UTS UNIVERSITY OF TECHNOLOGY SYDNEY

Underwater glider navigation techniques with ensemble-based estimation and streamline-based planning

by Kwun Yiu Cadmus To

Thesis submitted in fulfilment of the requirements for the degree of

Doctor of Philosophy

under the supervision of Prof. Robert Fitch,
Dr Chanyeol Yoo, and Dr Felix H. Kong

University of Technology Sydney
Faculty of Engineering and Information Technology

September 2023

Certificate of Original Authorship

I, Kwun Yiu Cadmus To, declare that this thesis, is submitted in fulfilment of the requirements for the award of Doctor of Philosophy, in the School of Mechanical and Mechatronic Engineering at the University of Technology Sydney.

This thesis is wholly my own work unless otherwise referenced or acknowledged. In addition, I certify that all information sources and literature used are indicated in the thesis.

I certify that the work in this thesis has not previously been submitted for a degree nor has it been submitted as part of the requirements for a degree at any other academic institution except as fully acknowledged within the text. This thesis is the result of a Collaborative Doctoral Research Degree program with The Defence Science and Technology Group of the Australian Department of Defence.

This research is supported by the Australian Government Research Training Program.

Kwun Yiu Cadmus To

September 2023

Underwater Glider Navigation Techniques with Ensemble-based Estimation and Streamline-based Planning

by

Kwun Yiu Cadmus To

A thesis submitted in fulfilment of the requirements for the degree of Doctor of Philosophy

Abstract

Underwater gliders are highly appealing for oceanic applications due to their endurance, allowing them to operate over hundreds of kilometres for several months. However, their limited thrust poses significant challenges for autonomous navigation, especially under the influence of ocean currents and the lack of reliable long-term forecasts. This thesis addresses three critical problems in estimation and path planning to reduce the reliance on remote supervision of glider operations.

The first problem focuses on flow field estimation with limited computational resources. We propose a novel algorithmic framework that leverages dedicated environmental estimation systems, such as those from the Australian Bureau of Meteorology (BOM), to extract recurring flow patterns from their ensemble outputs. These patterns serve as building blocks to reconstruct an estimate of environmental behaviour. This approach ensures constant computational time complexities for updating estimates from vehicle-measured flow velocities and providing flow velocity estimates at given positions, contrasting with Gaussian process (GP)-based approaches that can take cubic computational time for updates or estimates.

The second problem, which is not well-studied in the literature, involves determining fixed controls for underwater gliders to reach another position under the influence of

ocean currents, thus solving the broader path planning issue. We develop the foundations of streamline-based control theory, inspired by oceanographic concepts, allowing us to constrain the search for fixed vehicle velocities without disregarding feasible solutions. Through simulated environments using synthetic flow patterns, and the Eastern Australian Current (EAC) flow pattern based on data from the Australian BOM, we demonstrate that this constrained technique significantly improves path quality by over 35% compared to unconstrained searches with similar computation budgets.

The third problem examines new methods to improve upon the standard approach of generating feasible paths for vehicles in strong ocean currents, which typically involves extending a search tree using random controls. We propose measures of reachability and a filtered approach to connect to random positions more effectively, accounting for kinematic feasibility with streamline-based control theory. Simulations in environments with vortices show that our methods achieve higher connection rates than traditional Euclidean distance-based quantifications of reachability. Moreover, in scenarios using real forecast data from the Australian BOM, our methods produce initial solutions earlier than the standard random controls approach.

The significance of this work lies in providing an automated alternative to traditional underwater glider navigation workflows, transforming them into autonomous vehicles rather than mere tools. A robust automated navigation system for underwater gliders has the potential to revolutionise marine biology and oceanography research by simplifying multi-vehicle operations. Additionally, streamline-based control theory offers promising applications beyond oceanography, providing innovative solutions for disciplines dealing with similar flow fields, such as electromagnetic fields and gravitational fields.

Acknowledgements

I'd like to first acknowledge my supervisor Prof Robert Fitch and my co-supervisors Dr Chanyeol Yoo and Dr Felix Kong. Rob, you have an incredible ability to summarise dense technical information that never ceases to amaze me. I thank you for lifting the quality of our academic publications with this skill. Chanyeol, you taught me how to write academic publications in the first place. Felix, you taught me how to appreciate the value of precise communication. Thank you both for your time and patience guiding me throughout this journey. I'd like to also thank Dr Stuart Anstee who believed in me and have put me on this path. I appreciate the things I have learnt from this endeavour that I would have otherwise missed.

There are also some co-authors of some publications I'd like to thank: Dr James Lee, Dr Brian Lee, and (soon to be) Dr Stefan Kiss. Thank you all for working together with me, and putting up with me when I get nitpicky about details. I will never forget the times when we'd be working together on the same document at the same time.

I would also like to thank my fellow CDMRG members including Anthony, Diluka, Ed, Fred, Giuseppe, Jayant, Kavindie, Kosta, Lan, Mahdi, Mason, Mitchell, Nathan, Philip, and Vera, who spent time to read and share thoughts about academic papers. Special thanks to Graeme, Brian, Jen, and Andre who have taken/take even more extra time out of their schedules to organise the operation of this reading group, which has given us all the chance to develop our communication skills.

My heartfelt appreciation goes to my friends and family for their unwavering support and encouragement. Their belief in my abilities and their patience during my moments of stress have been a constant source of strength.

My advice for managing time for big tasks is the same as the one for git: "Commit early, commit often."

