsilon}$ is the vector part of the quaternion and η is the scalar part. This convention is used in [85], which is an excellent reference on quaternions to represent attitude. However, the left-handed convention is used for a quaternion, and the right-handed one is used here. This means that the right-to-left product transforms local to global. This is convenient, and found to be more intuitive. Given the convention in eq. (2.16)

the quaternion product is defined by eq. (2.17),

$$\boldsymbol{p} \otimes \boldsymbol{q} = \begin{bmatrix} p_\eta \boldsymbol{q}_\epsilon + q_\eta \boldsymbol{p}_\epsilon + \boldsymbol{p}_\epsilon \times \boldsymbol{q}_\epsilon \\ p_\eta q_\eta - \boldsymbol{p}_\epsilon^T \boldsymbol{q}_\epsilon \end{bmatrix}. \tag{2.17}$$

To perform a rotation of a vector in 3-space by a unit vector quaternion, quaternion inversion and conjugation needs to be defined. Quaternion conjugation is defined in eq. (2.18),

$$\boldsymbol{q}^* = \begin{bmatrix} -\boldsymbol{\epsilon} \\ \eta \end{bmatrix} \tag{2.18}$$

where * is the conjugate operator. Conveniently, if the quaternion is of unit magnitude, the inverse is equivalent to the conjugate of the quaternion. This makes inversion simple and efficient.

Additionally, quaternions can be pure. This means that the scalar part is zero. Therefore, with the right-handed convention and quaternion definition, rotating a vector in 3-space by a unit quaternion is defined by eq. (2.19),

$$\begin{bmatrix} \boldsymbol{x}_\mathcal{G} \\ 0 \end{bmatrix} = \boldsymbol{q} \otimes \begin{bmatrix} \boldsymbol{x}_\mathcal{L} \\ 0 \end{bmatrix} \otimes \boldsymbol{q}^* \tag{2.19}$$

where $\boldsymbol{x}_\mathcal{G}$ is the 3-space vector in the global reference frame, $\boldsymbol{x}_\mathcal{L}$ is the 3-space vector in the local reference frame, and $\boldsymbol{q}$ is the unit quaternion representation the local references frames orientation relative to the global reference frame. Equation (2.19) will allow the rotation of the acceleration vector $\boldsymbol{a}$, measured by the accelerometer, into the global reference frame; the reference frame the EKF is using. Additionally, the rotation from global to local frame uses the inverse of the quaternion to perform the rotation operation (eq. (2.19)),

$$\begin{bmatrix} \boldsymbol{x}_\mathcal{L} \\ 0 \end{bmatrix} = \boldsymbol{q}^* \otimes \begin{bmatrix} \boldsymbol{x}_\mathcal{G} \\ 0 \end{bmatrix} \otimes \boldsymbol{q} \tag{2.20}$$

where $\boldsymbol{q}$ is defined as the rotation of our local frame within the global frame; the convention in this report. This is also the convention used by the Gazebo simulator, and standard ROS geometry messages package. However, the ROS message refers to

η as w and the components of $\boldsymbol{\epsilon}$ directly as x, y, and z.

Finally, global reference frame quaternion rates in terms of the measured local angular rate becomes important in the formulation of the EKF and is defined by eq. (2.21) [40],

$$\dot{\boldsymbol{q}} = \frac{1}{2}\boldsymbol{q} \otimes \omega_{\mathcal{L}} \tag{2.21}$$

where $\omega_{\mathcal{L}}$ is the local reference frame 3-space angular rate. The angular rate is given by the gyroscopes.

2.12 Environment Modelling

Environment modelling covers two topics. First, Perlin noise, used in terrain generation and terrain augmentation. Both are capabilities provided by the ANT. Second, and finally, the mathematical description of the correlation between the terrain and the gravity anomaly. This relationship is used to exploit large databases of readily available bathymetry for GAN.

2.12.1 Perlin Noise

Perlin noise is a coherent pseudo-random noise. Coherent meaning the resulting function is smooth on smaller scales and pseudo-random on larger scales. Small and large are relative to how the Perlin noise function is parameterized. Two points close to each other will have similar values and two points far away from one another will be uncorrelated. Perlin noise is generated by sampling random numbers on a regular grid. These random numbers become the gradient of the function at those regular grid locations. The gradients are then interpolated using the fifth-order interpolation function in eq. (2.22) (from [42]),

$$f(t) = 6t^5 - 15t^4 + 10t^3 \tag{2.22}$$

where $f(t)$ is the interpolation function parameterized in terms of t. Using eq. (2.22) ensures that the first and second order derivatives are zero at both $t = 0$ and $t = 1$ corresponding to the regular grid. This provides at least first and second order continuity between neighbouring grid areas.

Perlin noise is heavily used in computer graphics for texture generation [42] [89], terrain generation, and other applications that benefit from a form of pseudo-random noise function that is continuous in at least its first two derivatives. Tian et al. use Perlin noise together with the fast Fourier transform (FFT) to simulate the ocean surface for computer graphics rendering [89]. It is common to find pseudo-random noise tools within most 3D graphics software [34].

Additionally, it can be used for generating missing terrain detail where real measurements are not detailed enough for creating a high-resolution model used in simulation [90]. For example, Martin et al. use Perlin noise for creating high-resolution asteroid models to enable simulation of spacecraft cameras and other sensors during spacecraft approach and landing on asteroids [90]. Perlin noise is used along with erosion modelling to supplement low resolution models of the asteroids.

An example of a single two-dimensional Perlin noise function is shown in Fig. 2.5. The Perlin noise function is N-dimensional so can be used in a two-dimensional spatial domain. It has a period of 2500 m and, inversely, a frequency of $1/2500\,m^{-1}$. The grid shows the intersections where the value of the gradient was pseudo-randomly generated by the definition of Perlin noise. In between these intersections the function is smoothed using eq. (2.22). When using Perlin noise to add realistic variation multiple layers of Perlin noise are often overlaid to produce the desired affects. These multiple superimposed layers are referred to as octaves. This is due to successive Perlin noise octaves having double the frequency of the previous layer. When discussing multiple superimposed octaves of Perlin noise there are a few mathematical concepts that are used to describe the relationship between successive Perlin noise octaves: lacunarity, persistence, base frequency, and base amplitude. Lacunarity is the ratio of successive octaves frequencies. The lacunarity defined recursively by eq. (2.23).

$$l = f_{k+1}/f_k \tag{2.23}$$

where f is the spatial frequency, and k is the index of the Perlin noise octave. Often the lacunarity is two (hence multiple layers are called octaves). Persistence is the

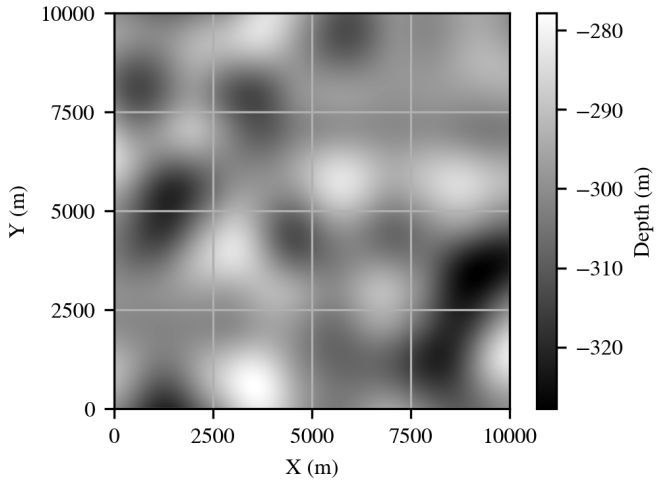

Figure (2.5) Adapted from Heubach and Seto [31]. A single octave two-dimensional Perlin noise function. The Perlin noise function has a period of 2500 m. The zero crossings occur at the intersections of the grid. The average value of this Perlin noise function has been adjusted from 0 m to −300 m in preparation of using it as a possible seascape.

ratio of successive octaves amplitudes. It is defined recursively by eq. (2.24)

$$r = A_{k+1}/A_k \tag{2.24}$$

where r is the persistence between successive Perlin noise layers, and A is the amplitude of the k^{th} Perlin noise layer. The base frequency is the frequency of the first Perlin noise layer. This first Perlin noise layer has the largest period and lowest frequency. Finally, the base amplitude is the amplitude of the first perlin noise layer. These two base properties define the initial value for the recursive definition of the lacunarity and persistence. An example Perlin noise function that contains five octaves, a lacunarity of 2 m, a persistence of 0.5 m, and a mean of −300 m is shown in Fig. 2.6. From Fig. 2.6 it is clear that successive octaves add to the detail within the terrain and significantly add to the realism of the terrain. The consequences of this are discussed further in section 4.1.3.

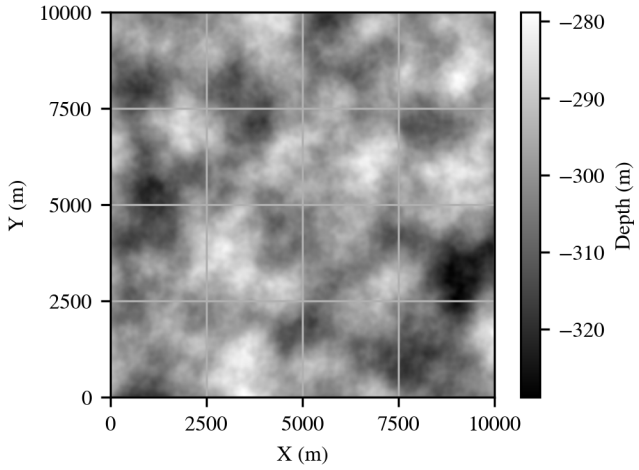

Figure (2.6) Five octaves of Perlin noise scaled to an area of 10 km. The five Perlin noise octaves have a lacunarity of 2, persistence of 0.5, base amplitude of 50 m, and base frequency of $1/2500m^{-1}$. Adapted from [31].

2.12.2 Correlation between Gravity Anomaly and Terrain

The correlation between the gravity anomaly and ocean bathymetry originates from the fundamental relationship between mass and gravity. The normal gravity is the gravitational acceleration calculated at the surface of the Earth's reference ellipsoid (e.g.the WGS 1984 reference ellipsoid used by GPS). There are multiple reference ellipsoids aiming to approximate the geoid — a more accurate model of the Earth that includes undulations. The geoid is an equipotential surface equivalent to the mean of the ocean's surface — if the oceans were static at equilibrium extended through continents. The geoid can only be discovered through extensive gravity measurements.

The difference between the gravity vector at the surface of the geoid, $\boldsymbol{g}_P$, and the gravity vector on the surface of the reference ellipsoid, $\boldsymbol{\gamma_Q}$, is defined as the gravity anomaly vector $\Delta\boldsymbol{g}$ [44, p.83]. The point Q is the projected along the ellipsoidal normal onto the reference ellipsoid to get point P. The difference in magnitude of $\Delta\boldsymbol{g}$ is referred to as the gravity anomaly [44, p.83].

To approximate the gravity anomaly, view the terrain as a distributed local source of mass variation — the same mass variation that gives the geoid its undulations. However, the terrain only gives the shape of the Earth's surface, not its internal

density variation. If constant density is assumed, then one can approximate the gravity anomaly with the ocean bathymetry which is readily available. This terrain correlation was used by Liu et al. in [43], Martine in [91], and Guo et al. in [92]. The basis of the terrain correlation equation can be derived using Newton's equation of gravity potential.

Given a distribution of n masses the gravitation potential at point p is defined by the sum of all contributions to the gravitational potential by each mass. Formally, this gives eq. (2.25),

$$V(\boldsymbol{p}) = -\sum_{i=1}^{n} \frac{Gm_i}{|\boldsymbol{p} - \boldsymbol{p}_i|} \tag{2.25}$$

where V is the gravitational potential associated with a finite mass distribution, $\boldsymbol{p}$ is the vector to the point at which the gravitational potential is being calculated, $\boldsymbol{p}_i$ is the vector to the centre of mass of the i^{th}point mass, m_i is the mass of the i^{th}point mass, G is the universal gravitational constant 6.674×10^{-11}, and i is the index identifying each point mass.

The acceleration, $\boldsymbol{a}$, due to this distributed mass can be calculated by taking the negative gradient of the gravitational potential V,

$$\boldsymbol{a} = -\nabla V(\boldsymbol{p}) \tag{2.26}$$

where ∇ is the gradient operator. Using eq. (2.26), derive the equation for the vertical acceleration a_z using eq. (2.29). Before performing the derivation two other variables need to be defined, the distance vector (eq. (2.27)) and the magnitude of the distance (eq. (2.28)):

$$\boldsymbol{r} = \boldsymbol{p} - \boldsymbol{p_i} \tag{2.27}$$

$$r = \left(\Delta x^2 + \Delta y^2 + \Delta z^2\right)^{1/2} \tag{2.28}$$

The derivation is shown in eq. (2.29). The derivation is shown for one coordinate dimension of the acceleration, a_z. The rest are similar and were omitted for

brevity. Equation (2.29a) shows the application of the partial derivative with respect to the vertical acceleration. The other partial derivatives resulting from the gradient operator were left out.

$$a_z = -\frac{\partial}{\partial z}\sum_{i=1}^{n}\frac{-Gm_i}{r} \tag{2.29a}$$

$$= -\frac{\partial}{\partial z}\sum_{i=1}^{n}\frac{-Gm_i}{\left(\Delta x^2+\Delta y^2+\Delta z^2\right)^{1/2}} \tag{2.29b}$$

$$= G\sum_{i=1}^{n} m_i \frac{\partial}{\partial z}\left(\Delta x^2+\Delta y^2+\Delta z^2\right)^{-1/2} \tag{2.29c}$$

$$= G\sum_{i=1}^{n} m_i \left(-\frac{1}{2}\left(\Delta x^2+\Delta y^2+\Delta z^2\right)^{-3/2}\right)\frac{\partial}{\partial z}\left(\Delta x^2+\Delta y^2+\Delta z^2\right) \tag{2.29d}$$

$$= G\sum_{i=1}^{n} m_i \left(-\frac{1}{2}\left(\Delta x^2+\Delta y^2+\Delta z^2\right)^{-3/2}\right)2\Delta z \tag{2.29e}$$

$$a_z = -G\sum_{i=1}^{n}\frac{m_i\Delta z}{r^3} \tag{2.29f}$$

Using eq. (2.28) the distance r can be replaced by its expanded form. A portion of that expanded form does vary with respect to z. However, the summation, the gravitational constant G, and the i^{th}mass, m_i, can be moved outside of the partial derivative as they do not vary with respect to z and can be treated as constants yielding eq. (2.29c). Use the chain rule to evaluate the partial derivative of the power term yielding eq. (2.29d). The second part of the chain rule reduces the final term (shown by eq. (2.29e)). Finally, what is left is the equation that gives the acceleration in the z direction based on the contribution of a collection of finite mass elements.

The derivation shows only the z coordinate dimension, the others are similar. The complete equation is shown in eq. (2.30).

$$\boldsymbol{a} = -G\sum_{i=1}^{n}\frac{m_i\boldsymbol{r}}{r^3} \tag{2.30a}$$

$$= -G\sum_{i=1}^{n}\frac{m_i\hat{\boldsymbol{r}}}{r^2} \tag{2.30b}$$

It is expected that the acceleration be proportional to the inverse square of the distance. Upon closer inspection, the distance r can be factored out using the property, $\boldsymbol{r} = r\hat{\boldsymbol{r}}$. Equation (2.30a) reduces to eq. (2.30b) to get the expected result. The acceleration due to gravity is proportional to the inverse of the square of the distance to the mass causing the acceleration. This provides confidence in the solution.

Additionally, a sanity check can be performed using dimensionless analysis to ensure units on one side of the equation are equivalent to the units on the other side of the equation. This process is shown in eq. (2.31),

$$G\left[\frac{Nm^2}{kg^2}\right], m_i\,[kg], \boldsymbol{r}\,[m], r\,[m] \tag{2.31a}$$

$$\frac{-Gm_i\boldsymbol{r}}{r^3} \tag{2.31b}$$

$$\rightarrow \left[\frac{Nm^2}{kg^2}\right][kg]\,[m]\left[\frac{1}{m^3}\right] \tag{2.31c}$$

$$\rightarrow \left[\frac{kg\cdot m\cdot m^2}{s^2\cdot kg^2}\right][kg]\,[m]\left[\frac{1}{m^3}\right] \tag{2.31d}$$

$$\rightarrow \left[\frac{m}{s^2}\right] \tag{2.31e}$$

where N is Newtons, m is meters, kg is kilograms, and s is seconds. Using the main equation eq. (2.30) and replacing the main variables with their respective units, and removing the summation (which has no effect on the units), yields eq. (2.31c). Then using the definition of a Newton, $[N] \rightarrow [kg\cdot m\cdot s^{-2}]$, one is left with eq. (2.31d). Reducing eq. (2.31d), the final units are the expected units of acceleration.

Equation (2.30) is almost the equation that would permit the derivation of the gravity gradient from the terrain using the correlation between the terrain and the gravity gradient. The i^{th}mass can be replaced by the density of the i^{th}mass element and its volume. The result is the final equation eq. (2.32), which allows the computation of the gravity acceleration due to a distribution of finite volumes that each have a distinct density and location.

$$a_z = -G\sum_{i=1}^{n}\frac{\rho_i\delta x_i\delta y_i\delta z_i\Delta z_i}{r^3} \tag{2.32}$$

2.13 Motivation

The work in this thesis aims to build on work by Pasnani and Seto in [28]. The work uses a particle filter to perform GAN using the global gravity anomaly database from the SCRIPPS institute. This database is limited to a resolution of 1 nm (nautical miles). By using the correlation between the terrain and the gravity anomaly the data within GEBCO can be exploited. The global GEBCO database has a resolution of approximately 400 m, about twice higher resolution than the gravity anomaly database. Since both methods use GAN, a particle filter, and similar sensors, future work fusing these two methods to improve the localization accuracy further is possible.

The review of the literature concludes with the choice of GAN using the gravity gradiometer as the sensor of choice. Bathymetry aided navigation requires the use of active sensors, and therefore does not meet the passivity requirement. The magnetic field has significant temporal variation [58], is not persistent, and requires complex filtering to remove the vehicle's affect on the magnetic field[63]. This makes magnetic aided navigation less attractive than GAN. The gravity field is persistent [44], and effectively unjammable [43].

A gravity gradiometer is desirable for GAN because the measurements are unaffected by the vehicle induced acceleration, thermal drift, and the Eötvös effect [68]. Using the gravity gradient also allows use of the correlation between bathymetry and the local gravity anomaly [43], enabling use of bathymetry for GAN.

Finally, a method for low-risk testing of the TAN algorithm was required. The AUV navigation testbed (ANT) requires an AUV model (provided by the UUV simulator [33], a physics-based simulator (Gazebo simulator), a library to manage the distributed robotics system (Robot Operating System (ROS)), an INS model (implemented in this thesis), and a realistic way to model environmental uncertainty (implemented in this thesis). Vehicle integration requires DCAF an autonomy framework developed for the IVER 3 (developed collaboratively during the work in this

thesis). The next section discusses the thesis methodology.

Chapter 3

Methodology

As previously discussed, GAN was chosen as the most promising method of TAN. GAN benefits from being geophysically persistent, temporally stable, passive, and having widely available maps. To support low-risk algorithm verification and future in-water tests various software tools where developed, independently, and collaboratively. This section covers the decision-making methods leading to this thesis.

The overall flow related to the work in this thesis is shown in a conceptual flow diagram in Figs. 3.1 to 3.4. Red represents problems or challenges that lead to green solutions which may lead to more challenges, and so on. The diagrams are brief and show a summary of the thought process that lead to the tools, and solutions developed in this thesis.

To start, the challenge of the thesis was to explore passive AUV navigation which is resistant to interference, passive, and uses readily available maps. Pure inertial navigation fulfills this requirement, however, its error growth is unbounded with time. The rate of the error growth can be slowed by aiding the pure inertial navigation with another measurement / sensor. Of the solutions explored in the literature review, TAN (specifically GAN) showed the most promise to aid pure inertial navigation. TAN could provide position references (landmarks) through measurements of features in Earth's magnetic field and gradient; seabed bathymetry; gravity anomaly, and gravity gradient. If such measurements are compared with *a priori* georeferenced maps then this may be a solution for long range underwater localization.

TAN approaches that use georeferenced bathymetric features employ acoustic sonars to sense the bathymetry. Therefore, this does not meet the requirement for a passive navigation system. This eliminates the left side of the diagram in Fig. 3.1. Magnetic based localization was not explored due to its semi-dynamic nature. Magnetic fields are in constant flux between the north and south poles switching (albeit over long times) so the maps would have to be updated with measurements, frequently. This eliminates the right branch in Fig. 3.1. Finally, GAN was determined to satisfy most of the initial requirements for passive, stable, and interference resistant underwater navigation.

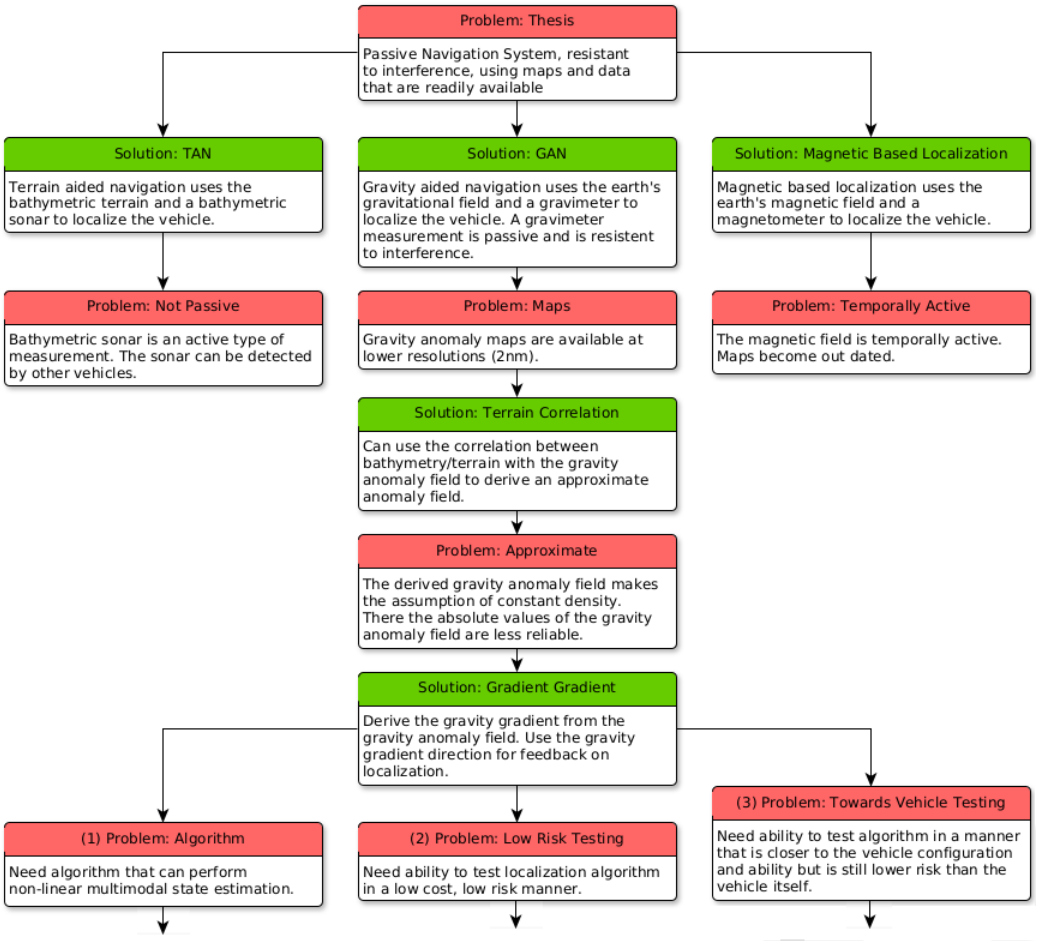

Figure (3.1) Highest level decision tree that maps the methodology for the thesis direction. This starts with the original problem, and includes decisions until the three branches that lead to Figs. 3.2 to 3.4. This includes a brief overview that justifies why gravity-aided navigation was chosen and leads into the main categories of the thesis.

GAN is passive because gravimeters and gravity gradiometers do not emit detectable energy into their environment to make their measurements. They are resistant to interference as changing the gravitational field notably would require large unrealistic mass changes. Gravity anomaly maps are available at a resolution of $1\,nm$ [93]. This is sufficient for long range navigation, however, it does not provide a high-fidelity environmental model to run simulations to develop navigation methodologies. The correlation between the terrain and gravity anomaly could be used to calculate the approximate gravity anomaly. This bases the gravity map on bathymetry maps which are readily available and at higher resolutions than $1\,nm$. Bathymetry map resolutions can range from 500 m globally [29] to 2 m, or better, for smaller geographical regions. The next issue to address is how to access readily available density information on the Earth's crust.

If the gravity gradient is used for navigation, then the local gravity anomaly is sufficient [43, 83], due to the gravity gradient (the first derivative of the gravity anomaly) favouring shallower, local terrain variations [94, 95]. Additionally, the gravity gradiometer is immune to vehicle-induced acceleration, thermal drift, and the Eötvös effect [68]. When the gravity gradient is used for navigation the absolute value of the gravity anomaly is of less consequence than the local variation (the gradient) [94]. This means the correlation between the terrain and the gravity anomaly can be used to calculate the local gravity anomaly [91] and then the gravity gradient is calculated. This gravity gradient map can be used for navigation [82], and the gravity gradient can be calculated to resolutions high enough to provide an environment model for a simulator. This leads to the next set of problems that each branch to their own decision tree as a continuation Figs. 3.2 to 3.4. The three challenges faced are: the navigation algorithm to use, how to perform low risk testing, and how to move toward vehicle testing (higher risk and more valuable testing).

Fig. 3.2 explores the choice of algorithm. GAN is inherently non-linear [6], and often multi-modal [5, p.30] (i.e. more than one hypothesis for the vehicle location). The algorithm must handle a high degree of non-linearity, as well as maintain multiple main hypotheses of the AUV position. The gravity gradient may not be distinctively

unique within the area so the algorithm must be able to address this. Additionally, there can be portions along an AUV's mission that have less variation in the anomaly field so the algorithm must address this as well. Historically, this motivated the TERCOM algorithm [22] and later particle filter based algorithms [45, 26, 27]. TERCOM had the advantage of very low computational requirements. However, due to its simplicity it often diverged which motivated particle filter-based algorithms (discussed further in section 4.4.1). The particle filter-based algorithm can represent an arbitrary state estimation probability (likelihood) distribution of the AUV position. The propagation of this distribution can be non-linear and multi-modal which fulfills the requirements for the localization problem. The choice of particle filter

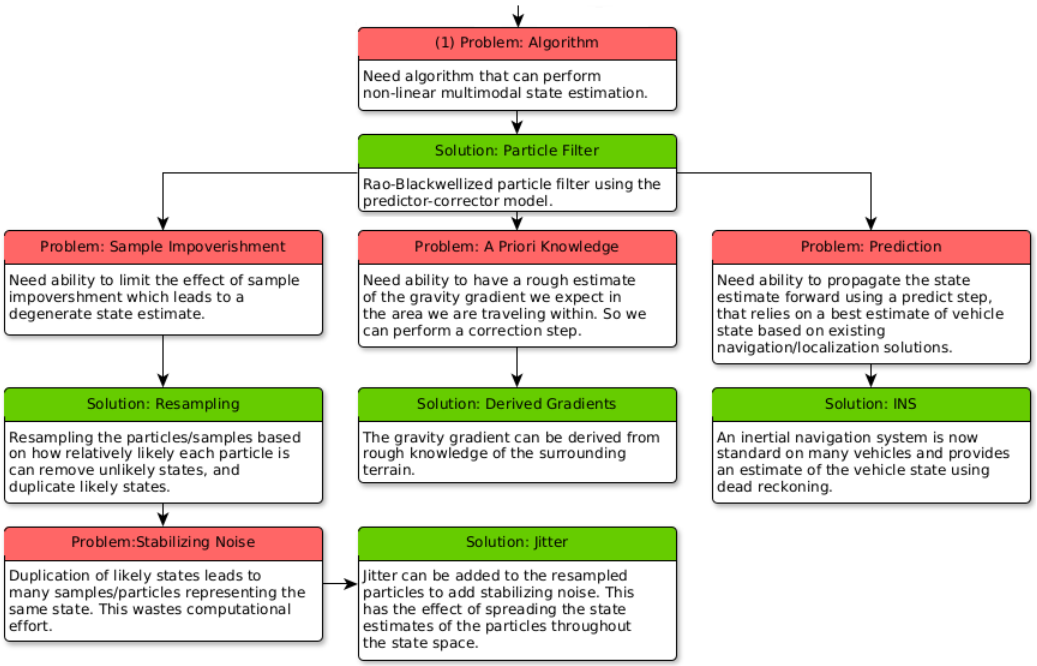

Figure (3.2) Continuation of decision branch (1) of the main decision tree in Fig. 3.1. This decision branch focuses on the state estimation algorithm used for vehicle localization using the gravity gradient field.

based localization led to three challenges, sample impoverishment, *a priori* information requirement, and how to perform the prediction step to propagate the particles.

Firstly, sampling impoverishment occurs when the probability of many particles representing state (location) estimate hypotheses is very small [96]. This wastes

computation power and is mitigated through resampling [96]. This process resamples particles, with the probability of choosing a particle proportional to its weight. The same particle can be chosen multiple times. This is discussed in more detail in section 4.4.1. This effectively multiplies particles with larger weight and favours the elimination of lower weighted (less likely hypotheses) ones. This leads next to the challenge of particle duplication. After resampling, higher weighted particles are multiplied. However, this leads to particles that share similar hypotheses, effectively wasting computation resources again. This is solved with stabilizing noise, or roughening [96], which injects sufficient noise to spread the duplicated particle out in the AUV's position state space without spreading them so much that the quality of the overall state estimate is reduced. This is discussed further in section 4.4.1.

Secondly, is the requirement for *a priori* information for the particle filter weighting or the correction step. The particle filter requires an estimate of the gravity gradient to weight the particles. The weight is calculated according to how likely the gradiometer measurement is, based on the particle's position estimate. This can be provided by using the derived gravity gradient map. Discussed in more detail in section 4.1.5).

Finally, the particle filter must propagate the particles forward, also referred to as the prediction step in state estimation. The INS state estimate can be used as the input to the particle filter. The INS, standard on many AUVs, is the basis of many navigation solutions. After a search, it does not appear that there are any readily available, high-fidelity INS model implementations that can be re-purposed for use in the Gazebo simulation, therefore one was developed by the author. This along with the implementation of the DVL model is discussed in more detail in section 4.1.1.

Next, the challenges and solutions created in the pursuit of low-risk testing is discussed in Fig. 3.3. Physics-based simulation are used for low-risk testing. This is mathematical modelling to approximate reality. The better the model the less difference there is between reality and the more appropriate it is to perform low-risk tests before moving to experiments. Determining which physics simulator to use was based on features, availability, open-source, and industrial readiness. Often industrial

readiness is in conflict with open-source readily available tools, however, that has slowly changed over time. Some of the most widely relied on technology of our era are open-source projects [97, 98]. Open-source allows companies, organizations, and individuals to collaborate on projects which they collectively own and benefit from. ROS is a development within the open-source community originally started by Willow Garage, and now the leading open-source robotics tool chain supporting numerous packages [99] (discussed more in section 3.1). ROS 2 robotics middleware made the switch from its own serialization and communications layer implementation to data distribution service (DDS) [100].

Data distribution service (DDS) is a data connectivity standard middleware protocol and API. It is the standard distributed systems middleware for the data sharing layer [101]. DDS is used across industries to manage large integrated distributed systems, e.g. Nav Canada uses DDS to run real-time traffic management for 3.3 million flights over 18 million square kilometres, National Aeronautics and Space Administration (NASA) uses DDS for their launch control system, and Raytheon uses DDS for interoperability between mission-critical military defence systems [101]. ROS has begun to merge the robotics research and industrial communities [102]. This is beneficial for the industry, and for the research community [102]. For this reason ROS and its complementary robotics simulator Gazebo were chosen. Designing a vehicle dynamics model, and vehicle controller were beyond the scope of this thesis; therefore, an openly available ROS and Gazebo simulator package UUV simulator was used [33].

The UUV simulator package provides the vehicle 3D model, vehicle dynamics model, and various vehicle controllers [33]. The torpedo-shaped style AUV model that the UUV simulator provides is a model for the ECA A9 [33]. The ECA A9 is the vehicle that most closely represents the vehicle kinematics and dynamics of the IVER 3; the most likely vehicle that the thesis work will use for future in-water trials.

Both the Gazebo and UUV simulator do not have an implementation of an AUV's navigation suite's INS. The INS provides dead-reckoning capabilities and is the basis of other navigation and localization algorithms [32, 41, 21]. This missing feature was addressed by implementing an EKF-based INS model, discussed further in section 4.1.1. The Gazebo simulator also does not have a gravimeter model (not

surprisingly) within in its simulated sensor suite.

A gravity gradiometer sensor model was implemented by deriving the gravity anomaly and from that the gravity gradient (discussed in section 4.1.6). The gravity gradiometer uses the ground truth of the AUV position and the gravity gradient map as well as additive Gaussian noise to model a measurement. The gravity gradiometer requires a detailed gravity anomaly map which was provided by the derivation of local gravity anomaly maps using the correlation between the terrain the gravity anomaly [82]. Density information (optional [82]) and bathymetry are used to derive the approximate gravity anomaly.

Density information can be generated using Perlin noise [42] (discussed in section 4.1.3). This enables full control over the amount of variation, and level of detail. This density information is included in the creation of the simulation environment, but not in the derivation of the *a priori* gravity gradient map the particle filter uses. This adds a realistic source of uncertainty. Bathymetry is readily available from the global GEBCO database [29]. The resolution is approximately 400 m. Creating a simulation environment using interpolated measurements from this bathymetry assumes there is no terrain variation between measurements. This is not realistic of course. The terrain can be augmented with specifically designed Perlin noise to add variation and detail between measurements without affecting the integrity of the original measurements [31]. Terrain augmentation is discussed in more detail in section 4.1.4.

Finally, the challenge of future vehicle testing is explored in Fig. 3.4. A hardware-in-the-loop (HITL) provides a low-risk test method that allows hardware integration and testing before integrating the algorithm on the actual vehicle. The ISL has two IVER 3 HITL emulators that can be used for this. However, the HITL does not have any high-level autonomy interface. Its existing interface requires the entire mission to be planned before launching the AUV. Additionally, there is no ROS interface. The vehicle does support a serial interface that provides waypoint based control capabilities. To address these deficits the ISL was sub-contracted to build the DRDC collaborative autonomous framework (DCAF). DCAF provides adaptive mission-planning capabilities, communications, survivability features, and obstacle

avoidance. The author directly developed the arbiter, the process that manages individual requests for vehicle actions from different sources within DCAF, and sends the highest priority action to the vehicle interface. It supports interrupts and resuming of previously interrupted actions. The author also directly developed the bounce survivability feature. The bounce behaviour adds a virtual boundary that the vehicle must stay within during the execution of its mission. These contributions to DCAF are discussed further in section 4.5.

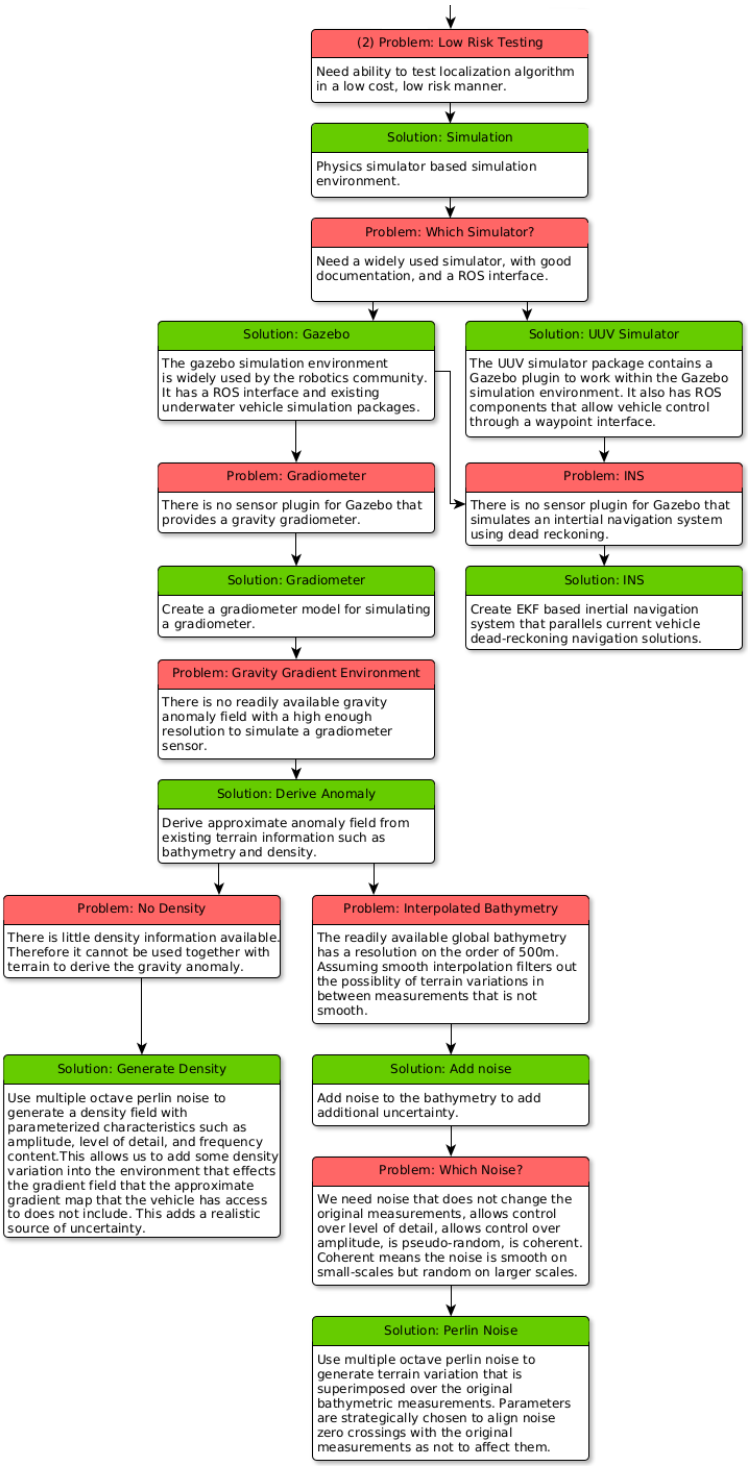

Figure (3.3) Continuation of decision branch (2) of the main decision tree in Fig. 3.1. This decision branch focuses on the computer simulations of the environment and peripherals needed to support the testing of the vehicle state estimation algorithm.

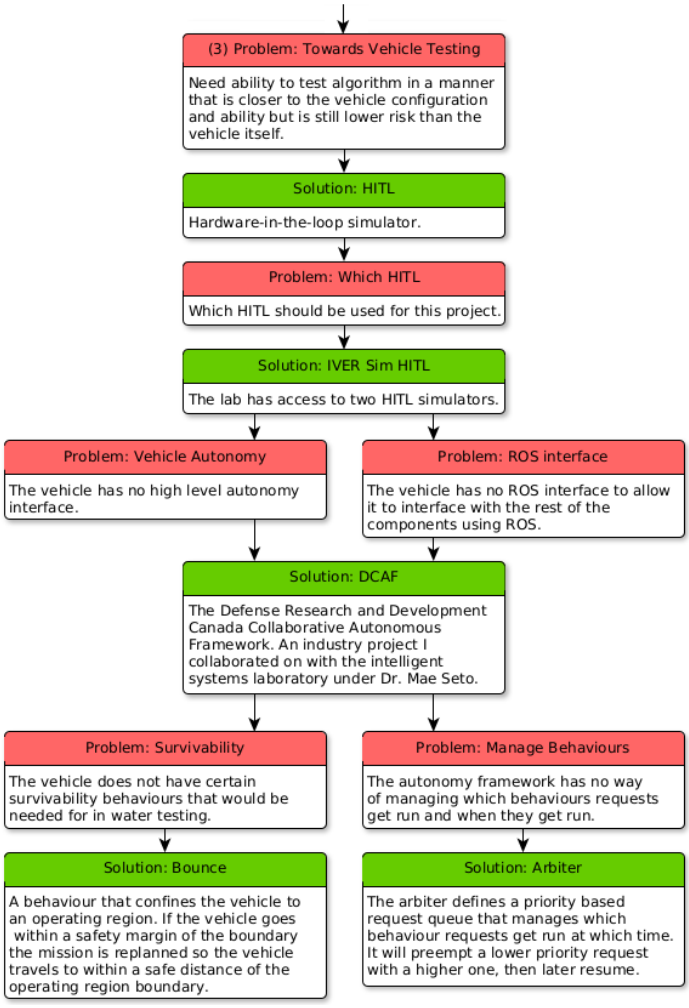

Figure (3.4) Continuation of branch (3) of the main decision tree in Fig. 3.4. This decision branch focuses on the beginnings of future work to test the algorithm in the hardware-in-the-loop simulator. This portion of the project was in collaboration with others in the Intelligent Systems Laboratory. Therefore, only the portions of DCAF directly developed in the thesis are indicated.

3.1 ROS and Gazebo

Over the past decade there has been a movement within the robotics research community towards open-source readily available tooling [99]. This follows the trend towards open-source in the parent field of computer science [97, 98, 103]. Open-source facilitates collaboration on tools and projects that are used by, but too resource intensive (not to mention redundant), for everyone to re-develop. This is a welcome development as there is now less shortage of available tools for the robotics research community [99, 104]. Additionally, well-led community driven projects are often better documented, supported and widely used. When large groups of students, researchers, and software engineers know a specific tool, they will prefer to use and improve that tool. At the same time widely used tools within an industry attracts the creation of course material, and courses for learning the tool in the first place. Together with community projects being accessible to all this is a reinforcing cycle that is a welcome addition to the robotics research community.

The use of common tools provides an incentive for researchers to share their tools with the community once created or improved. This has been done with parts of the thesis work already [31], and other parts will follow. To make the work sharable the choice was made to implement much of the work using ROS [99]. The ROS is a set of software libraries and tools that helps with developing robotics software [99]. There are large libraries of well vetted robotics algorithms, simulators, and middleware [105].

The ROS middleware provides the ability to create decoupled processes that interact through public interfaces. A node is a process in ROS (not always true, but good enough for understanding work within this thesis). The public interfaces can either be in the form of publish-subscribe (pub-sub) messages or service-client services. Messages are passed over topics which are given a unique string identifier domain-wide within ROS.

The pub-sub relationship is the simplest of the two. One or more publishers can publish messages to one or more subscribers on a unique topic. The publishers do not have to know about the subscribers, and vice versa. The publishers and subscribers are often in separate nodes but do not have to be. This relationship

allows the maximum decoupling as the publisher does not expect a response after publishing a message and the subscriber does not need to provide a response to receive a message (i.e. no hand-shaking is required). The best real-world analogy is a newspaper subscription. This relationship is best for nodes that need input from other nodes or need to provide information to other nodes within the system. For example, an altimeter would publish its measurements on a topic for other nodes to consume.

The server-client relationship has a server and a client. Again, these are often in separate processes. The interaction between server and client is captured in a public interface called a service. The service defines a request and a response message. A client will send a request message to the server and the server will respond with a response message. This relationship is best for nodes that provide a service for other nodes such as interpolation, geographic projection, etc.

ROS was chosen because it has a large community and ecosystem related to robotics. Distributed concurrency and performing tasks at the same time in different processes with interactions facilitated by messaging, is well suited for the thesis problem. To test the terrain-aided navigation algorithm one needs to run the simulation, a gradiometer model, projection algorithms, the particle filter, the inertial navigation system, the vehicle model, and the vehicle controller concurrently. ROS was developed with support for a robotics physics-based simulator called Gazebo. Additionally, a ROS package called UUV simulator, developed by Manhães et al., already existed [33]. This package provides a AUV vehicle dynamics model, as well as a vehicle waypoint controller; creating these was outside the scope of this thesis.