This big task took 902 hours.

Contents

List of Figures

List of Tables

List of Algorithms

List of Theorems and Definitions

Acronyms

1-norm Manhattan norm

2-norm Euclidean norm

ABIT* advanced batch informed tree

ADCP acoustic Doppler current profiler

AIS automatic identification system

AIT* adaptive informed tree

AOS adaptive optimistic steering

ARD automatic relevance determination

AUV autonomous underwater vehicle

BBD balanced box decomposition

BIT* batch informed tree

BOM Bureau of Meteorology

BVP boundary-value problem

CFD computational fluid dynamics

DCT discrete cosine transform

DMD dynamic mode decomposition

DO dynamically orthogonal

EAC Eastern Australian Current

EM expectation maximisation

EnKF ensemble Kalman filter

FMT fast marching tree

GP Gaussian process

GRBF Gaussian radial basis function

HPC high performance computing

iSST informed stable sparse rapidly exploring random tree

KF Kalman filter

KO kernel observer

LOOCV leave-one-out cross-validation

LS least squares

LSB lower speed bound

MCTS Monte Carlo tree search

MDP Markov decision process

MPC model predictive control

MPLB motion planning using lower bounds

NEMO nucleus for European modelling of the ocean

OUM ordered upwind method

PCA principal component analysis

PDE partial differential equation

PIV particle image velocimetry

POD proper orthogonal decomposition

PRM probabilistic roadmap

RMS root-mean-square

ROM reduced-order modelling

ROMS regional ocean modelling system

RRG rapidly exploring random graph

RRT rapidly exploring random tree

SE squared exponential

SE-ARD squared exponential-automatic relevance determination

SQP sequential quadratic programming

SST stable sparse rapidly exploring random tree

SVD singular value decomposition

SVM support vector machine

TP-BVP two-point boundary-value problem

TR-PIV time-resolved particle image velocimetry

VF-RRT vector field rapidly exploring random tree

WLS weighted least squares

List of Publications

1. **K. Y. C. To**, K. M. B. Lee, C. Yoo, S. Anstee and R. Fitch "Streamlines for motion planning in underwater currents," *Proc. of IEEE ICRA*, May 2019.

2. **K. Y. C. To**, J. J. H. Lee, C. Yoo, S. Anstee and R. Fitch "Streamline-based control of underwater gliders in 3D environments," *Proc. of IEEE CDC*, December, 2019.

3. S. H. Kiss, **K. Y. C. To**, C. Yoo, R. Fitch and A. Alempijevic "Minimally invasive social navigation," *Proc. of ARAA ACRA*, December, 2019.

4. **K. Y. C. To**, C. Yoo, S. Anstee and R. Fitch "Distance and steering heuristics for streamline-based flow field planning," *Proc. of IEEE ICRA*, May, 2020.

5. **K. Y. C. To**, F. H. Kong, K. M. B. Lee, C. Yoo, S. Anstee and R. Fitch "Estimation of spatially correlated ocean currents from ensemble forecasts and online measurements," *Proc. of IEEE ICRA*, May, 2021.

6. F. H. Kong, **K. Y. C. To**, G. Brassington, S. Anstee and R. Fitch "3D ensemble-based online oceanic flow field estimation for underwater glider path planning," *Proc. of IEEE/RSJ IROS*, September, 2021.

Chapter 1

Introduction

In recent decades, the exploration and utilisation of underwater environments have gained significant attention due to their vital role in understanding Earth's ecosystems, harnessing valuable resources, and facilitating scientific research. However, the hostile nature of these environments, characterised by high pressures, limited visibility, and constrained communication channels, presents formidable challenges to human intervention. As a result, the field of underwater robotics has emerged as a promising solution to address these challenges and to unlock the potential of underwater operations.

In particular, underwater gliders represent a transformative advancement in oceanographic research and technology. These uncrewed vehicles, such as the Slocum Glider in Fig. 1.1, are unique and efficient tools to gather data from the ocean's depths, surpassing traditional methods that have relied heavily on vessels that are expensive to run and labour-intensive deployments with limited coverage areas. By employing principles of buoyancy control and hydrodynamics for locomotion, underwater gliders are capable of conducting extended missions that span from weeks to months, covering thousands of kilometres. This aspect of the vehicle also happens to produce very little noise which lessens the disturbance to the ecosystem of the ocean, and makes them suitable for defence applications.

However, the method of propulsion leads to unique challenges specific to underwater gliders. The first is caused by the relatively low forward speed achievable. A vehicle with low propulsion speed implies that it is more affected by *advection*, the idea that the currents

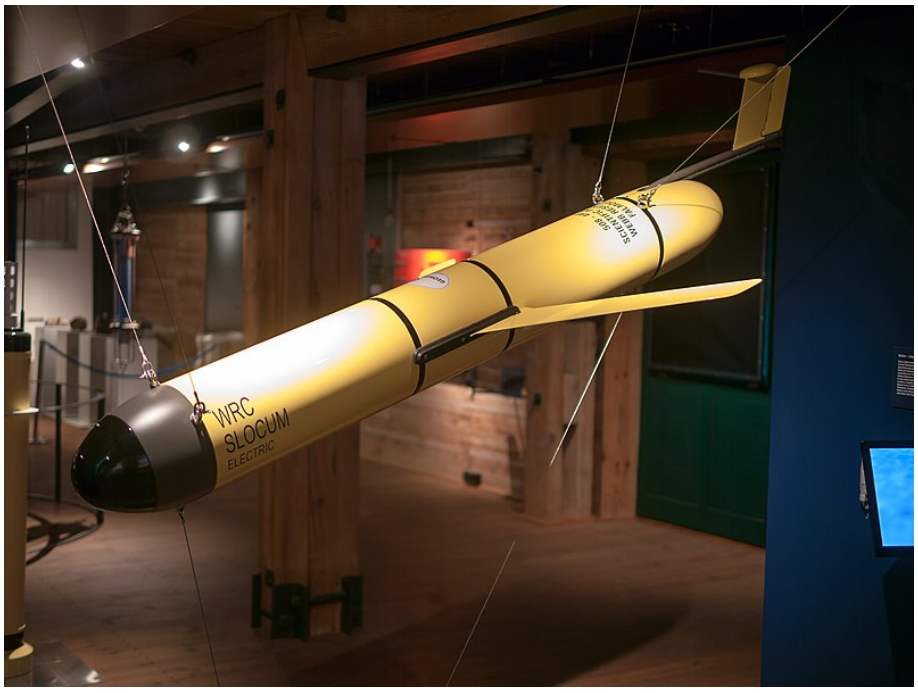


FIGURE 1.1: Photo of a Slocum Glider by Matti Blume is licensed under CC BY-SA 4.0

displace the vehicle from its "still-water" path, inhibiting and potentially preventing it from travelling in certain directions. The second is that underwater gliders have very limited communication bandwidth. Since electromagnetic waves do not travel through water effectively [1–3], underwater gliders must stay on the surface of the water to communicate. These attempts at communication can even fail sometimes due to poor sea states, and since gliders can lose a lot of travel progress by staying on the surface for too long, they are designed to continue onwards with potentially outdated travel plans. In practice, human operators are employed to task underwater gliders in order to avoid adverse currents, however this requires orchestrated coordination across multiple time zones for proper supervision.

In this thesis, we make developments towards a navigation system onboard underwater gliders in order to reduce the heavy reliance on remote supervision. Such a system will require a *planner*, a module that can account for ocean advection when re-routing the vehicle's plans to the goal, and an *estimator*, a module that can incorporate forecast data provided by high-performance computing systems and measurements from the vehicle's sensors. These modules should be computationally efficient to reduce power consumption,

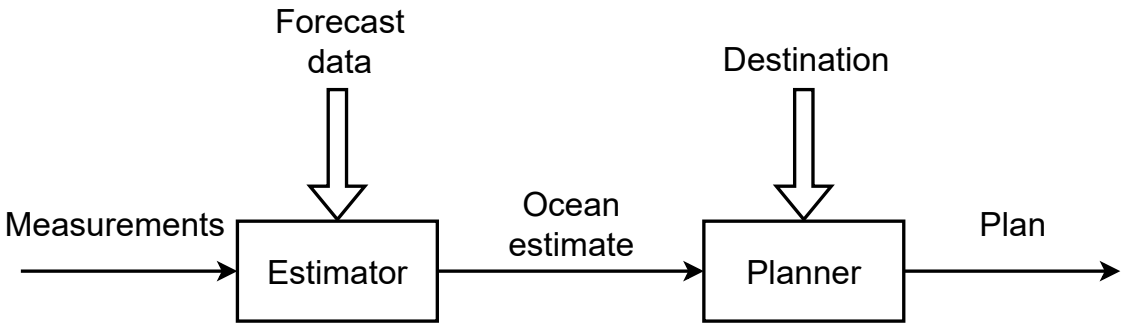


FIGURE 1.2: An example of a navigation system onboard an underwater glider

to quickly adapt to new information, and to run on the hardware constrained by the underwater glider's payload limit. A diagram of an example system using these modules is provided in Fig. 1.2. The three contributions of this thesis can be summarised as: 1) a estimator that leverages forecast data from dedicated algorithms and computation systems to quickly update the vehicle's understanding of the environment, 2) a novel approach to find faster the vehicle control necessary to reach another position under the influence of ocean currents, and 3) a novel approach to improve the connection rate of useful search algorithms, particularly in applications where the vehicle cannot travel faster than the ocean flow.

For the rest of this chapter, we will introduce the fundamental concepts for the work presented in this thesis at a high-level in order to describe the key issues that arise from achieving our goal outlined above. Then we describe the contributions of this thesis summarised above in more detail, and how the described issues are addressed. Finally we provide an outline of the thesis structure.

1.1 Underwater glider motion

The source of an underwater glider's propulsion derives from the change in weight within its hull. Water either floods into or is pumped out of a ballast tank which leads to a change in net vertical force on the vehicle. This is coordinated with a shift in weight, pitching the vehicle up or down allowing their wings to translate some of the vertical motion into forward motion. Whilst the energy consumption to vacate the ballast tank is

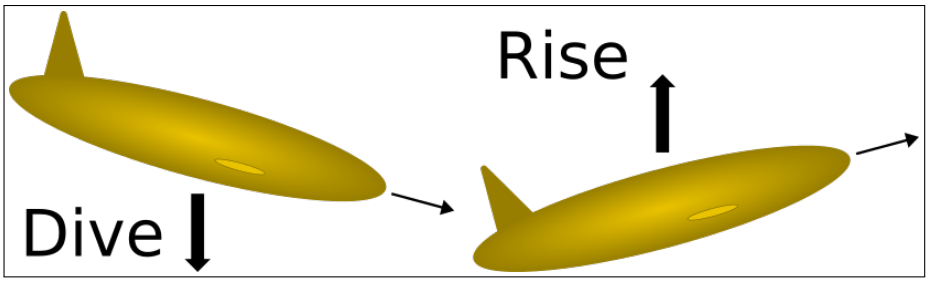


FIGURE 1.3: Underwater glider propulsion mechanism

relatively high, these manoeuvres occur infrequently enough that it consumes less energy than vehicle relying on other propulsion mechanisms such as propellers. Examples of the rising and diving motions that generate forward motion are depicted in Fig. 1.3.