The waypoint controller, part of the UUV simulator package [33], provides a high-level service go_to that accepts waypoint messages. The waypoint message includes control over the AUV goal position, goal depth, vehicle speed, and acceptance radius of the waypoint. This made it possible to write high-level testing code for the algorithm without re-implementing much of the lower-level functionality outside the thesis' scope (e.g. vehicle control, vehicle dynamics, and physics simulator).

All these tools used, together with the tools the author developed, are referred to

as the AUV navigation testbed (ANT). A testbed to enable and make AUV navigation research faster.

3.2 AUV Navigation Testbed

There are currently few tools directly for AUV terrain-aided navigation. In the course of the thesis work, several tools were used together, others re-purposed, and others developed. The portion of the tools developed for the thesis, plus the ones re-purposed, are called the ANT. Specifically, the ANT is the combined use of the ROS, the UUV simulator, the INS, the gravity gradiometer model, implementation of gravity anomaly derivation, the particle filter algorithm, and the ANT CLI. The ANT CLI provides much of the functionality as a command line interface. The capabilities of the ANT CLI have evolved to support the work in this thesis.

The ANT CLI was open-sourced as part of the second conference paper [31] published during this thesis work. The ANT is a portion of the thesis work contributions, excluding the gravity gradient localization algorithm, that works together with existing open-source readily available robotics tooling. The capabilities ANT CLI is discussed in detail in section 4.2.

The ANT was developed to support low-risk testing of AUV navigation algorithms. Its value is demonstrated during testing of the TAN algorithm developed within this thesis.

3.3 Terrain Aided Navigation (TAN) Algorithm

The TAN algorithm is an implementation of GAN. It uses the INS as input and the gravity gradient direction as position feedback. It builds on work by Pasnani and Seto [28]. Pasnani uses the gravity anomaly directly to perform GAN, the TAN algorithm uses the correlation of the gravity anomaly with the terrain to supplement lower resolution gravity anomaly maps with higher resolution bathymetry. This is demonstrated in section 5.6.

Fig. 3.5 is a summary figure of the source of uncertainty that affect the performance of the TAN algorithm. There are three main sources of uncertainty, the

unknown environmental density, the additional terrain variation or noise in between the bathymetry measurements, and the noise from the gravity gradiometer measurements. Section 5.6 discusses the results of the parameter study to explore the effects of environmental and sensor uncertainty on the performance of the TAN algorithm. There are three parameter studies that explore each of the sources of uncertainty and their effect on the particle filter algorithm's state estimate error. Additionally, there are trials of the INS model to verify that the INS replicates available INS navigation properties such as navigational uncertainty and error growth. Finally, work on the DCAF project is discussed and proposed as the next step in testing the TAN algorithm.

3.3.1 Trials Plan

The trials plan includes a verification of the performance characteristic of the INS model, DVL model, and three studies to explore the effect of environmental and sensor uncertainties on the performance of the TAN algorithm.

The first study, an AUV search mission, aims to verify that the INS model and DVL model replicates current state-of-the-art INSs like the Kearfott INS [8]. The AUV will run a survey mission within a simulated Beford Basin (Halifax, Nova Scotia) environment. The resulting uncertainty growth and state estimate error will be compared to existing INS performance. The implementation of this study is introduced in section 4.1.1, and the results and implications discussed in section 5.1.

Next, two parameter studies were performed to test the TAN algorithms' sensitivity to uncertainty within the environment (two of the three sources of uncertainty shown in Fig. 3.5). These parameter studies were completed and the data collected using the ROS parameter study (RPS) CLI (discussed in section 4.3). The parameter studies were performed with the ANT using ROS, Gazebo, and UUV simulator.

The first TAN algorithm study explores the effect of gravity gradiometer sensor noise on the quality of the TAN position state estimate. The study involves 80 runs of an 500 km AUV transit across an area of the ocean off the south-east coast of Nova Scotia. The gradiometer noise is varied systematically from 0 rad to 0.7 rad in

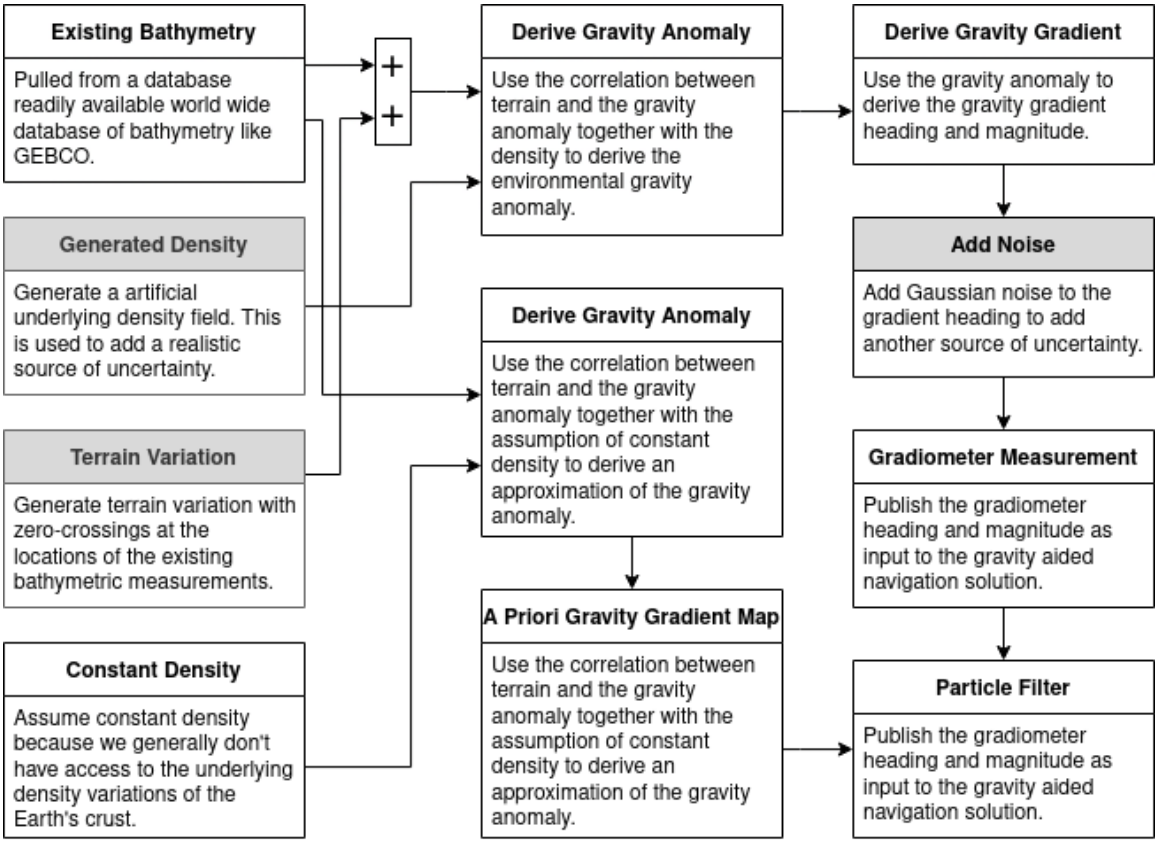

Figure (3.5) This figure is an overview of sources of uncertainty within the gravity aided localization algorithm. The main sources of uncertainty are highlighted with a grey header. These sources of uncertainty aim to make the simulation of the gravity aided localization algorithm closer to reality. The artificial density field does not actually need to be accurate as it is providing a source of uncertainty between what the *a priori* gravity gradient map contains versus what the environmental gravity gradient actually contains. The environmental gravity gradient is derived using an approximation of the gravity anomaly. The gravity anomaly is derived using augmented bathymetry and the the artificial density. The augmented bathymetry is the summation of the existing bathymetry field and artificial terrain variation in between measurements. The artificial terrain variation that is not included in the derivation of the *a priori* gravity gradient map adds an additional source of uncertainty to the input of the particle filter based gravity aided localization algorithm.

steps of 0.1 rad. This yields 8 sessions each containing 10 runs. The starting particle positions, starting INS position error, and the starting INS heading error are all pseudo-randomly initialized at the beginning of each run. The 10 runs then represent the expected variation in algorithmic performance subject to pressures from the gravity gradiometer sensor noise. The results of this study can be found in section 5.6.1.

The second TAN algorithm study explores the effect of the constant density assumption on the quality of the TAN state estimate. The study contains 50 runs of an 500 km AUV transit across an area of the ocean off the south-east coast of Nova Scotia. The density variation amplitude of the terrain is varied logarithmically from $50\,\mathrm{kg/m^3}$ to $800\,\mathrm{kg/m^3}$ in 5 steps. This yields 5 sessions with 10 runs each. The starting particle positions, starting INS position error, and starting INS heading error are pseudo-randomly initialized at the beginning of each run. The uncertainty spawns from the difference between the constant density assumption made to derive the *a priori* gravity gradient maps the AUV has access to and actual environment within the ANT which includes the effect of the density variation in its creation. Each session of 10 runs represent the resulting variation in TAN algorithmic performance when subjected to pressures due to the assumption of constant density when using the correlation between the gravity anomaly and the terrain to derive gravity gradient maps for use in navigation. The results of this study can be found in section 5.6.2.

After testing the TAN algorithm in the ANT the next steps are vehicle integration and in-water testing. The collaborative project DCAF, for which the author contributed to, was developed in support of future in-water testing of ROS based algorithms.

3.4 DRDC DCAF

Future work for the gravity gradient aided navigation leads to HITL and in-water testing. In preparation, the author collaborated on a DRDC project that is contracted out to the ISL. The aim of this project is to create a ROS based autonomous framework to enable backseat autonomy on the IVER 3 AUV. The work on DCAF includes, waypoint planning, survivability features, obstacle avoidance, mission control, acoustic communication, and vehicle interface. This is an ongoing collaborative

project. Each portion has a main developer and the support of others in the ISL when needed. The author's developed algorithms had the opportunity for in-water testing on the IVER3 AUV. However, delays due to the global pandemic have not made this possible.

The author was the main developer for the arbiter, the bounce survivability feature, and the technical documentation. The arbiter supports vehicle action interface. It manages separate vehicle action requests from other portions of the DCAF framework or the topside operator's computer and sends the highest priority actions on through the vehicle interface to the IVER 3 vehicle. The bounce behaviour supports a virtual boundary for the vehicle in which its entire mission must take place. The vehicle is blocked from moving outside this area within the scope of the mission. The implementation of these features as part of DCAF is discussed further in section 4.5.

The methodology section summaries the flow of the work within this thesis. It uses the challenges encountered as well as related research to justify methods used to address these challenges. ROS, the Gazebo simulator, and the UUV simulator package are introduced as methods to develop TAN algorithms. The choice of these tools is justified within the context of robotics research. These tools together with others developed during this thesis are presented as the method to perform low-risk testing of the TAN. This collection of tools is referred to as the ANT. Next, the methods used to verify the INS model and test the TAN algorithm within the ANT are introduced. Finally, development on the DCAF project is discussed as a method to move to vehicle integration and future in-water testing. The next section, the experiment section, will cover the implementation details of these methods.

Chapter 4

Experiment

This experiment section covers the components that make up the AUV navigation testbed (ANT), a brief over view of the peripheral functionality implemented by the ANT CLI. Then, details of the particle filter based TAN algorithm are covered. This is followed by the experiment section which covers the details of the DCAF project. Finally, the ROS parameter study CLI, used for all the parameter studies in this thesis, is discussed.

4.1 AUV Navigation Testbed

The ANT is a collection of tools to support AUV navigation. These tools have been developed or repurposed for the development of the TAN algorithm.

4.1.1 Inertial Navigation Modelling

An INS integrated on an AUV uses inertial and rotation sensors to estimate the vehicle's position from dead-reckoning. Dead-reckoning is a form of path integration, using the vehicle's velocity and elapsed time, to estimate the new or current vehicle position [32].

A basic INS uses the current orientation estimate and transforms them into the world frame where they are fused with the current estimate for a better position estimate. However, unless the rotation operation is part of the EKF, the effect of using an orientation with uncertainty on the state estimate is lost. Therefore, the rotation operation is included within the EKF, this make the state space representation and Jacobians more complex, but it provides the benefit of providing a better measure of the uncertainty of the state estimate.

In the context of AUV navigation, the INS is largely for dead-reckoning within the horizontal plane. An inclinometer provides an error bound for the vehicle's inclination, and the pressure sensor for the vehicle depth [3, 106]. When using the INS within this thesis an inclinometer is not used. However, depth is assumed to be known to a high accuracy, effectively turning a 3D localization problem into a 2D one in the horizontal plane [106]. The implemented INS uses the EKF framework for state estimation. The INS is aided by a DVL, giving a more direct measure of the speed-over-ground.

Prediction

Inertial measurements are used in place of a dynamic model so the integration of the inertial measurements yields an estimate of the AUV position, velocity, and attitude. The decision was made to do the integration within the EKF state transition function rather than outside of the EKF. The integration could be performed outside the EKF so more accurate numerical integration methods, like Runge-Kutta, could be applied. It would also allow the EKF to run at a slower prediction rate than the incoming measurements from the IMU. These additional features were deemed unnecessary to demonstrate the benefits of an external position update to bound the error growth of dead-reckoning. Therefore, the integration is performed within the EKF prediction step, requiring the EKF to run at 50 Hz.

The vehicle position, velocity, and attitude are the state variables for the EKF state estimation. The state vector is shown in eq. (4.1),

$$\boldsymbol{X} = \begin{bmatrix} \boldsymbol{x} \\ \dot{\boldsymbol{x}} \\ \boldsymbol{q} \end{bmatrix} \tag{4.1}$$

where $\boldsymbol{x}$ is the current position in world frame, $\dot{\boldsymbol{x}}$ is the current velocity in the world frame, and $\boldsymbol{q}$ is the unit quaternion representing the attitude of the body-centered vehicle frame relative to the world frame.

The discretized kinematic equation for position is shown in eq. (4.2),

$$\boldsymbol{x}_{k+1} = \frac{1}{2}\boldsymbol{a}_k T^2 + \dot{\boldsymbol{x}}_k T + \boldsymbol{x}_k \tag{4.2}$$

where T is the time step, $\boldsymbol{a}_k$ is the acceleration during the time step, $\dot{\boldsymbol{x}}$ is the velocity during the time step, and $\boldsymbol{x}_k$ is the previous position. Equation (4.2) assumes constant acceleration within a time step. This assumption is true for small time steps. For the 1/50 seconds time step used this is a good assumption because the vehicle dynamics / responses occur on a much longer time scale. Similar assumptions hold for the

discretized velocity, shown in eq. (4.3),

$$\dot{\boldsymbol{x}}_{k+1} = \boldsymbol{a}_k T + \dot{\boldsymbol{x}}_k. \tag{4.3}$$

The quaternion rate in eq. (2.21) can be discretized into the form of eq. (4.4) [85, p. 489],

$$\boldsymbol{q}_{k+1} = \frac{1}{2}\boldsymbol{\omega}_k \otimes \boldsymbol{q}_k \tag{4.4}$$

where $\boldsymbol{\omega}_k$ is the body-frame angular velocities relative to the world inertial frame (the gyroscope measurement). Expanded using the notations described in eq. (2.16) and the definition of the quaternion product (eq. (2.17)), the discretized equation for the quaternion can be derived as,

$$\begin{aligned} \boldsymbol{\epsilon}_{k+1} &= \frac{1}{2}T(\boldsymbol{\eta}_k\boldsymbol{\omega}_k + \boldsymbol{\epsilon}_k \times \boldsymbol{\omega}_k) + \boldsymbol{\epsilon}_k \\ \eta_{k+1} &= \frac{1}{2}T(-\boldsymbol{\epsilon}_k^T\boldsymbol{\omega}_k) + \eta_k. \end{aligned} \tag{4.5}$$

The discrete prediction equations define the non-linear prediction function f,

$$f(\boldsymbol{X}_{k-1}, \boldsymbol{a}_{\mathcal{G}}, \boldsymbol{\omega}_{\mathcal{L}}, T) = \begin{bmatrix} \boldsymbol{A}_{3\times1} \\ \boldsymbol{A}_{3\times1} \\ \boldsymbol{C}_{4\times1} \end{bmatrix} \tag{4.6}$$

where $\boldsymbol{A}_{3\times1}$ is defined by,

$$\boldsymbol{A}_{3\times1} = \begin{bmatrix} \frac{1}{2}a_x T^2 + \dot{x}T + x \\ \frac{1}{2}a_y T^2 + \dot{y}T + y \\ \frac{1}{2}a_z T^2 + \dot{z}T + z \end{bmatrix} \tag{4.7}$$

$\boldsymbol{B}_{3\times1}$ is defined by,

$$\boldsymbol{B}_{3\times1} = \begin{bmatrix} a_x T + \dot{x} \\ a_y T + \dot{y} \\ a_z T + \dot{z} \end{bmatrix} \tag{4.8}$$

and $\boldsymbol{C}_{4\times1}$ is defined by,

$$\boldsymbol{C}_{4\times1} = \begin{bmatrix} \frac{1}{2}\left(\epsilon_y\omega_z - \epsilon_z\omega_y + \eta\omega_x\right)T + \epsilon_x \\ \frac{1}{2}\left(-\epsilon_x\omega_z + \epsilon_z\omega_x + \eta\omega_y\right)T + \epsilon_y \\ \frac{1}{2}\left(\epsilon_x\omega_y - \epsilon_y\omega_x + \eta\omega_z\right)T + \epsilon_z \\ \frac{1}{2}\left(-\epsilon_x\omega_x - \epsilon_y\omega_y - \epsilon_z\omega_z\right)T + \eta \end{bmatrix}. \tag{4.9}$$

Equation (4.6) uses the world frame acceleration which can be calculated using the body frame acceleration, the current attitude estimate, and eq. (2.19) to yield eq. (4.10) and eq. (4.11).

$$\boldsymbol{a}_{\mathcal{G}} = \begin{bmatrix} a_x \\ a_y \\ a_z \end{bmatrix}_{\boldsymbol{a}_{\mathcal{G}}} \tag{4.10}$$

$$\boldsymbol{a}_{\mathcal{G}} = \begin{bmatrix} \left(a_x\left(\epsilon_x^2 - \epsilon_y^2 - \epsilon_z^2 + \eta^2\right) + a_y\left(2\epsilon_x\epsilon_y - 2\epsilon_z\eta\right) + a_z\left(2\epsilon_x\epsilon_z + 2\epsilon_y\eta\right)\right) \\ \left(a_x\left(2\epsilon_x\epsilon_y + 2\epsilon_z\eta\right) + a_y\left(-\epsilon_x^2 + \epsilon_y^2 - \epsilon_z^2 + \eta^2\right) + a_z\left(-2\epsilon_x\eta + 2\epsilon_y\epsilon_z\right)\right) \\ \left(a_x\left(2\epsilon_x\epsilon_z - 2\epsilon_y\eta\right) + a_y\left(2\epsilon_x\eta + 2\epsilon_y\epsilon_z\right) + a_z\left(-\epsilon_x^2 - \epsilon_y^2 + \epsilon_z^2 + \eta^2\right) - g\right) \end{bmatrix}_{\boldsymbol{a}_{\mathcal{L}}} \tag{4.11}$$

Equation (4.6) is used in eq. (2.4) predict the state $\boldsymbol{X}_{k|k-1}$. To calculate the predicted or a priori covariance matrix, eq. (2.5) is used. The transition function Jacobian F is derived using its definition (eq. (2.6)), and the non-linear transition function (eq. (4.6)), yielding eq. (4.12),

$$\boldsymbol{F}_{k-1}(\boldsymbol{X}_{k|k-1}, \boldsymbol{a}_{\mathcal{L}}, \boldsymbol{\omega}_{\mathcal{L}}, T) = \begin{bmatrix} \boldsymbol{A}_{6\times6} & \boldsymbol{B}_{6\times4} \\ \boldsymbol{0}_{4\times6} & \boldsymbol{C}_{4\times4} \end{bmatrix} \tag{4.12}$$

where $\boldsymbol{A}_{6\times6}$ is defined by,

$$\boldsymbol{A}_{6\times6} = \begin{bmatrix} 1 & 0 & 0 & T & 0 & 0 \\ 0 & 1 & 0 & 0 & T & 0 \\ 0 & 0 & 1 & 0 & 0 & T \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{4.13}$$

$\boldsymbol{B}_{6\times4}$ is defined by,

$$\boldsymbol{B}_{6\times4} = \begin{bmatrix} \frac{\partial f_x}{\partial \epsilon_x} & \frac{\partial f_x}{\partial \epsilon_y} & \frac{\partial f_x}{\partial \epsilon_z} & \frac{\partial f_x}{\partial \eta} \\ \frac{\partial f_y}{\partial \epsilon_x} & \frac{\partial f_y}{\partial \epsilon_y} & \frac{\partial f_y}{\partial \epsilon_z} & \frac{\partial f_y}{\partial \eta} \\ \frac{\partial f_z}{\partial \epsilon_x} & \frac{\partial f_z}{\partial \epsilon_y} & \frac{\partial f_z}{\partial \epsilon_z} & \frac{\partial f_z}{\partial \eta} \\ \frac{\partial f_{\dot{x}}}{\partial \epsilon_x} & \frac{\partial f_{\dot{x}}}{\partial \epsilon_y} & \frac{\partial f_{\dot{x}}}{\partial \epsilon_z} & \frac{\partial f_{\dot{x}}}{\partial \eta} \\ \frac{\partial f_{\dot{y}}}{\partial \epsilon_x} & \frac{\partial f_{\dot{y}}}{\partial \epsilon_y} & \frac{\partial f_{\dot{y}}}{\partial \epsilon_z} & \frac{\partial f_{\dot{y}}}{\partial \eta} \\ \frac{\partial f_{\dot{z}}}{\partial \epsilon_x} & \frac{\partial f_{\dot{z}}}{\partial \epsilon_y} & \frac{\partial f_{\dot{z}}}{\partial \epsilon_z} & \frac{\partial f_{\dot{z}}}{\partial \eta} \end{bmatrix} \tag{4.14}$$

and $\boldsymbol{C}_{4\times4}$ is defined by,

$$\boldsymbol{C}_{4\times4} = \begin{bmatrix} 1 & \frac{T\omega_z}{2} & -\frac{T\omega_y}{2} & \frac{T\omega_x}{2} \\ -\frac{T\omega_z}{2} & 1 & \frac{T\omega_x}{2} & \frac{T\omega_y}{2} \\ \frac{T\omega_y}{2} & -\frac{T\omega_x}{2} & 1 & \frac{T\omega_z}{2} \\ -\frac{T\omega_x}{2} & -\frac{T\omega_y}{2} & -\frac{T\omega_z}{2} & 1 \end{bmatrix}. \tag{4.15}$$

$$\begin{aligned} \frac{\partial f_x}{\partial \epsilon_x} &= \frac{1}{2}T^2\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\ \frac{\partial f_x}{\partial \epsilon_y} &= \frac{1}{2}T^2\left(2\epsilon_x a_y - 2\epsilon_y a_x + 2\eta a_z\right) \\ \frac{\partial f_x}{\partial \epsilon_z} &= \frac{1}{2}T^2\left(2\epsilon_x a_z - 2\epsilon_z a_x - 2\eta a_y\right) \\ \frac{\partial f_x}{\partial \eta} &= \frac{1}{2}T^2\left(2\epsilon_y a_z - 2\epsilon_z a_y + 2\eta a_x\right) \end{aligned} \tag{4.16}$$

$$\begin{aligned}
\frac{\partial f_y}{\partial \epsilon_x} &= \frac{1}{2}T^2\left(-2\epsilon_x a_y + 2\epsilon_y a_x - 2\eta a_z\right) \\
\frac{\partial f_y}{\partial \epsilon_y} &= \frac{1}{2}T^2\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\
\frac{\partial f_y}{\partial \epsilon_z} &= \frac{1}{2}T^2\left(2\epsilon_y a_z - 2\epsilon_z a_y + 2\eta a_x\right) \\
\frac{\partial f_y}{\partial \eta} &= \frac{1}{2}T^2\left(-2\epsilon_x a_z + 2\epsilon_z a_x + 2\eta a_y\right)
\end{aligned} \tag{4.17}$$

$$\begin{aligned}
\frac{\partial f_z}{\partial \epsilon_x} &= \frac{1}{2}T^2\left(-2\epsilon_x a_z + 2\epsilon_z a_x + 2\eta a_y\right) \\
\frac{\partial f_z}{\partial \epsilon_y} &= \frac{1}{2}T^2\left(-2\epsilon_y a_z + 2\epsilon_z a_y - 2\eta a_x\right) \\
\frac{\partial f_z}{\partial \epsilon_z} &= \frac{1}{2}T^2\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\
\frac{\partial f_z}{\partial \eta} &= \frac{1}{2}T^2\left(2\epsilon_x a_y - 2\epsilon_y a_x + 2\eta a_z\right)
\end{aligned} \tag{4.18}$$

$$\begin{aligned}
\frac{\partial f_{\dot{x}}}{\partial \epsilon_x} &= T\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\
\frac{\partial f_{\dot{x}}}{\partial \epsilon_y} &= T\left(2\epsilon_x a_y - 2\epsilon_y a_x + 2\eta a_z\right) \\
\frac{\partial f_{\dot{x}}}{\partial \epsilon_z} &= T\left(2\epsilon_x a_z - 2\epsilon_z a_x - 2\eta a_y\right) \\
\frac{\partial f_{\dot{x}}}{\partial \eta} &= T\left(2\epsilon_y a_z - 2\epsilon_z a_y + 2\eta a_x\right)
\end{aligned} \tag{4.19}$$

$$\begin{aligned}
\frac{\partial f_{\dot{y}}}{\partial \epsilon_x} &= T\left(-2\epsilon_x a_y + 2\epsilon_y a_x - 2\eta a_z\right) \\
\frac{\partial f_{\dot{y}}}{\partial \epsilon_y} &= T\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\
\frac{\partial f_{\dot{y}}}{\partial \epsilon_z} &= T\left(2\epsilon_y a_z - 2\epsilon_z a_y + 2\eta a_x\right) \\
\frac{\partial f_{\dot{y}}}{\partial \eta} &= T\left(-2\epsilon_x a_z + 2\epsilon_z a_x + 2\eta a_y\right)
\end{aligned} \tag{4.20}$$

$$\begin{aligned}
\frac{\partial f_{\dot{z}}}{\partial \epsilon_x} &= T\left(-2\epsilon_x a_z + 2\epsilon_z a_x + 2\eta a_y\right) \\
\frac{\partial f_{\dot{z}}}{\partial \epsilon_y} &= T\left(-2\epsilon_y a_z + 2\epsilon_z a_y - 2\eta a_x\right) \\
\frac{\partial f_{\dot{z}}}{\partial \epsilon_z} &= T\left(2\epsilon_x a_x + 2\epsilon_y a_y + 2\epsilon_z a_z\right) \\
\frac{\partial f_{\dot{z}}}{\partial \eta} &= T\left(2\epsilon_x a_y - 2\epsilon_y a_x + 2\eta a_z\right)
\end{aligned} \tag{4.21}$$

where T is the time step, all accelerations $\boldsymbol{a}$ and angular rates $\boldsymbol{\omega}$ are in the local or body frame $\mathcal{L}$, and the state variables are evaluated at time step $k-1$. Additionally, because the covariance matrix of the IMU sensor is in the body frame, the covariance must be rotated into the world frame. This violates the additive noise assumption as the noise in the world frame is no longer additive. Therefore, L, the Jacobian of the process noise covariance Q, is derived using eq. (2.7), yielding eq. (4.22),

$$\boldsymbol{L}_{k-1}(\boldsymbol{X}_{k-1}, T) = \begin{bmatrix} \boldsymbol{A}_{6\,\mathrm{x}\,3} & \boldsymbol{0}_{6\,\mathrm{x}\,3} \\ \boldsymbol{0}_{4\,\mathrm{x}\,3} & \boldsymbol{B}_{4\,\mathrm{x}\,3} \end{bmatrix} \tag{4.22}$$

$$\boldsymbol{A}_{6\,\mathrm{x}\,3} = \begin{bmatrix} \frac{T^2\left(\epsilon_x^2 - \epsilon_y^2 - \epsilon_z^2 + \eta^2\right)}{2} & \frac{T^2(2\epsilon_x\epsilon_y - 2\epsilon_z\eta)}{2} & \frac{T^2(2\epsilon_x\epsilon_z + 2\epsilon_y\eta)}{2} \\ \frac{T^2(2\epsilon_x\epsilon_y + 2\epsilon_z\eta)}{2} & \frac{T^2\left(-\epsilon_x^2 + \epsilon_y^2 - \epsilon_z^2 + \eta^2\right)}{2} & \frac{T^2(-2\epsilon_x\eta + 2\epsilon_y\epsilon_z)}{2} \\ \frac{T^2(2\epsilon_x\epsilon_z - 2\epsilon_y\eta)}{2} & \frac{T^2(2\epsilon_x\eta + 2\epsilon_y\epsilon_z)}{2} & \frac{T^2\left(-\epsilon_x^2 - \epsilon_y^2 + \epsilon_z^2 + \eta^2\right)}{2} \\ T\left(\epsilon_x^2 - \epsilon_y^2 - \epsilon_z^2 + \eta^2\right) & T\left(2\epsilon_x\epsilon_y - 2\epsilon_z\eta\right) & T\left(2\epsilon_x\epsilon_z + 2\epsilon_y\eta\right) \\ T\left(2\epsilon_x\epsilon_y + 2\epsilon_z\eta\right) & T\left(-\epsilon_x^2 + \epsilon_y^2 - \epsilon_z^2 + \eta^2\right) & T\left(-2\epsilon_x\eta + 2\epsilon_y\epsilon_z\right) \\ T\left(2\epsilon_x\epsilon_z - 2\epsilon_y\eta\right) & T\left(2\epsilon_x\eta + 2\epsilon_y\epsilon_z\right) & T\left(-\epsilon_x^2 - \epsilon_y^2 + \epsilon_z^2 + \eta^2\right) \end{bmatrix} \tag{4.23}$$

$$\boldsymbol{B}_{4\,\mathrm{x}\,3} = \begin{bmatrix} \frac{T\eta}{2} & -\frac{T\epsilon_z}{2} & \frac{T\epsilon_y}{2} \\ \frac{T\epsilon_z}{2} & \frac{T\eta}{2} & -\frac{T\epsilon_x}{2} \\ -\frac{T\epsilon_y}{2} & \frac{T\epsilon_x}{2} & \frac{T\eta}{2} \\ -\frac{T\epsilon_x}{2} & -\frac{T\epsilon_y}{2} & -\frac{T\epsilon_z}{2} \end{bmatrix} \tag{4.24}$$

where the state variables are evaluated at time step $k-1$. Finally, the process noise covariance matrix is defined by the variance of the accelerometers and the gyroscopes (eq. (4.25)),

$$\boldsymbol{Q_k}(\boldsymbol{a}_{\mathcal{L}}, \boldsymbol{\omega}_{\mathcal{L}}) = \begin{bmatrix} \sigma_{a_x}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{a_y}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{a_z}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{\omega_x}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\omega_y}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\omega_z}^2 \end{bmatrix} \tag{4.25}$$

where $\sigma_{a_x}^2$ is the accelerometer variance in the body frame x axis, and the $\sigma_{\omega_x}^2$ is the gyroscope variance around the body frame x axis. In this implementation, the assumption is that all accelerometer variances are the same and similarly all gyroscope variances are the same.

The predict step is triggered by a message published by the IMU. This occurs at a rate of 50 Hz. Therefore, the nominal mean time step of the EKF prediction cycle is about 20 ms. In this implementation the message time stamp is used to determine the time step between consecutive messages.

DVL Update

To bound the velocity error growth a direct measurement of the velocity-over-ground is fused in the EKF update step. The measurement equation yields the expected measurement given the current state estimate,

$$\tilde{\boldsymbol{h}}_{dvl}(\boldsymbol{X}_{k|k-1}) = \begin{bmatrix} -2\dot{x}\epsilon_x\epsilon_z - 2\dot{x}\epsilon_y\eta + 2\dot{y}\epsilon_x\eta - 2\dot{y}\epsilon_y\epsilon_z + \dot{z}\epsilon_x^2 + \dot{z}\epsilon_y^2 - \dot{z}\epsilon_z^2 - \dot{z}\eta^2 \\ 2\dot{x}\epsilon_x\epsilon_y - 2\dot{x}\epsilon_z\eta - \dot{y}\epsilon_x^2 + \dot{y}\epsilon_y^2 - \dot{y}\epsilon_z^2 + \dot{y}\eta^2 + 2\dot{z}\epsilon_x\eta + 2\dot{z}\epsilon_y\epsilon_z \\ \dot{x}\epsilon_x^2 - \dot{x}\epsilon_y^2 - \dot{x}\epsilon_z^2 + \dot{x}\eta^2 + 2\dot{y}\epsilon_x\epsilon_y + 2\dot{y}\epsilon_z\eta + 2\dot{z}\epsilon_x\epsilon_z - 2\dot{z}\epsilon_y\eta \end{bmatrix} \tag{4.26}$$

where $\tilde{\boldsymbol{h}}$ is the expected measurement in the DVL reference frame. Equation (4.26) is derived by applying consecutive rotations to the global velocity state estimate. One rotation is from the world frame to IMU/body frame and the other is from the body frame to the INS frame. Equation (2.20) is used to perform the consecutive rotations

to yield eq. (4.27),

$$\begin{bmatrix} \dot{\boldsymbol{x}}_{dvl} \\ 0 \end{bmatrix} = \boldsymbol{q}^*_{dvl/\mathcal{L}} \otimes \left(\boldsymbol{q}^* \otimes \begin{bmatrix} \dot{\boldsymbol{x}}_{\mathcal{G}} \\ 0 \end{bmatrix} \otimes \boldsymbol{q} \right) \otimes \boldsymbol{q}_{dvl/\mathcal{L}} \tag{4.27}$$

where $\dot{\boldsymbol{x}}_{dvl}$ is the expected measurement, $\boldsymbol{q}_{dvl/\mathcal{L}}$ is the orientation of the DVL frame relative to the body frame defined by eq. (4.28), $\boldsymbol{q}$ is the estimated orientation of the body frame relative to the world frame, and $\dot{\boldsymbol{x}}_{\mathcal{G}}$ is the velocity estimate in the world frame. Both $\boldsymbol{q}$ and $\dot{\boldsymbol{x}}_{\mathcal{G}}$ are part of the AUV state.

$$\boldsymbol{q}_{dvl/\mathcal{L}} = \begin{bmatrix} 0 \\ \frac{\sqrt{2}}{2} \\ 0 \\ \frac{\sqrt{2}}{2} \end{bmatrix} \tag{4.28}$$

The Jacobian for the DVL measurement update is defined by eq. (4.29).

$$\boldsymbol{H}_k(\boldsymbol{X}_{k|k-1}) = \begin{bmatrix} 0 & 0 & 0 & \frac{\partial h_{\dot{x}}}{\partial \dot{x}} & \frac{\partial h_{\dot{x}}}{\partial \dot{y}} & \frac{\partial h_{\dot{x}}}{\partial \dot{z}} & \frac{\partial h_{\dot{x}}}{\partial \epsilon_x} & \frac{\partial h_{\dot{x}}}{\partial \epsilon_y} & \frac{\partial h_{\dot{x}}}{\partial \epsilon_z} & \frac{\partial h_{\dot{x}}}{\partial \eta} \\ 0 & 0 & 0 & \frac{\partial h_{\dot{y}}}{\partial \dot{x}} & \frac{\partial h_{\dot{y}}}{\partial \dot{y}} & \frac{\partial h_{\dot{y}}}{\partial \dot{z}} & \frac{\partial h_{\dot{y}}}{\partial \epsilon_x} & \frac{\partial h_{\dot{y}}}{\partial \epsilon_y} & \frac{\partial h_{\dot{y}}}{\partial \epsilon_z} & \frac{\partial h_{\dot{y}}}{\partial \eta} \\ 0 & 0 & 0 & \frac{\partial h_{\dot{z}}}{\partial \dot{x}} & \frac{\partial h_{\dot{z}}}{\partial \dot{y}} & \frac{\partial h_{\dot{z}}}{\partial \dot{z}} & \frac{\partial h_{\dot{z}}}{\partial \epsilon_x} & \frac{\partial h_{\dot{z}}}{\partial \epsilon_y} & \frac{\partial h_{\dot{z}}}{\partial \epsilon_z} & \frac{\partial h_{\dot{z}}}{\partial \eta} \end{bmatrix} \tag{4.29}$$

$$\begin{aligned} \frac{\partial h_{\dot{x}}}{\partial \dot{x}} &= -2\epsilon_x\epsilon_z - 2\epsilon_y\eta \\ \frac{\partial h_{\dot{x}}}{\partial \dot{y}} &= 2\epsilon_x\eta - 2\epsilon_y\epsilon_z \\ \frac{\partial h_{\dot{x}}}{\partial \dot{z}} &= \epsilon_x^2 + \epsilon_y^2 - \epsilon_z^2 - \eta^2 \\ \frac{\partial h_{\dot{x}}}{\partial \epsilon_x} &= -2\dot{x}\epsilon_z + 2\dot{y}\eta + 2\dot{z}\epsilon_x \\ \frac{\partial h_{\dot{x}}}{\partial \epsilon_y} &= -2\dot{x}\eta - 2\dot{y}\epsilon_z + 2\dot{z}\epsilon_y \\ \frac{\partial h_{\dot{x}}}{\partial \epsilon_z} &= -2\dot{x}\epsilon_x - 2\dot{y}\epsilon_y - 2\dot{z}\epsilon_z \\ \frac{\partial h_{\dot{x}}}{\partial \eta} &= -2\dot{x}\epsilon_y + 2\dot{y}\epsilon_x - 2\dot{z}\eta \end{aligned} \tag{4.30}$$

$$
\begin{aligned}
\frac{\partial h_{\dot{y}}}{\partial \dot{x}} &= 2\epsilon_x\epsilon_y - 2\epsilon_z\eta \\
\frac{\partial h_{\dot{y}}}{\partial \dot{y}} &= -\epsilon_x^2 + \epsilon_y^2 - \epsilon_z^2 + \eta^2 \\
\frac{\partial h_{\dot{y}}}{\partial \dot{z}} &= 2\epsilon_x\eta + 2\epsilon_y\epsilon_z \\
\frac{\partial h_{\dot{y}}}{\partial \epsilon_x} &= 2\dot{x}\epsilon_y - 2\dot{y}\epsilon_x + 2\dot{z}\eta \\
\frac{\partial h_{\dot{y}}}{\partial \epsilon_y} &= 2\dot{x}\epsilon_x + 2\dot{y}\epsilon_y + 2\dot{z}\epsilon_z \\
\frac{\partial h_{\dot{y}}}{\partial \epsilon_z} &= -2\dot{x}\eta - 2\dot{y}\epsilon_z + 2\dot{z}\epsilon_y \\
\frac{\partial h_{\dot{y}}}{\partial \eta} &= -2\dot{x}\epsilon_z + 2\dot{y}\eta + 2\dot{z}\epsilon_x
\end{aligned}
\tag{4.31}
$$

$$
\begin{aligned}
\frac{\partial h_{\dot{z}}}{\partial \dot{x}} &= \epsilon_x^2 - \epsilon_y^2 - \epsilon_z^2 + \eta^2 \\
\frac{\partial h_{\dot{z}}}{\partial \dot{y}} &= 2\epsilon_x\epsilon_y + 2\epsilon_z\eta \\
\frac{\partial h_{\dot{z}}}{\partial \dot{z}} &= 2\epsilon_x\epsilon_z - 2\epsilon_y\eta \\
\frac{\partial h_{\dot{z}}}{\partial \epsilon_x} &= 2\dot{x}\epsilon_x + 2\dot{y}\epsilon_y + 2\dot{z}\epsilon_z \\
\frac{\partial h_{\dot{z}}}{\partial \epsilon_y} &= -2\dot{x}\epsilon_y + 2\dot{y}\epsilon_x - 2\dot{z}\eta \\
\frac{\partial h_{\dot{z}}}{\partial \epsilon_z} &= -2\dot{x}\epsilon_z + 2\dot{y}\eta + 2\dot{z}\epsilon_x \\
\frac{\partial h_{\dot{z}}}{\partial \eta} &= 2\dot{x}\eta + 2\dot{y}\epsilon_z - 2\dot{z}\epsilon_y
\end{aligned}
\tag{4.32}
$$

The covariance update requires the Jacobian $\boldsymbol{M}_k$ (eq. (2.14)) and the DVL measurement noise matrix to be defined. The residual $\tilde{\boldsymbol{v}}_k$ is in the DVL reference frame, therefore, the additive noise assumption holds and $\boldsymbol{M}_k$ becomes,

$$
\boldsymbol{M}_k = \boldsymbol{I}_{3x3} \tag{4.33}
$$

the identity matrix. This simplifies Equations 2.9 and 2.12. Finally, the DVL measurement noise covariance matrix is defined by,

$$\boldsymbol{R}_k(\dot{\boldsymbol{x}}_{\mathcal{L}_{dvl}}) = \begin{bmatrix} \sigma^2_{\dot{x}_{dvl}} & 0 & 0 \\ 0 & \sigma^2_{\dot{y}_{dvl}} & 0 \\ 0 & 0 & \sigma^2_{\dot{z}_{dvl}} \end{bmatrix} \tag{4.34}$$

where $\sigma^2_{\dot{x}_{dvl}}$ is variance of the velocity measurement in the x direction within the DVL frame of reference. This implementation assumes the variance in all three axes are the same.

This concludes the details of the INS implementation. The INS model provides the basis of AUV dead-reckoning within the ANT. The next part of the ANT covered is the environment model.

4.1.2 Gazebo World Models

The Gazebo simulation environment supports mesh files for the creation of a simulation environment. These 3D object files can contain meshes with visual textures and shading. This is supported by the underlying 3D rendering engine. The 3D objects are used in visualization and physics simulation. This facilitates simulating sensors in a physical environment provided by the physics-based engines in Gazebo.

In-water testing is an extension of the work in this thesis. The nearest, and often used, body of water in the Halifax Area is Bedford Basin. To mitigate risk for future in-water tests the vehicle, its behaviours, and performance are first verified in a simulated environment. This provided the motivation to create the underwater Bedford Basin environment within Gazebo.

The Bedford Basin Gazebo world was created using the functionality of the ANT CLI. The original bathymetric 2 m resolution data provided by Canadian hydrographic service (CHS) [107] was merged with the lower resolution DEM provided by the government of Nova Scotia. The resulting raster covers a 7 km by 7 km area encasing the Bedford Basin.

This raster was transformed into a Gazebo world model using the ANT CLI developed as part of the thesis work presented and published at the Oceans 2021 conference in San Diego [31]. The tool makes heavy use of raster [35], numerical [108], and graphical tools [36] available as Python libraries, as well as the Blender Python API [34]. Blender is an open-source widely used 3D modelling and animation software [34]. This pipeline allows any raster, in a geospatial data abstraction library (GDAL) supported format, to be loaded by the tool and turned into 3D world model supported by the Gazebo simulator. The tool can resize, re-sample, scale, and transform the raster before turning it into the final 3D mesh and configuration files expected by Gazebo. The Blender Python API is used to overlay a colour map generated by matplotlib, a Python plotting library, onto the 3D mesh. This allows easier visualization of the height variation in the final Gazebo world model. This pipeline saves time and reduces the probability of mistakes when creating a Gazebo world model using raster data. It reduced what would normally take a researcher a day (not including the time to learn all the tools involved in the process) to a few minutes.

The results of this raster to Gazebo world model pipeline will be discussed in detail in 5.2.

4.1.3 Terrain Generation

Control over the detail and composition of the terrain used for the TAN algorithm enables testing of terrain variation and uncertainty in algorithmic performance. It also allows creation of synthetic density fields to fill in density information for which there are no available detailed measurements. This adds to the environmental uncertainty the TAN algorithm is subjected to — consequently making its simulations more realistic.