An underwater glider's journey typically consists of *legs*, each of which involve alternating diving and rising motions between two specified depths towards a waypoint, roughly following a zigzag or sawtooth trajectory. In each leg, an underwater glider may surface for communications for a human operator to confirm progress. The time between each surface is usually set to an hour to allow for a reasonable amount of advancement before subjecting the vehicle to stronger currents at the surface for communication. Some underwater glider systems can account for some of the advecting effect of ocean currents, which appears to use a single estimate of the local flow velocity computed from unexpected drift that is computed at each surfacing event.

In this thesis, we make approximations that reduce the computational demand to be executed on hardware onboard an underwater glider, both of which are justified by the long time scales that the underwater glider operate in.

Kinematic approximation with steady-state dynamics The motion of the underwater glider in water is directly controlled by selections of propulsion velocity, the result of its steady-state dynamics

Holonomic vehicle approximation The vehicle can instantaneously change its propulsion velocity relative to the water

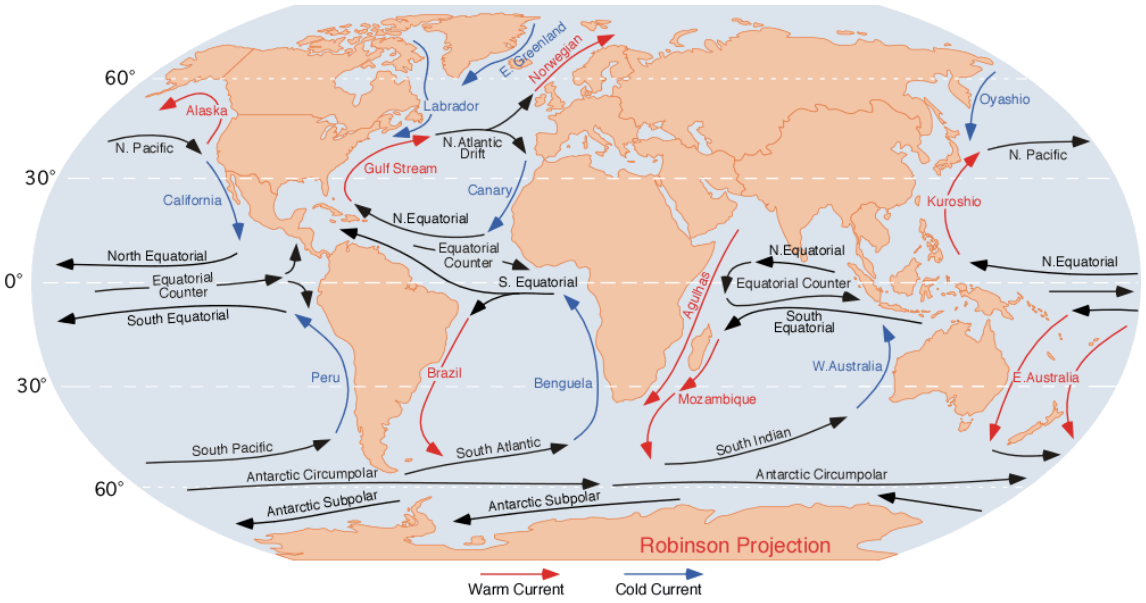


FIGURE 1.4: Illustration of ocean surface currents by Dr Michael Pidwirny. The image is from the public domain under US Code Title 17, Chapter 1, §§105 and 106.

1.2 Flow field estimation

The most significant aspect of the ocean that affects an underwater glider's motion is the velocity of its currents. This motivates the need to estimate the corresponding vector field, or *flow field*.

In general, estimation involves deriving some information from data using a mathematical model of some process. The highly non-linear dynamics of the ocean, such as the surface currents shown in Fig. 1.4, has lead to a long history of research in oceanography to estimate and predict its behaviour. A famous set of equations lie at the heart of this work called the Navier-Stokes equations, solvable by numerical computation that demands high computation resources in most cases. This problem is compounded by the large discrepancy between a lack of available data from real world measurements, and the sheer size of the ocean.

To account for various sources of uncertainty, *ensembles* of estimations are often generated in a fashion reminiscent of particle filters [4, 5]. This involves repeated estimations using different slight perturbations of measurements, and sometimes even varying models to capture different potential outcomes which can be interpreted as "possible realities". An

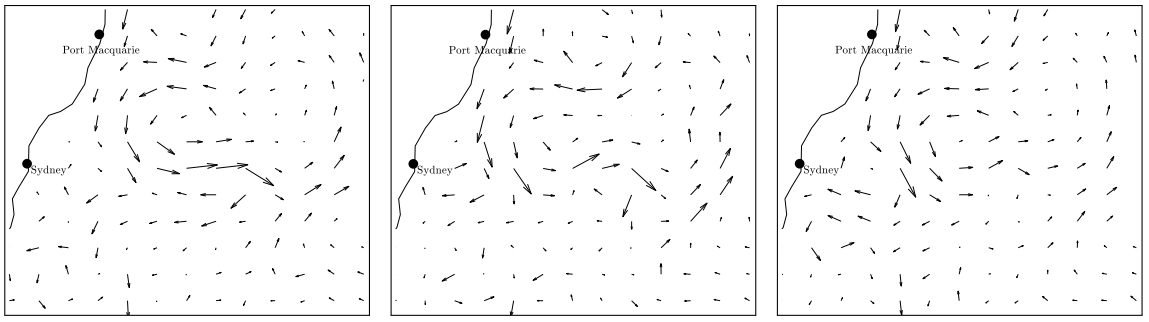


FIGURE 1.5: A few different flow fields away from the coast of New South Wales, Australia, based on ensemble prediction data from the Australian Bureau of Meteorology (BOM)

example of different predicted flow fields is shown in Fig. 1.5. This technique is extensively used in weather and climate forecasting in conjunction with human experts to identify extreme weather meteorological events [6].

Despite using dedicated computationally powerful hardware, the resolution of the estimation solution is still constrained by the computational demands. Oceanographers often employ certain approximations to simplify computation as they can afford less detail in some aspects in order to study other aspects in greater detail. In this thesis, we leverage these ideas from oceanography to enable estimation and planning capabilities using the hardware on underwater gliders with limited computational power.

1.3 Path planning in flow fields

The path planning problem is about finding a sequence of intermediate states a system can transition through to reach a given final state. A sequence of these states is called a *path*. In many applications, we are also interested in finding the most optimal path that either maximises some value function, or minimises some cost function, e.g. finding a time-optimal path between two point.

Optimal motion planning in flow fields can be viewed as an instance of the long-standing *Zermelo's navigation problem* [7], for which there is no known general analytical solution. There has been a wide range of research focusing on various aspects of this problem that

are loosely categorised into four types that are discussed in Sec. 2.2. The type that is most relevant to this thesis is the sampling-based approach.

The development of sampling-based path planners has gained a lot of traction in research recently, potentially due to their desirable algorithmic properties, flexibility of application, and the algorithmic simplicity. In these algorithms, a graph is systematically constructed using states that are sampled either using a random or deterministic sequence that guides the construction of a graph of states are considered for the solution. Some of the desirable algorithmic properties that appear in sampling-based path planning and their loose definitions are as follows:

Probabilistic completeness The probability of the algorithm finding a solution if it exists approaches 100% as the number of samples increases.

Asymptotic optimality A property of an algorithm describing its ability to produce solutions converging to a "robustly feasible"[1] optimal solution if it exists as the number of samples approaches infinity.

Anytime The algorithm can be terminated early for a feasible solution. Typically the algorithm does not have a solution until a period of time has elapsed.

An algorithm with this combination of properties is useful for underwater gliders because in practice it is more important to have an sub-optimal solution earlier that is feasible, rather than having an outdated optimal solution that is no longer feasible. The combination of the anytime and asymptotically optimal properties implies that the computation time can be adaptively committed to potentially find better solutions. This is in contrast to many other approaches where the required computation time can only be roughly estimated from previous attempts. If better paths are required, the algorithm needs to completely restart with new parameters, e.g. approaches that utilise a pre-defined grid resolution. In this thesis, we are interested in the application of these types of sampling-based planners for underwater gliders.

[1] In the context of path planning, a *robustly feasible* path means that there is always a deformation that can be applied at any point along the path that still corresponds to a feasible path.

1.4 Scope of thesis

For automated underwater glider navigation, the vehicle must possess an understanding of its environment and make informed decisions regarding its movements while considering the influence of the environment. These tasks essentially boil down to estimation and planning challenges. This research centres on addressing issues in these two domains, particularly within strong 2D time-invariant flow fields, all while maintaining a stringent requirement for low computational overhead. This line of work is suitable to the interface of some underwater gliders which in practice only enable control in two dimensions. Some work has also been done on a special case of the 3D environment which we refer to as 2.5D flow fields, which are 3D flow fields where all flow velocities are purely horizontal. This investigation further explores a different paradigm of control found in existing work [8] that generate underwater glider trajectories which do not necessarily follow the typical sawtooth trajectory.

While this thesis aims to improve navigational capabilities to underwater gliders by addressing a range of challenges, it is important to acknowledge that certain aspects lie beyond the scope of this work. We do not directly address the navigation issues associated with the time-variant flow fields, nor the rigorous probabilistic representation of the flow field estimate which enables the use of planners that can produce paths with some level of safety. With these boundaries established, we pose the main issues that this thesis considered in the following sections.

1.4.1 Ensemble-informed flow field estimation

Whilst it is infeasible to generate an estimate of the flow field in the underwater glider as accurate as the ones from oceanography that leverage large and computationally powerful hardware, we can utilise their generated ensemble predictions to assist with flow field estimation. However, it is not clear how a single estimate of the flow field can be obtained from a discrete set of ensemble members. Each ensemble member is intended to be a feasible possibility of reality, but none of them are likely to be exactly correct. Another concern is that each ensemble member contains a discrete set of position and velocity

pairs extracted from a model of a flow field; however what is the flow velocity at a position between the given set of data? We aim to create a continuous flow field representation that integrates both ensemble forecasts and direct measurements from the vehicle.