Perlin noise can also be used to generate pseudo realistic terrain to enhance the realism of simulations. This was previously done by others to simulate a spacecraft landing on an asteroid for the purpose of designing the spacecraft to explore the asteroid [90]. Perlin noise was used by Martin et al. [90] along with erosion modelling to supplement low-resolution terrain information of the modelled asteroids.

Similar techniques were used in the thesis. Perlin noise was used to augment

bathymetry and generate density fields to increase the fidelity of the simulation environment for the AUV long-distance transit operations. The augmented bathymetry add detailed terrain variation on top of the approximately 0.5 km resolution bathymetry available from the GEBCO database [29].

The ANT CLI was written to provide this functionality. It provides the ability to generate terrain to a set of specifications. The number of octaves, the base period, base amplitude, lacunarity, persistence, size, mean value, and origin can be set during generation. Additionally, the smallest feature size that is guaranteed can be set using a command line option. This implicitly chooses the number of octaves, n_{octave}, required to guarantee a minimum feature size of, s, within the terrain. The equation used to calculate the number of octaves is,

$$n_{octave} = \lceil \frac{\log(s) - \log(p_o)}{\log(1/l)} + 1 \rceil \tag{4.35}$$

where p_o is the period of the first Perlin noise layer, and l its lacunarity. This function was derived for use in the ANT CLI tool-chain.

This extends the possible use of the terrain generation tool to simulation of side-scan sonar within a ocean environment. The minimum feature size s for simulating the returns for side-scan sonar would be determined by the frequency used by the side-scan sonar. Side-scan sonar was not used within the work for this thesis. However, this increases the utility of this tool for other researchers. The tool also supports using a similar process to add noise to an existing lower-resolution map to increase the level of detail in a realistic way. This is referred to as terrain augmentation within this thesis.

4.1.4 Terrain Augmentation

The ANT CLI supports terrain generation. It reads an existing raster and adds Perlin noise layers on top of the existing raster. The raster retains its georeferencing information. Perlin noise is especially suited for this application because if the lacunarity is 2, the grid on which the gradients are sampled, and therefore the zero value crossings, line up with each other. The locations of the grid intersections of the

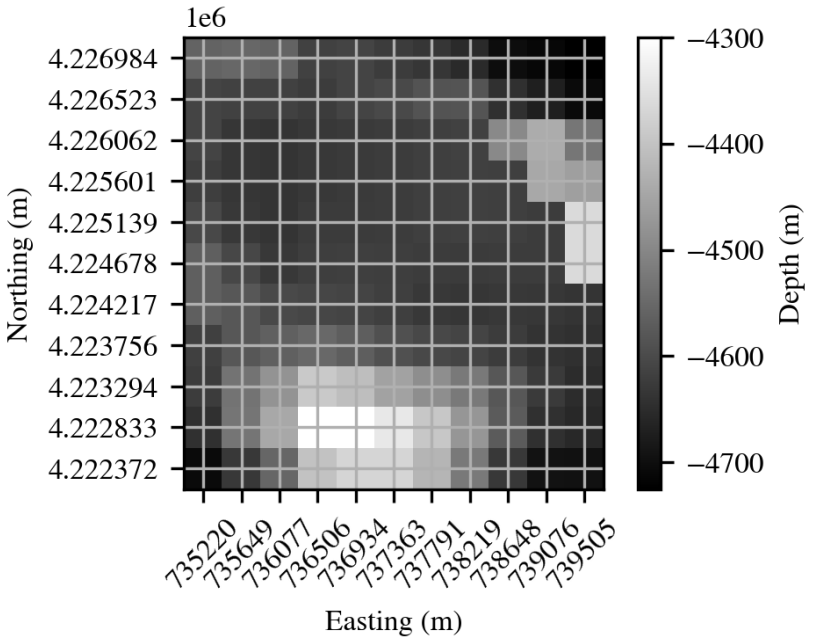

Figure (4.1) Adapted from [31]. An extracted raster from the GEBCO database. The resolution is approximately 450 m. The raster is georeferenced using the UTM Zone 20N NAD83 coordinate reference system.

first Perlin noise function are guaranteed to be zero. An affine transformation can be used to line the Perlin noise layers up with the lower resolution data. The result is additional noise in between the original measurements. This is much more realistic than assuming the terrain varies linearly between measurements when using a raster to generate an environment model.

For example, from Heubach and Seto [31] a 5 km by 5 km raster was extracted from the GEBCO database (shown in Fig. 4.1). It was chosen because it had interesting features (two seamounts). The raster has a resolution of about 450 m. Creating a simulation environment from this raster would normally involve interpolating between measurements. Whether this is linear or cubic the resulting seascape is smooth with no finer details. Depending on the application this may be sufficient. However, if this is used as the environment it assumes the measurements are unaffected by noise, and the terrain variation in between measurements is known. The more realistic approach taken in this thesis is to augment the terrain with noise and to use the augmented terrain as the simulation environment. The original measurements are used as input

to the TAN algorithm. This adds realistic uncertainty to the information that the TAN algorithm has about the environment.

An example of this augmented terrain is shown in Fig. 4.2. From Fig. 4.2, the

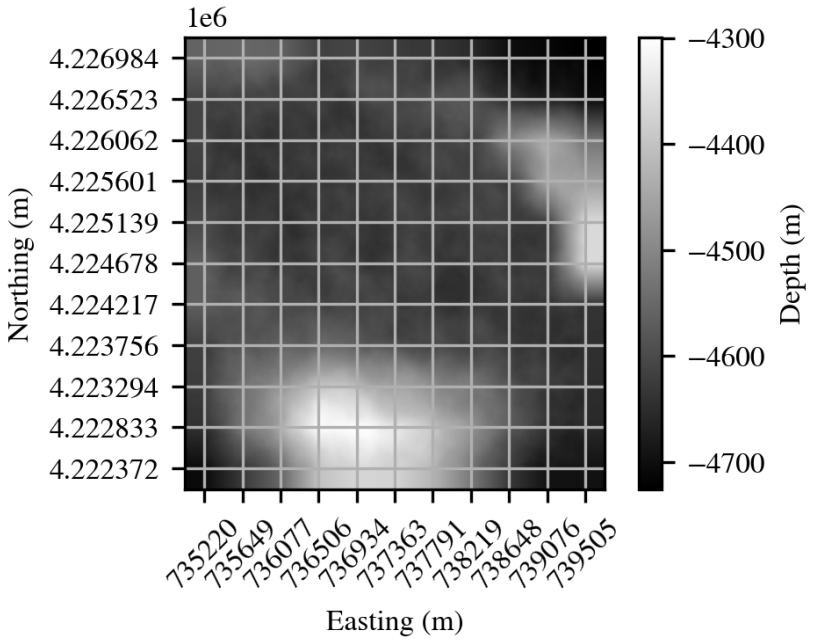

Figure (4.2) Adapted from [31]. An extracted raster from the GEBCO database. It has been augmented with five octaves of Perlin noise, with a lacunarity of 2 and persistence of 0.5.

details added by the Perlin noise is subtle. It adds realism and additional uncertainty when the original GEBCO measurements are used for the navigation or localization algorithm. Augmenting the terrain leaves the original measurements at the original measurement location unchanged. This assumption is verified and discussed in section 5.4. The terrain augmentation feature is not used for testing the TAN algorithm. The resolution of the bathymetry was sufficient to test the navigation over long-distances.

4.1.5 Terrain to Gravity Anomaly Conversion

The approximation of the gravity anomaly used in the ANT makes use of eqn 2.32, introduced in section 2.12.2. This is possible because the gravity anomaly created

by mass attenuates by the inverse square law. The gravity gradient, being the first derivative of the gravity anomaly, attenuates at the inverse cube of the distance, making it even more sensitive to local mass variations [82, 94]. Therefore, the local mass variation has the largest impact on the gravity gradient. This is assumption is used by Liu et al. in [43]. Additionally, since the interest is in the gravity gradient, for navigation purposes, the absolute value of the gravity anomaly becomes irrelevant for navigation [94]. Its variation must be within the dynamic range of the gradiometer (approximately 1 E [94], this depends on platform and will only improve with time).

The local anomaly calculation is posed using matrix algebra to exploit available matrix algebra tools and ultimately reduce the computation time. The local gravity anomaly is approximated using a raster of the bathymetry as input. Optionally, a density raster can also be supplied. If the density raster is not used the constant density assumption is made. This replaces the entire density matrix with the identical value of $2670\,\mathrm{kg\,m^{-3}}$ (An approximate value of the outer crustal density). A single average value for the density of the Earth's crust is really difficult to find. This value is sufficient. The gravity anomaly does not need to be absolutely accurate — only relatively accurate. Next, it is assumed the average density of the seawater is $1027\,\mathrm{kg\,m^{-3}}$ [109]. The density of sea water depends on temperature and salinity [109], however, the depth has the largest effect on density variation in seawater and can be modelled [110]. Extensive modelling of the seawater density variation is outside the scope of this thesis, so the constant value will be used.

A few properties can be set to affect the calculation of the local gravity anomaly field. Calculation depth, target window size, and target resolution can be set. The calculation depth is the depth at which local gravity anomaly is calculated. The target window size is used to determine the horizon for which mass elements are deemed to negligibly impact the gravity anomaly, and the target resolution determines the window movement step size.

A single local gravity anomaly is calculated by selecting a single position to determine the approximate local anomaly. Equation (2.32) is used, where the density ρ is determined by either the constant density assumption or a value from the density raster. The top-down area $\delta x \cdot \delta y$ of a finite volume is determined by the resolution

of the bathymetry raster. The height of the finite volume, δz, is set by the ocean depth. The difference in depth between the centre of the finite volume of ocean water and the calculation depth yields Δz. The Euclidean difference between the centre of the finite volume and the calculation position yields the distance r. Together these parameters provide the means to calculate the local anomaly at a single point by performing a summation of all the contributions to the single local anomaly from each finite volume within the designated window size. This window can be swept across the raster to calculate local anomaly field.

The approximate local anomaly field will be smaller in its geographical bounds than the input raster. This is due to the window size (horizon). A local anomaly can only be calculated at the centre of a window, therefore, the local anomaly field will be half a window size smaller in the direction of all the Cartesian directions. The resulting gravity anomaly maps are shown and discussed in further detail in section 5.5.

There are significant limitations to this approach. First, the gravity anomaly cannot be compared to measured gravity anomalies directly. It would be more accurate to call the calculated gravity anomaly the relative gravity anomaly, because it is relative to the local area in which it was calculated. This means it cannot be directly used for navigation as the measured gravity anomaly. The gravity anomaly is different than the relative gravity anomaly and would lead to quick divergence of the state estimate. However, if only the gradient of the local anomaly is used then the absolute difference between measured and calculated becomes irrelevant [95]. The gravity gradient can be measured with a gravity gradiometer, and compared to the gradient of the calculated local gravity anomaly for AUV navigation [43, 92, 82].

4.1.6 Gravity Gradiometer Model

The gravity gradiometer model used within the ANT is defined by eq. (4.36),

$$\begin{bmatrix} \|\nabla g\| \\ \angle \nabla g \end{bmatrix} = \begin{bmatrix} \sqrt{\left(\frac{\partial g_z}{\partial x}\right)^2 + \left(\frac{\partial g_z}{\partial y}\right)^2} + \mathcal{N}(\mathbf{0}, \sigma_m) \\ \arctan 2\left(\frac{\frac{\partial g_z}{\partial y}}{\frac{\partial g_z}{\partial x}}\right) + \mathcal{N}(\mathbf{0}, \sigma_\theta) \end{bmatrix} \tag{4.36}$$

where $\|\nabla g\|$ is the planar magnitude of the gravity gradient, $\angle\nabla g$ is the gravity gradient's heading planar direction, $\mathcal{N}$ is the normal distribution, g_z is the vertical component of the gravity gradient, σ_m is the standard deviation of the Gaussian noise applied to the magnitude of the gravity gradient, σ_θ is the standard deviation of the Gaussian noise applied to the gravity gradient heading, x is the coordinate direction in-line with the cardinaleast direction, y is the coordinate direction in-line with the north cardinal direction. The study that explores the effect of gravity gradiometer heading noise on the performance of the particle filter implementation of GAN is discussed in section 5.6.1.

The gravity gradiometer has limitations. It does not include the tidal effect, the vehicle-induced acceleration, the Eötvös effect, or thermal drift. The latter three effects are negligible if the sensors within the gravity gradiometer are precisely calibrated together. Additionally, the gravity gradiometer ignores the third dimension of the gravity gradient. This was to make the implementation simpler, and fit within the scope of the thesis. A more realistic gravity gradiometer would give the full three-dimensional gravity vector. Finally, the uncertainty due to noise would be much more complex. There would be uncertainty associated with all the sensors within the gradiometer and additional uncertainty from the combination of their measurements. However, this model suffices in enabling exploration of the implementation of a GAN algorithm. Increasing the complexity, realism, and accuracy of these models is part of the possible continuation of the work in this thesis.

This concludes the discussion of the implementation of the main parts of the ANT. Much of the covered functionality is used by running the ANT CLI. The summary of this tool is covered next, along with additional functionality not yet covered.

4.2 AUV Navigation Testbed Command Line Interface

The ANT CLI contains much of the peripheral tool-chain that helps create an underwater simulation environment quickly. The CLI supports various operations related to the ANT including: visualizing Perlin noise, augmenting existing terrain, resampling a raster, cropping a raster to a square, generating density for a raster, generating terrain, deriving an gravity anomaly field from bathymetry and density, displaying

a geotiff, translating a raster, generating traces from the diagonal of GeoTIFF files, and converting a raster to a Gazebo world model. Some of these functionalities have already been discussed in detail. Others will be discussed in this section.

4.2.1 Visualizing Perlin Noise

To help with the understanding of how Perlin noise layers interact and how certain parameters affect the roughness, and detail of generated terrain, visualization functionality was added to the ANT CLI. The command accepts the target resolution, size, base period, mean, amplitude, number of octaves, lacunarity, persistence, and a random seed as options. The target resolution determines the resolution at which the superimposed Perlin noise layers are sampled, the size determines the scaling factor applied to the Perlin noise function, the base period is the inverse of the Perlin noise gradient grid spatial sampling frequency, the mean is the transformation for all values of the Perlin noise function, amplitude is the multiplication factor for all values of the sampled function, the number of octaves determines the number of Perlin noise layers, the lacunarity the ratio of frequencies of successive Perlin noise layers, persistence the ratio of the amplitude of successive Perlin noise layers, and the random seed allows the creator to control what see is used to initialize the pseudo-random number generator.

The Perlin noise functions are generated using these constraints then plotted and shown using matplotlib a Python plotting library [36]. This leads to fast iteration on which parameters deliver the effect desired by the user.

4.2.2 Terrain Augmentation

The terrain augmentation command provides the interface to augment existing terrain with Perlin noise. This functionality was covered in section 4.1.4.

4.2.3 Raster Re-sampling

The re-sampling command eases working with large rasters that may have much higher or lower resolution than one would like for their application. This is especially important when importing very large maps into Gazebo which has a hard limit for

how many data points it can handle. This command uses built-in functionality from the rasterio Python library [35] and is really there for convenience. There are a lot of other tools, such as GDAL, that provide this same functionality.

4.2.4 Crop Raster to Square

This command crops an existing raster to a square. This is important when working with Gazebo which has trouble when working with digital elevation models that are not square. Additionally, a square raster is a prerequisite for Perlin noise terrain augmentation.

4.2.5 Generate Density for Raster

This command generates a density field using Perlin noise. The input parameters for the Perlin noise function can be specified. Additionally, this command uses the georeferencing information from the input raster to georeference the generated density field to the same geographical area as the original raster. This allows the density field to mimic density measurements taken in that geographical area. More discussion about generated density fields can be found in section 5.3

4.2.6 Generate Terrain as Raster

This command provides terrain generation capabilities similar to the Perlin noise visualization command. However, instead of the plot being produced, the result is a GeoTIFF raster file. This can be used in any geographic information system (GIS) tool, or the other ANT CLI commands that support a raster as an input such as the raster to Gazebo world command.

4.2.7 Derive Gravity Anomaly Field

This command uses the correlation between the gravity anomaly and the terrain to derive a local gravity anomaly field. This commands implements the equations and processes first introduced in section 2.12.2. This command supports including a crustal density field in the derivation of the gravity anomaly. A finite element approach is used to derive the local gravity anomaly. The resulting local gravity

anomaly maps are discussed further in section 5.5.

4.2.8 Show GeoTIFF

This command supports plotting an existing GeoTIFF raster. Some simple properties of the resulting figure can be changed such as x label, y label, x tick spacing, y tick spacing, and x multiplication factor, y multiplication factor, and an output path at which to save the figure. This command speeds up making quick plots of existing GeoTIFF rasters, and is used to produce figures found throughout this thesis.

4.2.9 Translate Raster

This command takes a input raster path and applies a simple transformation to the location of the raster. Useful to be able to apply a false origin to place the centre or a corner of the raster at 0,0.

4.2.10 Compare Slices

This command takes in any number of .stl, raster, and .dae files and determines the trace along the main diagonal. This is really helpful when verifying that a transformation from a raster to standard triangle language (STL) file was completed correctly. A slice of all the main diagonals is plotted for comparison. It makes errors in conversion much easier to detect.

4.2.11 Raster to Gazebo World

This command provides the pipeline from raster to Gazebo world model. The destination folder can be specified, so can the size, the height exaggeration, and the colour map to apply as a visual effect. Any colour map name supported by matplotlib will work. An example of height exaggeration can be see in section 5.2. The world size parameter is useful for scaling a raster that covers a really large area, down to a smaller area that Gazebo supports. Gazebo is limited in the size of model it supports. Gazebo has support for DEM, however, there were numerous collision and visual problems. Therefore, the world models are loaded in as Collada files.

Collada is a 3D model interchange format used by numerous 3D modelling software. It allows interchange of 3D models between different software. The 3D model was created within Blender by importing the original raster. Then the raster is converted to a mesh, rendering properties are set, and the colour map overlay is added. This is exported as a Collada file that Gazebo supports. This reduces a process that was error prone, tedious, and took the better part of a day down to a few minutes of run time for a raster containing millions of measurements.

4.3 ROS Parameter Study CLI

In the process of performing the parameter studies within the ROS and Gazebo environments the lack of a formalized approach to running studies was discovered. This led to the necessity of developing a Python library that formalizes an approach to parameter studies within the ROS ecosystem. The Python library, ros_parameter_study (rps for short), provides the means of organizing multiple ROS launch files, command line environments, rosrun commands, topic bag data collection, and stopping conditions to perform a parameter study. The capabilities of the ROS parameter study CLI and the accompanying RPS library are described in detail in appendix E.

The ROS parameter study tool may not be an academic contribution in the strictest sense, however, the package will be released to the academic and ROS communities as an open-source package. This contributes to the efficiency of others that use the package to reduce their own work load. The work in this thesis would not have been possible without the work of others sharing their written software, and benefited greatly from the open-source community. Therefore, sharing this work is significant.

This concludes the discussion about the functionality provided by the ANT, and related tooling. Next, the implementation details of the particle filter based TAN algorithm are discussed.

4.4 Terrain Aided Navigation Algorithm

The overview of the TAN algorithm is shown in Fig. 4.3. The TAN algorithm uses the particle filter framework to perform state estimation of the position and heading of the vehicle in two-dimensions. The particles are propagated forward in the prediction state using the input from the INS, and the particles are weighted based on the likelihood that the measured gradiometer heading would be seen at the location of the particles hypothesis.

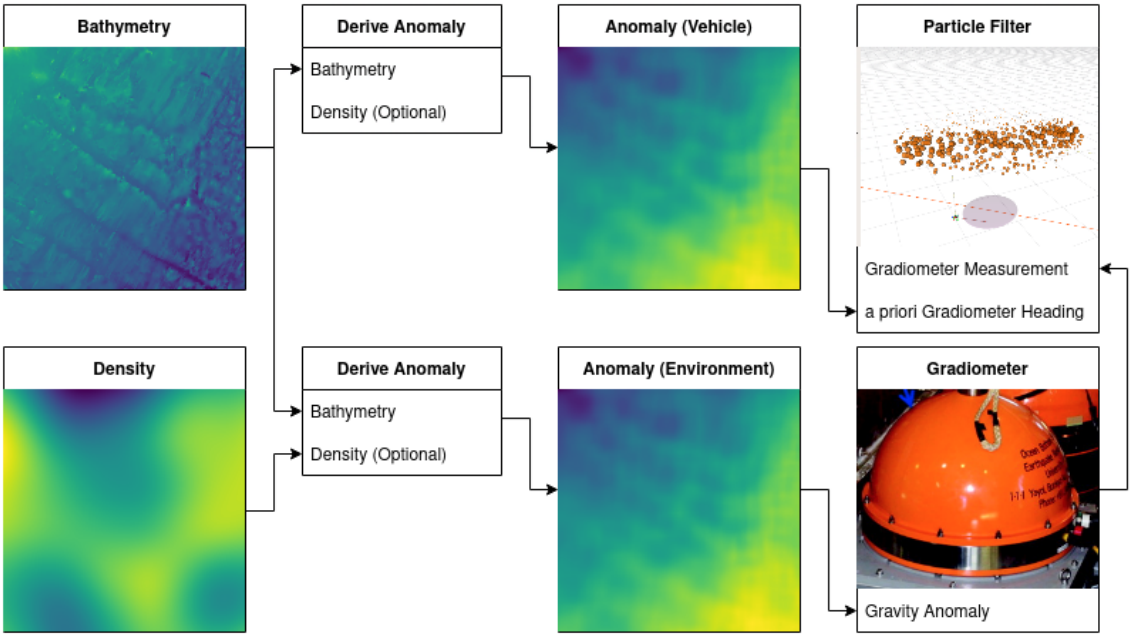

Figure (4.3) An overview of the gravity-aided navigation implemented. The bathymetry from readily available sources like GEBCO is used with a generated density field to derive the gravity anomaly field used as the environment in simulations. The gravity anomaly field that the particle filter uses to evaluate the likelihood of a gradiometer heading measurement is derived from the original bathymetry with the constant density assumption.

The TAN algorithm requires *a priori* bathymetry information which can be converted to gravity gradient maps exploiting the correlation between the terrain and the gravity anomaly, discussed in section 2.12.2. The simulated gradiometer uses a similarly derived gravity anomaly map. However, the environment gravity anomaly includes the influence of the terrain density which is unknown to the vehicle state estimation. This explores the validity of the constant density assumption the particle

filter uses to perform TAN.

4.4.1 Particle Filter

The TAN algorithm is implemented using a particle filter. A particle filter or sequential Monte Carlo method uses a set of samples to represent the posterior state estimate distribution. These samples are referred to as particles, each one representing a single state hypothesis. A particle filter uses a propagation, weighting, and resampling cycle.

Propagation

The particle filter uses the INS as the basis of its propagation or prediction step.

$$\begin{bmatrix} x^{k+1} \\ y^{k+1} \\ \theta^{k+1} \end{bmatrix} = \begin{bmatrix} x^k + \Delta d_{ins} \cos(\theta^{k+1}) \\ y^k + \Delta d_{ins} \sin(\theta^{k+1}) \\ \theta^k + \Delta\theta^{ins} \end{bmatrix} \tag{4.37}$$

where Δd_{ins} and $\Delta\theta_{ins}$ are given by,

$$\begin{bmatrix} \Delta d_{ins} \\ \Delta\theta_{ins} \end{bmatrix} = \begin{bmatrix} \sqrt{(x^k_{ins} - x^{k-1}_{ins})^2 + (y^k_{ins} - y^{k-1}_{ins})^2} \\ \theta^k_{ins} - \theta^{k-1}_{ins} \end{bmatrix} \tag{4.38}$$

and where x^k_{ins} is the INS position estimate in the x direction, y^k_{ins} is the INS position estimate in the y direction,

Weighting

The weight assigned to each particle is given by the likelihood of the gradiometer measurement θ_m, given the gradiometer direction at the particle's location x_i, y_i retrieved from the vehicle's *a priori* gravity gradient map. The likelihood function is not a simple Gaussian, as other dimensions in the state estimate, because a heading is defined on an interval of 2π and one would expect the likelihood function to wrap around just as the angle does. This behaviour can be approximated using eq. (4.39).

$$\mathcal{L}(\theta_m|\theta_i,\sigma_m) = \frac{1}{\sigma_m\sqrt{2\pi}}\left(\mathrm{e}^{-\frac{1}{2}(\frac{\theta_m-\theta_i}{\sigma_m})^2} + \sum_{j=1}^{n}\left(\mathrm{e}^{-\frac{1}{2}(\frac{\theta_m-\theta_i+2\pi j}{\sigma_m})^2} + \mathrm{e}^{-\frac{1}{2}(\frac{\theta_m-\theta_i-2\pi j}{\sigma_m})^2}\right)\right) \tag{4.39}$$

where θ_m is the gradiometer's gravity gradient direction measurement, σ_m is the standard deviation of the gradiometer's gravity gradient direction measurement, and θ_i is the gravity gradient direction at the location of the i^{th} particle given by the vehicle's *a prior* gravity gradient map. θ_i is evaluated by taking the position (x_i, y_i) of the i^{th} particle and looking up the value of the gravity gradient direction at that location in the vehicle's *a priori* gravity gradient map.

Equation (4.39) repeats the likelihood function every $\pm 2\pi j$ for j from 1 to n. The number of repetitions depends on the standard deviation of the Gaussian function. The total area under the distribution can be estimated to less than 1% error by calculating n to capture at least the area within three standard deviations of the distribution. n is calculated using eq. (4.40):

$$n = \left\lceil \frac{3\sigma_m}{2\pi} \right\rceil. \tag{4.40}$$

Once the relative likelihood is calculated, each particle's weight is updated using eq. (4.41),

$$w_i^{k+1} = w_i^k \cdot \mathcal{L}(\theta_m|\theta_i,\sigma_m) \tag{4.41}$$

where w_i^{k+1} is the updated particle weight, w_i^k is the previous particle weight, and the relative likelihood is given by eq. (4.39).

Following the weight update, the particle weights must be normalized. Normalization is completed using eq. (4.42),

$$w_i^{k+1} = \frac{w_i^k}{\sum_i w_i}. \tag{4.42}$$

Over time some particles can become insignificant because of their extremely low weight. Ideally, these particles would be dropped and replaced by particles spawned

near particles with higher weights. This is achieved with particle resampling.

Resampling

The particle filter uses resampling to increase the particles filter's resolution in areas with heavily weighted particles. There are numerous resampling schemes used in particle filters [96]. This thesis focused on multinomial resampling. Multinomial resampling is applied when the effective sample size is below a threshold. The effective sample size is a measure used to describe how many particles are effectively describing the particle distribution. To avoid particle degeneracy, where particles have very low weights and have no effect on the weighted distribution anymore, the resampling step is used. The measure of effective sample size used is described in eq. (4.43) [96, p. 85],

$$\text{ESS} = \frac{1}{\sum_i w_i^2} \tag{4.43}$$

where w_i is the weight of the i^{th} particle, and ESS is the effective sample size. The threshold used is given by eq. (4.44),

$$n_{eff} < \frac{N_p}{2} \tag{4.44}$$

where N_p is the total number of particles.

When the threshold is met the particles are resampled using the multinomial resampling scheme [111] [96]. The particles are resampled from a discrete distribution where the probability of sampling a particle is directly proportional to its weight w_i. After, the weights are reinitialized to,

$$w_i = \frac{1}{N_p} \tag{4.45}$$

where N_p is the total number of particles. Without stabilizing noise resampling can lead to particle duplication.

Stabilizing Noise

The goal of stabilizing noise is to inject enough noise to spread the particles out in the state space without sacrificing particle filter convergence. Stabilization noise is also referred to as jitter. Equation (4.46) shows the jitter vector that is added to each particle's state estimate.

$$\begin{bmatrix} j_x \\ j_y \\ j_\theta \end{bmatrix}_i = \begin{bmatrix} f_x \cdot \mathcal{N}(\mathbf{0}, \sigma_x) \\ f_y \cdot \mathcal{N}(\mathbf{0}, \sigma_y) \\ f_\theta \cdot \mathcal{N}(\mathbf{0}, \sigma_\theta) \end{bmatrix}_i \tag{4.46}$$

where i is the particle index, j_x is the jitter in the x coordinate direction, j_y is the jitter in the y coordinate direction, j_θ is the jitter added the heading estimate, f_x is the multiplicative factor in the x coordinate direction, f_y is the multiplicative factor in the y coordinate direction, f_θ is the multiplicative factor for the heading estimate, σ_x is the standard deviation of all the particle x position estimates, σ_y is the standard deviation of all the particle y position estimates, σ_θ is the standard deviation of all the particle heading estimates.

The jitter vector is added directly to the state estimate. The sampling from the normal distribution is done separately for every particle estimate. This provides enough jitter to keep particles from representing the same hypotheses while maintaining particle filter convergence.

After testing the AUV navigation algorithm within the ANT. The next steps are vehicle integration, and in-water testing. The next section explains the details of the developments towards these objectives, the DCAF project.

4.5 DRDC Collaborative Autonomous Framework

The goal for testing the TAN algorithm is in-water testing. The intermediate step for this is to integrate the algorithm on the vehicle. A HITL significantly accelerates this integration by replicating the software stack running on the actual vehicle. The ISL has access to two HITL that run the full vehicle software stack. Additionally, it has an environment simulator that supports a range of environmental, inertial, and

survivability sensors. However, it does not include a gravity related sensor like a gravimeter or gravity gradiometer. Therefore, the main value of using the HITL is integration of the TAN algorithm code and supporting peripherals onto a software replicate of the vehicle.

The DCAF project is ongoing and the work was collaborative. The author was the main developer for the bounce behaviour, the arbiter, and the software documentation. The implementation details of the bounce behaviour, and the arbiter will now be discussed.

4.5.1 Bounce Behaviour

The bounce behaviour constrains the vehicle to a virtual volume defined by the boundaries (vertical walls) of a rectangular prism. In practice this is implemented by constraining the vehicle in a horizontal operating region and a operating depth range. The aim of of this behaviour is reducing the chances of vehicle loss if another behaviour tries to submit an action that would place the vehicle outside its allowed operating region. The bounce behaviour has one of the highest priorities, second only to the obstacle avoidance behaviour (see section 4.5.2). The bounce behaviour is called bounce because the vehicle is confined to the operating region by "bouncing" off the virtual boundaries.

Vertical Bounce

The vertical bounce behaviour constrains the vehicle to a range of depths. This can be used to keep the vehicle from going below a certain depth if another behaviour commands it to dive deeper. The parameters defining the behaviour are summarized in table 4.1. The vertical bounce behaviour is triggered when either eqs. (4.47) and (4.48) become true.

$$d > d_{max} - d_{max\_buffer} \tag{4.47}$$

$$d < d_{min} + d_{min\_buffer} \tag{4.48}$$

Table (4.1) A summary of the parameters that the vertical bounce behaviour supports and their descriptions.

Parameter	Description
max_depth	The maximum depth the vehicle is permitted to dive to.
min_depth	The minimum depth the vehicle is permitted to surface to.
max_depth_buffer	The safety buffer used to trigger the bounce behaviour when the vehicle is close to reaching the max_depth.
min_depth_buffer	The minimum safety buffer used to trigger the bounce behaviour when the vehicle is close to reaching the min_depth.

where d is the AUV depth. When triggered the vertical bounce behaviour requests a high priority depth set point at the buffer distance from the depth bounds.

Horizontal Bounce

The horizontal bounce behaviour constrains the vehicle to an operating region. The implementation of the horizontal bounce behaviour makes use of the operating region boundary, and operating region buffer size to calculate an inner safety boundary that triggers the bounce behaviour when exited by the AUV. The parameters supported by the horizontal bounce behaviour are described in table 4.2. The horizontal bounce behaviour is triggered when the vehicle exits the inner safety boundary calculated using an offset, defined by the operating_region_buffer, from the operating region boundary. A plan to get the vehicle back within the operating region is calculated using geometry. Two circles with a radius of the vehicle's turn radius are created. Both are placed so they are coincident with the vehicle position and tangent to the vehicle heading. The intersection with the inner boundary is calculated for both circles (shown in Fig. 4.4). The shortest path along the circles, in the direction of the vehicle, back into the safety region is chosen. The waypoint corresponding to the intersection with the safety boundary and the shortest path is sent to the arbiter as a high priority waypoint action. Algorithmic details of this planning are included in appendix D.

Table (4.2) A summary of the parameters that the horizontal bounce behaviour supports and their descriptions.

Parameter	Description
operating_region_bounds	A string conforming to the comma separated value (.csv) format to define a sequence of four points. These points are in latitude longitude format and defined the corners of the rectangular operating region.
operating_region_buffer	The safety margin or buffer used to calculate an area within the operating region that is no closer than the buffer value to the boundary defined by the operating region.
turn_radius	The vehicle turn radius to use to plan the vehicle path back into the inner boundary region. Should be less than half of the buffer value.

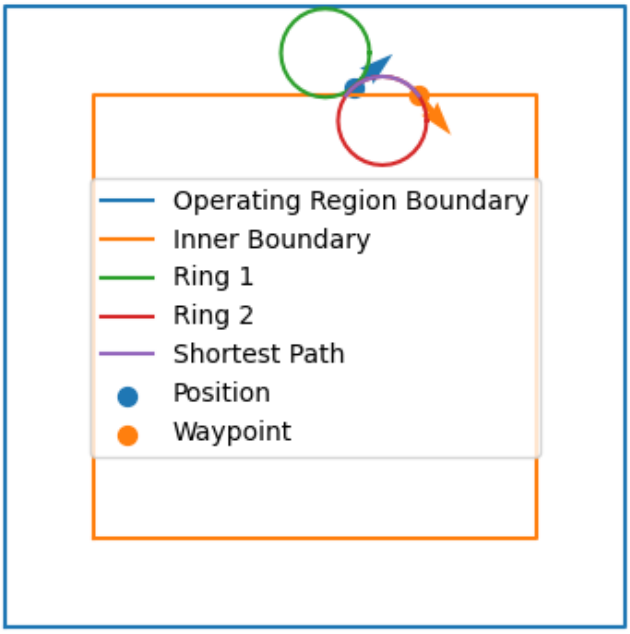

Figure (4.4) The top-left corner of the operating region is shown. The outer and inner boundary are shown as the straight black lines. The distance between these two lines is defined by the buffer parameter. The current vehicle position that triggered the horizontal bounce behaviour is shown with a blue dot and an vector for the vehicle heading. The circles used for path planning are shown as the black circles. Finally, the orange dot and vector represent the waypoint sent to the arbiter, by the bounce behaviour, to get the vehicle back to safety.

The planning algorithm uses computational geometry to determine the quickest, safest path back to the operating region for the kinematically limited AUV. Therefore, the algorithm is computationally efficient to run. The planning algorithm does not consider the vehicle dynamics or kinematics within the algorithm. The non-holonomic constraints of the vehicle are only addressed by the user-provided vehicle turn radius. There is no guarantee that the vehicle can perform the motion that is planned given its current momentum. This can only be addressed by being conservative with the vehicle turn radius and setting it to the worst case scenario. Since the bounce behaviour is a behaviour that is used only when the vehicle is performing anomalously, this is deemed to have minimal performance impacts to the vehicles nominal operation.

4.5.2 Vehicle Behaviour Arbitration

The vehicle behaviour arbitration is conducted by the arbiter process. The arbiter supports a ROS action interface. All behaviours within DCAF use this action interface to request an action from the vehicle. The action interface supports servo and waypoint commands. Servo commands change the vehicle heading set point for a specified period of time. The waypoint command sets the vehicle controller's waypoint objective. The goal of the arbiter is to take competing behaviours action priorities and make a decision on which action to submit to the vehicle, when to interrupt actions with other more important ones, when to resume previously interrupted actions, and finally to pass along cancel requests for actions that are no longer needed.

The arbiter is fundamentally implemented as a priority queue. Adding a behaviour to DCAF requires adding a behaviour ID to the configuration for the arbiter along with the behaviour priority. For example, the waypoint planner for search patterns is the lowest priority, the bounce behaviour is of higher priority, and currently the obstacle avoidance behaviour has the highest priority. Interrupting and resuming an action is supported by controlling the conditions under which items get pushed and popped off the priority queue.

The priority queue is implemented using the C++ standard library's implementation of heap algorithms. Every time a request for action is issued to the arbiter through the action interface, it is pushed onto the heap using `std:push_heap`. When

there are multiple actions in the queue, the one with the highest priority is selected. If there is a conflict between multiple actions with the same priority the actions are selected on a first-in-first-out (FIFO) basis. The arbiter keeps track of the current active action, without removing it from the queue yet. Once the action is deemed complete the arbiter pops action from the queue and starts the next one. If the action at any time is cancelled by the behaviour then the arbiter relays the cancel request to the vehicle, pops the action from the queue, and begins the next action in the queue.

The arbiter supports interrupts and resuming interrupted actions. When the arbiter is currently executing a lower priority action and receives a request for a higher priority action it will send the higher priority action request to the vehicle for execution. It will not remove the lower priority action from the queue, as lower priority action will be resumed once the higher priority action is complete. An interrupt is needed when the arbiter detects that the action at the front of the queue does not match the active action anymore. This means that a higher priority action has been received during the other actions execution. The arbiter must now be interrupted to service the higher priority action. The arbiter relies heavily on the max heap algorithm.

C++'s max heap algorithm is used because it provides the majority of the functionality required from the arbiter, is part of the standard library, is tested and is reliable. Any additional functionality for the arbiter, like the action interface, is provided by ROS [99]. This together makes the arbiter more maintainable and robust then using a custom algorithm. In the future the arbiter may need to support concurrent actions that do not conflict with each other on the vehicle. Such as taking measurements and diving. The arbiter can be changed to meet these requirements once necessary.

This concludes the discussion of the author's main contributions to the DCAF project, and the experiment section. The results and verification of these implementations will now be discussed.

Chapter 5

Results and Discussion

This chapter presents and discusses the results. First, the verification of the INS model is presented and discussed. Second, results related to environment modelling are discussed — Gazebo world models, terrain generation, and terrain augmentation, and gravity maps. Third, and finally, the results of the parameter studies using the particle filter based TAN algorithm are discussed.

5.1 Inertial Navigation Modelling

The verification of the INS model is presented in this section. The simulations demonstrate certain aspects of the INS/DVL system well, and will be further discussed. Table 5.1 provides a qualitative representation of each simulation.

Table (5.1) Summary of qualitative and quantitative aspects of each INS model simulation.

Simulation Number	Gyroscope Deviation	Initial Heading Deviation	DVL Update	Orientation Update
1	$0.01\,\mathrm{rad\,s^{-1}}$	≈ 0	Yes	No
2	$0.01\,\mathrm{rad\,s^{-1}}$	5°	Yes	No
3	$0.01\,\mathrm{rad\,s^{-1}}$	5°	Yes	Yes

In all simulations the AUV performs the "lawnmower" or boustrophedon survey mission over an area in Bedford Basin as shown in Fig. 5.1. This pattern is used for survey missions to provide overlapping swaths of sensor coverage (e.g. mine counter-measures and bathymetric mapping). Note, this bathymetric map within the simulation and Gazebo does not observe the east-north-up convention. The results will be presented with the positive y-axis in the south direction. As shown, the path length is approximately 1700 m and starts at (500,750). The transect lengths are 200 m, and the spacing is 60 m. This area was chosen because of the distinctive sea mount feature at the start of the mission and less variations later in the mission. The DVL was modelled with a standard deviation of $0.3\,\mathrm{m\,s^{-1}}$, to emulate the performance of the Kearfott SeaNav -24™sea navigation unit. The SeaNav -24 is a fully integrated sea navigation unit that has a bounded orientation error of 1 mrad and a position error of 0.1% of distance travelled circular error. Simulation three emulates

the SeaNav -24 performance; however, the first two simulations explore the effects of having a less fully integrated state-of-the-art navigation solution.

All simulations use a gyroscope with a covariance matrix defined in the body-centered reference frame,

$$\mathbf{\Sigma}_g = \begin{bmatrix} 0.0001 & 0 & 0 \\ 0 & 0.0001 & 0 \\ 0 & 0 & 0.0001 \end{bmatrix}. \tag{5.1}$$

This variance was chosen for demonstration purposes based on the localization error ellipses within the mission. Similarly, the initial position deviation is 1 m and the initial velocity deviation is 0.1 m for all simulations. This facilitates comparisons between simulations and attribution of observed effects on the localization uncertainty. The position uncertainty will be presented as an ellipse representing the 95% confidence error ellipse of the multivariate Gaussian distribution for the x, y position.

Simulation 1, explores the localization of the INS/DVL solution without an initial heading error. This is unrealistic, but serves as a baseline to highlight the impact of the integrated gyroscope on the position uncertainty. The results of simulation 1 are shown in Fig. 5.2. The results show that integrating the measured angular velocities to determine the estimated heading, and using the heading to derive the rotation from world frame to the body-centered frame, creates unbounded error in the position estimation. The integration of the DVL measurements contributes to the uncertainty but not the orientation or the error ellipse shape, as the DVL's measurement uncertainty is simulated identically for all three axes. The final position uncertainty has a covariance of,

$$\mathbf{\Sigma} = \begin{bmatrix} 22 & -9 \\ -9 & 65 \end{bmatrix} \tag{5.2}$$

resulting in an error ellipse with major and minor axes lengths of 40 m and 22 m, respectively.

Simulation 2, more realistically, includes the initial heading uncertainty. A small

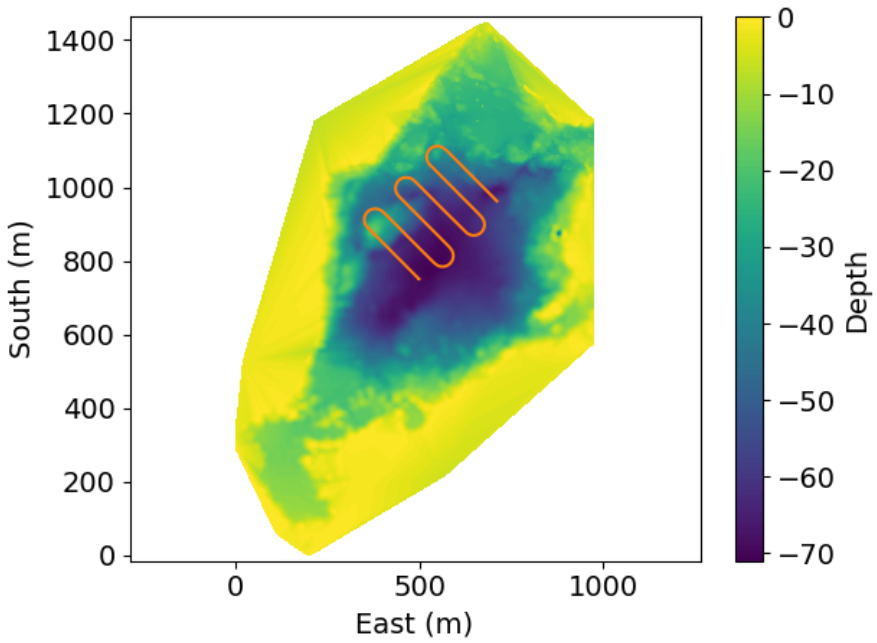

Figure (5.1) Bedford Basin, Halifax NS, Canada bathymetry, showing the search pattern used for the simulations.