1.4.2 Steering with persistent controls

Sampling-based path planning algorithms tend to share functions known as *primitive functions* that are commonly required for the application to particular problem settings. One of these is called `Steer`, a function to determine the control required to traverse between two states. The availability of such a function enables algorithms to be used such as probabilistic roadmap (PRM) [9] to generate a search graph to can be queried multiple times with different starting and ending states, or the optimal variant of rapidly exploring random tree (RRT) [10] which involves a "rewiring" step that restructures the search graph to preserve previous computational efforts.

For our application, we are interested in the use of *persistent controls*, each of which is a propulsion velocity that is maintained for a specified duration corresponding to the steady state dynamics of underwater gliders. However, under the advection effects of the ocean's currents, there is no analytical solution to determine the persistent control that is needed on a general flow field. A numerical approach is to apply a technique reminiscent to the *shooting method*, which in our case is a search over velocity space. This is a computationally expensive process involving forward integrations for a fixed horizon of steps that is then repeated for each of a set of control values to solve the underlying two-point boundary-value problem (TP-BVP). This high computational cost limits the effectiveness of the overall algorithm and as a result limits the solution quality or geographical scale of problem instances that can be feasibly solved. We are interested in a method to efficiently determine the persistent control to traverse between two states.

1.4.3 Advection reachability problem

A consequence of underwater glider operation in *strong* flow fields, i.e. flow fields that contains flow velocities with magnitudes greater than or equal to the vehicle's maximum

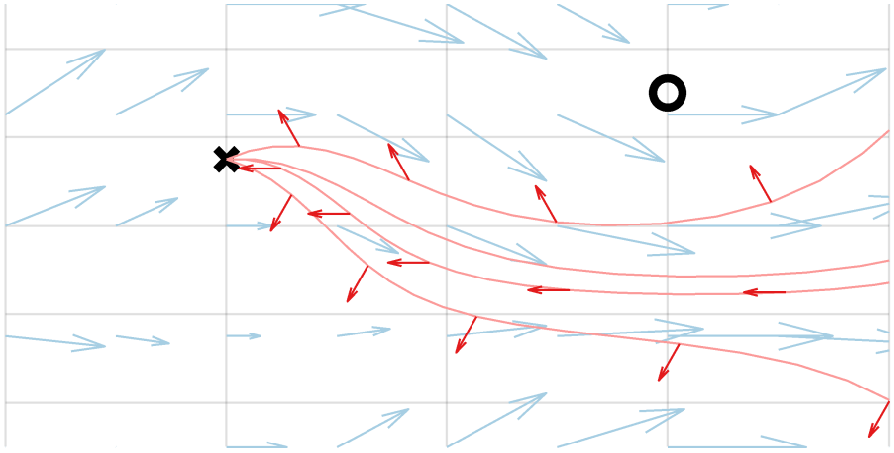


FIGURE 1.6: The limited reachable space of a vehicle (black cross) in a strong flow field (blue arrows) attempting to reach a goal (black circle). Four trajectories (pink lines) are shown: three with maximum velocity in the direction of the red arrows, and one with no control applied (no arrows).

propulsion speed, is that the overall system is non-holonomic, despite the holonomic vehicle approximation. It is more difficult to plan in this type of system, as the constrained travel direction of the vehicle affects the reachability of the vehicle and leads to undesirable consequences if standard implementations of primitive functions are used. One example of this is the primitive function `Nearest` for algorithms like RRT. In these algorithms, this function can be interpreted as a way to "find the node from the search graph from which it is most likely to reach a given state". However, the standard implementation chooses the node with the lowest Euclidean distance to the given state which can lead to choices that cannot connect with the given state. For example in Fig. 1.6, the vehicle at the cross is subject to advection from the flow field and is actually unable to reach the circle. This is referred to as the *advection reachability problem*.

There are kinodynamic sampling-based path planners can be applied for these kinematically constrained systems, however in these algorithms the samples are used less as nodes in the constructed graph, and used more as a way to select an existing node. This diminishes the effectiveness of sampling techniques, e.g. deterministic sampling for search graphs with more balanced coverage [11] and biased sampling towards path nodes for faster convergence [12]. We are interested an alternate implementation of the primitive function `Nearest` that leads to the addition of graph nodes more similar to the samples.

1.5 Contributions

The main contribution of this thesis is the development of new algorithmic techniques that are designed to reduce remote supervision of underwater glider navigation. Central to the efficiency of these innovations is the utilisation of the *incompressibility approximation* from oceanography, as well as the deliberate omission of explicit modelling of kinetic aspects of the problem. Below, we enumerate our proposed algorithms and routines, specifically tailored to tackle the identified challenges:

1. A novel flow field estimation algorithm which is a recursive Bayesian estimator of a compressed model to efficiently integrate online measurements. The compressed model is formed as a linear combination of continuous basis flow fields, which are extracted from the ensemble data through kernel methods and the singular value decomposition (SVD). Intuitively, this reframes the estimation problem to the estimation of basis flow field contributions, or the "presence" of reoccurring spatially correlated flow fields found in the forecast data. This compact representation avoids explicit modelling of physical interactions and achieves high computational efficiency.

2. A `Steer` function that utilises novel streamline-based control theory that enables a search space reduction without disregarding any feasible velocity to traverse from one point to another in 2D incompressible flow fields. The underlying constraint is derived from a natural consequence of treating the vehicle's propulsion velocity as a flow field, which can be superimposed with the environment flow field for a flow field that describes the overall motion of the vehicle. We also show how this idea can extend to the 2.5D case, and argue that a probabilistically complete sampling-based path planner that uses this primitive function remains to be probabilistically complete. This contribution significantly improves the computational efficiency of algorithms that utilise a `Steer` function, either reducing the required computation time or improving the solution quality in the same allotted time.

3. Two novel `Nearest` functions that better reflect the boundary for the advection reachability problem following two different principles. The first approach proposes an alternate distance measure that quantifies the reachability between two points by

augmenting the usual notion of distance with a term that describes a minimum speed for feasible connection. The second approach instead explores the idea of the progressive elimination of candidates before choosing the physically closest option. We confirm in simulation that these approaches clearly outperform the standard choice of Euclidean distance. Furthermore we find that more consistent search trees can be generated compared to the best practice in the kinodynamic RRT framework [13], which generated a search graph that heavily depends on the location of the root node and system dynamics.

1.6 Thesis outline

The remainder of the thesis is organised as follows:

Chapter 2 surveys relevant literature, establishing a frame of reference for existing work and highlighting interesting findings, and aspects related to our study.

Chapter 3 presents our work on computationally efficient flow field estimation from ensemble forecast data in 2D and 2.5D incompressible flow fields.

Chapter 4 presents the streamline-based control theory and its derivation to efficiently search for the control necessary to traverse between two points in 2D and 2.5D incompressible flow fields.

Chapter 5 presents a study of the two proposed approaches to address the advection reachability problem.

Chapter 6 concludes the thesis including immediate future work that can further developments towards the vision of an onboard navigation system to reduce reliance on remote supervision, along with other potential research in other applications.

Chapter 2

Related work

In this chapter, we discuss recent literature to understand the current state of research in flow field navigation. In Sec. 2.1 we review relevant modelling techniques for flow field estimation, and identify a promising real-time approach that can take advantage of the information present in the output of ensemble methods. In Sec. 2.2, we compare various approaches for path planning in flow fields, and establish that sampling-based planning algorithms to be the most suitable class of algorithms for our problem, as some of them are shown to have desirable properties for onboard underwater gliders computation. Finally in Sec. 2.3, we provide an overview of different types of sampling-based path planning algorithms and some of the underlying theory.

2.1 Flow field modelling

Flow field estimation is the problem of estimating a high number of variables over a continuous space. To estimate ocean flow, a key component is the model which serves as a necessary level of abstraction to keep computation demands of the algorithm low. Modelling constrains the possible solutions of the estimation process which can also reduce the amount of measurements required to fully determine the flow. In Sec. 2.1.1, we will first establish some basic concepts from oceanography. Then in Sec. 2.1.2, we will briefly study the tools that oceanographers use to predict future states of the ocean. We then

move onto modelling techniques that focus more on patterns found in data: reduced-order modelling in Sec. 2.1.3 and kernel-based modelling in Sec. 2.1.4. The findings in this section are summarised in Sec. 2.1.5.

2.1.1 Ocean modelling

The fundamental component of physics-based flow field modelling is the Navier-Stokes equations. These equations enforce *conservation of momentum*, and has successfully described fluid flow for the past couple of centuries [14]. Effective application of these equations requires an oceanographer to also consider the boundary conditions of the fluid, and variables that affect density such as temperature and salinity of the water.

Furthermore, Navier-Stokes equations are a set of partial differential equations (PDEs) that do not have analytical solutions so they are typically solved through numerical approaches such as computational fluid dynamics (CFD) which involves discretisation of the continuous system. However, due to the generality of the Navier-Stokes equations, resulting solutions can sometimes describe superfluous dynamics [14] which are typically caused by numerical issues.

Oceanographers often apply approximations to address these issues, facilitating their analysis in the phenomena of interest. Below we list some approximations that are used:

Incompressibility approximation Divergence of the fluid is assumed to be zero [15].

Boussinesq approximation Assumes that the density of the fluid is constant except in the buoyancy term of the Navier-Stokes equations. Flow field velocities should be small relative to the speed of sound c_{fluid}[1] and the vertical speeds should be small relative to $c_{\text{fluid}}^2/g_{\text{Earth}}$ [14, 15, 17].

Hydrostatic approximation Assumes that pressure is only caused by the amount of fluid above it. This disregards the pressure caused by the forces from fluid movement. The approximation is poor when vertical accelerations are not small relative to g_{Earth}

[1] Roughly 1500 m/s in seawater [16]

or when the ratio of vertical to horizontal lengthscales of the motion of interest is much less than $1:10$ [14, 15].

Geostrophic approximation Only considers the major sources of dynamics in the deep sea away from boundaries like coasts, sea surface, and the seafloor. Considers the pressure gradient across depth and Coriolis forces with the hydrostatic approximation. Should only be used for spatiotemporal scales larger than 50 km and 3 days [17].

Thin-shell approximation Ignores the ocean's depth. This is a reasonable assumption since the ocean depth is negligible compared to the Earth's radius [15].

Traditional approximation Takes the thin-shell approximation and assumes that the Earth is sphere with the same volume of water in the ocean. This affects the computation of distance between two points and is a reasonable approximation since Earth is approximately spherical [14].

Rigid lid approximation Assumes the surface height of the ocean relative to the centre of the Earth does not change. Generally used to remove high frequency fluctuations of motion on the ocean's surface. Whilst it is computationally advantageous in some aspects, it introduces a variety of issues so the approximation is becoming less popular [14].