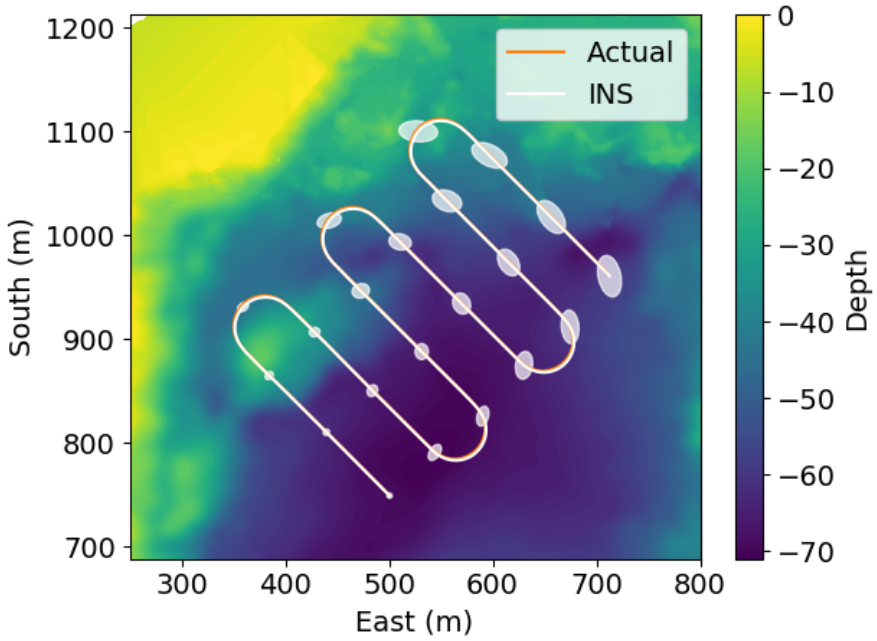

Figure (5.2) Simulation Scenario 1: No initial heading uncertainty. The ellipses represent the 95% confidence boundary of the Gaussian uncertainty distribution.

heading error of 5° was chosen. From Fig. 5.3, it is clear that the initial heading error has significant impact on the uncertainty of the final position. This is due to the heading error also affecting the position error. The INS implementation includes the reference rotation frame from the body frame to the world frame within the EKF. Therefore, heading uncertainty will have effects on the localization accuracy of the INS. It also causes the ratio of the error ellipse's major to minor axes to increase

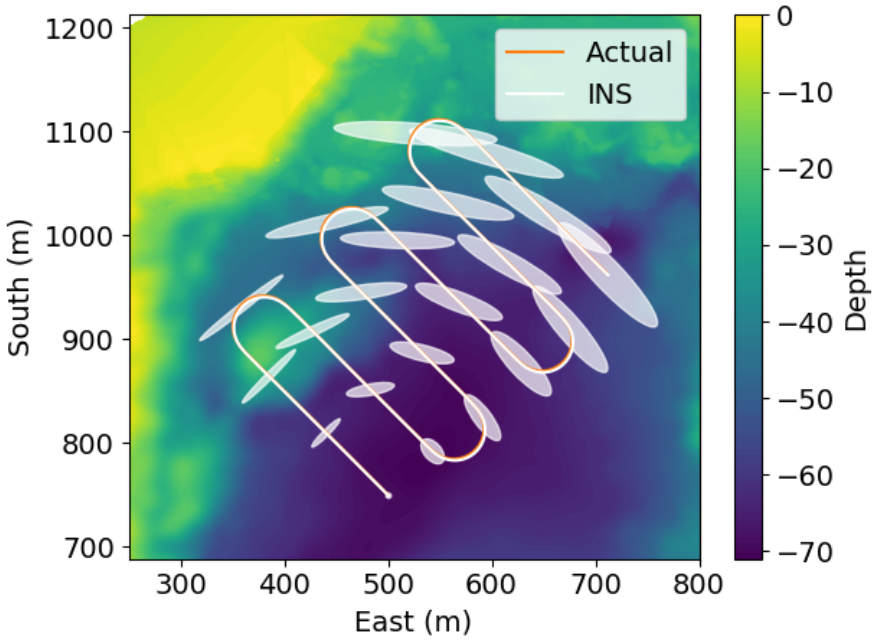

Figure (5.3) Simulation Scenario 2: 0.09 rad (5°) initial heading uncertainty. The ellipses represent the 95% confidence boundary of the Gaussian uncertainty distribution.

significantly from 2 to 5. This localization solution has no external reference for its orientation; therefore, the heading uncertainty introduced by the initial heading error will persist. The final position uncertainty has a covariance of,

$$\mathbf{\Sigma} = \begin{bmatrix} 381 & -368 \\ -368 & 421 \end{bmatrix} \tag{5.3}$$

resulting in an error ellipse with major and minor axes of 136 m and 28 m, respectively.

Finally, it is useful to simulate (examine) a commercial full navigation solution like

the Kearfott SeaNav -24. Simulation 3 shows this with an additional EKF orientation update that provides the orientation to within 0.001 rad. The results of simulation 4 are shown in Fig. 5.4. The initial uncertainty disappears with the first orientation

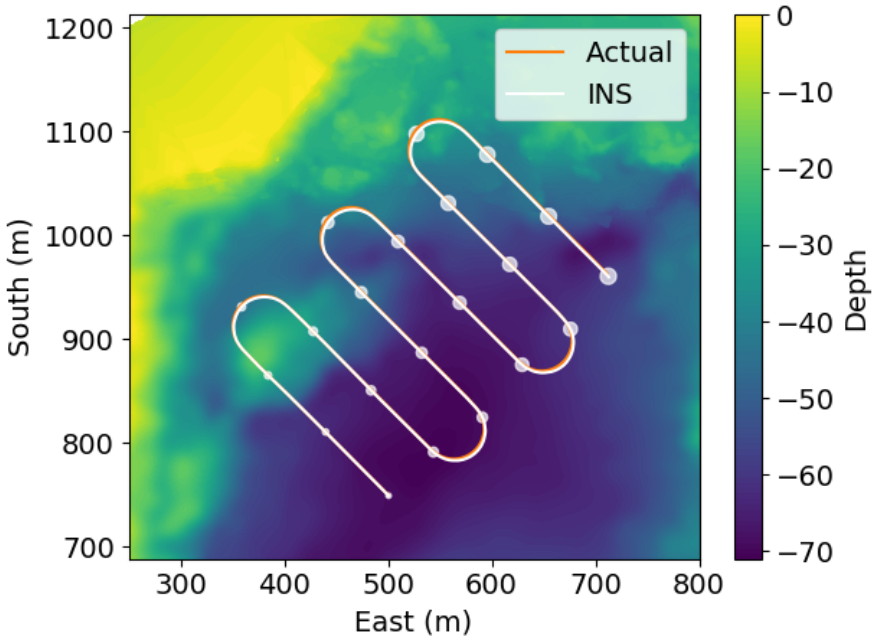

Figure (5.4) Simulation Scenario 3: Simulating the uncertainty characteristics of the Kearfott SeaNav -24

update by the EKF. The way the external orientation update is performed by the Kearfott SeaNav -24 is not important if the error specifications provided by Kearfott are trusted, unless the magnetic readings are unreliable and the unit relies on them for an external orientation reference. It is assumed this is not the case in Bedford Basin. The error ellipse shape, which is characteristic of heading error affecting the position error through reference frame rotation, disappears due to the simulated external orientation reference. Additionally, the position uncertainty grows much slower, due to the lack of heading error growth. The final position uncertainty covariance is,

$$\mathbf{\Sigma} = \begin{bmatrix} 10 & 6 \\ 6 & 10 \end{bmatrix} \tag{5.4}$$

resulting in an error ellipse with major and minor axes of 15 m and 15 m, respectively.

The Kearfott SeaNav -24 states that the circular error probable (CEP) is approximately 0.1% of distance travelled. The search path is approximately 1700 m long; therefore when the expected position error of 1.7 m is converted to the 95% error ellipse of radius 3.5 m, it is within a factor of two from the simulated error ellipse radius of 7.5 m. Therefore, the implemented INS/DVL navigation solution could be used to emulate the Kearfott SeaNav -24 navigation solution.

It was shown that the INS/DVL localization solution is configurable to explore the effects of initial heading error, heading error due to gyroscopic measurement integration, and error due to DVL velocity measurement integration. The position error of the unaided INS remains unbounded. This demonstrates the importance of externally referenced position updates to bound the error growth of the position estimate.

The INS model provides dead-reckoning emulation capabilities of commercially available INSs within the ANT. Next, are sections discussing the results of environment modelling capabilities of the ANT, such as the generation of Gazebo world models, terrain generation, terrain augmentation, and local gravity anomaly derivation.

5.2 Gazebo World Models

The ANT CLI provides the data pipeline to use a georeferenced raster to create a world model compatible with the Gazebo simulator, this was introduced in section 4.2. This ANT CLI tool was used to create the Gazebo world used for the vehicle simulation to collect the bathymetry trace in section 5.4. The resulting model within the Gazebo environment is shown in Fig. 5.5. Fig. 5.5 shows the 5 km by 5 km area from the GEBCO database. The model contains all the configuration files, a 3D mesh, and visual rendering properties that allow Gazebo to render it correctly. The visual rendering is slightly discontinuous in this model because visual mesh smoothing was not added through the Blender API when the model was first created. However, the Bedford Basin Gazebo world shows the full capabilities of the raster to Gazebo world model pipeline.

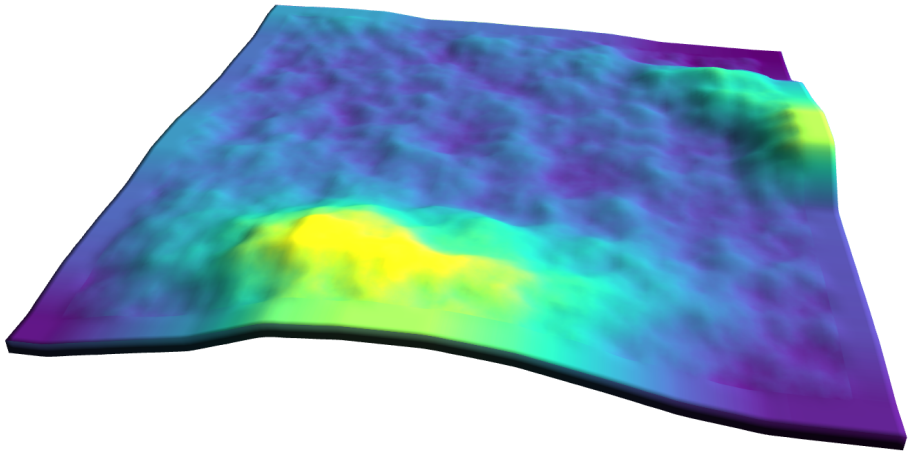

Figure (5.5) The Gazebo world model of the sea mounts first introduced in section 4.1.4.

The Bedford Basin Gazebo world first introduced in section 4.1.2, is shown in Fig. 5.6. Fig. 5.6 shows the rendered visual mesh. This model also includes collision support. The height is exaggerated by a factor of ten for visual effect only in this thesis. This Gazebo world model is useful because of its high resolution (2 m) [107], and its close geographical proximity to the Lab. The Bedford Basin is one of main locations for in-water testing by Halifax researchers.

The Bedford Basin model is already used by other researchers in the ISL for collaborative robotic localization research. This demonstrates the need for a pipeline from real-world raster measurements to the Gazebo world model that is easy to use.

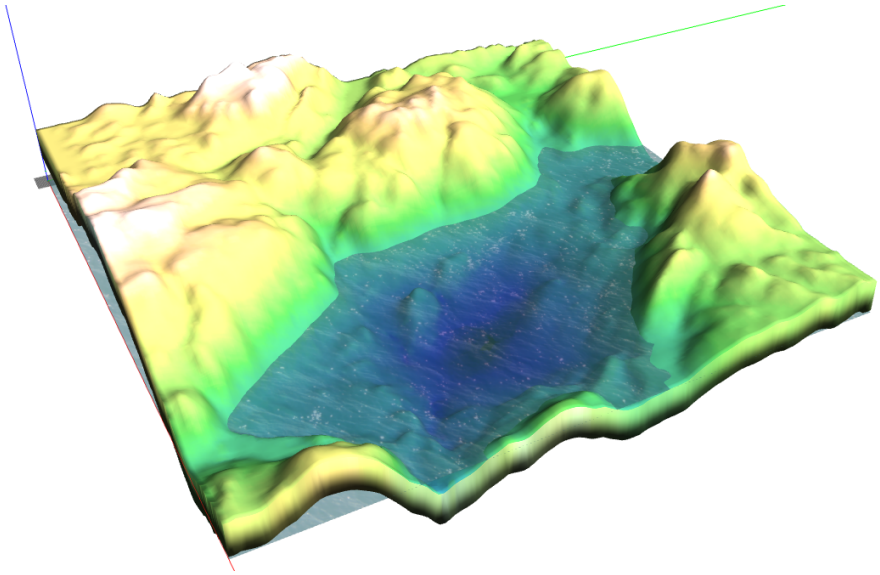

Figure (5.6) The Bedford Basin Gazebo world model. The height is exaggerated by a factor of ten. This helps visualize the height variation. The colour map represents the ocean depth. The model covers a 7 km by 7 km area surrounding the Bedford Basin, Halifax, Nova Scotia, Canada.

5.3 Terrain Generation

Terrain generation was used to generate missing crustal density. Synthetic terrain generation is implemented in the ANT CLI (discussed in section 4.2) and uses the process described in section 4.1.3. The generated density is used within the second parameter study which looks at the effect of the constant density assumption on the performance of the TAN algorithm, which will be discussed in section 5.6.2. An example of the generated density field is shown in Fig. 5.7. The GEBCO bathymetry georeferencing was used to georeference the density to have the two rasters overlap. Both were used to generate the gravity anomaly. Appendix B shows the other generated density rasters used within the density uncertainty parameter study, and contains more examples of generated density.

The limitations of terrain generation ("terrain" is loosely used because it was used to generate density) using Perlin noise are numerous. Perlin noise is usually homogeneous in its variation throughout. This means there is not more or less detail in one area or another. Overcoming this limitation is discussed later in section 6.1. Perlin noise also has less irregular features than real density variation. Density can

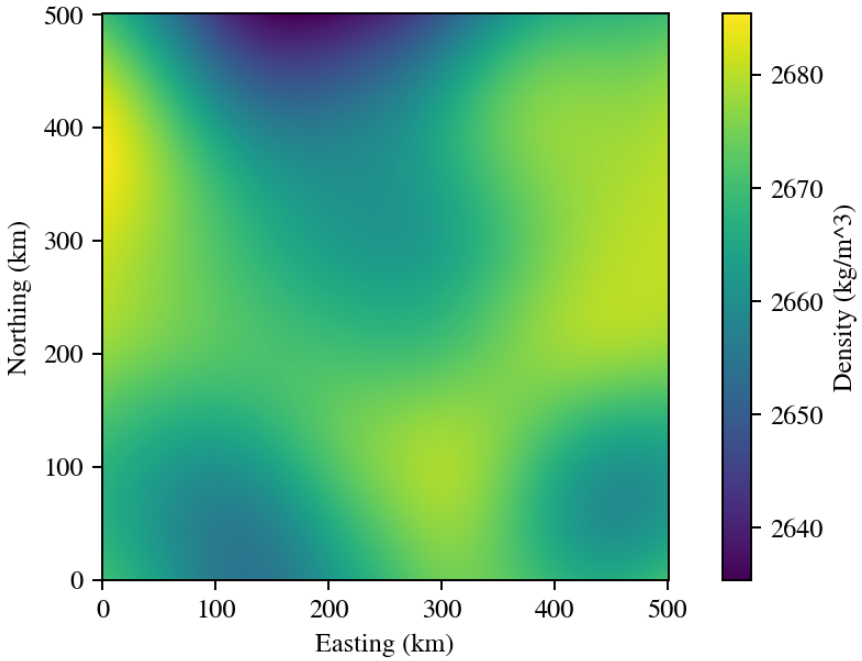

Figure (5.7) Synthetic density variation within the earths crust generated using the Perlin noise. The density is comprised of two Perlin noise octaves, a lacunarity of 2, and a persistence of 0.5. The base period is 500 km, base amplitude is 50 kg/m^3, and mean value is 2670 kg/m^3.

change abruptly and irregularly for a number of reasons, including but not limited to ore deposits, sub-ocean floor oil deposits, and different mineral formations. This is one of the main limitations of using Perlin noise to generate the density variation.

Where measurements are available, it is better to use Perlin noise, or what this thesis refers to as terrain generation, to add missing details to the terrain. This combines the realism and benefits of actual measurements while making up for the missing detail of the available measurements. This is referred to as terrain augmentation.

5.4 Terrain Augmentation

Terrain augmentation of a lower resolution terrain raster does not affect the original terrain measurements when adding noise. It only adds terrain variations in between original measurements. It also maintains the georeferencing that the original raster possessed. The ANT CLI includes this functionality. This was discussed in section 4.1.4. These claims are verified by completing an AUV mission through an ocean environment defined by the raster first introduced in section 4.1.4. The path is shown in Fig. 5.8. The bathymetry encountered during the AUV mission must match the original bathymetry data at all 11 intersections of the grid. This is where the Perlin noise layers must all have a value of zero. The results of this transect are shown in Fig. 5.9. Fig. 5.9 shows that terrain augmentation using Perlin noise layers to create the ocean terrain was successful. The results of the terrain augmentation create more realistic terrain variation between original measurements than linear interpolation —without impacting their integrity. This is shown by the augmented bathymetry trace directly passing through all original bathymetric measurements. The level of variation in between bathymetry measurements in the augmented terrain is completely under the user's control. This demonstrates the power of this method to add visual realism, environmental texture, and environmental uncertainty into a simulation environment.

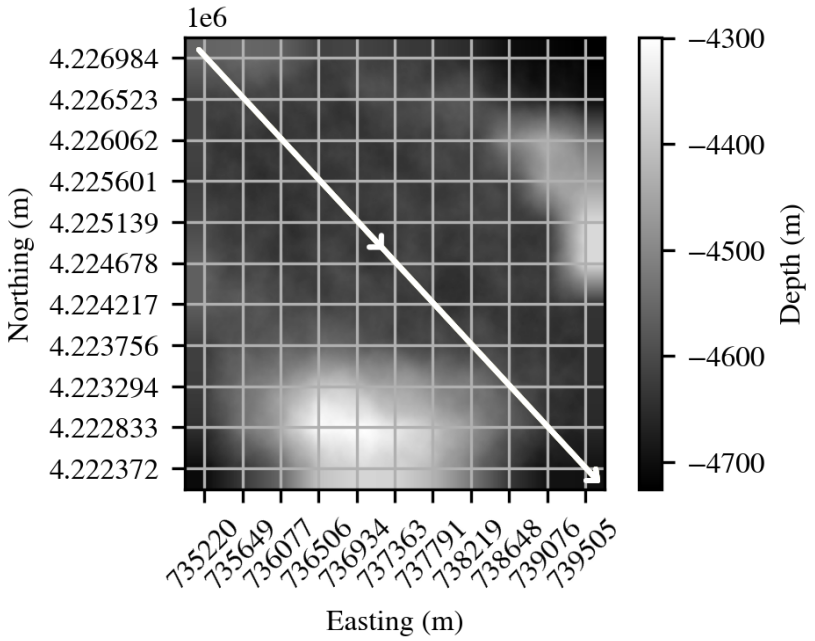

Figure (5.8) Adapted from Heubach and Seto [31]. The white path shows the AUV mission path for which the bathymetry transect depths are recorded. The direction of travel is indicated on the figure (top-left to bottom-right). This trace crosses 11 grid intersections where the original bathymetry measurements should be unchanged.

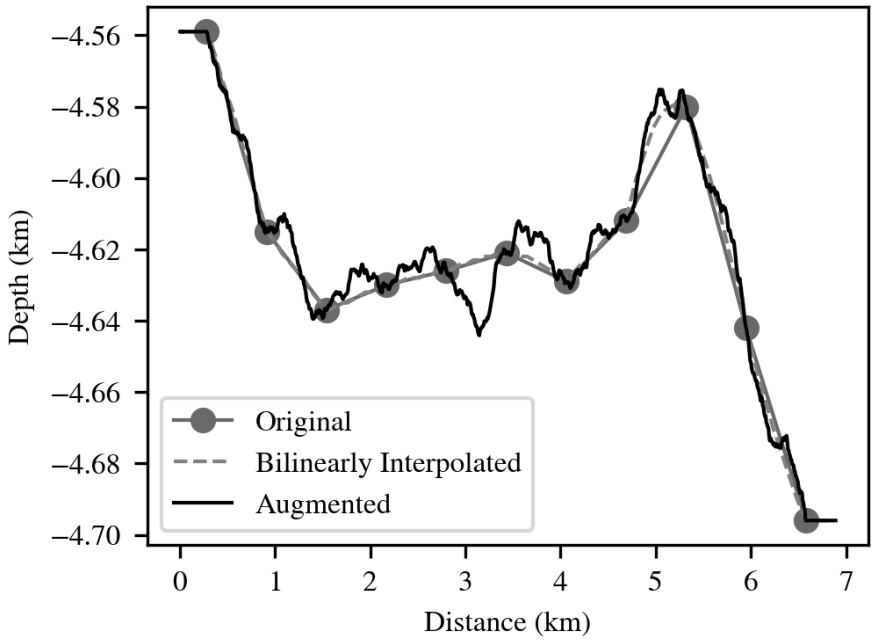

Figure (5.9) Adapted from Heubach and Seto [31]. The bathymetry trace along the AUV vehicle path shown in Fig. 5.8. The ocean depth is on the y-axis, the distance the AUV over-ground is shown on the x-axis. The markers show the original bathymetry measurements from the GEBCO database. The dashed line shows the ocean depth along the AUV travel path if ocean environment were to be generated using bi-linearly interpolation of the original measurements. The dark line shows the ocean depth along the AUV path if the Perlin noise layers are used to augment the original measurements to create the ocean environment.

5.5 Gravity Maps

The correlation of the gravity anomaly with the terrain (Fig. 5.10) and density (Fig. 5.7) enables calculation of the relative gravity anomaly with methods first introduced in section 4.1.5. The resulting relative gravity anomaly is shown in Fig. 5.11.

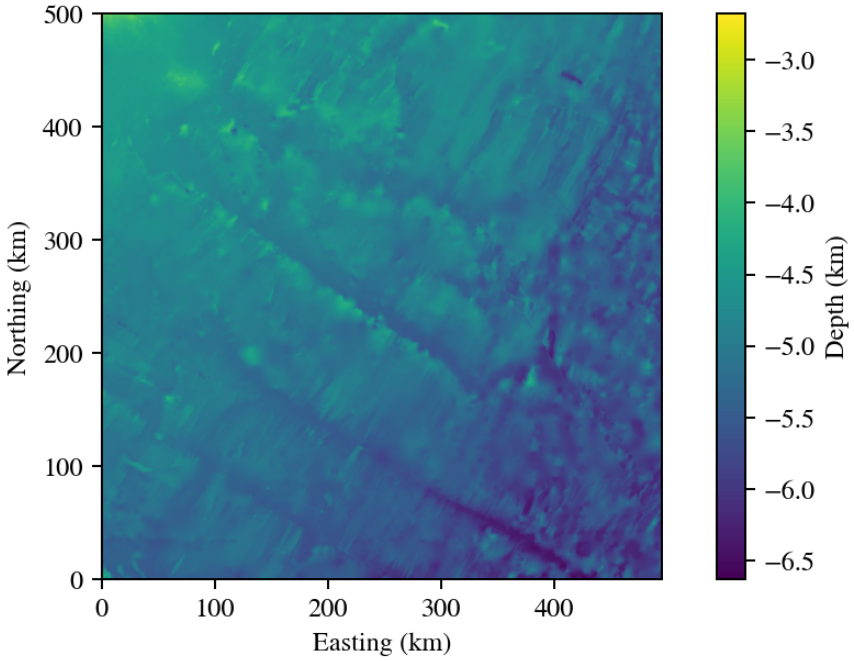

Figure (5.10) An area of ocean to the south of Nova Scotia. The area covers an area of 500 km by 500 km. It is georeferenced using the UTM Zone 20N coordinate reference system which uses the WGS84 datum. The depth ranges from 3 km in the north-west corner close to the continental shelf of North America, to 6.5 km in some of the trenches. This area is used for all of the parameter studies used to explore the effect of uncertainty on the performance of the particle filter based TAN algorithm.

The gravity anomaly variation within Fig. 5.11 is well within the capability of state-of-the-art navigation quality gravimeters (approximately 0.1 mGal [68]). A gravity gradiometer is being simulated for use in the ANT for the TAN algorithm by using the derived gravity gradient field. This is derived from Fig. 5.10, and shown in Fig. 5.12. Fig. 5.12 shows the magnitude of the gravity gradient. The gravity gradient magnitude is well within the abilities of existing gravity gradiometers (1 E [94]). Therefore, the particle filter TAN algorithm using the gravity gradient as positional

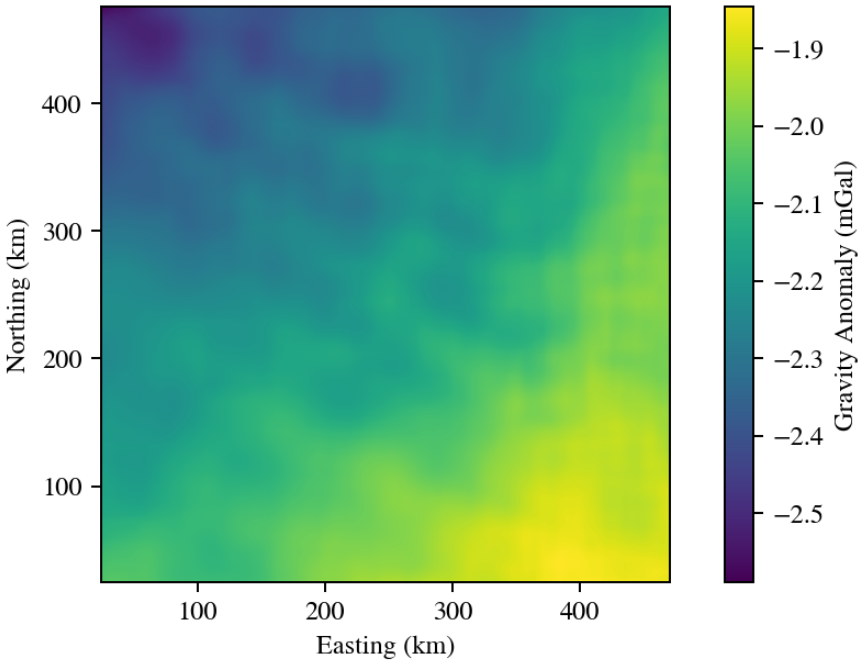

Figure (5.11) The calculated gravity anomaly for the area shown in Fig. 5.10 and the density shown in Fig. 5.7. The gravity anomaly is displayed in mGal (1 mGal = 1×10^{-3}cm/s^2). The gravity anomaly was calculated using the method described in section 4.1.5 using a window size of 50 km.

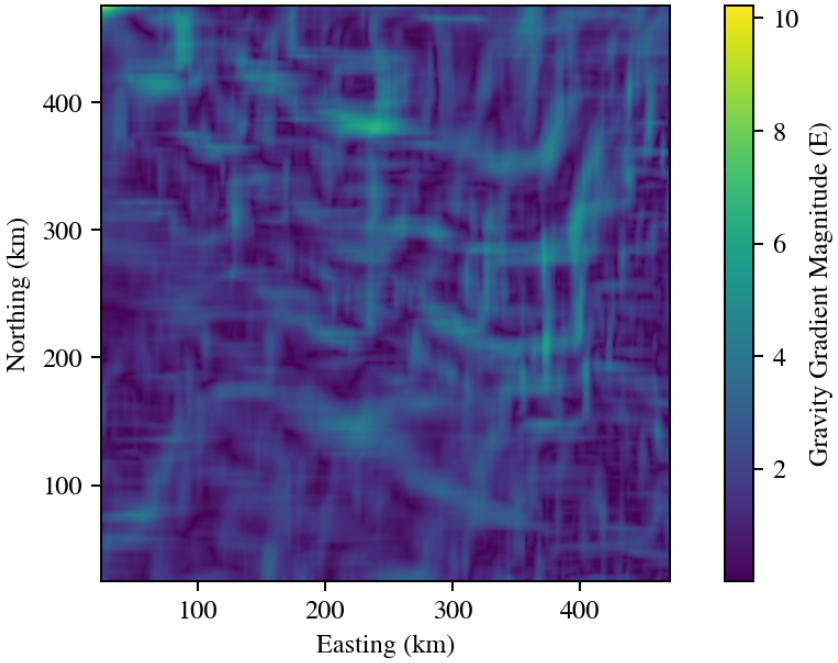

Figure (5.12) The gravity gradient calculated from eq. (2.32). The magnitude of the gravity gradient is on the order of $1\,E$ ($1\,E = 10 \times 10^{-9}\,\mathrm{s}^{-2}$). The area covered is the same as the gravity anomaly in Fig. 5.11.

feedback is within current gravity gradiometer sensor capabilities. This moves the algorithm into the domain of algorithms that could realistically be implemented on an AUV given the platform was large enough to host the gravity gradiometer.

This concludes the results of environmental modelling solution capabilities provides by the ANT. Next, the ANT is used to perform two parameter studies on the performance of the TAN algorithm, compared with dead-reckoning subjected to uncertainty.

5.6 TAN Algorithm

5.6.1 Gradiometer Heading Noise

The first test of particle filter algorithm performance was applying Gaussian noise to the gradiometer's heading measurement. The heading measurement is used by the particle filter algorithm to determine the state estimate along with the INS position. When these two position estimates, INS and particle filter position (TAN position),

are compared they will be referenced by their respective ROS topics, \eca_a9\pose_ins, and \eca_a9\pose_tan.

Table 5.2 shows the first parameter study setup. There are eight sessions each with ten runs. Each session has a unique parameter configuration, and each run has a unique random initialization of initial uncertainties and particle locations. There are a total of eight sessions spanning a range of gradiometer heading noise standard deviations - from 0 rad to 0.7 rad (0° to 40°). A standard noise range for gradiometer heading was not readily available, due to the variation in types of gravity gradiometers [94]. Therefore, the noise range encompasses a vast range of noise settings to be conservative — the noise causes the heading to point orthogonal to the actual heading about 5% of the time with a noise standard deviation of 0.7 rad.

Table (5.2) A summary of the parameter study on the effect of Gaussian gradiometer heading noise on the performance of the particle filter algorithm. Each session has ten runs with random initialization for particle filter, and INS localization and heading errors.

Session	σ (rad)	Runs
0	0	10
1	0.1	10
2	0.2	10
3	0.3	10
4	0.4	10
5	0.5	10
6	0.6	10
7	0.7	10

The random initialization of particles uses a uniform distribution, even though the INS estimate is normally distributed. The circular uniform distribution covers an area defined by three times the standard deviation of the normal distribution of the INS estimate. This almost guarantees (99.7%) the vehicle is within the initial area covered by particles. The uniform distribution is used to maximize coverage, and refrain from unnecessarily biasing the particle initialization. Additionally, uniform distributions were used to initialize the actual error of the INS. This maximizes coverage of error cases. This is distinct from the uncertainty estimate that the INS uses, which is normally distributed.

This study includes a total of 80 runs. Approximately nine ended in particle degeneracy within the particle filter's localization estimate. Six of these degenerate cases occurred in the zero noise session. All other sessions had one or less degenerate cases within the ten runs of each session. The zero noise case degeneracy occurred because the particle filter requires stabilizing noise, or the illusion of stabilizing noise, to remain stable. The first run was especially affected because the standard deviation of the particle filter's probability distributions, from which samples were evaluated, where set to approximately match the value of the gradiometer's heading standard deviation. For the first session this was approximately zero. This causes the particle filter to converge prematurely, and become degenerate. The position estimate then degenerates to produce NaN's which means all the particles collapse onto one position. The first session is not representative of using the particle filter in practice where the uncertainty is certainly not close to zero. It was performed as a baseline.

The environment used for this study is an approximately 500 km by 500 km area of the south-east coast of Nova Scotia, Canada (Fig. 5.13). The top-left corner of the area just touches the continental shelf off the coast of Nova Scotia. The area was chosen because it contains bathymetric features, however, the shape and size of the features is not unique within the area. The raster defining this area is from the GEBCO database. The resolution of the bathymetry is approximately 460 m.

Fig. 5.13 shows the XY paths of the INS position estimate. The paths are shown as on overlay on a colour map of the ocean bathymetry. The paths show the scale of the parameter study. The vehicle paths start in the lower-left hand corner and travel to the upper-right hand corner during the run. The paths begin to spread out slightly (not visible given the scale of the plot). This is due to the random initialization of the INS position error. This position error has a spread of about 4 km. This is shown in more detail in Fig. 5.14. This is not realistic, however, testing outside the particle filter localization algorithm determines performance under non-nominal conditions.

The initial INS error is sampled from a uniform distribution, previously justified, hence the spread at the beginning (bottom left) Fig. 5.13. The INS estimate paths in Fig. 5.13 converge at the top-right because the INS position estimate is used by the

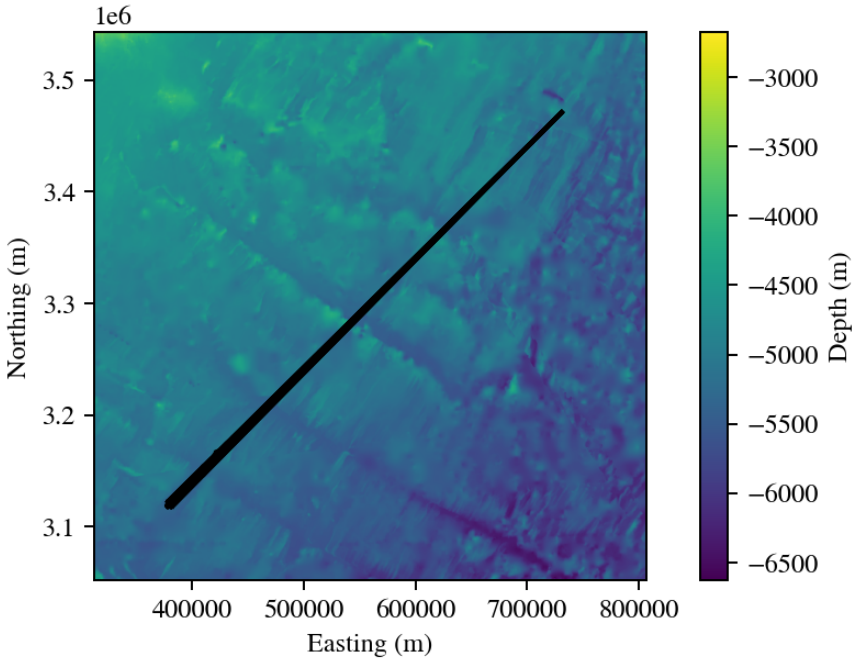

Figure (5.13) All eighty XY paths of the vehicle's INS position estimate are overlaid on the colour map of the ocean bathymetry raster. The initial position spread is about 4 km and shown in more detail in Fig. 5.14. The initial heading error of about 0.1 radians (6 deg) and shown in more detail in Fig. 5.16. Time elapses from the bottom-left point to the top-right.

vehicle waypoint controller.

The initial INS position estimates are shown in Fig. 5.14. Each point in the distributions is at approximately uniform range from the actual vehicle position 69 km northing, 69 km easting. The uniform distribution is coverage of initial error that the INS has, this is not to be confused with the initial position uncertainty of the INS which is Gaussian distributed. The spread of the initial INS error is approximately 3 km. This matches and validates the initialization procedure which randomly initializes the INS position error from two uniform distributions along the northing and easting axes. Each uniform distribution has a range from -3 km to +3 km matching the results seen in Fig. 5.14. The initial Euclidean error of the INS position estimate is randomlt initialized as shown in Fig. 5.15.

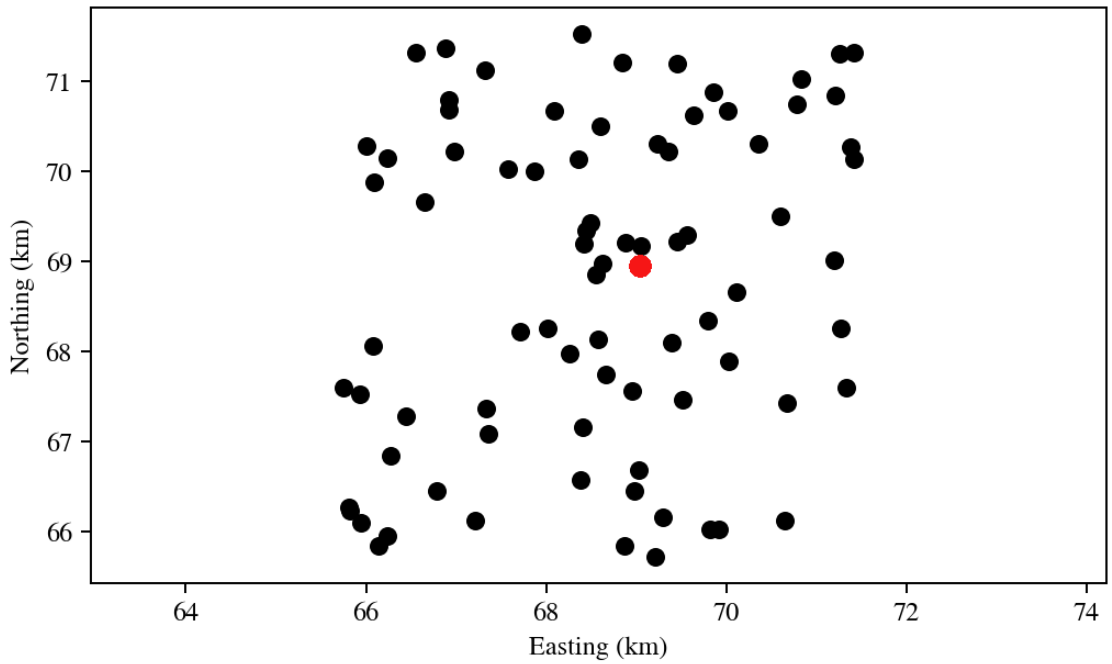

Figure (5.14) The randomly initialized positions of the INS. The actual starting position of the vehicle is 69 km northing, 69 km easting, shown by the red dot. The spread is approximately 4 km.

Fig. 5.15 shows the initial distribution of the INS position estimate error for each session is evenly distributed enough that the runs should capture the full range of possible starting position error conditions for each gradiometer noise setting.

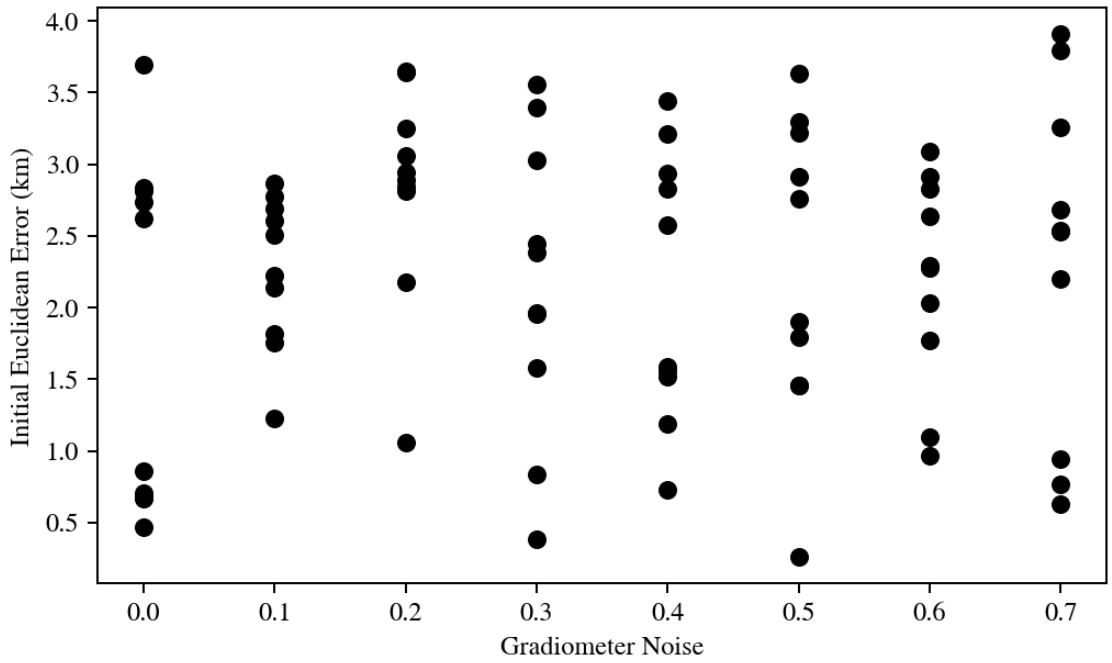

Figure (5.15) The randomly initialized initial INS Euclidean position errors for each session of ten runs with the same gradiometer noise setting. The error range is approximately 4 km The spread is approximately uniform.

Fig. 5.16 shows the spread of the INS initial heading estimate error. The distribution of the initial heading error for each gradiometer noise setting is uniformly distributed over a range of 0.2 rad (11°), from −0.1 rad to 0.1 rad. The Euclidean and heading error s are both evenly distributed over a reasonable range to give confidence in the particle filter performance results when compared against the dead-reckoning only (INS).

For each session's worth of multiple runs with identical gradiometer noise setting, the error distribution of the particle filter and the dead-reckoned position estimate is compared. A representative figure was chosen with a gradiometer noise standard deviation that is in the middle of the study's dynamic range. Session two is highlighted here as a typical case. The rest of the results for the gradiometer noise study are in appendix A. Session two compares the particle filter and dead-reckoned position estimate performance with a gradiometer heading noise of 0.2 rad (11°). Fig. 5.17 contains all runs that did not have problems with degeneracy.

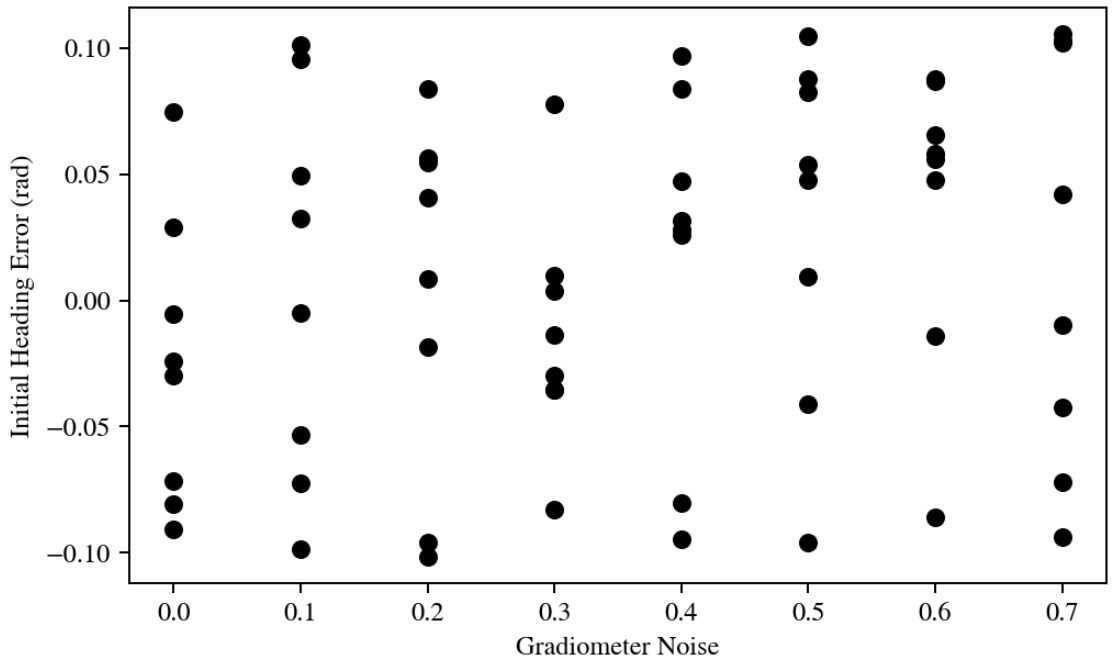

Figure (5.16) The random initialized initial INS heading errors for each session of ten runs with the same gradiometer noise setting. The error range is approximately 0.1 radians (6°). The spread is approximately uniform across the heading error range.

The degeneracy was a problem for the first session. Six of the ten runs ended in degeneracy. This is due to the first session having a gradiometer heading noise standard deviation of zero and the particle filter having an artificially low 0.001 rad noise standard deviation. This standard deviation is used to determine the distributions that particles are weighted and sampled from. At very low standard deviations this leads to instability in the particle filter and the particle hypotheses degenerate into a single estimate. This is prevented by keeping the particle filter probability distributions large enough to avoid degeneracy. Additional noise stabilization techniques are another way to avoid this.

Of the other runs within the session, half had one degenerate case, the rest had none. This shows the first session is exceptional and does not contribute much insight into how uncertainty affects the AUV position estimate. It does the opposite, showing that artificially not capturing uncertainty at all is one of the reasons for particle degeneracy. The particle filter must represent the uncertainty within the system and artificially reducing uncertainty leads to a worse or degenerate position estimate.

From Fig. 5.17 it is clear the INS position estimate error is unbounded in time. The particle filter that implements TAN shows that once position information, in the form of the gradiometer heading direction derived from terrain maps, is incorporated into the position estimate the position estimate error is bounded. The difference is large enough as highlighted in Fig. 5.18 which shows just the TAN position estimate error.

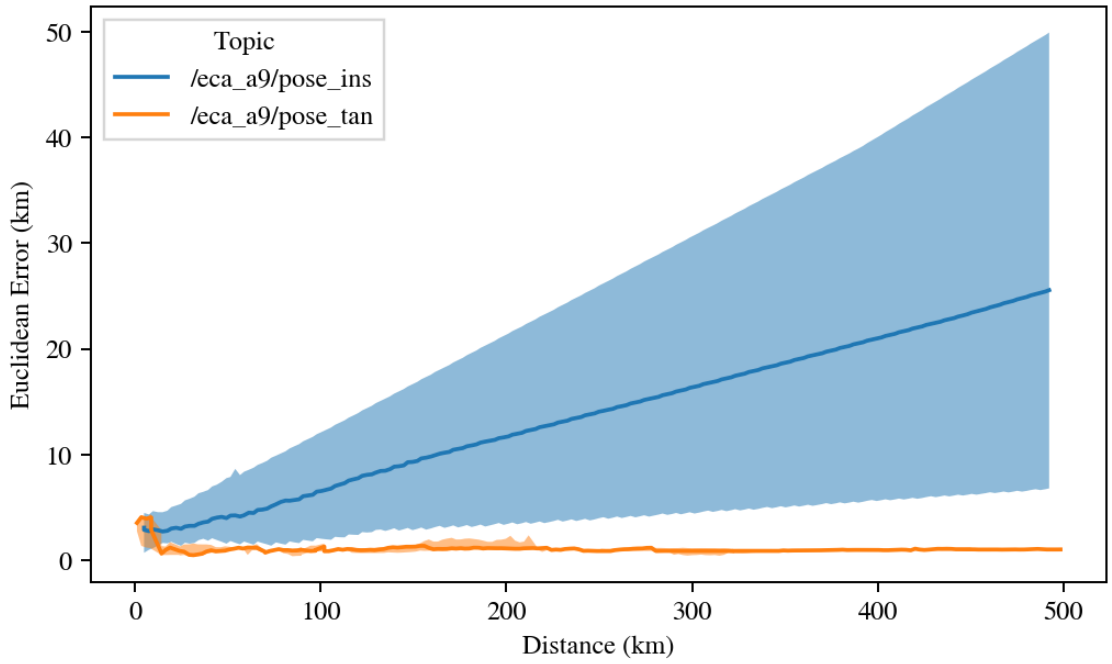

Figure (5.17) Comparison of the Euclidean error of the INS and particle filter based localization solutions. The comparison uses a representative session, session 2. The gradiometer noise standard deviation for session 2 is 0.2 rad (11°). One run of the ten was excluded in this plot because the particles during that run degenerated enough to cause the position estimate to become not a number about half way through. The dark line represents the median case for collection of runs, the filled area the total spread. The particle filter spread is hard to visualize in the comparison plot so it is plotted on its own in Fig. 5.18.

Fig. 5.18 shows that the particle filter corrects for the initial position error in the INS which is approximately 10 km. Just after, it maintains an error of approximately 1 km for the duration of the transit by the AUV. There is significant variation between runs as seen by the shaded area covering all runs within this session. Some of this variation is due to the particle filter estimate being based on random sampling and

stabilizing noise. So some variation between runs is expected. Additionally, the variation in the position estimate error are greater during some parts of the journey than others. This is due to the underlying limitation of using the terrain as the basis of the gradiometer heading estimate. During a mission there will be terrain sections where there is less terrain variation which may lead to sections of larger position error versus lesser variation later in the mission.

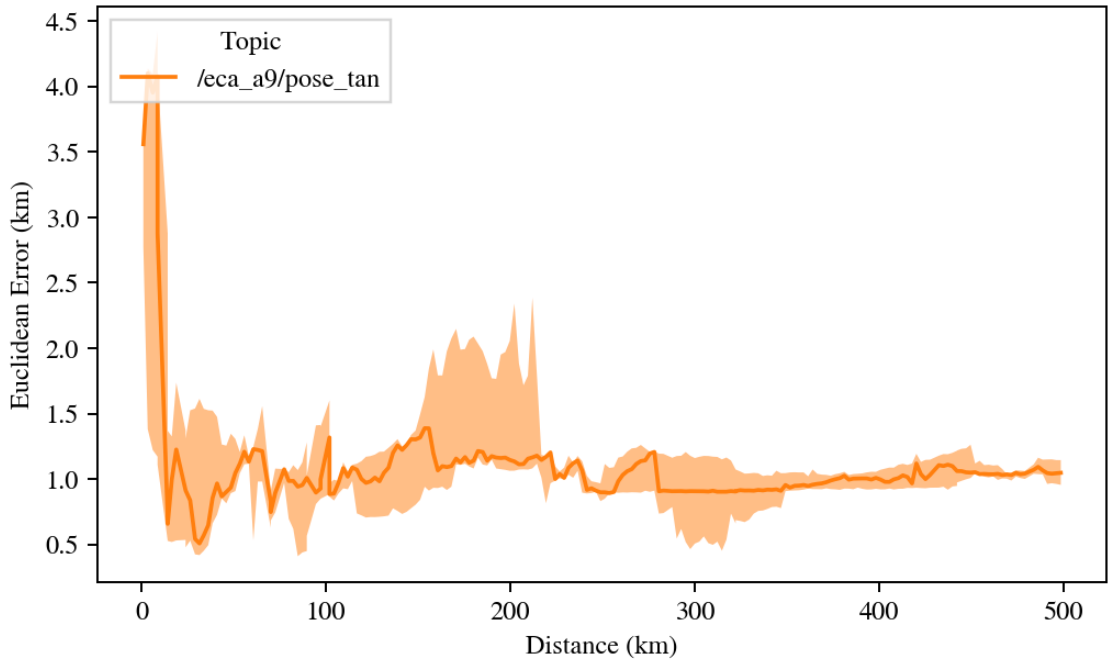

Figure (5.18) The particle filter based localization performance for session 2. Session 2 has a standard deviation of 0.2 rad (11°). The dark line represents the median Euclidean error, the shaded area encompasses the entire spread of all runs within the session. The particle filter performance is compared based on distance travelled by the vehicle.

Additionally, Fig. 5.18 shows the median error is around 1 km. This reflects the limitation of using a map with a limited resolution of 1 km. This means any particle position hypotheses that are closer than 1 km will be equally probable and consequently given the same particle weight. Therefore, the estimate convergence stagnates.

When comparing the INS positioning performance with the particle filter using the median, the particle filter out-performs the INS estimate by a factor of 25 over

the 500 km mission — from 25 km median position error to 1 km position error. Of course this is not quite a fair comparison because the INS estimate does not contain position feedback and therefore has unbounded error growth. Extending this mission to an arbitrary length would increase this factor. This shows how important it is to have a source of position feedback in the vehicle state estimate. There may be more to be learned from the difference between sessions within the study.

When comparing successive session's results (gradiometer noise increase from session 0 to session 7) (Figs. A.1 to A.16 are found in appendix A) it is apparent that the higher the gradiometer heading noise the larger and more frequent the fluctuations of the particle filter estimate. Higher gradiometer heading noise causes the particle filter estimate to vary more than it otherwise would. Additionally, the variation of the particle filter estimate grows with higher noise. This is expected, but particle filter median performance hovers around the 1 km error. This leads to the conclusion that within this study the noise limit causing the particle filter to consistently diverge was not reached. This is a limitation of this study.

Another limitation is the uncertainty from the constant density assumption. Leading to the next study. A study that is more representative of a scenario where the vehicle has access to *a priori* gravity gradient maps from actual measurements of the gravity gradient (similar to the work in [28]) rather than a gravity gradient map derived from the bathymetry making the assumption crustal density is constant. Using the constant seafloor density assumption to generate the *a priori* map and the environment model ignores the uncertainty resulting from the non-constant seafloor density in the real environment.

5.6.2 Seafloor Density Uncertainty

The density uncertainty parameter study explores how the assumption of constant seafloor density affects the performance of the particle filter (TAN) algorithm. This was done by varying the amplitude of density variations as input to the environmental gravity anomaly model. The goal of this study is to test the sensitivity of the particle filter algorithm to varying levels of density variations within the Earth's crust when making the assumption of constant density. This assumption is often required due to

a realistic lack of detailed density information about the Earth's crust.

The particle initialization and INS error initialization are kept consistent with methods used in the previous study (section 5.6.1).

The parameter study contains 50 runs. The density variation amplitude is varied logarithmically from $50\,\mathrm{kg\,m^{-3}}$ to $800\,\mathrm{kg\,m^{-3}}$. This is an overly inclusive range that is large enough to make conclusions about the effect of the constant density assumption on the performance of the TAN algorithm. The density amplitude is used as input to the method used for the generation of synthetic density, discussed in section 4.1.3 and section 5.3. The gravity anomaly field within the simulator was derived from the density field, and the GEBCO bathymetry. The *a priori* maps the AUV particle filter (TAN) algorithm has access to only use the GEBCO bathymetry and assume constant density. The greater the density amplitude variation the greater the difference between the maps used for navigation and the environment. The are 10 runs per session, where each session represents one parameter setting (density amplitude is a single value). There are a total of 5 sessions, the summary for which is shown in table 5.3.

Table (5.3) A summary of the parameter study on the effect of density amplitude variations on the performance of the gradiometer aided localization.

Session	Density Amplitude (kg/m^3)	Runs
0	50	10
1	100	10
2	200	10
3	400	10
4	800	10

The environment used for this study is the same as the gradiometer heading noise study, shown in Fig. 5.13. The path travelled by the vehicles are similar. For each run the vehicle was spawned in the environment in the same location. Additionally, the INS was initialized with a pseudo-random initial positional error, and heading error. The amount of error is unrealistic at approximately 4 km in diameter. However, this amount of error is used to test whether the particle filter algorithm successfully

converges given a higher degree of uncertainty about the initial position of the vehicle. The distribution of initial INS starting positions is shown in Fig. 5.19. Fig. 5.19 shows

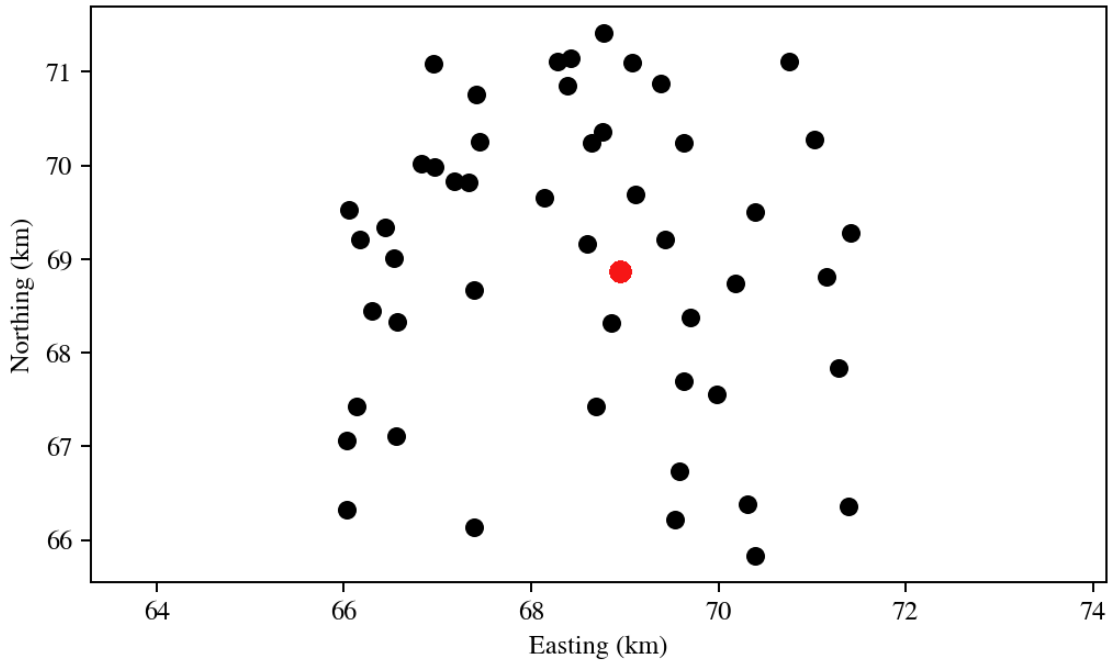

Figure (5.19) The randomly initialized positions of the INS. The actual starting position of the vehicle is 69 km northing, 69 km easting, shown by the red dot. The spread is approximately 8 km.

the spread is approximately uniform and is about 8 km. This provides confidence that the study's initial error conditions provides realistic coverage of possible position error conditions in the INS. The INS initial position Euclidean error is shown per session in Fig. 5.20. Fig. 5.20 shows that the distribution of initial position Euclidean error is approximately uniform for each session. The provides confidence that the runs in each session, which will be compared, have coverage of most possible dead-reckoning INS initial error conditions. The spread of initial heading error is especially important because over long distances small heading errors lead to large position estimate errors.

Fig. 5.21 shows the distribution of absolute heading errors in the initial dead-reckoning (INS) heading estimate. Again the initial heading error is approximately evenly distributed across the range of -0.1 rad to +0.1 rad for each session. The

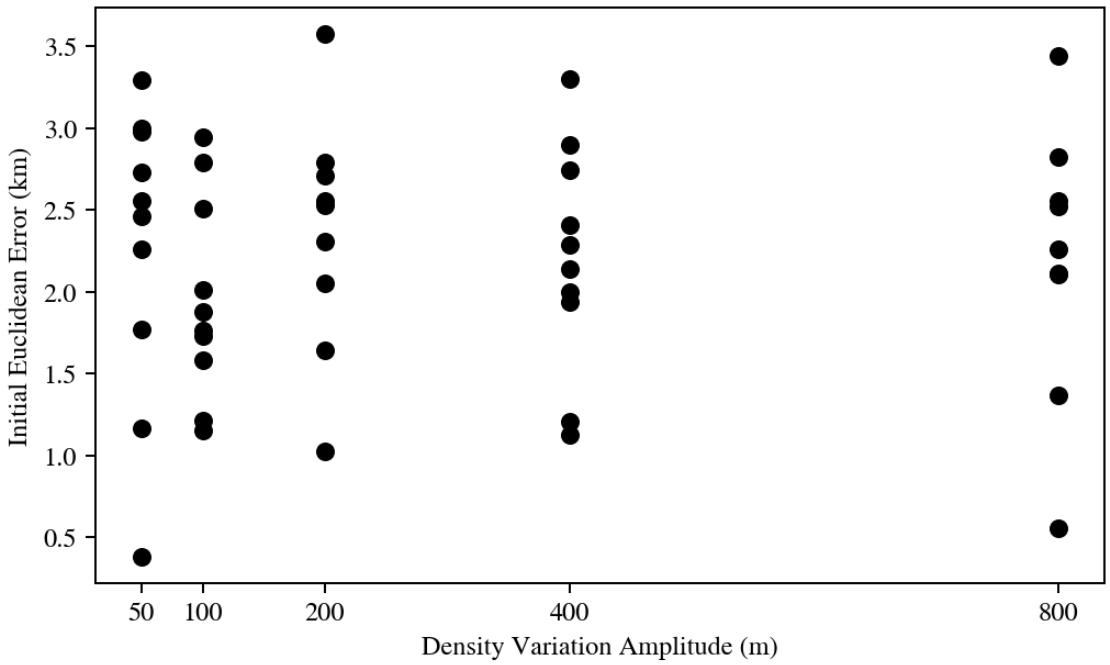

Figure (5.20) The randomly initialized initial INS euclidean position errors for each session. The absolute error range is approximately 4 km. The spread is approximately uniform.

coverage of initial error conditions for the INS is approximately evenly distributed for positional and heading error. This observation holds across sessions.

The particle filter (TAN) algorithm position error estimate is compared with dead-reckoning performance in Fig. 5.22. Session 2 was chosen as a representative example of the performance comparison. For each run the particle filter was pseudo-randomly initialized. The gravity gradiometer sensor heading noise standard deviation was set to a constant 0.2 rad for all sessions. Fig. 5.22 shows the median accumulated position error during the 500 km mission is approximately 30 km. This compares with median error of 1 km for the particle filter TAN algorithm. The particle filter successfully converges within 30 km, then varies around the position error of 1 km. This is due to the limitation of the resolution of the map used for navigation. The re-sampled resolution of the gravity gradient map used by the AUV is 1 km. Since the gravity gradiometer measurement is compared to the gravity gradient map, any measurement within 1 km use the same gravity gradient map value to weight the

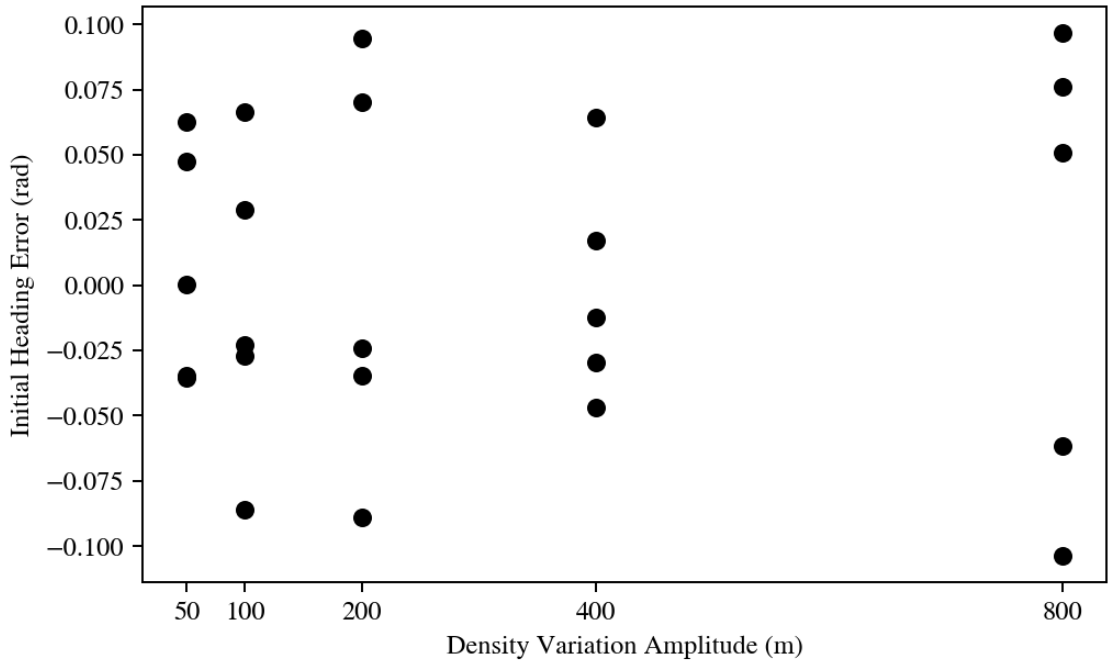

Figure (5.21) The random initialized initial INS heading errors for each session of ten runs with the same gradiometer noise setting. The error range is approximately 0.2 radians (12°). The spread is approximately uniform across the heading error range.

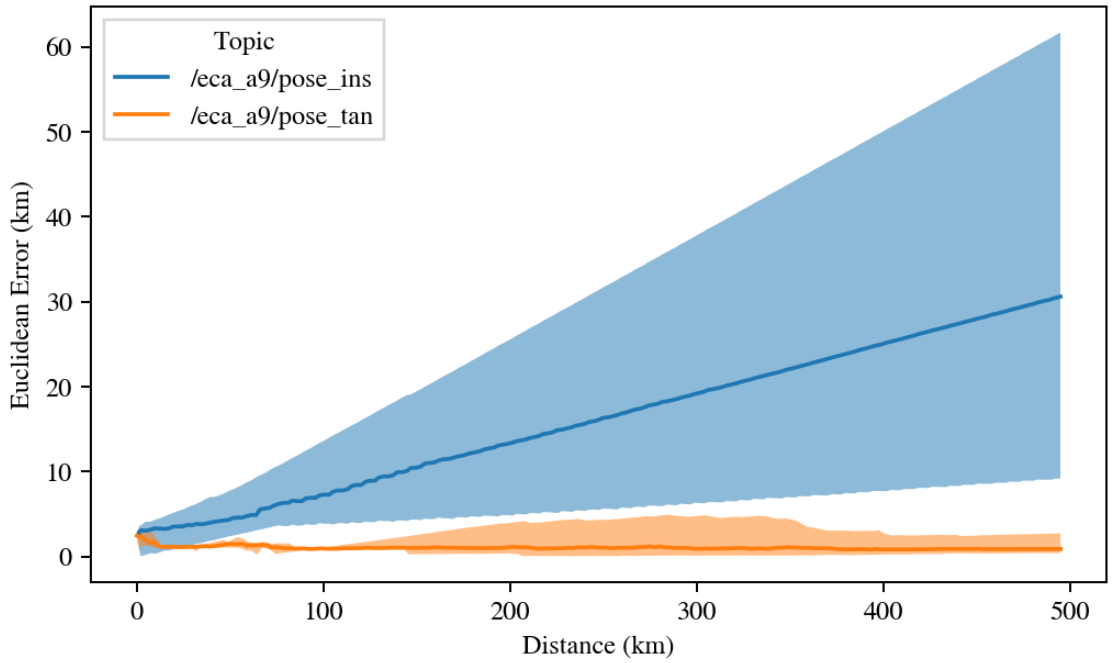

Figure (5.22) Comparison of the Euclidean error of the INS and particle filter based localization solutions. The comparison uses a representative session, session 2. The gradiometer noise standard deviation for session 2 is 0.2 rad (11°). The dark line represents the median case for collection of runs, the filled area the total spread. The particle filter spread is hard to visualize in the comparison plot so it is plotted on its own in Fig. 5.23.

particles, this effectively sets a hard limit on the accuracy of the position estimate of the particle filter.

The zoomed in particle filter performance of session 2 is shown in Fig. 5.23. Fig. 5.23 shows that the particle filter algorithm performance varies significantly. One

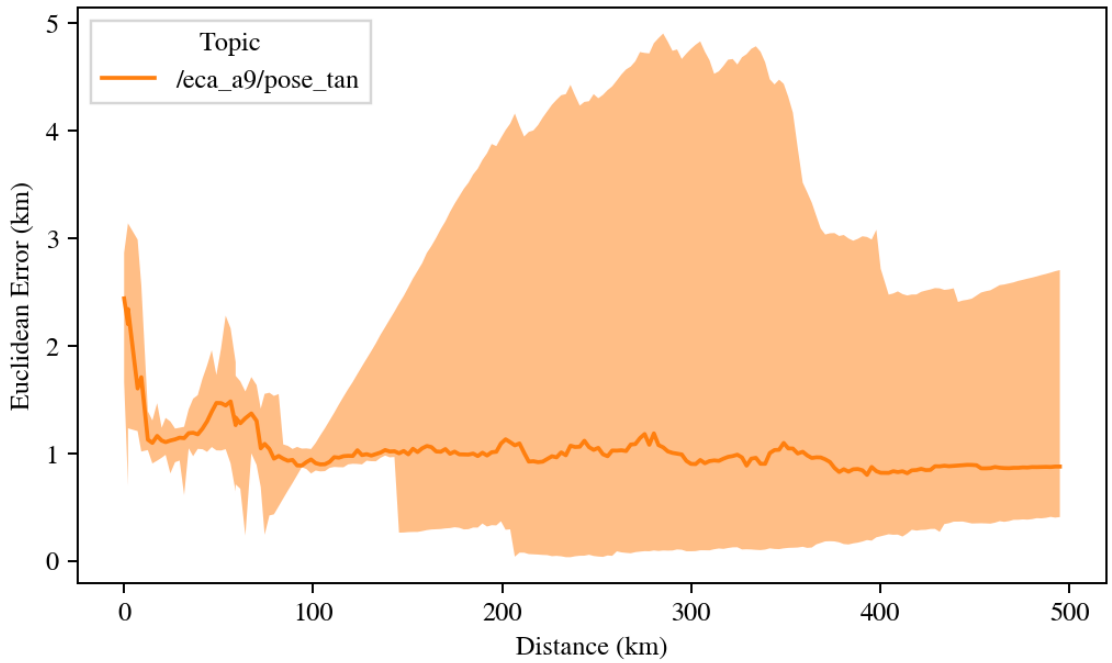

Figure (5.23) Only the particle filter positional Euclidean error during session 2. Session 2 had a density amplitude variation of 200 kg m^{-3}. The gravity gradiometer heading noise has a standard deviation of 0.2 rad. The dark line is the median and the shaded area is envelope for the maximum and minimum. The particle filter estimate converges within 30 km and then varies around 1 km error.

run diverges around 100 km, shown by the shaded area. This shows there is room for improvement for how robust the particle filter algorithm is. There are similar cases within other sessions within this study.

When comparing sessions within this study with increase density amplitude (included in appendix B) there is an increase of median performance roughness when increasing the density amplitude variation. However, even with very high density variation the particle filter still converges and stays converged for the majority of the runs within each session. This is due to the difference between navigating using the

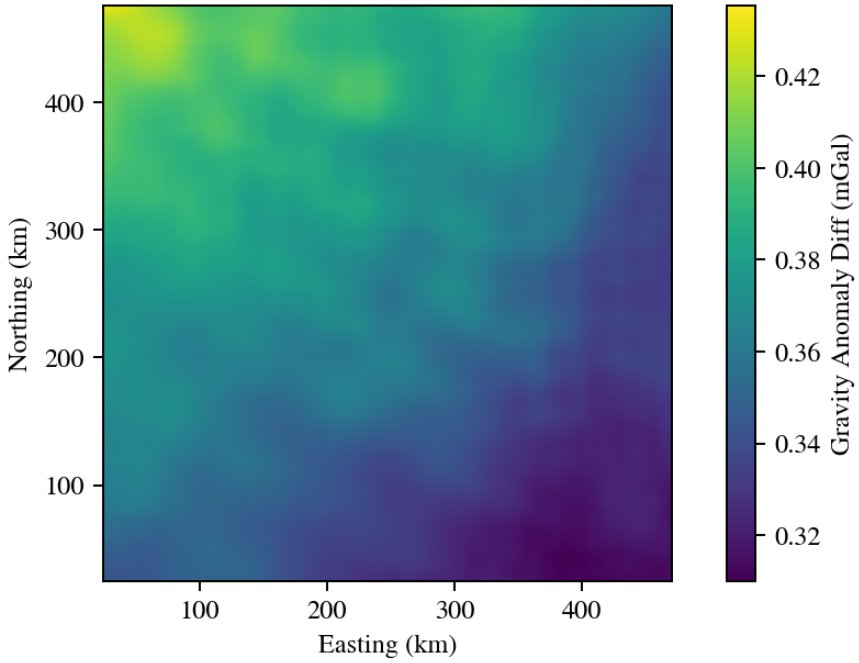

Figure (5.24) The difference in the gravity anomaly that the constant density assumption makes. The gravity anomaly difference has units of mGal ($1\,\text{Gal} = 1\,\text{cm/s}^2$).

absolute gravity anomaly and the gravity gradiometer heading. The magnitude of the gravity anomaly may change significantly with changes of density (shown in Fig. 5.24), but the direction of gradient is much more likely to remain unchanged.

This study shows that the TAN algorithm is quite robust when subjected to uncertainties in the environmental density by making the constant density assumption. The TAN algorithm successfully converges and maintains a position estimate 30 times more accurate than the dead-reckoned position estimate over a distance of 500 km. The TAN algorithm limits the error growth experienced by dead-reckoning by successfully incorporating the gravity gradient heading into the state estimate as position feedback.

There are other forms of density variation that have not been explored in this study. This include the level of detail in the density field, the roughness, and discontinuities. These are limitations of this parameter study, and should be addressed in future work. However, this study gives a basis to the rationale of assuming constant

density when using the gravity gradient for navigation.

This concludes the results chapter. The results demonstrate the ability of the INS model to emulate various configurations of a commercially available navigation-grade INS. The simulations used to demonstrate the capabilities of the INS model are performed in the Bedford Basin Gazebo model. This further justifies its development. The unbounded error growth shows the necessity of position feedback. For this the TAN algorithm was proposed.

Next, environment modelling results were discussed. Terrain generation was used to create a synthetic density field to use in the second parameter study of the TAN algorithm. Terrain augmentation was not used within this thesis, however, its potential to be of use to other researchers [31] justifies its existence.

Finally, the TAN algorithm was tested in the ANT. The first parameter study of the TAN algorithm demonstrated the particle filter was robust to gravity gradiometer heading sensor noise. The particle filter converged quickly and stayed converged throughout the AUV mission. The particle filter successfully limited the error growth of the INS, and out-performed dead-reckoning by a factor of 30 —limited only by the gravity gradient map resolution. Additionally, the second parameter study demonstrated that the constant density assumption used when using bathymetry as an input to gravity gradiometer based TAN is justified. Density amplitude variation was shown to have little effect on the performance of the particle filter algorithm. This demonstrates that bathymetry could successfully be used as a replacement for maps of the gravity anomaly for GAN, as mentioned by [82], and used in [43].

Chapter 6

Conclusion

This thesis describes the process of developing a TAN algorithm that is passive, resistant to interference, limits dead-reckoning error growth, and uses readily-available data. To achieve this objective a collection of tools were developed, referred to as the ANT, to support low-risking testing of the TAN algorithm using the ROS and Gazebo simulator ecosystem. Additionally, the development of DCAF enables eventual vehicle integration and in-water testing of the TAN. The TAN algorithm uses a particle filter based state estimation algorithm, similar to work by Pasnani and Seto [28], but using the correlation between the bathymetry and the gravity anomaly to use the gravity gradient as position feedback. This exploits the increased availability of *a priori* bathymetry maps which are also of higher resolution. The proposed approach also enables the possibility of using both gravity anomaly maps and bathymetry maps together and fuse the position feedback within the particle filter state estimate to increase the localization accuracy even further. This is left to future work.

The TAN algorithm performance was compared to dead-reckoning. The INS model that implements dead-reckoning was covered in detail in section 4.1.1. The INS error properties were chosen to simulate the error properties of current state-of-the-art INS. The desired error characteristics where successfully demonstrated in section 5.1. The INS model was shown to successfully capture difference in error growth between the cross-track and along-track error position. The along-track error grows from the integration of acceleration, whereas the cross-track error grows from the integration of acceleration and angular velocity. The cross-track error was shown to grow faster than the along-track error, as expected in section 5.1.

The particle filter (TAN) algorithm limits the unbounded growth due to dead-reckoning and out performs dead-reckoning by a factor of 25 over a 500 km AUV mission. The particle filter algorithm is robust against gravity gradiometer heading noise, converging with heading noise of up to 0.7 rad (40°), demonstrated in section 5.6.1. Less than 10% of the particle filter runs showed the beginning of divergence during the first half of the AUV mission, with the majority converging again during the second half of the journey. Even during the significant increase in localization error during this partial divergence the particle filter algorithm was able to recover successfully. This demonstrated the robustness of the particle filter algorithm.

Terrain generation was then used to design density data for use in testing the assumption of constant density when deriving the gravity anomaly from the bathymetry. In section 5.6.2, it was shown that varying the density amplitude had little effect on the performance of the TAN algorithm. The TAN algorithm still converged within 15 km and maintained a median localization accuracy of approximately 1 km. This shows support for the assumption of constant density often made when using the correlation between terrain and gravity anomaly for GAN ([43, 92]).

All parameter studies discussed during this thesis where performed using the ROS parameter study (RPS) tool. RPS enabled parameter studies using an existing ROS based implementation of the TAN algorithm and the Gazebo simulator. The RPS tool also made working with the data output from ROS and Gazebo simulator possible using the widely used Python language. This tool did not exist within the ROS package ecosystem and will be contributed back to the research community to allow others to benefit from the work in this thesis.

The ANT CLI packages the tools that enabled the work in this thesis. This includes terrain generation, terrain augmentation, derivation of the gravity anomaly using the correlation between the terrain and the gravity and anomaly, and a pipeline from a raster data set to a Gazebo world model. The tools enabled the research in this thesis and have been open-sourced and released to the research community as part of the presentation of the work in this thesis at the OCEANS 2021 conference in San Diego. The ANT is already being used by other researchers within the ISL.

The work in this thesis demonstrated four main contributions. First, a TAN algorithm that is passive, resistant to interference, limits dead-reckoning error growth, and uses readily-available measurements, Second, an INS model implementation that has error growth properties of current state-of-the-art INS and can be used for simulating AUV dead-reckoning. Third, the development of a set of tools (the ANT, and RPS) to enable and ease underwater AUV research, especially in the field of TAN.

Each contribution worked towards the goal of AUV localization accuracy. Localization accuracy is fundamental to the challenge of underwater navigation and exploration. The difficulty of underwater localization comes from the underwater

domain, which limits communication speed and bandwidth due to the nature of the acoustic signal medium, as radio frequencies experience high attenuation, making them impractical. A navigation solution that relies on above ground, GNSS, is not applicable for underwater operations unless frequent vehicle surfacing is allowed by the mission. Therefore, during underwater operation, vehicles rely on dead-reckoning, which has unbounded position error growth.

TAN is one of the solutions to the localization problem for the underwater domain, particularly useful for long-distance AUV navigation. After all, AUVss are a platform for further underwater sensing, making it possible to explore an environment that is still relatively unexplored. Higher accuracy localization is a fundamental enabler for this.

6.1 Future Work

The next steps for the work presented in this thesis is vehicle integration. There was significant progress in the development of DCAF, a ROS based autonomy framework for the IVER 3. The DCAF project is in the stage of on-vehicle testing. Once the limitations and bugs within DCAF have been identified and work to eliminate them completed, DCAF would be ready for testing the TAN algorithm.

First, it is recommended that the TAN algorithm implemented on the vehicle should use feedback from the bathymetry, using sensors such as the on-board altimeter. Current gravity gradiometers do not fit on vehicle's the size of the IVER 3. Once the particle filter is proven to be robust under realistic data conditions during in-water testing on the IVER 3, the TAN algorithm should be implemented on an AUV that has the space for a gravity gradiometer.

Current gravity measurement technology is in early stages of technological readiness, and provides its own challenges for vehicle integration related to power usage, space requirements, mechanical integration, and software integration [112]. The host platform would need to be large enough to support a gravity gradiometer, such as the Explorer class vehicle. This would be a significant undertaking, and would require a team of researchers and technicians. Verification that the gravity gradiometer works,

and produces measurements comparable to existing verified data on board the AUV is recommended. The sensor integration would include precise gradiometer calibration to eliminate measurement errors from vehicle induced acceleration, the Eötvös effect, and thermal drift, as well as quantifying the effect of tides and determining the necessity of filtering these effects from the measurements.

A possible way to reduce the time to testing of the TAN algorithm would be ship-based testing using the gravity gradiometer that would later be integrated into the AUV. This would allow a significant amount of testing of the algorithm and enable discovery of sensor limitations before committing to AUV integration. Once satisfactory results have been achieved the gravity gradiometer can then be integrated into the AUV.

On completion of the sensor integration, the gravity gradient based TAN algorithm (demonstrated within this thesis) should be integrated on to the vehicle and tested on the water surface. This would enable comparison to the absolute GPS position. Close agreement between the GPS position and the TAN state estimate position would support the use of the TAN algorithm for long-range underwater AUV missions.

References

[1] Christopher R. German et al. "A Long Term Vision for Long-Range Ship-Free Deep Ocean Operations: Persistent Presence through Coordination of Autonomous Surface Vehicles and Autonomous Underwater Vehicles". In: *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Southampton, United Kingdom: IEEE, Sept. 2012, pp. 1–7. DOI: 10.1109/AUV.2012.6380753.

[2] Robert Panish and Mikell Taylor. "Achieving High Navigation Accuracy Using Inertial Navigation Systems in Autonomous Underwater Vehicles". In: *OCEANS 2011 IEEE - Spain*. Santander, Spain: IEEE, June 2011, pp. 1–7. ISBN: 978-1-4577-0086-6. DOI: 10.1109/Oceans-Spain.2011.6003517.

[3] Liam Paull et al. "AUV Navigation and Localization: A Review". In: *IEEE Journal of Oceanic Engineering* 39.1 (Jan. 2014), pp. 131–149. ISSN: 0364-9059, 1558-1691. DOI: 10.1109/JOE.2013.2278891.

[4] Liu Lanbo, Zhou Shengli, and Cui Jun-Hong. "Prospects and Problems of Wireless Communication for Underwater Sensor Networks". In: *Wireless Communications and Mobile Computing* 8.8 (2008), pp. 977–994. ISSN: 1530-8677. DOI: 10.1002/wcm.654.

[5] Kjetil Bergh Ånonsen. "Advances in Terrain Aided Navigation for Underwater Vehicles". In: (2010), p. 157.

[6] José Melo and Aníbal Matos. "Survey on Advances on Terrain Based Navigation for Autonomous Underwater Vehicles". In: *Ocean Engineering* 139 (July 2017), pp. 250–264. ISSN: 00298018. DOI: 10.1016/j.oceaneng.2017.04.047.

[7] Warren S. Flenniken Iv, John H. Wall, and David M. Bevly. "Characterization of Various IMU Error Sources and the Effect on Navigation Performance". In: 2005.

[8] Kearfott. *Specifications - Kearfott SeaNav INS*. 2020.

[9] Chris Kaminski et al. "12 Days under Ice: An Historic AUV Deployment in the Canadian High Arctic". In: *2010 IEEE/OES Autonomous Underwater Vehicles*. Monterey, CA, USA: IEEE, Sept. 2010, pp. 1–11. ISBN: 978-1-61284-980-5. DOI: 10.1109/AUV.2010.5779651.

[10] Sebastian Carreno et al. “A Survey on Terrain Based Navigation for AUVs”. In: *OCEANS 2010 MTS/IEEE SEATTLE*. Seattle, WA: IEEE, Sept. 2010, pp. 1–7. ISBN: 978-1-4244-4332-1. DOI: 10.1109/OCEANS.2010.5664372.

[11] M.B. Larsen. “Synthetic Long Baseline Navigation of Underwater Vehicles”. In: *OCEANS 2000 MTS/IEEE Conference and Exhibition. Conference Proceedings (Cat. No.00CH37158)*. Vol. 3. Sept. 2000, 2043–2050 vol.3. DOI: 10.1109/OCEANS.2000.882240.

[12] Liam Paull, Mae Seto, and John Leonard. “Decentralized Cooperative Trajectory Estimation for Autonomous Underwater Vehicles”. In: *IEEE International Conference on Intelligent Robots and Systems* (June 2015). DOI: 10.1109/IROS.2014.6942559.

[13] David Pick et al. “Uncertainty Analysis of Ultra-Short- and Long- Baseline Localization Systems for Autonomous Underwater Vehicles”. In: *OCEANS 2018 MTS/IEEE Charleston*. Oct. 2018, pp. 1–6. DOI: 10.1109/OCEANS.2018.8604760.

[14] Baozhi Chen and Dario Pompili. “Minimizing Position Uncertainty for Under-Ice Autonomous Underwater Vehicles”. In: *Computer Networks* 57.18 (Dec. 2013), pp. 3840–3854. ISSN: 13891286. DOI: 10.1016/j.comnet.2013.09.009.

[15] Brian Claus and Ralf Bachmayer. “Terrain-Aided Navigation for an Underwater Glider”. In: *Journal of Field Robotics* 32.7 (Oct. 2015), pp. 935–951. ISSN: 15564959. DOI: 10.1002/rob.21563.

[16] K.B. Anonsen and O. Hallingstad. “Terrain Aided Underwater Navigation Using Point Mass and Particle Filters”. In: *2006 IEEE/ION Position, Location, And Navigation Symposium*. Coronado, CA: IEEE, 2006, pp. 1027–1035. ISBN: 978-0-7803-9454-4. DOI: 10.1109/PLANS.2006.1650705.

[17] Parth Pasnani and Mae L. Seto. “Terrain-Based Localization and Mapping for Autonomous Underwater Vehicles Using Particle Filters with Marine Gravity Anomalies”. In: *IFAC-PapersOnLine* 51.29 (2018), pp. 354–359. ISSN: 24058963. DOI: 10.1016/j.ifacol.2018.09.498.

[18] Vladimir Djapic et al. “Challenges in Underwater Navigation: Exploring Magnetic Sensors Anomaly Sensing and Navigation”. In: *2015 IEEE Sensors Applications Symposium (SAS)*. Zadar, Croatia: IEEE, Apr. 2015, pp. 1–6. ISBN: 978-1-4799-6117-7. DOI: 10.1109/SAS.2015.7133638.

[19] Hua mu et al. “Geomagnetic Surface Navigation Using Adaptive EKF”. In: June 2007, pp. 2821–2825. DOI: 10.1109/ICIEA.2007.4318926.

[20] Bo Wang et al. "Improved Particle Filter-Based Matching Method With Gravity Sample Vector for Underwater Gravity-Aided Navigation". In: *IEEE Transactions on Industrial Electronics* 68.6 (June 2021), pp. 5206–5216. ISSN: 1557-9948. DOI: 10.1109/TIE.2020.2988227.

[21] Hubiao Wang et al. "Location Accuracy of INS/Gravity-Integrated Navigation System on the Basis of Ocean Experiment and Simulation". In: *Sensors (Basel, Switzerland)* 17.12 (Dec. 2017), p. 2961. ISSN: 1424-8220. DOI: 10.3390/s17122961.

[22] Jr. Cannon, Carl Mark W., and Joseph W. *TERCOM Performance: Analysis and Simulation:* tech. rep. Fort Belvoir, VA: Defense Technical Information Center, June 1974. DOI: 10.21236/AD0783804.

[23] Kedong Wang et al. "Matching Error of the Iterative Closest Contour Point Algorithm for Terrain-Aided Navigation". In: *Aerospace Science and Technology* 73 (Feb. 2018), pp. 210–222. ISSN: 12709638. DOI: 10.1016/j.ast.2017.12.010.

[24] Fanming Liu et al. "Application of Kalman Filter Algorithm in Gravity-Aided Navigation System". In: *2011 IEEE International Conference on Mechatronics and Automation*. Beijing, China: IEEE, Aug. 2011, pp. 2322–2326. ISBN: 978-1-4244-8113-2. DOI: 10.1109/ICMA.2011.5986348.

[25] Lin Wu, Jie Ma, and Jinwen Tian. "A Self-Adaptive Unscented Kalman Filtering for Underwater Gravity Aided Navigation". In: *IEEE/ION Position, Location and Navigation Symposium*. Indian Wells, CA, USA: IEEE, May 2010, pp. 142–145. ISBN: 978-1-4244-5036-7. DOI: 10.1109/PLANS.2010.5507294.

[26] Burak Turan and Ali Turker Kutay. "Particle Filter Studies on Terrain Referenced Navigation". In: *2016 IEEE/ION Position, Location and Navigation Symposium (PLANS)*. Savannah, GA: IEEE, Apr. 2016, pp. 949–954. ISBN: 978-1-5090-2042-3. DOI: 10.1109/PLANS.2016.7479793.

[27] Stephen Barkby et al. "Bathymetric Particle Filter SLAM Using Trajectory Maps". In: *The International Journal of Robotics Research* 31.12 (Oct. 2012), pp. 1409–1430. ISSN: 0278-3649, 1741-3176. DOI: 10.1177/0278364912459666.

[28] P Pasnani, M Seto, and J Gu. "Long Range Underwater Localization and Navigation Using Gravity-Based Measurements". In: (2020), p. 8.

[29] British Oceanographic Data Centre. *GEBCO - The General Bathymetric Chart of the Oceans*. https://www.gebco.net/.

[30] F. Heubach and M. L. Seto. "Extended Range AUV Localization and Navigation Aided by Gravity Anomalies and Bathymetry". In: *2020 IEEE/OES Autonomous Underwater Vehicles Symposium (AUV)*. Sept. 2020, pp. 1–6. DOI: 10.1109/AUV50043.2020.9267918.

[31] Franz Heubach and Mae Seto. "Generation of Augmented Bathymetry to Aid Development of Terrain-Aided Autonomous Underwater Vehicle Localization and Navigation Approaches". In: *OCEANS 2021: San Diego – Porto*. Sept. 2021, pp. 1–6. DOI: 10.23919/OCEANS44145.2021.9705861.

[32] B. Barshan and H.F. Durrant-Whyte. "Inertial Navigation Systems for Mobile Robots". In: *IEEE Transactions on Robotics and Automation* 11.3 (June 1995), pp. 328–342. ISSN: 1042296X. DOI: 10.1109/70.388775.

[33] Musa Morena Marcusso Manhães et al. "UUV Simulator: A Gazebo-based Package for Underwater Intervention and Multi-Robot Simulation". In: *OCEANS 2016 MTS/IEEE Monterey*. IEEE, Sept. 2016. DOI: 10.1109/oceans.2016.7761080.

[34] Blender Foundation. *Blender*. The Blender Project.

[35] *Rasterio*. Mapbox. 2018.

[36] *Matplotlib*. Matplotlib Development Team. 2022.

[37] Paul D. Groves. *Principles of GNSS, Inertial, and Multi-sensor Integrated Navigation Systems*. 2008.

[38] Chris Roman and Hanumant Singh. "A Self-Consistent Bathymetric Mapping Algorithm". In: *Journal of Field Robotics* 24.1-2 (Jan. 2007), pp. 23–50. ISSN: 15564959, 15564967. DOI: 10.1002/rob.20164.

[39] Adrian Ratter and Claude Sammut. "Local Map Based Graph SLAM with Hierarchical Loop Closure and Optimisation". In: (2015), p. 10.

[40] Joan Solà. "Quaternion Kinematics for the Error-State Kalman Filter". In: *arXiv:1711.02508 [cs]* (Nov. 2017). arXiv: 1711.02508 [cs].

[41] M. Karimi, M. Bozorg, and A. R. Khayatian. "A Comparison of DVL/INS Fusion by UKF and EKF to Localize an Autonomous Underwater Vehicle". In: *2013 First RSI/ISM International Conference on Robotics and Mechatronics (ICRoM)*. Tehran: IEEE, Feb. 2013, pp. 62–67. ISBN: 978-1-4673-5811-8 978-1-4673-5809-5 978-1-4673-5810-1. DOI: 10.1109/ICRoM.2013.6510082.

[42] Ken Perlin. "Improving Noise". In: *ACM Transactions on Graphics* 21.3 (July 2002), pp. 681–682. ISSN: 0730-0301. DOI: 10.1145/566654.566636.

[43] Fanming Liu et al. "Integrated Navigation System Based on Correlation between Gravity Gradient and Terrain". In: *2009 International Joint Conference on Computational Sciences and Optimization*. Sanya, Hainan, China: IEEE, Apr. 2009, pp. 289–293. ISBN: 978-0-7695-3605-7. DOI: 10.1109/CSO.2009.98.

[44] Weikko Heiskanen and Helmut Moritz. *Physical Geodesy*. W. H. Freeman, 1967.

[45] Sebastian Thrun. "Particle Filters in Robotics". In: (2002), p. 9.

[46] José Melo and Aníbal Matos. "A Data-driven Particle Filter for Terrain Based Navigation of Sensor-limited Autonomous Underwater Vehicles". In: *Asian Journal of Control* 21.4 (2019), pp. 1659–1670. ISSN: 1561-8625. DOI: 10.1002/asjc.2107.

[47] Paul A. Miller et al. "Autonomous Underwater Vehicle Navigation". In: *IEEE Journal of Oceanic Engineering* 35.3 (July 2010), pp. 663–678. ISSN: 0364-9059, 1558-1691, 2373-7786. DOI: 10.1109/JOE.2010.2052691.

[48] Meng Wu and Jian Yao. "Adaptive UKF-SLAM Based on Magnetic Gradient Inversion Method for Underwater Navigation". In: (2015), p. 5.

[49] S. Julier, J. Uhlmann, and H.F. Durrant-Whyte. "A New Method for the Nonlinear Transformation of Means and Covariances in Filters and Estimators". In: *IEEE Transactions on Automatic Control* 45.3 (Mar. 2000), pp. 477–482. ISSN: 1558-2523. DOI: 10.1109/9.847726.

[50] Kevin Murphy and Stuart Russell. "Rao-Blackwellised Particle Filtering for Dynamic Bayesian Networks". In: *Sequential Monte Carlo Methods in Practice*. Ed. by Arnaud Doucet, Nando Freitas, and Neil Gordon. New York, NY: Springer New York, 2001, pp. 499–515. ISBN: 978-1-4419-2887-0 978-1-4757-3437-9. DOI: 10.1007/978-1-4757-3437-9_24.

[51] Shandor Dektor and Stephen Rock. "Improving Robustness of Terrain-Relative Navigation for AUVs in Regions with Flat Terrain". In: *2012 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Southampton, United Kingdom: IEEE, Sept. 2012, pp. 1–7. ISBN: 978-1-4577-2056-7 978-1-4577-2055-0 978-1-4577-2054-3. DOI: 10.1109/AUV.2012.6380751.

[52] Peter W. Kimball and Stephen M. Rock. "Mapping of Translating, Rotating Icebergs With an Autonomous Underwater Vehicle". In: *IEEE Journal of Oceanic Engineering* 40.1 (Jan. 2015), pp. 196–208. ISSN: 0364-9059, 1558-1691, 2373-7786. DOI: 10.1109/JOE.2014.2300396.

[53] Stephen M. Chaves et al. "Pose-Graph SLAM for Underwater Navigation". In: *Sensing and Control for Autonomous Vehicles*. Ed. by Thor I. Fossen, Kristin Y. Pettersen, and Henk Nijmeijer. Vol. 474. Cham: Springer International Publishing, 2017, pp. 143–160. ISBN: 978-3-319-55371-9 978-3-319-55372-6. DOI: 10.1007/978-3-319-55372-6_7.

[54] Liam Paull et al. *Encyclopedia of Robotics: Chapter 11*. Springer Berlin Heidelberg, 2018. ISBN: 978-3-642-41610-1.

[55] Doupadi Bandara et al. "Technologies for Under-Ice AUV Navigation". In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Tokyo, Japan: IEEE, Nov. 2016, pp. 108–114. ISBN: 978-1-5090-2442-1. DOI: 10.1109/AUV.2016.7778657.

[56] Di Qiu et al. "Underwater Navigation Using Location-Dependent Signatures". In: *2012 IEEE Aerospace Conference*. Big Sky, MT: IEEE, Mar. 2012, pp. 1–9. ISBN: 978-1-4577-0557-1 978-1-4577-0556-4 978-1-4577-0555-7. DOI: 10.1109/AERO.2012.6187192.

[57] Joao Quintas, Francisco Curado Teixeira, and Antonio Pascoal. "Magnetic Signal Processing Methods with Application to Geophysical Navigation of Marine Robotic Vehicles". In: *OCEANS 2016 MTS/IEEE Monterey*. Monterey, CA, USA: IEEE, Sept. 2016, pp. 1–8. ISBN: 978-1-5090-1537-5. DOI: 10.1109/OCEANS.2016.7761322.

[58] Naomi Kato and Toshihide Shigetomi. "Underwater Navigation for Long-Range Autonomous Underwater Vehicles Using Geomagnetic and Bathymetric Information". In: *Advanced Robotics* 23.7-8 (Jan. 2009), pp. 787–803. ISSN: 0169-1864, 1568-5535. DOI: 10.1163/156855309X443016.

[59] Izabela Bodus-Olkowska and Natalia Wawrzyniak. "Hydrographic Imaging for Underwater Environment Modelling". In: *2017 18th International Radar Symposium (IRS)*. Prague, Czech Republic: IEEE, June 2017, pp. 1–10. ISBN: 978-3-7369-9343-3. DOI: 10.23919/IRS.2017.8008211.

[60] G.C. Bishop. "Gravitational Field Maps and Navigational Errors". In: *Proceedings of the 2000 International Symposium on Underwater Technology (Cat. No.00EX418)*. Tokyo, Japan: IEEE, 2000, pp. 149–154. ISBN: 978-0-7803-6378-6. DOI: 10.1109/UT.2000.852532.

[61] Fanming Liu et al. "Navigability Analysis Based on Gravity Map Statistical Characteristics". In: *2012 IEEE International Conference on Mechatronics and Automation*. Chengdu, China: IEEE, Aug. 2012, pp. 522–526. ISBN: 978-1-4673-1278-3 978-1-4673-1275-2 978-1-4673-1277-6. DOI: 10.1109/ICMA.2012.6283162.

[62] Colin M. MacKenzie, Mae L. Seto, and Yajun Pan. "Extracting Seafloor Elevations from Side-Scan Sonar Imagery for SLAM Data Association". In: *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*. Halifax, NS, Canada: IEEE, May 2015, pp. 332–336. ISBN: 978-1-4799-5827-6 978-1-4799-5829-0. DOI: 10.1109/CCECE.2015.7129298.

[63] Aaron Canciani and John Raquet. "Airborne Magnetic Anomaly Navigation". In: *IEEE Transactions on Aerospace and Electronic Systems* 53.1 (Feb. 2017), pp. 67–80. ISSN: 0018-9251. DOI: 10.1109/TAES.2017.2649238.

[64] *IAGA V-MOD Geomagnetic Field Modeling: International Geomagnetic Reference Field IGRF-13*. https://www.ngdc.noaa.gov/IAGA/vmod/igrf.html.

[65] William J. Hinze, R. Von Frese, and Afif H. Saad. *Gravity and Magnetic Exploration: Principles, Practices, and Applications*. New York: Cambridge University Press, 2013. ISBN: 978-0-521-87101-3.

[66] Peter Hood. "History of Aeromagnetic Surveying in Canada". In: *The Leading Edge* 26.11 (Nov. 2007), pp. 1384–1392. ISSN: 1070-485X, 1938-3789. DOI: 10.1190/1.2805759.

[67] Gunther Kletetschka et al. "Localization of the Chelyabinsk Meteorite From Magnetic Field Survey and GPS Data". In: *IEEE Sensors Journal* 15.9 (Sept. 2015), pp. 4875–4881. ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2015.2435252.

[68] Masanao Shinohara et al. "Development of an Underwater Gravity Measurement System Using Autonomous Underwater Vehicle for Exploration of Seafloor Deposits". In: *OCEANS 2015 - Genova*. Genova, Italy: IEEE, May 2015, pp. 1–7. ISBN: 978-1-4799-8736-8. DOI: 10.1109/OCEANS-Genova.2015.7271487.

[69] *GRACE Fact Sheet*. https://earthobservatory.nasa.gov/features/GRACE/page3.php. Text.Article. Mar. 2004.

[70] "Eötvös Effect". In: *Wikipedia* (Dec. 2019).

[71] L. V. Kiselev et al. "Autonomous Underwater Robot as an Ideal Platform for Marine Gravity Surveys". In: *2017 24th Saint Petersburg International Conference on Integrated Navigation Systems (ICINS)*. Saint Petersburg, Russia: IEEE, May 2017, pp. 1–4. DOI: 10.23919/ICINS.2017.7995685.

[72] Yong Wang et al. "Technology of Gravity Aided Inertial Navigation System and Its Trial in South China Sea". In: *IET Radar, Sonar & Navigation* 10.5 (June 2016), pp. 862–869. ISSN: 1751-8784, 1751-8792. DOI: 10.1049/iet-rsn.2014.0419.

[73] L3 Harris. *IVER3 Autonomous Underwater Vehicle (AUV)*. 2019.

[74] Takemi Ishihara et al. "Development of an Underwater Gravity Measurement System with Autonomous Underwater Vehicle for Marine Mineral Exploration". In: *2016 Techno-Ocean (Techno-Ocean)*. Kobe, Japan: IEEE, 2016, pp. 127–133. ISBN: 978-1-5090-2445-2. DOI: 10.1109/Techno-Ocean.2016.7890633.

[75] David T. Sandwell and Walter H. F. Smith. "Marine Gravity Anomaly from Geosat and ERS 1 Satellite Altimetry". In: *Journal of Geophysical Research: Solid Earth* 102.B5 (May 1997), pp. 10039–10054. ISSN: 01480227. DOI: 10.1029/96JB03223.

[76] James C. Kinsey, Maurice A. Tivey, and Dana R. Yoerger. "Toward High-Spatial Resolution Gravity Surveying of the Mid-Ocean Ridges with Autonomous Underwater Vehicles". In: *OCEANS 2008*. Quebec City, QC, Canada: IEEE, 2008, pp. 1–10. ISBN: 978-1-4244-2619-5. DOI: 10.1109/OCEANS.2008.5152005.

[77] Zu Yan et al. "Modeling Local Gravity Anomaly Self-Adaption Quotient Reference Maps for Underwater Autonomous Navigation". In: *2014 IEEE 26th International Conference on Tools with Artificial Intelligence*. Limassol, Cyprus: IEEE, Nov. 2014, pp. 945–949. ISBN: 978-1-4799-6572-4. DOI: 10.1109/ICTAI.2014.143.

[78] Stephen Barkby et al. "A Featureless Approach to Efficient Bathymetric SLAM Using Distributed Particle Mapping". In: *Journal of Field Robotics* 28.1 (Jan. 2011), pp. 19–39. ISSN: 15564959. DOI: 10.1002/rob.20382.

[79] Momotaz Begum, George I. Mann, and Raymond Gosine. "An Evolutionary SLAM Algorithm for Mobile Robots". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Beijing, China: IEEE, Oct. 2006, pp. 4066–4071. ISBN: 978-1-4244-0258-8 978-1-4244-0259-5. DOI: 10.1109/IROS.2006.281870.

[80] Stephen Barkby et al. "Incorporating Prior Maps with Bathymetric Distributed Particle SLAM for Improved AUV Navigation and Mapping". In: (2009), p. 7.

[81] Yurong Han et al. "An Improved TERCOM-Based Algorithm for Gravity-Aided Navigation". In: *IEEE Sensors Journal* 16.8 (Apr. 2016), pp. 2537–2544. ISSN: 1530-437X, 1558-1748, 2379-9153. DOI: 10.1109/JSEN.2016.2518686.

[82] A. Jircitano, J. White, and D. Dosch. "Gravity Based Navigation of AUVs". In: *Symposium on Autonomous Underwater Vehicle Technology*. June 1990, pp. 177–180. DOI: 10.1109/AUV.1990.110453.

[83] Ling Xiong, Jie Ma, and Jin-wen Tian. "Gravity Gradient Aided Position Approach Based on EKF and NN". In: *Proceedings of 2011 Cross Strait Quad-Regional Radio Science and Wireless Technology Conference*. Harbin, Heilongjiang, China: IEEE, July 2011, pp. 1347–1350. ISBN: 978-1-4244-9792-8. DOI: 10.1109/CSQRWC.2011.6037213.

[84] Yuhong Yang, Junchuan Zhou, and Otmar Loffeld. "Quaternion-Based Kalman Filtering on INS/GPS". In: (), p. 8.

[85] De Ruiter. *Spacecraft Dynamics and Control : An Introduction*. 2012.

[86] James M. Maley. *Multiplicative Quaternion Extended Kalman Filtering for Nonspinning Guided Projectiles:* tech. rep. Fort Belvoir, VA: Defense Technical Information Center, July 2013. DOI: 10.21236/ADA588831.

[87] Richard S. Bucy and Peter D. Joseph. *Filtering For Stochastic Processes With Applications To Guidance*. New edition edition. Providence, R.I: American Mathematical Society, Mar. 2005. ISBN: 978-0-8218-3782-5.

[88] Renato Zanetti and Kyle J. DeMars. "Joseph Formulation of Unscented and Quadrature Filters with Application to Consider States". In: *Journal of Guidance, Control, and Dynamics* 36.6 (Nov. 2013), pp. 1860–1864. ISSN: 0731-5090, 1533-3884. DOI: 10.2514/1.59935.

[89] Li Tian. "Ocean Wave Simulation by the Mix of FFT and Perlin Noise". In: (2014), p. 4.

[90] I. Martin et al. "Asteroid Modeling for Testing Spacecraft Approach and Landing". In: *IEEE Computer Graphics and Applications* 34.4 (July 2014), pp. 52–62. ISSN: 1558-1756. DOI: 10.1109/MCG.2014.22.

[91] J. E. Martine. *Relating Geoid Anomalies, Gravity Anomalies and Ocean Topography*. Tech. rep. Fort Belvoir, VA: Defense Technical Information Center, Dec. 1983. DOI: 10.21236/ADA140112.

[92] Y. Guo, L. Xiong, and X. Cheng. "Modeling and Analysis of Gravity and Gravity Gradient Based on Terrain Anomaly". In: *2018 13th IEEE Conference on Industrial Electronics and Applications (ICIEA)*. May 2018, pp. 1957–1961. DOI: 10.1109/ICIEA.2018.8398029.

[93] *Scripps Institution of Oceanography, UC San Diego*. https://scripps.ucsd.edu/.

[94] Dan DiFrancesco et al. "Gravity Gradiometry – Today and Tomorrow". In: *11th SAGA Biennial Technical Meeting and Exhibition*. Swaziland, South Africa, European Association of Geoscientists & Engineers, 2009. DOI: 10.3997/2214-4609-pdb.241.difrancesco_paper1.

[95] Daniel DiFrancesco et al. “Gravity Gradiometer Systems – Advances and Challenges”. In: *Geophysical Prospecting* 57.4 (2009), pp. 615–623. ISSN: 1365-2478. DOI: 10.1111/j.1365-2478.2008.00764.x.

[96] T. Li, M. Bolic, and P. M. Djuric. “Resampling Methods for Particle Filtering: Classification, Implementation, and Strategies”. In: *IEEE Signal Processing Magazine* 32.3 (May 2015), pp. 70–86. ISSN: 1558-0792. DOI: 10.1109/MSP.2014.2330626.

[97] The Linux Foundation. *The Linux Kernel Archives*. Linux Kernel Organization. 2022.

[98] Python Software Foundation. *Python.Org*. The Python Software Foundation. 2022.

[99] Open Robotics. *ROS*. Open Robotics. 2022.

[100] William Woodall. *ROS on DDS*. https://design.ros2.org/articles/ros_on_dds.html. 2019.

[101] Object Management Group. *DDS Portal – Data Distribution Services*. https://www.dds-foundation.org/. 2021.

[102] *ROS-Industrial*. https://rosindustrial.org. 2022.

[103] *Docker*. Docker. 2022.

[104] Open Source Robotics Foundation. *Gazebo*. http://gazebosim.org/.

[105] Open Robotics. *ROS Index*. https://index.ros.org/packages/. 2022.

[106] Georgios Salavasidis et al. “Terrain Aided Navigation for Long Range AUV Operations at Arctic Latitudes”. In: *2016 IEEE/OES Autonomous Underwater Vehicles (AUV)*. Tokyo, Japan: IEEE, Nov. 2016, pp. 115–123. ISBN: 978-1-5090-2442-1. DOI: 10.1109/AUV.2016.7778658.

[107] Fisheries and Oceans Canada Government of Canada. *Nautical Charts and Services*. https://www.charts.gc.ca/index-eng.html. July 2019.

[108] *NumPy*. NumPy. 2022.

[109] National Earth Science Teachers Association (NESTA). *Density of Ocean Water*. https://windows2universe.org/earth/Water/density.html. 2010.

[110] Robert Tenzer, Pavel Novák, and Vladislav Gladkikh. “On the Accuracy of the Bathymetry-Generated Gravitational Field Quantities for a Depth-Dependent Seawater Density Distribution”. In: *Studia Geophysica et Geodaetica* 55.4 (Aug. 2011), p. 609. ISSN: 1573-1626. DOI: 10.1007/s11200-010-0074-y.

[111] Jeroen D. Hol, Thomas B. Schon, and Fredrik Gustafsson. "On Resampling Algorithms for Particle Filters". In: *2006 IEEE Nonlinear Statistical Signal Processing Workshop*. Cambridge, UK: IEEE, Sept. 2006, pp. 79–82. ISBN: 978-1-4244-0579-4 978-1-4244-0581-7. DOI: 10.1109/NSSPW.2006.4378824.

[112] Takemi Ishihara et al. "High-Resolution Gravity Measurement Aboard an Autonomous Underwater Vehicle". In: *GEOPHYSICS* 83.6 (Nov. 2018), G119–G135. ISSN: 0016-8033, 1942-2156. DOI: 10.1190/geo2018-0090.1.

Appendix A

Gradiometer Noise Particle Filter Results

This appendix contains all the plots comparing positional estimate error of the particle filter solution (TAN) versus only dead reckoning (INS) for the gradiometer noise parameter study. This study looks at the effect of gravity gradiometer heading noise on the positional estimate error. Figs. A.1 to A.16. Detailed discussion of this study can be found in section 5.6.1. Each figure in this appendix summarizes the performance spread of one session. All runs in the same session have the same gradiometer noise heading standard deviation. The summary of the study's session is shown in table 5.2.

For each session there are two figures. The first figure shows the comparison between positional estimate error between the particle filter and dead reckoning. The second shows only the positional estimate error of the particle filter because it is hard to see details of the spread when plotted on the same plot as dead reckoning. The

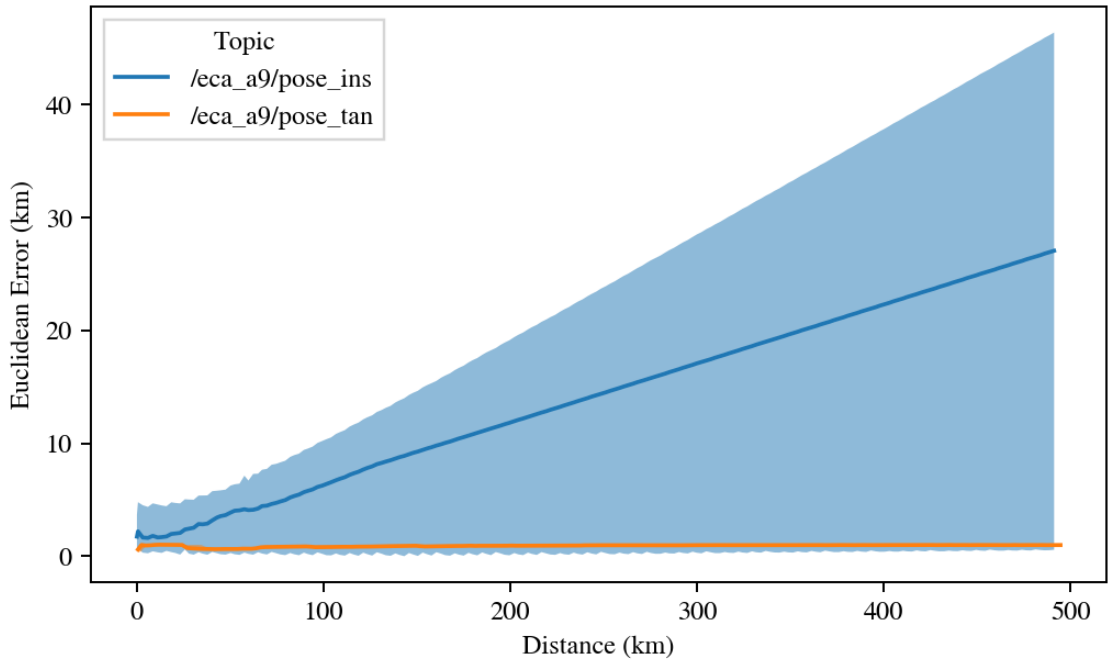

Figure (A.1) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 0 is shown. The gradiometer heading noise for this session was 0 rad. The first session is rather unrealistic and only 6 of the 10 runs finished with problems of particle degeneracy. This is due to the particle filter requiring some stabilizing noise to avoid degeneracy. A noise setting of 0 does not achieve this. This was expected and holds little value for application to reality.

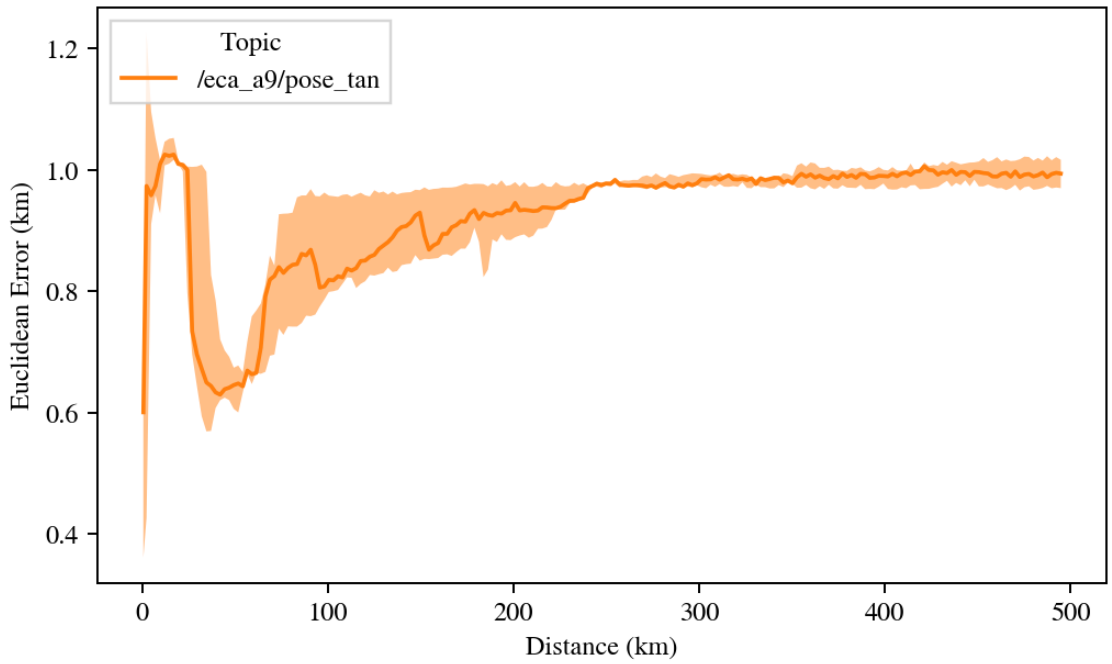

Figure (A.2) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 0. The gradiometer heading noise for this session was 0 rad. Note that a noise setting of 0 is unrealistic and caused 6 out of 10 of the particle filter runs to end in degeneracy. This was expected.

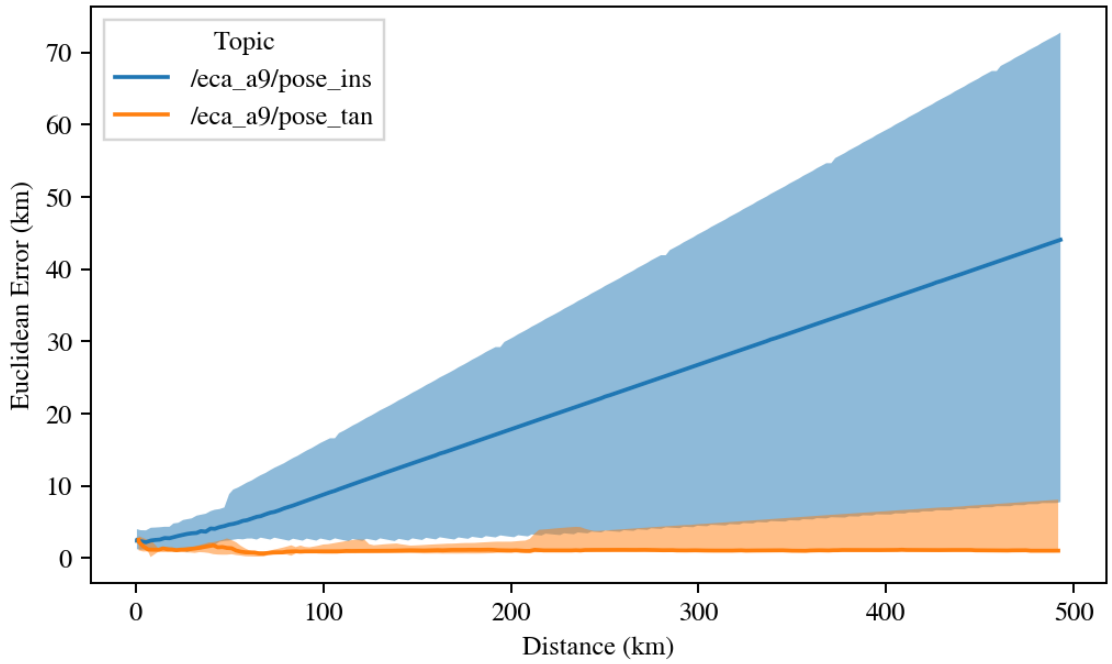

Figure (A.3) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 1 is shown. The gradiometer heading noise for this session was 0.1 rad.

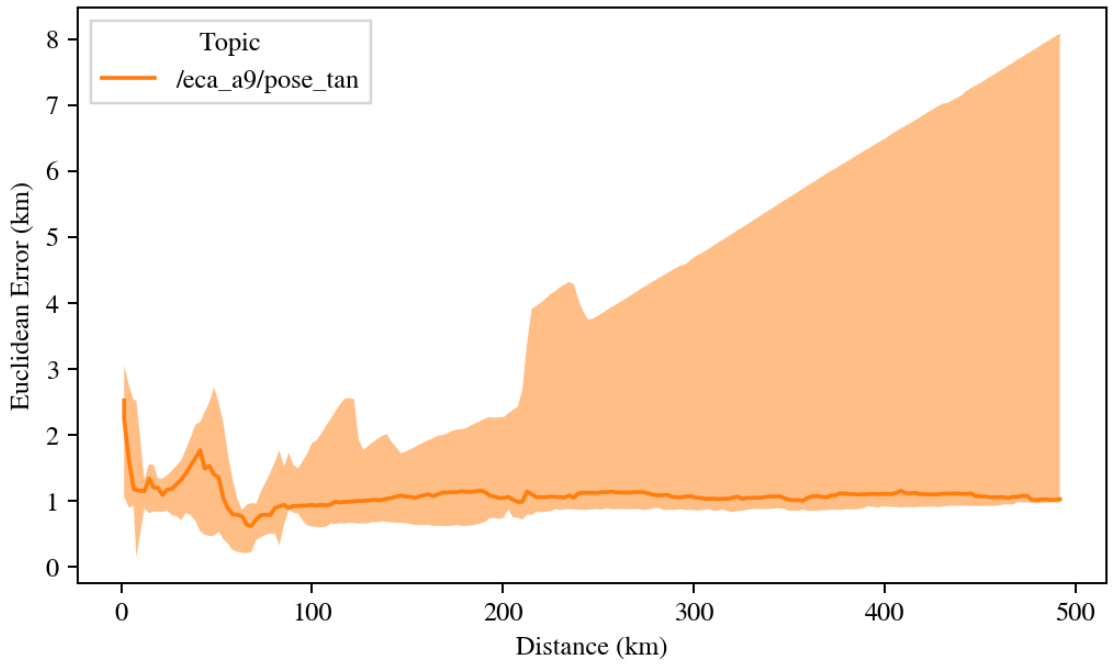

Figure (A.4) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 1. The gradiometer heading noise for this session was 0.1 rad.

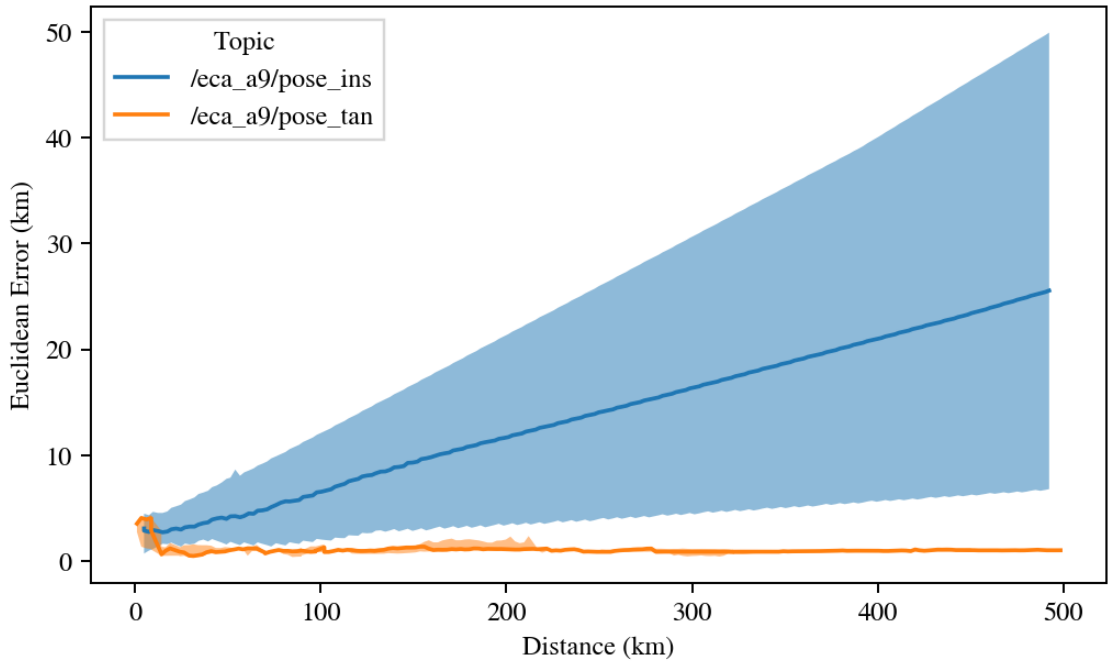

Figure (A.5) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 2 is shown. The gradiometer heading noise for this session was 0.2 rad.

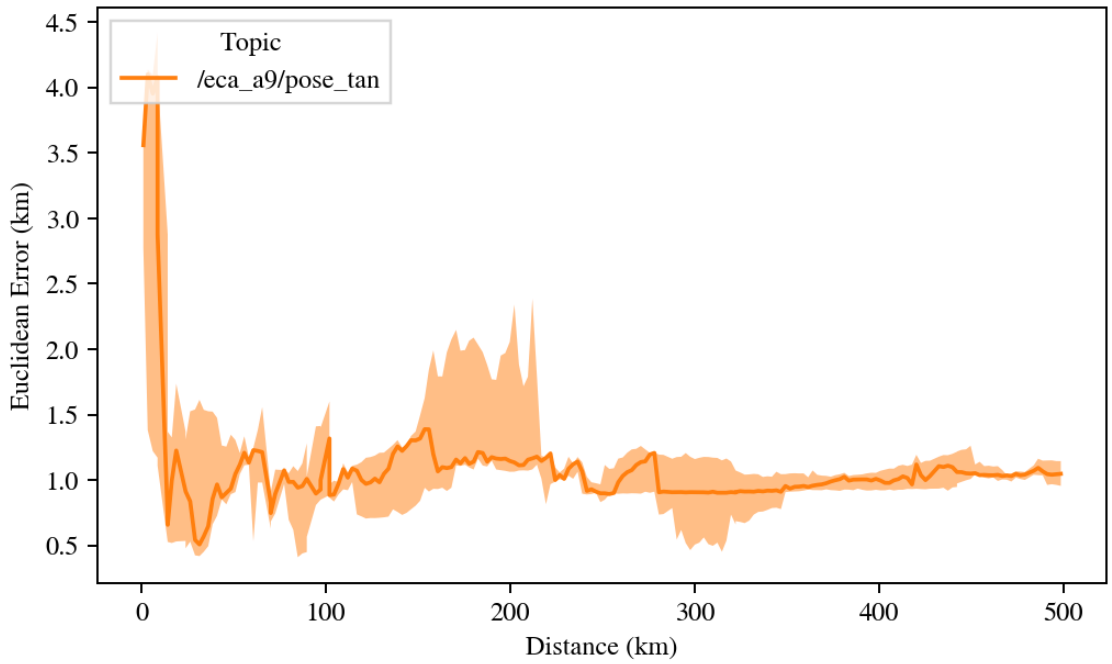

Figure (A.6) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 2. The gradiometer heading noise for this session was 0.2 rad.

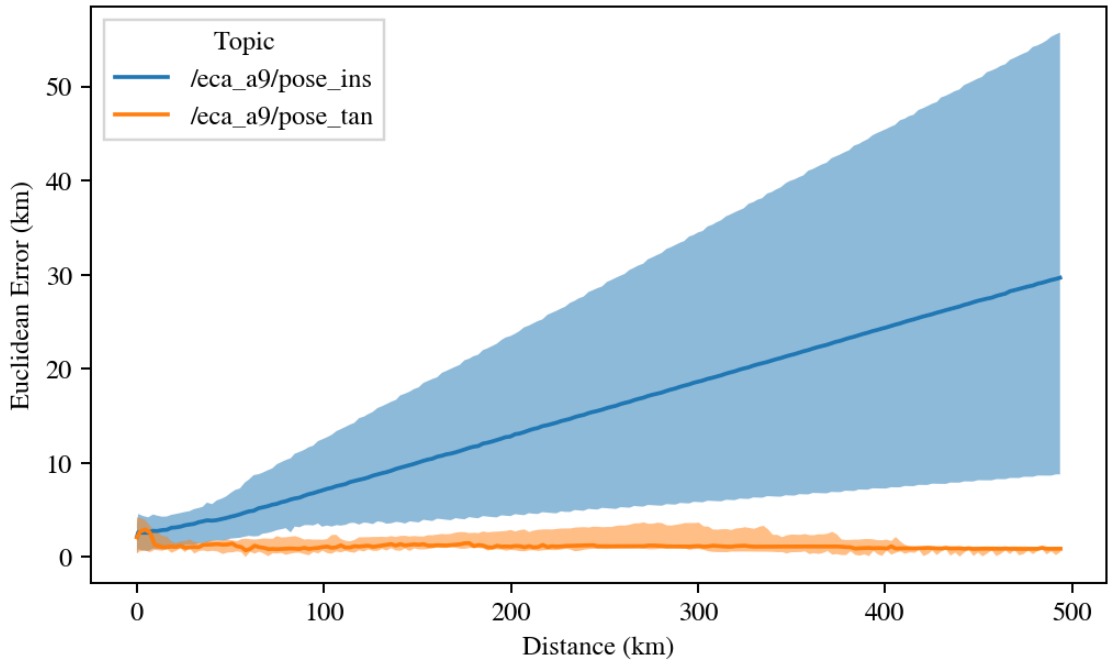

Figure (A.7) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 3 is shown. The gradiometer heading noise for this session was 0.3 rad.

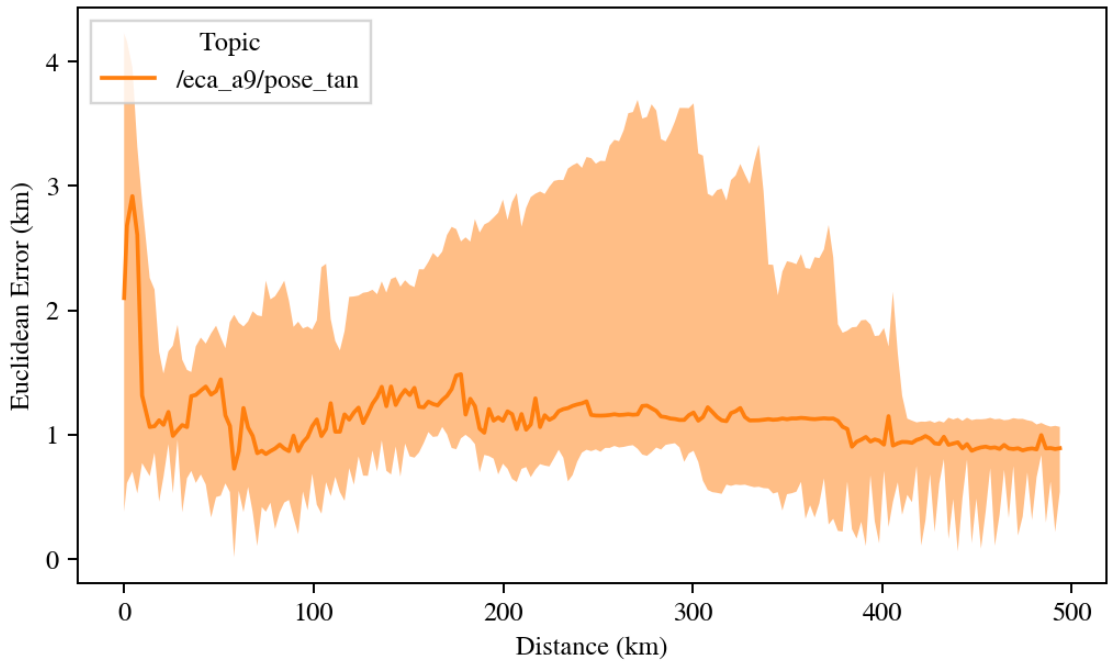

Figure (A.8) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 3. The gradiometer heading noise for this session was 0.3 rad.

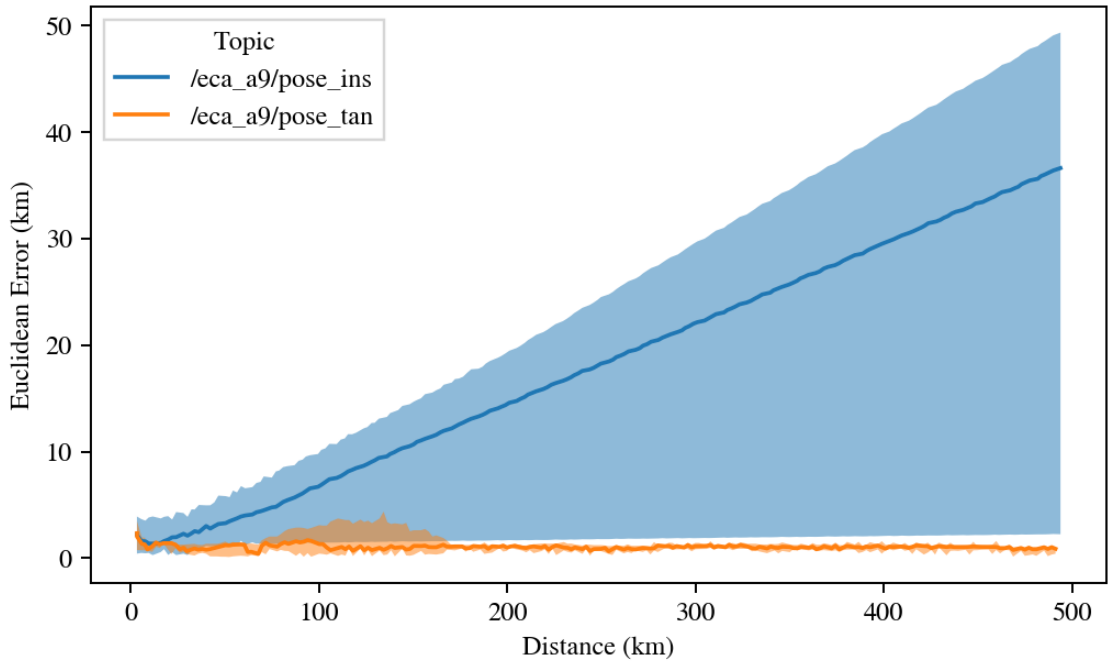

Figure (A.9) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 4 is shown. The gradiometer heading noise for this session was 0.4 rad.

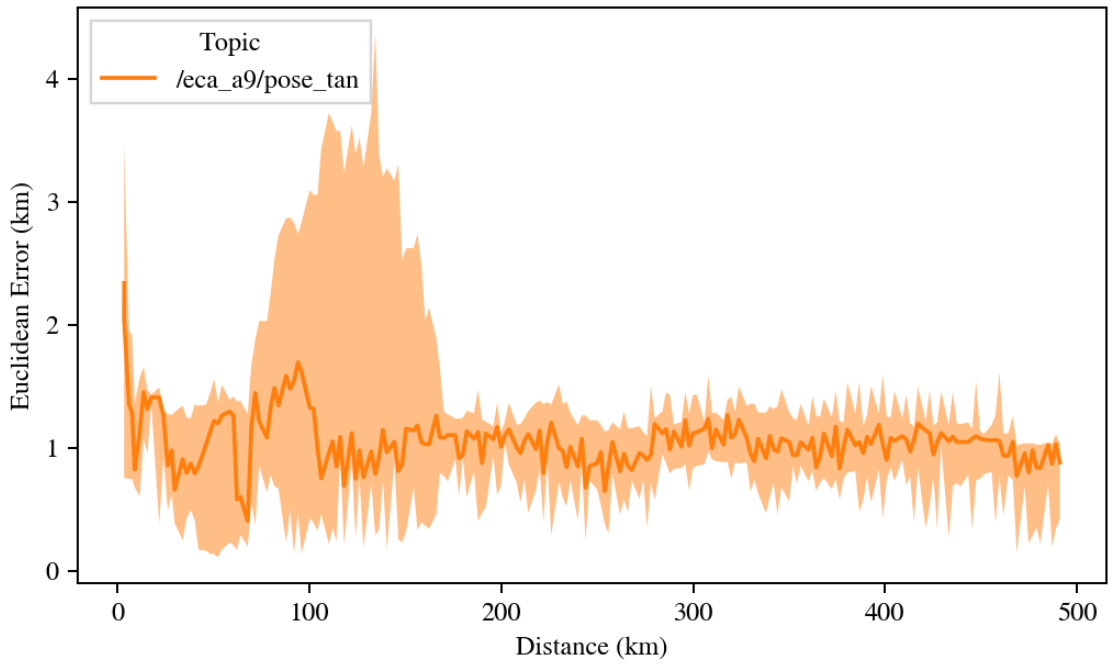

Figure (A.10) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 4. The gradiometer heading noise for this session was 0.4 rad.

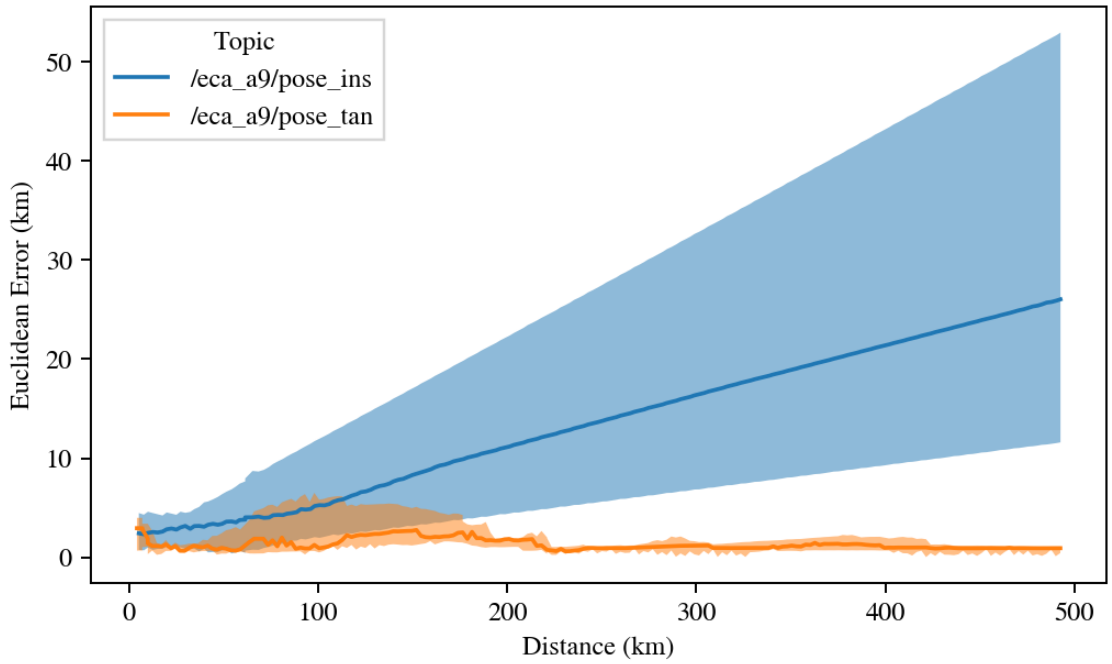

Figure (A.11) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 5 is shown. The gradiometer heading noise for this session was 0.5 rad.

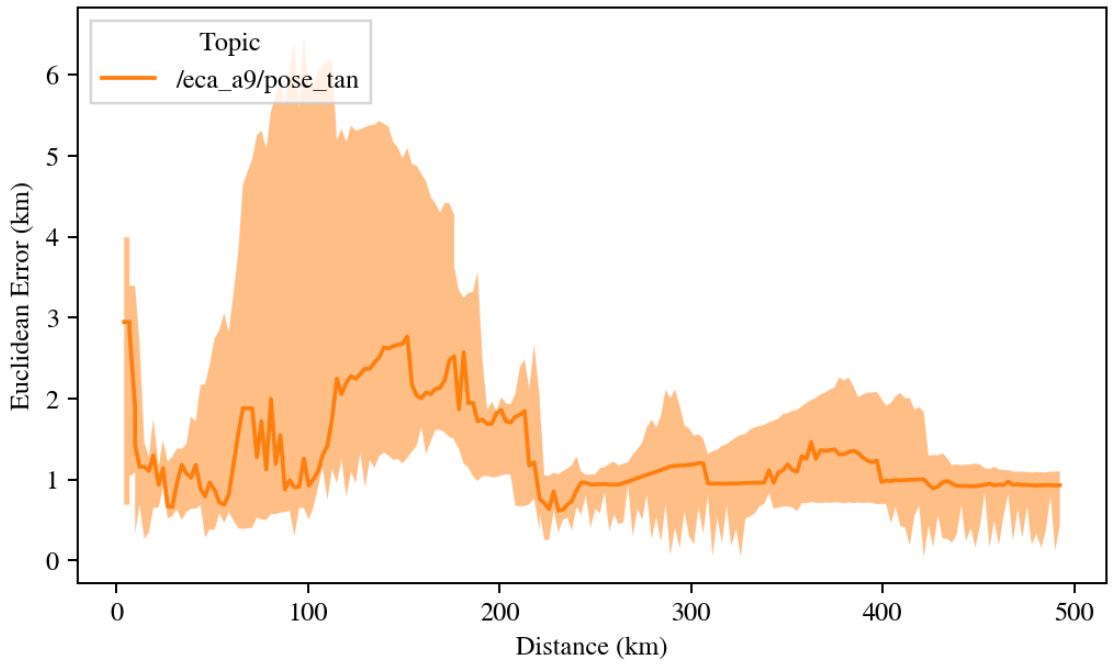

Figure (A.12) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 5. The gradiometer heading noise for this session was 0.5 rad.

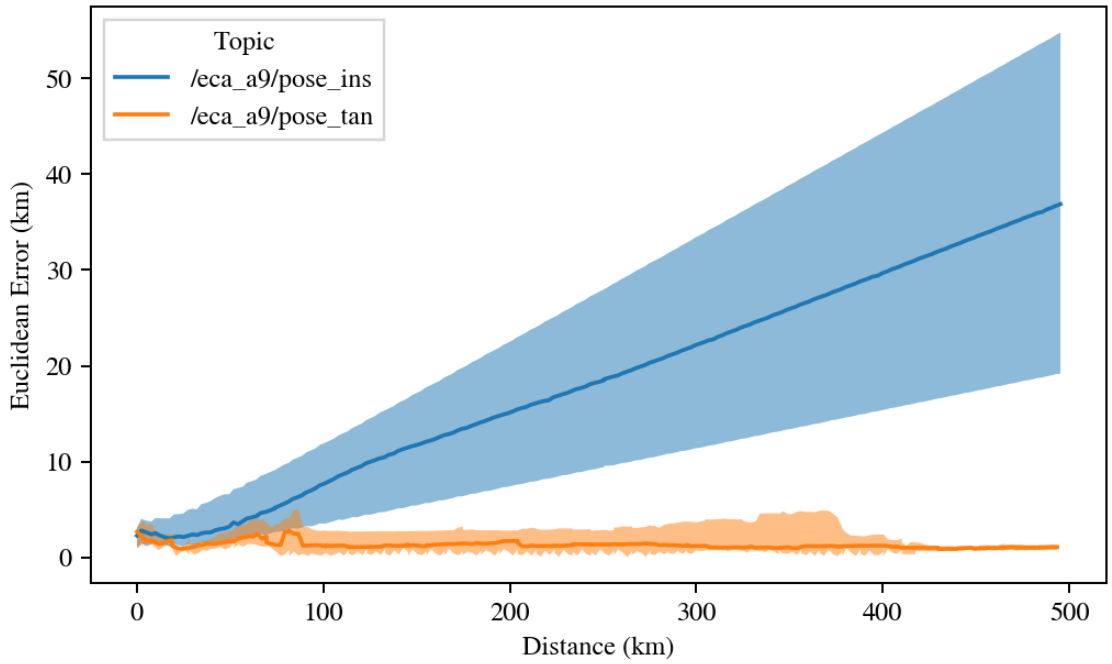

Figure (A.13) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 6 is shown. The gradiometer heading noise for this session was 0.6 rad.

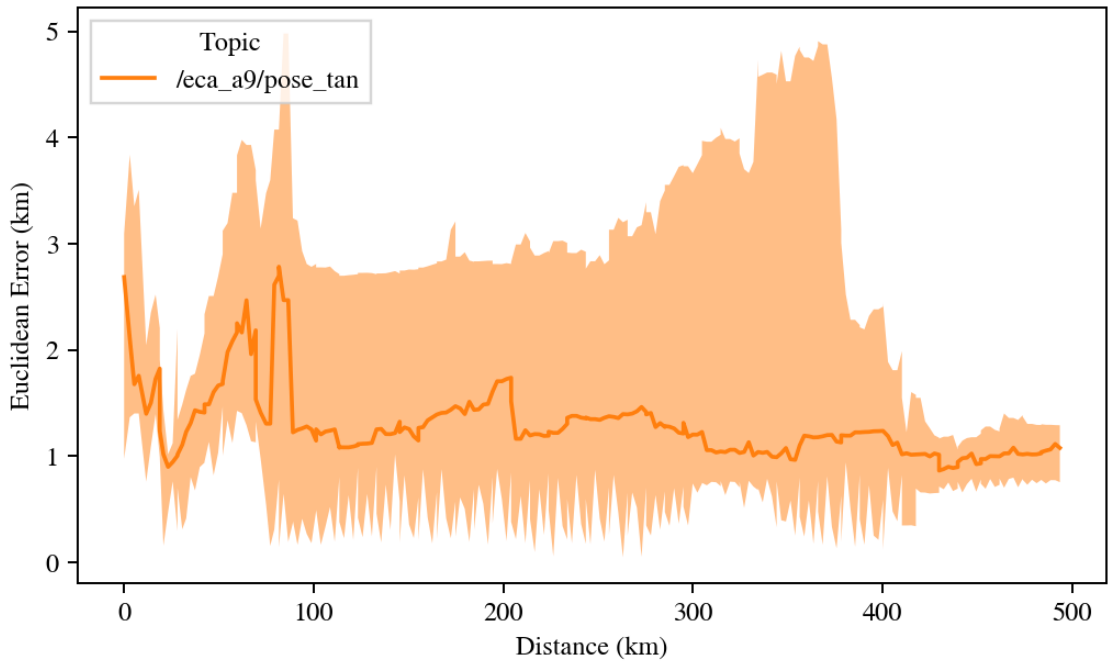

Figure (A.14) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 6. The gradiometer heading noise for this session was 0.6 rad.

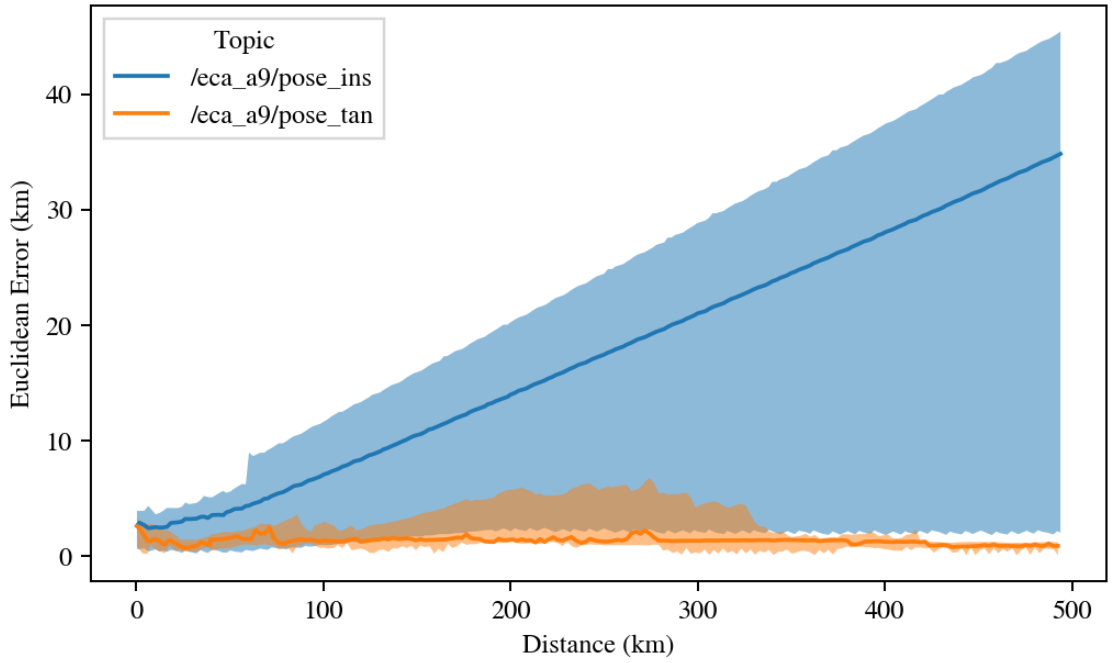

Figure (A.15) The comparison of positional estimate Euclidean error between the particle filter (TAN) and dead reckoning (INS). The legend shows the topic names within ROS. The dark line represents the median Euclidean positional estimate error. The lighter shaded area represents the bounds of the Euclidean positional estimate error (minimum and maximum). The comparison is made by distance over ground covered by the vehicle. Session 7 is shown. The gradiometer heading noise for this session was 0.7 rad.

measure of error used is Euclidean error.

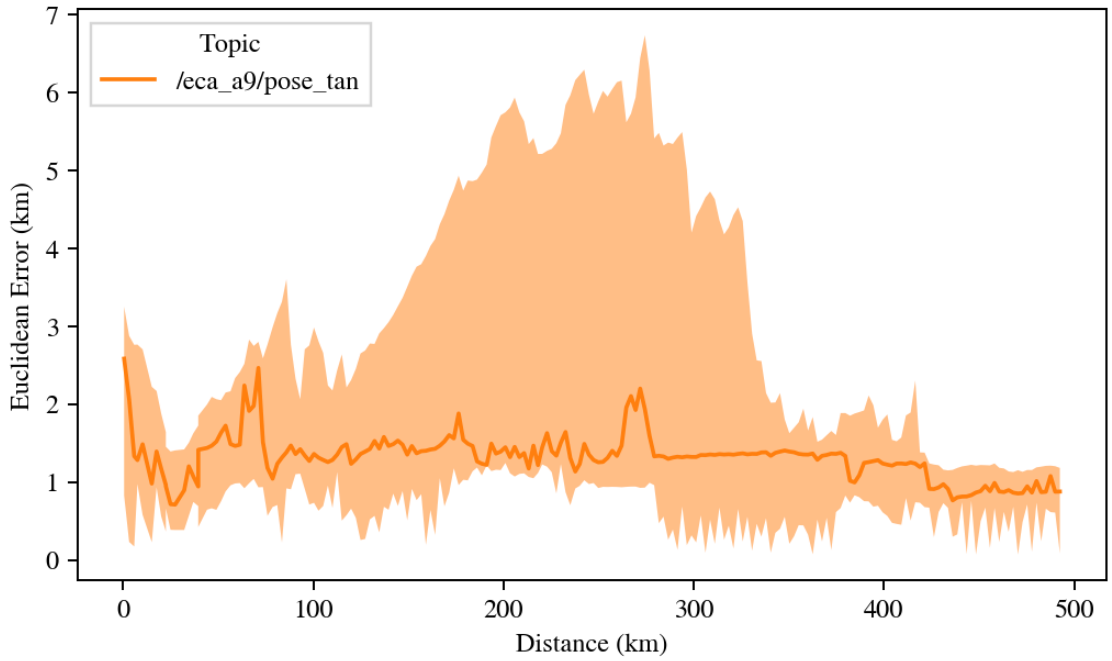

Figure (A.16) The zoomed in plot of the positional estimate Euclidean error of the particle filter (TAN). The legend shows the topic names within ROS. The dark line represents the median positional estimate Euclidean error. The lighter shaded area shows the maximum in minimum errors within session 7. The gradiometer heading noise for this session was 0.7 rad.

Appendix B

Density Amplitude Study

This appendix shows all the positional error spreads associated with the particle filter algorithm (TAN) compared with dead reckoning (INS) for the density uncertainty study. For this study the density amplitude variation was varied logarithmically. The generated density maps used to derive the gravity gradient used as positional feedback for the TAN algorithm are show in Figs. B.1 to B.15. A representative example, related discussion, and implications are discussed in section 5.6.2.

B.1 Density Maps

The density maps shown in Figs. B.1 to B.5 vary from a density amplitude variation of $50\,\mathrm{kg/m^3}$ to $800\,\mathrm{kg/m^3}$ in five steps. The density was generated using two octaves of Perlin noise. The base period of the Perlin noise octaves was set to 500 km.

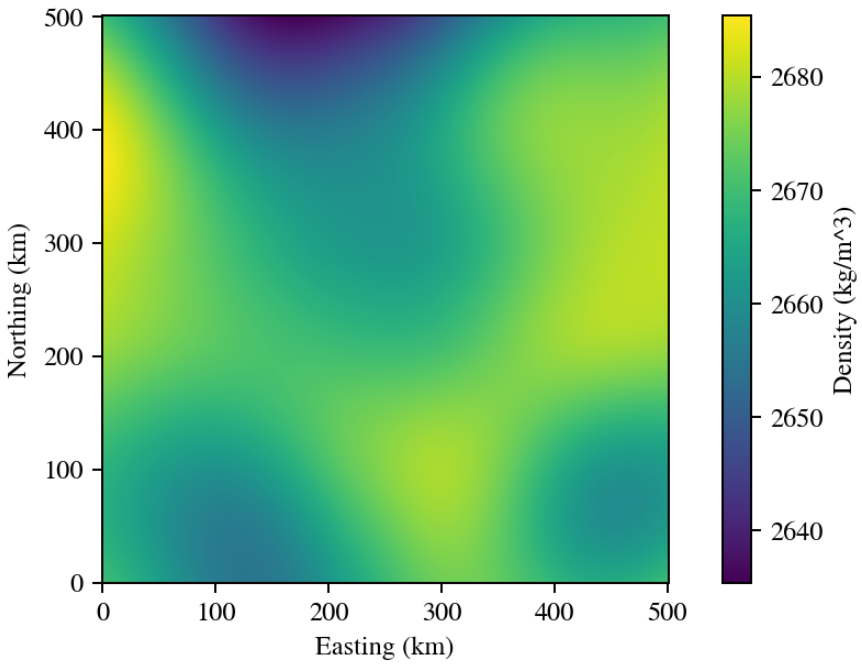

Figure (B.1) The generated density map using layers of Perlin noise. This density field contains two octaves of Perlin noise, configured with a base period of 500 km, a lacunarity of 2, and a persistence of 0.5. The density varies over an area of 500 km by 500 km. It is georeferenced to the same GEBCO area used for the study, shown in Fig. 5.13. However, it is shown with a false origin that is relative to the south-west corner of the GEBCO bathymetry. The base amplitude for the Perlin noise used to generate this density was set to 50 kg/m^3.

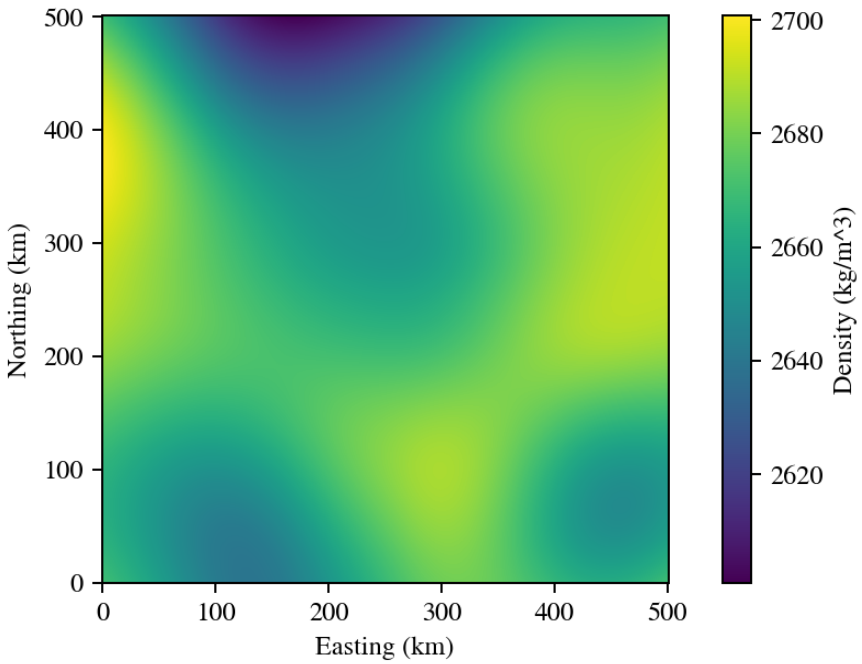

Figure (B.2) The generated density map using layers of Perlin noise. This density field contains two octaves of Perlin noise, configured with a base period of 500 km, a lacunarity of 2, and a persistence of 0.5. The density varies over an area of 500 km by 500 km. It is georeferenced to the same GEBCO area used for the study, shown in Fig. 5.13. However, it is shown with a false origin that is relative to the south-west corner of the GEBCO bathymetry. The base amplitude for the Perlin noise used to generate this density was set to 100 kg/m^3.

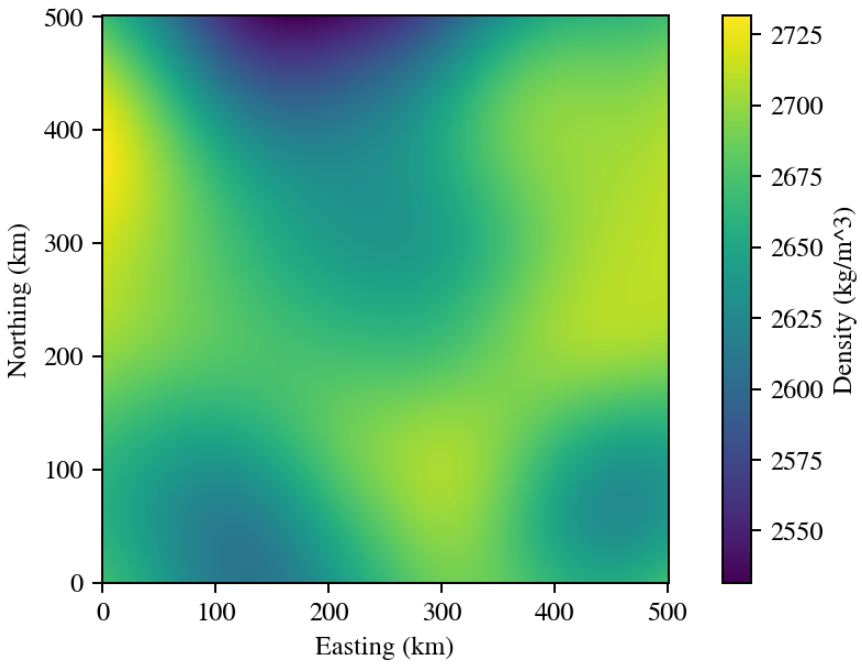

Figure (B.3) The generated density map using layers of Perlin noise. This density field contains two octaves of Perlin noise, configured with a base period of 500 km, a lacunarity of 2, and a persistence of 0.5. The density varies over an area of 500 km by 500 km. It is georeferenced to the same GEBCO area used for the study, shown in Fig. 5.13. However, it is shown with a false origin that is relative to the south-west corner of the GEBCO bathymetry. The base amplitude for the Perlin noise used to generate this density was set to 200 kg/m^3.

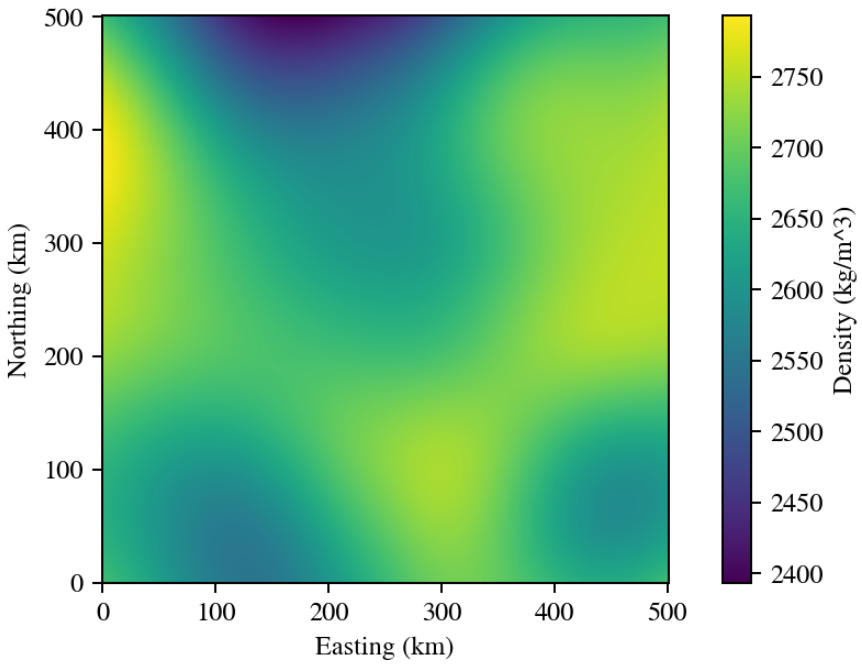

Figure (B.4) The generated density map using layers of Perlin noise. This density field contains two octaves of Perlin noise, configured with a base period of 500 km, a lacunarity of 2, and a persistence of 0.5. The density varies over an area of 500 km by 500 km. It is georeferenced to the same GEBCO area used for the study, shown in Fig. 5.13. However, it is shown with a false origin that is relative to the south-west corner of the GEBCO bathymetry. The base amplitude for the Perlin noise used to generate this density was set to 400 kg/m^3.

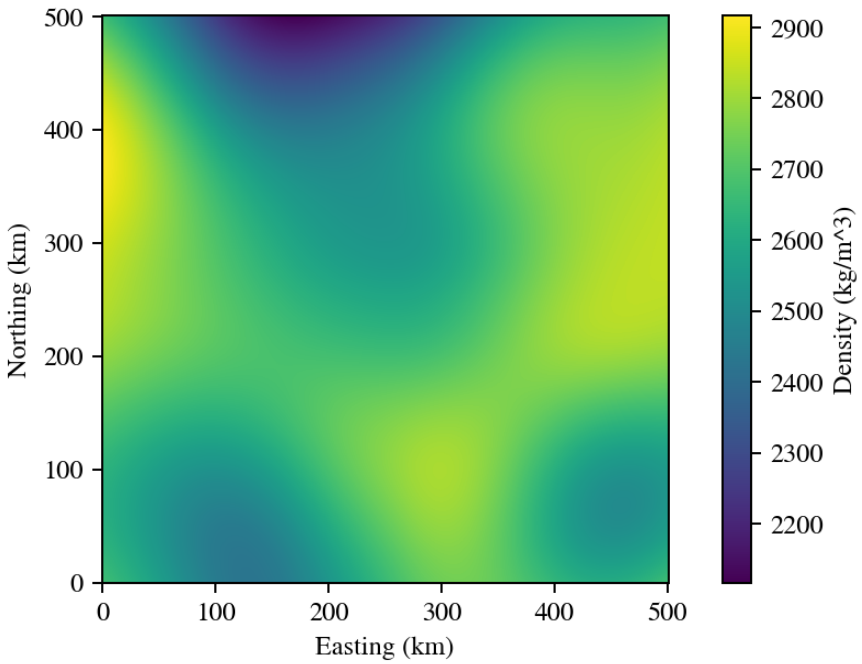

Figure (B.5) The generated density map using layers of Perlin noise. This density field contains two octaves of Perlin noise, configured with a base period of 500 km, a lacunarity of 2, and a persistence of 0.5. The density varies over an area of 500 km by 500 km. It is georeferenced to the same GEBCO area used for the study, shown in Fig. 5.13. However, it is shown with a false origin that is relative to the south-west corner of the GEBCO bathymetry. The base amplitude for the Perlin noise used to generate this density was set to 800 kg/m^3.

B.2 Particle Filter Results

Figs. B.6 to B.15 show all the results for the comparison between dead reckoning INS and particle filter TAN performance during the density uncertainty parameter study. The study varies the amplitude of the crustal density logarithmically to determine the effect of the constant density assumption on the performance of the TAN algorithm. Within the study there were 5 sessions with 10 runs each. The density amplitude was varied from $50\,\mathrm{kg/m^3}$ to $800\,\mathrm{kg/m^3}$. The results and implications of this study are discussed in section 5.6.2.

For each session there are two plots. One shows the comparison between the positional Euclidean error of the dead-reckoning (INS) and the particle filter (TAN). The other the zoomed in version of the TAN algorithms performance. This is because the details of the performance of the TAN algorithm are hard to see when compared with the dead-reckoning performance.

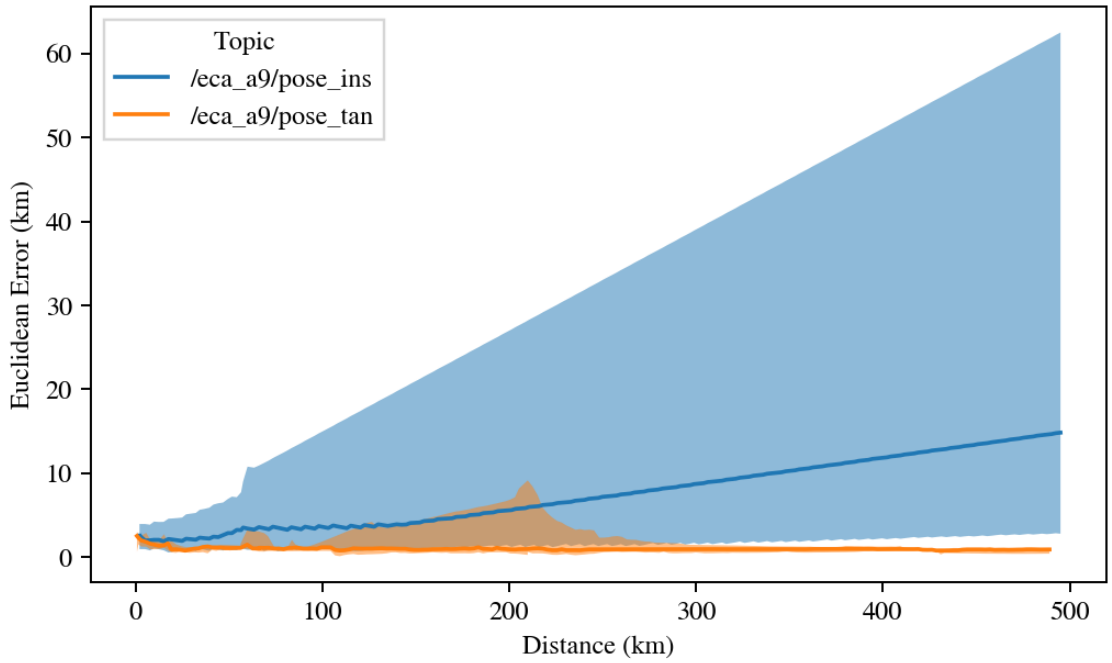

Figure (B.6) The comparison between the positional Euclidean error of dead-reckoning (INS) and the particle filter algorithm (TAN). The dark line is the median error of the positional estimate, the shaded area is the spread of the error of the positional estimate for all the runs within session 0. The runs are compared by the distance the vehicle has travelled, shown on the x-axis. The density variation of this session was $50\,\mathrm{kg/m^3}$.

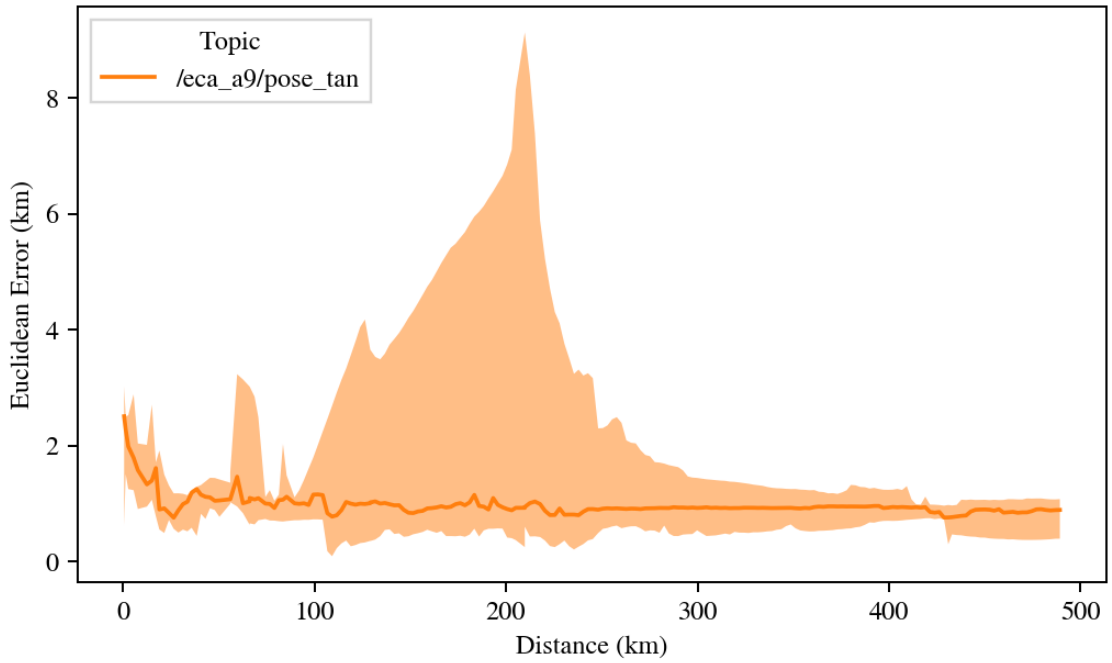

Figure (B.7) The positional estimate Euclidean error for the particle filter algorithm (TAN). The dark line is the median positional estimate Euclidean error, and the shaded area is the spread of the error within session 0. The density variation of this session was $50\,\mathrm{kg/m^3}$. All runs within a session are compared based on AUV distance travelled, shown on the x-axis.

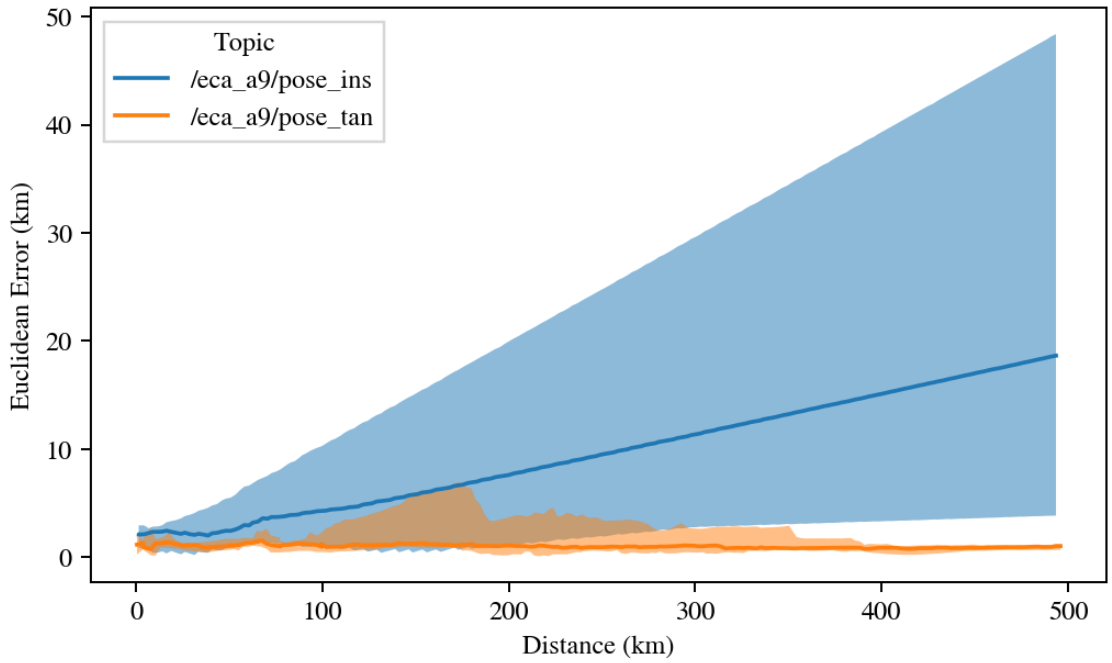

Figure (B.8) The comparison between the positional Euclidean error of dead-reckoning (INS) and the particle filter algorithm (TAN). The dark line is the median error of the positional estimate, the shaded area is the spread of the error of the positional estimate for all the runs within session 1. The runs are compared by the distance the vehicle has travelled, shown on the x-axis. The density variation of this session was $100\,\mathrm{kg/m^3}$.

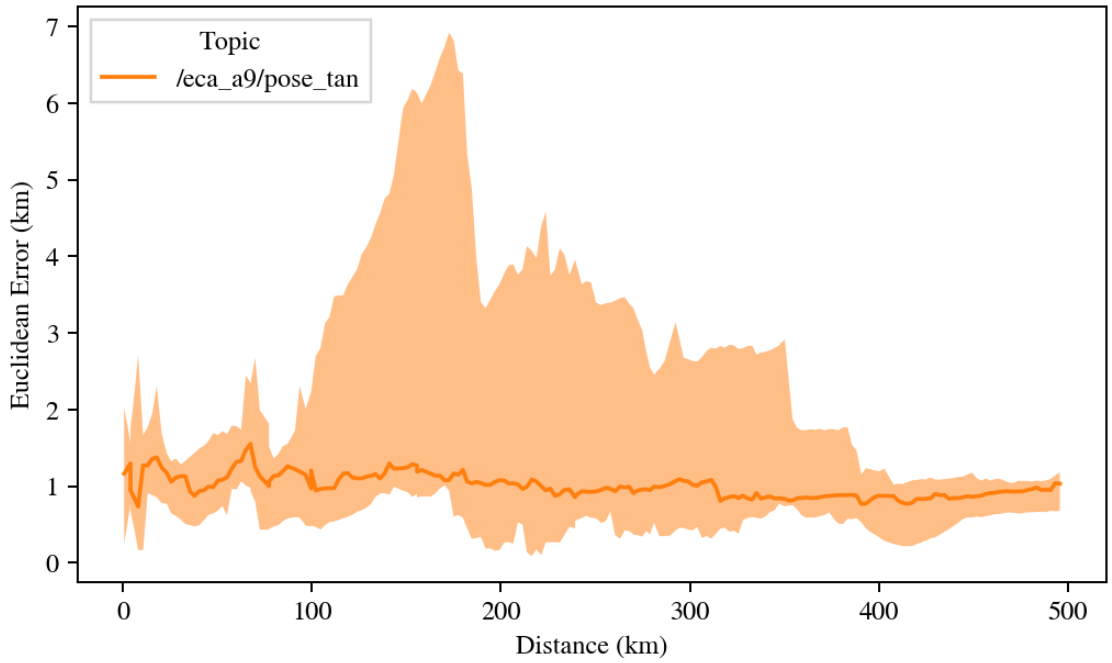

Figure (B.9) The positional estimate Euclidean error for the particle filter algorithm (TAN). The dark line is the median positional estimate Euclidean error, and the shaded area is the spread of the error within session 1. The density variation of this session was $100\,\mathrm{kg/m^3}$. All runs within a session are compared based on AUV distance travelled, shown on the x-axis.

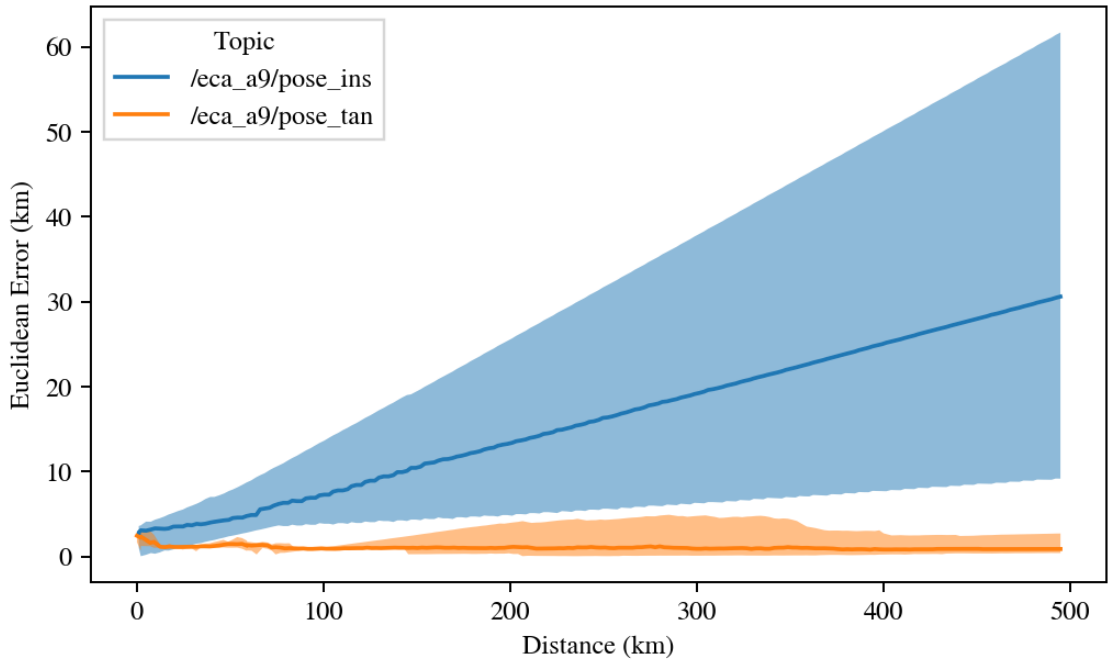

Figure (B.10) The comparison between the positional Euclidean error of dead-reckoning (INS) and the particle filter algorithm (TAN). The dark line is the median error of the positional estimate, the shaded area is the spread of the error of the positional estimate for all the runs within session 2. The runs are compared by the distance the vehicle has travelled, shown on the x-axis. The density variation of this session was $200\,\mathrm{kg/m^3}$.

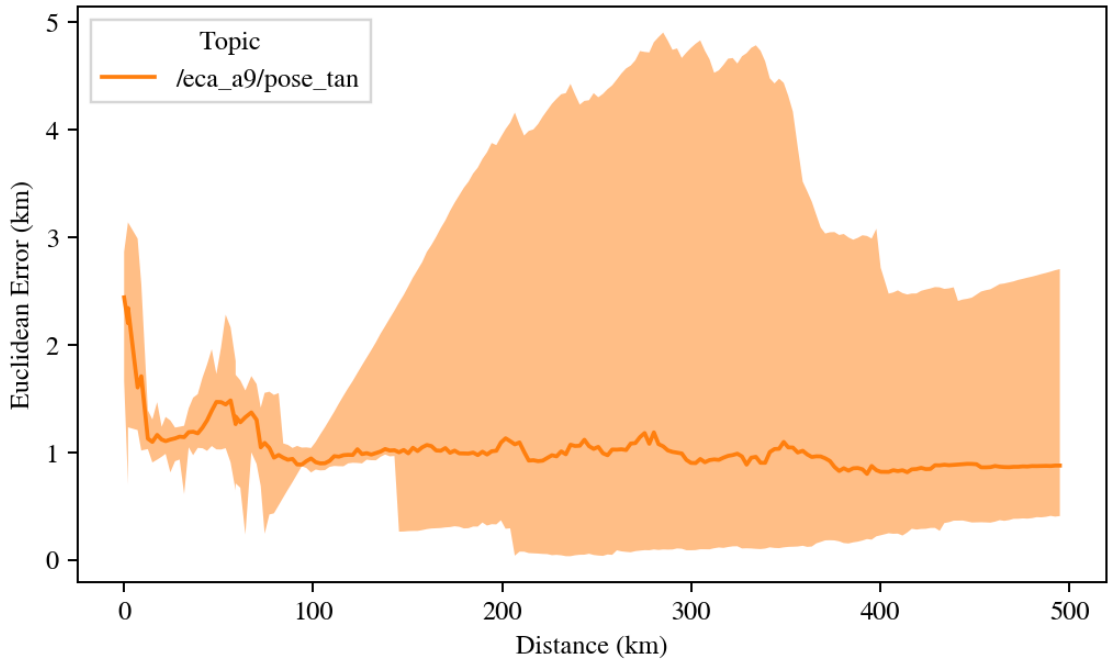

Figure (B.11) The positional estimate Euclidean error for the particle filter algorithm (TAN). The dark line is the median positional estimate Euclidean error, and the shaded area is the spread of the error within session 2. The density variation of this session was $200\,\mathrm{kg/m^3}$. All runs within a session are compared based on AUV distance travelled, shown on the x-axis.

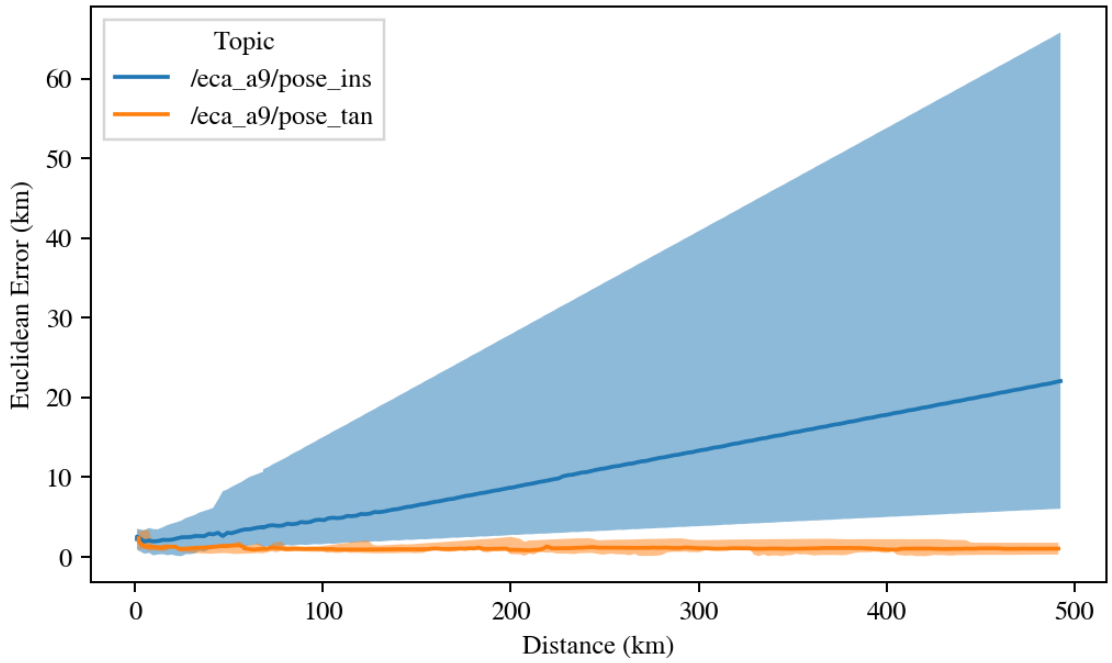

Figure (B.12) The comparison between the positional Euclidean error of dead-reckoning (INS) and the particle filter algorithm (TAN). The dark line is the median error of the positional estimate, the shaded area is the spread of the error of the positional estimate for all the runs within session 3. The runs are compared by the distance the vehicle has travelled, shown on the x-axis. The density variation of this session was $400\,\mathrm{kg/m^3}$.

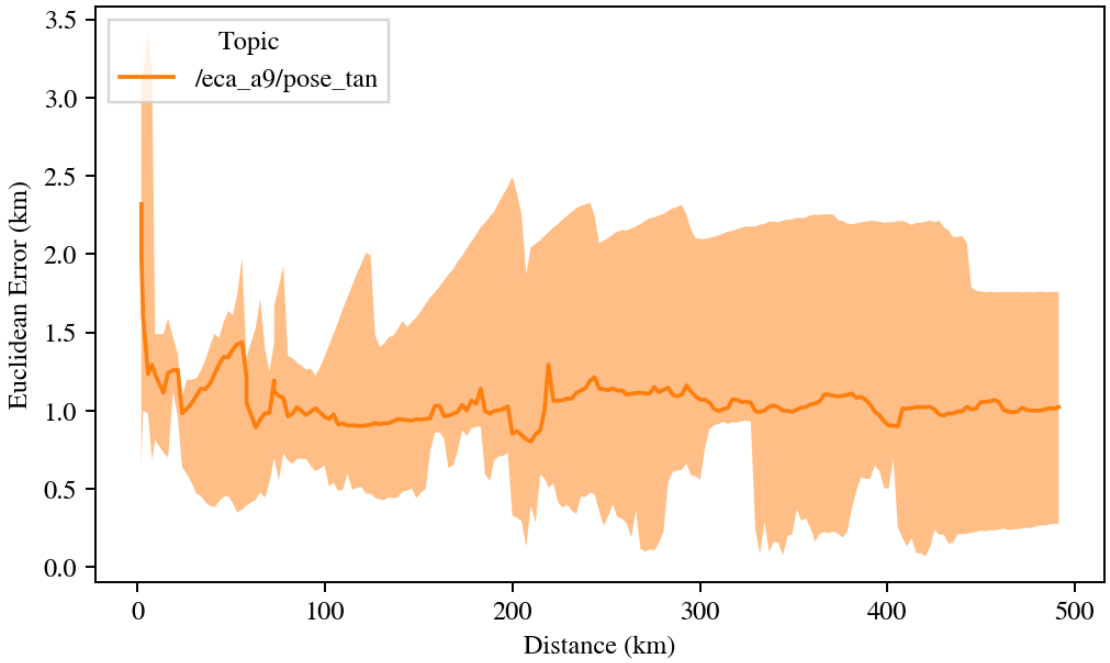

Figure (B.13) The positional estimate Euclidean error for the particle filter algorithm (TAN). The dark line is the median positional estimate Euclidean error, and the shaded area is the spread of the error within session 3. The density variation of this session was $400\,\mathrm{kg/m^3}$. All runs within a session are compared based on AUV distance travelled, shown on the x-axis.

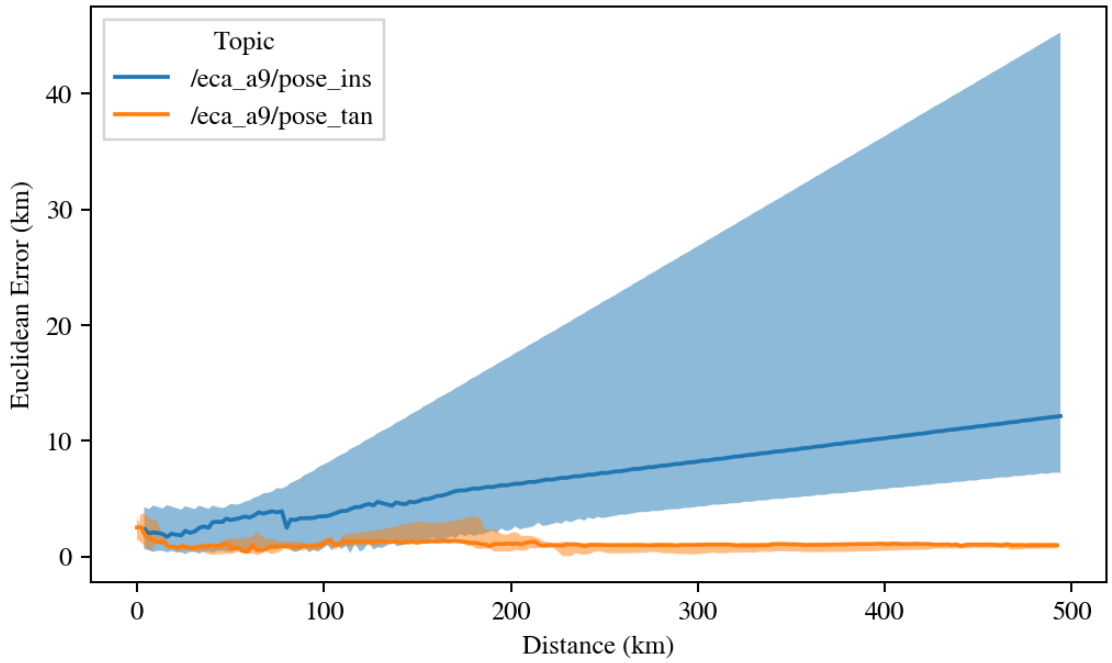

Figure (B.14) The comparison between the positional Euclidean error of dead-reckoning (INS) and the particle filter algorithm (TAN). The dark line is the median error of the positional estimate, the shaded area is the spread of the error of the positional estimate for all the runs within session 4. The runs are compared by the distance the vehicle has travelled, shown on the x-axis. The density variation of this session was $800\,\mathrm{kg/m^3}$.

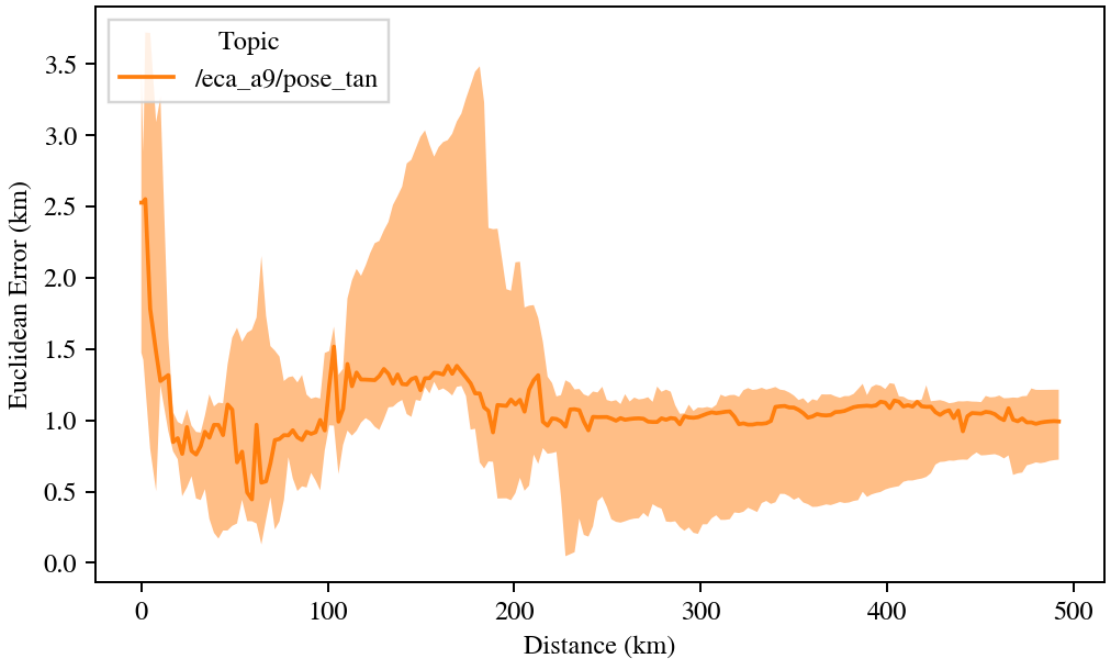

Figure (B.15) The positional estimate Euclidean error for the particle filter algorithm (TAN). The dark line is the median positional estimate Euclidean error, and the shaded area is the spread of the error within session 4. The density variation of this session was $800\,\mathrm{kg/m^3}$. All runs within a session are compared based on AUV distance travelled, shown on the x-axis.

Appendix C

Local Gravity Anomaly from Terrain in Python

Listing 1 shows the Python code implementation of the conversion from terrain to local gravity anomaly using the correlation between the gravity anomaly and the terrain. This algorithm is explained in more detail in sections 2.12.2 and 4.1.5.

```
1  import numpy as np
2  from tqdm import tqdm
3
4
5  def local_anomaly(bathy, bathy_xj, bathy_yi, interpolation_depth, i, j, m, n,
   ↪  density):
6      """
7      Calculate the local gravity anomaly using a window on a data set.
8
9      This calculated anomaly is not any official measure of the gravity anomaly.
10     It is a simplified version that approximates the gravity anomaly gradients
11     but is not correct in the sense of absolute values.
12
13     The function assumes a few parameter values. An average crustal density of
14     2670 kg/m^3, an average seawater density of 1027 kg/m^3, a universal gravity
15     constant of 6.674e-11 m^3 kg^-1 s^-2.
16
17     :param      bathy:                The bathymetry value matrix. The value at
18                                       row, col must correspond with the
19                                       coordinate (x[col], y[row]).
```

```
20      :type       bathy:                d x p np.array
21      :param      bathy_xj:             The x coordinate vector. Length must be
22                                        the same as the columns in bathy value
23                                        matrix.
24      :type       bathy_xj:             1 x p np.array
25      :param      bathy_yi:             The y coordinate vector. Length must be
26                                        the same as the rows in bathy value
27                                        matrix.
28      :type       bathy_yi:             1 x d np.array
29      :param      interpolation_depth:  The height at which the gravity anomaly
30                                        should be calculated. Positive heights are
31                                        above the water surface. Negative heights
32                                        are below the water surface.
33      :type       interpolation_depth:  float
34      :param      i:                    The starting row of the window.
35      :type       i:                    int
36      :param      j:                    The starting column of the window.
37      :type       j:                    int
38      :param      m:                    The number of rows in the window.
39      :type       m:                    int
40      :param      n:                    The number of columns in the window.
41      :type       n:                    int
42      :param      density:              The density value matrix. The value at
43                                        row, col must correspond with the
44                                        coordinate (x[col], y[row]). The average
45                                        crustal density is used if no density
46                                        matrix is provided.
47      :type       density:              d x p np.array
48
49      :returns:   A tuple of length 3. (x coordinate, y coordinate, gravity
50                  anomaly)
51      :rtype:     tuple (float, float, float)
52      """
53
54      crustal_density = 2670  # kg/m^3
55      # The density of sea water.
56      seawater_density = 1027  # kg/m^3
57      # The universal gravitation constant.
```

```
58      G = 6.674e-11  # m^3 kg^-1 s^-2
59      area = np.abs((bathy_xj[1] - bathy_xj[0]) * (bathy_yi[1] - bathy_yi[0]))
60
61      # m and n must be odd
62      # i, j, m, n are all in terms of indices.
63      # the location that the anomaly is being calculated at.
64      min_res = max(abs(bathy_xj[1] - bathy_xj[0]), abs(bathy_yi[1] - bathy_yi[0]))
65      x, y = bathy_xj[j + (floor(n / 2))], bathy_yi[i + (floor(m / 2))]
66      xx, yy = np.meshgrid(bathy_xj[j : j + n], bathy_yi[i : i + m])
67      dxx = np.abs(xx - x)
68      dyy = np.abs(yy - y)
69      height = bathy[i : i + m, j : j + n]
70      dzz = interpolation_depth - height / 2.0
71      if density is None:
72          density = crustal_density * np.ones(dxx.shape)
73
74      r = np.sqrt(np.square(dxx) + np.square(dyy) + np.square(dzz))
75      effective_density = np.zeros(dxx.shape)
76      np.putmask(effective_density, height >= 0, density)
77      np.putmask(effective_density, height < 0, seawater_density - density)
78
79      # Remove calculations using terrain that is too close
80      too_close_mask = r < min_res / 2.0
81      r[too_close_mask] = np.nan
82      if np.sum(too_close_mask) > 1:
83          print("WARNING: Threw out {} cells because they were to close.")
84          print(np.sum(too_close_mask))
85
86      volume = area * np.abs(height)
87      dzz = np.abs(dzz)
88      effective_mass = effective_density * volume
89      gravity_contribution = G * effective_mass * dzz / r**3
90      gravity_contribution[np.isnan(gravity_contribution)] = 0
91      return x, y, np.sum(gravity_contribution)
92
93
94  def local_anomaly_sweep(
95      bathy, bathy_xj, bathy_yi, interpolation_depth, m, n, istep, jstep, density
96  ):
```

```
97      """
98      Perform a sweep across a larger matrix to calculate multiple local gravity
99      anomalies using a moving window.
100
101     :param      bathy:                The bathymetry value matrix. The value at
102                                       row, col must correspond with the
103                                       coordinate (x[col], y[row]).
104     :type       bathy:                d x p np.array
105     :param      bathy_xj:             The x coordinate vector. Length must be
106                                       the same as the columns in bathy value
107                                       matrix.
108     :type       bathy_xj:             1 x p np.array
109     :param      bathy_yi:             The y coordinate vector. Length must be
110                                       the same as the rows in bathy value
111                                       matrix.
112     :type       bathy_yi:             1 x d np.array
113     :param      interpolation_depth:  The height at which the gravity anomaly
114                                       should be calculated. Positive heights are
115                                       above the water surface. Negative heights
116                                       are below the water surface.
117     :type       interpolation_depth:  float
118     :param      m:                    The number of rows in the window.
119     :type       m:                    int
120     :param      n:                    The number of columns in the window.
121     :type       n:                    int
122     :param      istep:                The number of rows to skip for every
123                                       window step.
124     :type       istep:                int
125     :param      jstep:                The number of columns to skip for every
126                                       window step.
127     :type       jstep:                int
128     :param      density:              The density value matrix. The value at
129                                       row, col must correspond with the
130                                       coordinate (x[col], y[row]). The average
131                                       crustal density is used if no density
132                                       matrix is provided.
133     :type       density:              d x p np.array
134
135     :returns:   A tuple of length 3. (x coordinate vector, y coordinate vector,
```

```
136                 anomaly value matrix)
137         :rtype:     tuple (1 x q np.array, 1 x r np.array, r x q np.array)
138         """
139         ivec = range(0, bathy.shape[0] - m, istep)
140         jvec = range(0, bathy.shape[1] - n, jstep)
141 
142         # Populate the anomaly field
143         anomaly = np.zeros((len(ivec), len(jvec)))
144         xs = list()
145         ys = list()
146 
147         # Calculate the local anomaly
148         for i, ibathy in tqdm(
149             enumerate(ivec), desc="Calculating anomalies", total=len(ivec),
                ↪  unit="anomaly"
150         ):
151             for j, jbathy in enumerate(jvec):
152                 x, y, anomaly[i, j] = local_anomaly(
153                     bathy,
154                     bathy_xj,
155                     bathy_yi,
156                     interpolation_depth,
157                     ibathy,
158                     jbathy,
159                     m,
160                     n,
161                     density,
162                 )
163                 if i == 0:
164                     xs.append(x)
165             ys.append(y)
166 
167         return np.asarray(xs), np.asarray(ys), anomaly
```

(1) A Python implementation of deriving the gravity anomaly from the correlation between gravity and the terrain. The implementation makes heavy use of Python's matrix library numpy. This speeds up the processing time substantially.

Appendix D

Horizontal Bounce Behaviour Algorithm

The horizontal bounce behaviour uses computational geometry to plan a quick safe pat back to within a safe distance of the operating region boundary. The horizontal bounce behaviour and the reasons for it existing are discussed in section 4.5.1.

The horizontal bounce behaviour makes the use of the geometry engine geometry engine open-source (GEOS), and linear algebra library (Eigen for C++, Numpy for Python). The GEOS library has bindings for Python as well as a C++ API. The final implementation that is now part of the DCAF framework is written in C++.

The horizontal bounce behaviour once triggered is as follows:

1. Determine the unit vector that defines the current vehicle heading.

$$\hat{u}_{vehicle} = [\cos(\theta_{vehicle}), \sin(\theta_{vehicle})]^T \tag{D.1}$$

2. Determine the two vectors that are perpendicular to $\hat{u}$, that will be used to define the direction of the circle's two centres.

$$R = \begin{bmatrix} \cos(\pi/2) & -\sin(\pi/2) \\ \sin(\pi/2) & \cos(\pi/2) \end{bmatrix} \hat{u}_1 = R \cdot \hat{u}_{vehicle} \hat{u}_2 = R^T \cdot \hat{u}_{vehicle} \tag{D.2}$$

3. Determine the centres of the two circles.

$$c_1 = x_{vehicle} + r \cdot \hat{u}_1 \tag{D.3}$$

$$c_2 = x_{vehicle} + r \cdot \hat{u}_2 \tag{D.4}$$

4. Create high resolution line ring defining the circles, C_1, C_2, using the circle centres c_1 and c_2 and the turn radius r.

5. Use the polygon defining the inner boundary area, B_{inner}, to compute the line paths, p_1 and p_2, that define the parts of C_1 and C_2 that are outside of the polygon B_{inner}.

$$[p_1, p_2] = B_{inner} \setminus (C_1 \cup C_2) \tag{D.5}$$

6. Determine the shortest parth length between p_1 and p_2 and assign it to p_{min}.

$$p_{min} = \text{minlength}(p_1, p_2) \tag{D.6}$$

7. Determine the intersection of the line path p_{min} with the inner boundary exterior by determining which boundary point for the line path p_{min} is furthest from the vehicle position $x_{vehicle}$, as one of the boundary points is approximately the vehicle position itself. Assign the resulting point to x_{wp}.

8. Determine the waypoint heading by taking the last two points in the line path p_{min} and determine the unit vector, $\hat{u}_{wp}$, representing vehicle heading at the waypoint x_{wp}.

9. Determine the vehicle heading, θ_{wp}, using the unit vector, $\hat{u}_{wp}$.

$$\theta_{wp} = \arctan 2(\hat{u}_{wp_y}, \hat{u}_{wp_x}) \tag{D.7}$$

10. The active waypoint is replaces with the waypoint determined by the bounce

behaviour, w_{wp} and θ_{wp} leading the vehicle back to a safe distance from the operating region boundary.

Appendix E

The ROS Parameter Study CLI

The ROS parameter study CLI is a tool developed for running parameter studies with existing ROS systems. This CLI along with the accompanying **rps** library (developed for loading the results of the parameter studies) were used for all parameter studies within this thesis. The RPS package's CLI interface has two commands. The first is **run**, the second is **extract**.

E.0.1 The Run Command

run is used to launch a full parameter study using a configuration file as input. A full example of the configuration file that was used for the gradiometer noise study can be found in appendix E.1. The merits of the configuration file and what part of the ROS parameter study features they support will be discussed in detail in this section.

For the parameter studies in this thesis a study consists of multiple sessions each containing multiple runs. A session's runs all share the same setting configuration, such as gradiometer noise standard deviation. A run refers to one execution of the simulation environment coupled with the ROS nodes. A session can contain one or more runs. This is useful when internal parameters for a node are randomly initialized. For example, the initial INS error and the initial particle positions for the particle filter. After all runs are complete the runs within a session contain the variation due to random initialization and the variation between sessions is due to the main parameters varied during the study.

The run command loads in the configuration file and then creates a set of bash commands that will be executed for a specific run. For each run a set of processes are launched according to the configuration file. These processes are run as child processes of the main ROS parameter study program, and their status is tracked with a progress bar to show overall study progress and individual run progress.

There are seven main keywords within the configuration file: env, processes, runs, launch, commands, record, and termination. Each will be explained with its snippet from the sample configuration file from appendix E.1.

The env section supports setting up a bash environment before any of the launch commands are run. This allows sourcing the needed ROS environment setup scripts. A few macros are supported, one of them, ${}, is shown in listing 2. This macro supports using bash environment variables within the configuration file. The ${} macro was written written to help avoid hard-coding in paths to files within the configuration file.

```
1 env:
2   scripts:
3     - ${HOME}/workspace/ros1_ws/devel/setup.sh
4
```

(2) A portion of the ROS parameter study configuration file showing the usage of the env section.

The processes and runs sections each support a single number for configuration (shown in listing 3). The processes field enables executing multiple runs in parallel. This number should be adjusted based on how powerful your computer is. If not keep it configured at 1. The runs field determines how many runs will be completed for each session. If there are no random initialized parameters internal to the nodes you are launching then keep this at 1. If there is randomly initialized internal parameters within nodes use a number large enough for reasonable coverage of the expected spread.

```
5 processes: 1
6
7 runs: 10
8
```

(3) A portion of the ROS parameter study configuration file showing the usage of the processes and runs section.

The launch field supports numerous entries each corresponds to an existing launch file (shown in listing 4). Each entry under the launch field supports the identifying name, package name, launch file name, and args fields. The name field is used to identify this launch file in the output log files that ROS parameter study gathers after each run, the package field for the name of the ROS package, the file field for the launch file name (expected to be within the launch folder inside the ROS package), and the args field which supports any number of key value pairs for setting launch file arguments.

```
launch:
  - name: gazebo
    package: ds_gazebo_worlds
    file: world.launch
    args:
      world_name: $(find ds_gazebo_worlds)/worlds/gebco_01.world
      gui: false

  - name: model_upload
    package: ds_uuv_nav
    file: eca_a9.launch
    args:
      x: 700.
      y: 700.
      z: -50.
      yaw: 0.7071

  - name: ins
    package: ds_localization
    file: ins.launch

  - name: controller
    package: ds_uuv_nav
    file: controller.launch

  - name: maps
    package: ds_localization
```

```
36      file: map_gebco_01.launch
37      args:
38        bathymetry_gt_geotiff_path: $(find
          ↪ ds_gazebo_worlds)/models/gebco_01/bathymetry.tif
39        anomaly_gt_geotiff_path: $(find
          ↪ ds_gazebo_worlds)/models/gebco_01/anomaly-const-density.tif
40        anomaly_map_geotiff_path: $(find
          ↪ ds_gazebo_worlds)/models/gebco_01/anomaly-const-density.tif
41
42    - name: tan
43      package: ds_localization
44      file: tan.launch
45      args:
46        update_frequency: 1.
47        heading_jitter_factor: 0.5
48        x_jitter_factor: 0.5
49        y_jitter_factor: 0.5
50        gradient_std_dev: [$(linspace 0.001, 0.7, 8)]
51
52    - name: sensors
53      package: ds_sensors
54      file: sensors.launch
55      args:
56        gradiometer_frequency: 0.5
57        gradiometer_magnitude_std_dev: 0.0
58        gradiometer_heading_std_dev: [$(linspace 0, 0.7, 8)]
59
60    - name: rviz
61      package: ds_uuv_nav
62      file: rviz.launch
63
```

(4) A portion of the ROS parameter study configuration file showing the usage of the launch section. It also shows usage of the linspace and find macro.

The are a few examples of other macros within listing 4. They all have the syntax $(cmd param1, param2, ...). The first is the find macro (shown on line 38). It works exactly like the find macro in ROS launch files. It will be replaced by the full path to the ROS package using the rospkg Python package to perform the lookup. This

avoids having to hard-code the location of ROS packages within the configuration file. The next macro is linspace (shown on line 50). It expects exactly three arguments. It behaves exactly like Python package numpy's linspace function because this function is used for this macro. The macro will be replaced by numbers, separated by commas so they are valid Yaml, from the first parameter to the last parameter in exactly the last parameters amount of steps. For example parameters $1, 2, 3$ would yield $1.0, 1.5, 2.0$. There is another similar macro, logspace, that has the same behaviour but instead of a linear step it takes a number of logarithmic spaced steps. These two macros make it easier to calculate evenly spaced variation of parameters linear or logarithmic for the parameter study.

If a launch file parameter specified using the key value pairs under the args field is a list of values it must have the same length as all other parameters that have a list of values. This is what determines how many sessions the study will have. In our example there are eight sessions because the gradient standard deviation noise and particle filter setting are varied together.

The commands section provides the ability to execute any command along with the execution of a run (shown in listing 5).

```
64 commands:
65   - name: go_to
66     cmd: "rosservice call /eca_a9/go_to \"{
67                                             waypoint: {
68                                               header: {
69                                                 seq: 0,
70                                                 stamp: {
71                                                   secs: 0,
72                                                   nsecs: 0
73                                                 },
74                                                 frame_id: ''
75                                               },
76                                               point: {
77                                                 x: 4300.,
78                                                 y: 4300.,
79                                                 z: -50.0
80                                               },
```

```
                                                max_forward_speed: 3.0,
                                                heading_offset: 0.0,
                                                use_fixed_heading: false,
                                                radius_of_acceptance: 10.0
                                              },
                                              max_forward_speed: 3.0,
                                              interpolator: 'lipb'
                                            }\""
    delay: 25.
```

(5) A portion of the ROS parameter study configuration file showing the usage of the commands section.

The name field provides the identifier for collecting log files for the process resulting from this command, the cmd field is the actual command to run, and the delay field is used to delay the execution of the command for a specified amount of seconds from the start of a run. In this case it is used to issue the go_to command to the ECA vehicle waypoint controller once all other processes have initialized. The 25 seconds was chosen by trial an error based on the workstation the simulation was done on.

The record section is used to explicitly configure which topics are recorded by rosbag during each run (shown in listing 6). The namespace field configures the namespace that the topics are under, and the topics field contains all the topic names in a list. The bag files can later be extracted to csv files and organized based on the topic names using the command **extract**. lst:rps.config.commands).

```
record:
  namespace: 'eca_a9'
  topics:
    - pose_gt
    - pose_ins
    - altimeter
    - pose_tan
    - tan/pose2d
    - gravity_gradiometer
```

(6) A portion of the ROS parameter study configuration file showing the usage of the record section.

Finally, the termination section allows configuring which node will monitor the run and determine when the run is complete (shown in listing 7). The package field is the name of the ROS package, the node field is the name of the file to run, the args fields supports key value pairs of node arguments, the namespace field determines the namespace the node will be pushed into when it is started (required when the topics the node is monitoring are within a namespace), and the timeout field which will abort the run if the run takes longer than this amount of seconds.

```
101 termination:
102   package: ds_uuv_nav
103   node: monitor_goal.py
104   args:
105     x: 4300
106     y: 4300
107     tolerance: 30
108     topic: pose_ins
109   namespace: eca_a9
110   timeout: 100000
```

(7) A portion of the ROS parameter study configuration file showing the usage of the termination section.

The termination criteria node requires further explanation. The node can be written to monitor any topics it has access to and can determine progress however it sees fit. This allows the termination criteria to be very flexible and custom for a specific application. The only requirement is for the application to conform to a specific protocol on its stdout. It must only print out a measure of the error as a number followed by a new line, and when it deems the termination criteria has been met print out the string *Done*. The error measure can be any measure whatsoever. The first value read from standard out is used as the starting error and each consecutive read error is compared to the initial error to determine progress and display this using the progress bars. When *Done* is read from stdout, the progress bar is set to complete and managed sub-processes are killed. This ends the run, closes log files, closes active bag files, and shuts down all nodes (including the Gazebo simulation environment).

The results of a run are stored in the output directory. This can be controlled by

the current working directory when using the ROS parameter study CLI. An example of the output directory file structure is shown in listing 8.

```
├── 00_00_00_2021-12-13-14-56-29.bag
├── 00_00_01_2021-12-13-15-32-06.bag
├── 00_00_02_2021-12-13-16-08-53.bag
├── 00_00_03_2021-12-13-16-46-34.bag
...
├── 00_input_configuration.yml
├── 00_parsed_configuration.yml
...
├── 03_00_00_2022-01-11-17-58-31.bag
├── 03_00_01_2022-01-11-19-30-31.bag
├── 03_00_02_2022-01-11-21-01-57.bag
...
├── 03_07_07_2022-01-16-15-20-50.bag
├── 03_07_08_2022-01-16-16-03-04.bag
├── 03_07_09_2022-01-16-16-45-36.bag
├── 03_input_configuration.yml
├── 03_parsed_configuration.yml
├── log
...
│      ├── 03_00_00
│      │      ├── controller.log.txt
│      │      ├── gazebo.log.txt
│      │      ├── go_to.log.txt
│      │      ├── ins.log.txt
│      │      ├── maps.log.txt
│      │      ├── model_upload.log.txt
│      │      ├── rosbag.log.txt
│      │      ├── rviz.log.txt
│      │      ├── sensors.log.txt
```

```
│       │       └── tan.log.txt
│       ├── 03_00_01
│       │       ├── controller.log.txt
│       │       ...
│       │       └── tan.log.txt
...
│       ├── 03_07_09
│       │       ├── controller.log.txt
│       │       ├── gazebo.log.txt
│       │       ├── go_to.log.txt
│       │       ├── ins.log.txt
│       │       ├── maps.log.txt
│       │       ├── model_upload.log.txt
│       │       ├── rosbag.log.txt
│       │       ├── roscore.log.txt
│       │       ├── rviz.log.txt
│       │       ├── sensors.log.txt
│       │       └── tan.log.txt
│       └── 03_main.log.txt
```

(8) The directory structure of the output directory for the parameter study. Each run is given a unique identifier compromised of the study ID, session ID, and run ID (01_03_09 for study 1, session 3, and run 9, where 0 is the starting number). Ellipses show missing output (the over 2000 files in this directory don't fit into a reasonable example)

Each run is identified by a unique ID comprised of a combination of the study ID, session ID, and the run ID. For example, study 0, session 3, run 5 would have the ID 00_03_05. This ID is later used to read the extracted .csv files for data analysis and plotting. Ellipses show missing file for brevity. Listing 8 shows the the output directory after running four studies. There are ten runs in each of the eight sessions within each study. Leading to 80 runs per study. For each study there is the input configuration file which is a copy of the configuration file used for that study. There is also a copy of the parsed configuration file that is the configuration file with the macros applied (file paths, and numbers have replace the original macros to make

the Yaml file valid Yaml). These two files are useful for debugging errors in the configuration file. Log files are also collected to help with debugging.

All processes including the main processes are logged into files within the log directory. The files are grouped by their unique ID. Each log directory contains files for that specific run organized by the names given within the configuration file using the name field. The main process is logged to a file identified with the name main. These log files contain all the information one would usually see printed to the terminal when running a launch file. Very helpful for when the fiftieth of eighty runs crashes after two days and one needs to investigate why. Now that the bag files exist the **extract** command is required.

E.0.2 The extract command

The extract command was written to extract all topics within the record section of the configuration file into .csv files that can be read into any language of choice, in this case Python. For a large study this can take half an hour or more. The resulting directory structure is shown in listing 9.

```
...
├── 00_input_configuration.yml
├── 00_parsed_configuration.yml
...
├── 03_00_00_2022-01-11-17-58-31.bag
├── 03_00_00_|eca_a9|altimeter.csv
├── 03_00_00_|eca_a9|gravity_gradiometer.csv
├── 03_00_00_|eca_a9|pose_gt.csv
├── 03_00_00_|eca_a9|pose_ins.csv
├── 03_00_00_|eca_a9|pose_tan.csv
├── 03_00_00_|eca_a9|tan|pose2d.csv
├── 03_00_01_2022-01-11-19-30-31.bag
...
```

(9) Abbreviated layout of the results directory after the extract command is run. The .csv files are the result of the extract command.

Each topic has been extracted to a .csv file with the unique ID of the run with

the full topic name. Forward slashes are not valid in file names so they have been replaced by the — symbol. This structure allows rps to intelligently read in the csv files in to organized pandas data frames (the Python data analysis library used for data analysis).

The rps library

The ROS parameter study package comes with both a command line interface to run and extract the parameter study, as well as a importable library to prepare the data for analysis. A example of the rps library's usage is shown in listing 10.

```
1  from pathlib import Path
2  import ros_parameter_study as rps
3
4  directories = [
5      Path("./gebco_01/gradiometer_noise/results"),
6  ]
7  globs = ["03*.csv"]
8  data = rps.DataAccessor(directories, globs)
9  study = data.with_topics(
10     [
11         "/eca_a9/pose_tan",
12         "/eca_a9/tan/pose2d",
13         "/eca_a9/pose_ins",
14         "/eca_a9/pose_gt",
15     ],
16     subsample_factors=[
17         10,
18         1,
19         100,
20         100,
21     ],
22 )
```

(10) Code snippet to briefly show the power of the rps library. Once imported the rps library gives access to the DataAccessor which can be used to control the loading in of the data generated by the parameter study.

Line 2 shows the way that the library is imported. Line 4-6 defines the directories to look in for the extracted files from the parameter study. Line 7 defines the globs

used for each directory. A glob is a way of string matching file names. The glob 03*.csv matches all the .csv files that are part of the study three. Line 8 initializes the DataAccessor class defined in the rps library. It uses the directory and glob and find all matching .csv files organize them by study, session, and run. It then determines which topics are available for each run and provides one main method with_topics that allows custom loading of the files using pandas built-in read_csv function.

Line 9-22 shows an example usage of the with_topics method of the DataAccessor class. It accepts a list of topic names that you would like to have accessible, as well as supporting a different sub-sample factor for each topic. If some topics are collected and recorded at 10 Hz and a run takes 30 min, sub-sampling during the file read process can save a lot of time when you really only need a data point at most every 10 s. The with_topic method does not load the data into pandas data frames yet. The data is lazy loaded when the study is looped through (shown in listing 11).

```
24 from functools import partial
25
26 for session in study.sessions():
27     for run in session.runs():
28         dfs_by_topic = run.to_comparable_dataframes(
29             key="%time",
30             in_place_transformers=[
31                 partial(
32                     add_run_relative_column_by_key,
33                     key="%time",
34                     output_key="run_time",
35                 ),
36                 partial(
37                     add_distance_column,
38                     output_key="distance",
39                     x_fields=["field.pose.pose.position.x", "field.x"],
40                     y_fields=["field.pose.pose.position.y", "field.y"],
41                 ),
42                 partial(
43                     add_run_relative_column_by_key,
44                     key="distance",
45                     output_key="run_distance",
```

```
46                 ),
47             ],
48         )
49
50         tan_df = dfs_by_topic["/eca_a9/pose_tan"]
51         gt_df = dfs_by_topic["/eca_a9/pose_gt"]
52         tan_x = tan_df["field.pose.pose.position.x"].to_numpy()
53         gt_x = gt_df["field.pose.pose.position.x"].to_numpy()
54         error = tan_x - gt_x
```

(11) Code snippet to briefly show the usage of the lazy data loading using the rps library. The rps library supports making data streams at different collection rates directly comparable using rps convenience functions.

The study object API makes use of Python's standard iterator protocol. Line 26-27 show how to loop over sessions and runs. The run object supports one main method to_comparable_dataframes. This method performs the loading of the data, the in place transformation of the resulting data frames, and makes sure the data frames of different topics are directly comparable using a a column that all topics have. The key keyword argument on line 29 determines which column is used to make the data frames comparable. Behind the scenes this enforces that all topics have that column in their tabular data, that enough tabular data exists to compare it to other tables, that all tables are sub-sampled to equalize the frequency with the slowest frequency data, and that all tables have equal number of entries. For example when the key is time, afterwards all the topics within a run are directly comparable (you can add and subtract the arrays directly and this is valid) using the absolute time stamp within the table. After this step a number of in place transformers supported by the rps library can be applied.

An in place transformer is just a function that takes in the dfs_by_topic as an argument an applies any in place transformation to some or all of the run's topic's tabular data. There are three examples of in place transformers used in listing 11. The add_run_relative_column_by_key compares all the topics within a run using a a column they all share, in this case %time, and adds a column with the run relative column corresponding to that key in a different column. The is useful to make multiple runs comparable because the original time column is absolute time. Subtracting the

earliest time that any of the topics within a run contain yields a run relative time that enables comparison between runs that happened at different absolute times. This same function can be used to create a run relative column by a different key such as distance travelled.

Different runs, though they may travel the same approximate distance, can take very different amounts of time to simulate. This leads to the relative run time being less of a good column to compare different runs with. Therefore, the run relative distance can be used once it is created by the other in place transformer. This final way of comparing multiple different runs was used to created the error spread comparison figures within section 5.6.

This is a brief overview of rps package. This package was developed by me as a necessity for running parameter studies within the ROS environment. A package providing this functionality did not exist. The amount of development time was substantial, however, the parameter studies would have taken substantially more time without it. This tool will also be released to the research community. The rps package is separate from the rest of my work and therefore can be released as a package on its own. The rps package will provide a tool that was missing from the robotics research community. Additionally, this package will be used within our own lab by other students and researcher who wish to perform a parameter study within the ROS environment, the main tool-chain used within the ISL.

E.1 Full Configuration File

This section contains the interrupted example of a ROS parameter study configuration file for the gradiometer noise parameter study.

```
1 env:
2   scripts:
3     - ${HOME}/workspace/ros1_ws/devel/setup.sh
4
5 processes: 1
6
7 runs: 10
8
```

```yaml
launch:
  - name: gazebo
    package: ds_gazebo_worlds
    file: world.launch
    args:
      world_name: $(find ds_gazebo_worlds)/worlds/gebco_01.world
      gui: false

  - name: model_upload
    package: ds_uuv_nav
    file: eca_a9.launch
    args:
      x: 700.
      y: 700.
      z: -50.
      yaw: 0.7071

  - name: ins
    package: ds_localization
    file: ins.launch

  - name: controller
    package: ds_uuv_nav
    file: controller.launch

  - name: maps
    package: ds_localization
    file: map_gebco_01.launch
    args:
      bathymetry_gt_geotiff_path: $(find
      ↪ ds_gazebo_worlds)/models/gebco_01/bathymetry.tif
      anomaly_gt_geotiff_path: $(find
      ↪ ds_gazebo_worlds)/models/gebco_01/anomaly-const-density.tif
      anomaly_map_geotiff_path: $(find
      ↪ ds_gazebo_worlds)/models/gebco_01/anomaly-const-density.tif

  - name: tan
    package: ds_localization
    file: tan.launch
```

```
45      args:
46        update_frequency: 1.
47        heading_jitter_factor: 0.5
48        x_jitter_factor: 0.5
49        y_jitter_factor: 0.5
50        gradient_std_dev: [$(linspace 0.001, 0.7, 8)]
51
52    - name: sensors
53      package: ds_sensors
54      file: sensors.launch
55      args:
56        gradiometer_frequency: 0.5
57        gradiometer_magnitude_std_dev: 0.0
58        gradiometer_heading_std_dev: [$(linspace 0, 0.7, 8)]
59
60    - name: rviz
61      package: ds_uuv_nav
62      file: rviz.launch
63
64  commands:
65    - name: go_to
66      cmd: "rosservice call /eca_a9/go_to \"{
67                                              waypoint: {
68                                                header: {
69                                                  seq: 0,
70                                                  stamp: {
71                                                    secs: 0,
72                                                    nsecs: 0
73                                                  },
74                                                  frame_id: ''
75                                                },
76                                                point: {
77                                                  x: 4300.,
78                                                  y: 4300.,
79                                                  z: -50.0
80                                                },
81                                                max_forward_speed: 3.0,
82                                                heading_offset: 0.0,
83                                                use_fixed_heading: false,
```

```
84                                          radius_of_acceptance: 10.0
85                                        },
86                                        max_forward_speed: 3.0,
87                                        interpolator: 'lipb'
88                                      }\""
89      delay: 25.
90
91  record:
92    namespace: 'eca_a9'
93    topics:
94      - pose_gt
95      - pose_ins
96      - altimeter
97      - pose_tan
98      - tan/pose2d
99      - gravity_gradiometer
100
101 termination:
102   package: ds_uuv_nav
103   node: monitor_goal.py
104   args:
105     x: 4300
106     y: 4300
107     tolerance: 30
108     topic: pose_ins
109   namespace: eca_a9
110   timeout: 100000
```

(12) An example configuration file for the parameter study CLI. This configuration file uses the Yaml file format and is easily readable in Python.