Turbulent closure hypothesis Models the fluctuating term in the numerical computation as a function of the mean flow. This fluctuating term is not trivial to compute otherwise so choosing the form of this term is known as the *closure problem* [15, 18].

Deep-water approximation Approximates the value of a hyperbolic tangent function in the computation of the wave frequency due to the water depth being much larger than the wave length [17].

Shallow-water approximation Approximates the value of a hyperbolic tangent function in the computation of the wave frequency due to the water depth being much smaller than the wave length [17].

Different combinations of these assumptions are used depending on the application (critical variables, scale, and expected phenomenon) [14, 15] to reduce the impact on solution

accuracy. Regional ocean modelling system (ROMS) [19], and nucleus for European modelling of the ocean (NEMO) [15] are two examples of numerical computation tools that are tailored for ocean environments.

To mitigate the computational demands of our flow field estimation algorithm, we opt not to delve into the intricacies of modelling the kinetic aspects of fluid flow. Instead, we find kinematic approximations more useful, such as the incompressibility and rigid lid approximations, which can be less computationally intensive to apply.

2.1.2 Ensemble forecasting

Fluid dynamics are highly non-linear systems and are described as *chaotic*, suggesting that small deviations lead to increasingly large effects over time. The growth of this error can be repressed by incorporating measurements, however some of this error can be left unchecked due to measurements with high noise, or simply the lack of available measurements. This motivates the need to understand the uncertainty of a solution. A way to represent an ocean forecast along with uncertainty is to use *ensemble forecasting*, which effectively predicts many possible realities. Each of these are referred to as an *ensemble members* and are initialised with different values. This type of forecasting is also common practice in meteorology for the purposes of weather predictions and disaster prevention. It is common practice to defer to human experts to interpret the forecast in preparation of responses to critical phenomena such as extreme weather [6].

Ensemble forecasting is generally achieved through the use of the ensemble Kalman filter (EnKF) [20–22]. Each ensemble member describes an entire ocean state, which can include quantities such as flow velocities, temperature, and salinity. These members are propagated through time independently. The mean of the ensemble can be interpreted as the best estimate, and the spread of the members around the mean can be interpreted as the variance in the ensemble. Measurements are incorporated through Bayesian updates which accounts for the ensemble uncertainty and measurement uncertainty.

Whilst ensemble forecasting is able to capture uncertainty as variations of ocean dynamics, the forward propagation of each ocean state is typically done through numerically solving

PDEs for each ensemble member. This is very computationally intensive and impractical to be performed onboard small robotic systems. Other techniques are especially required for underwater gliders to produce a flow field estimate in a timely manner, even at the cost of estimation accuracy.

2.1.3 Reduced-order modelling

Modelling is a process of describing the mathematical relationship between independent and dependent variables. The idea of *reduced-order modelling (ROM)* is to reduce the number of variables to describe this relationship. Instead of representing flow quantities at multiple positions of interest, one class of ROM techniques involves the modelling the data as linear combination of a *library* of elements, or simply *elements*. Flow field estimation using ROM then boils down to a problem of estimating *weights*, the coefficients of these elements. This draws parallels to related to data compression and sparse coding problems where libraries are chosen to re-express data more efficiently as sparse sets of elements.

The library elements can either be selected based on domain-specific knowledge or *machine learning*, which comes from the analysis of collected data. We outline some commonly used choices of elements below.

2.1.3.1 Discrete cosine transform

A widely used example of ROM outside flow field estimation is the discrete cosine transform (DCT) [23] for JPEG images [24] which makes use of elements based on cosine functions of different frequencies. JPEG images achieve data compression by using a predefined set of elements described in its standards so it only the weights need to be stored. This type of element is shown to be less effective for flow field modelling by Bai et al. [25], potentially because the elements do not reflect realistic flow fields. We discuss the other type of element in their study next.

2.1.3.2 Proper orthogonal decomposition

Proper orthogonal decomposition (POD) is one of most popular techniques for ROM. It reduces dimensionality whilst minimising the reconstruction error based on the Euclidean norm (2-norm) [26]. This technique appears in many fields using different names [27, 28], the most relevant of which is the principal component analysis (PCA) and the singular value decomposition (SVD).

In data analysis, PCA identifies their elements as orthogonal unit vectors from the data, each of which describe different correlations of data variation. As a result, the elements describe the different patterns in the data, commonly referred to as *modes*. If the data is rearranged as a matrix $\boldsymbol{B}$ such that the columns contain different samples of data and the rows correspond to each variable, we can find the modes through the SVD

$$\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H, \tag{2.1}$$

where $(\cdot)^H$ is the conjugate transpose operator, the columns of $\boldsymbol{U}$ and $\boldsymbol{V}$ are the left- and right-singular vectors, and the values in the diagonal of $\boldsymbol{\Sigma}$ are the singular values. The columns of $\boldsymbol{U}$ correspond to modes, and the columns of

$$\boldsymbol{W} = \boldsymbol{\Sigma}\boldsymbol{V}^H, \tag{2.2}$$

are the weights corresponding to each sample in $\boldsymbol{B}$.

The SVD is also useful because it is unique [27] and it provides a natural ordering to its elements using the corresponding singular values. A larger singular value indicates a greater significance of the corresponding element in the reconstruction of the data. This also allows us to *truncate* the ROM by removing the weights and elements corresponding to low singular values, which are usually associated with high-frequency noise.

In flow field estimation, the POD technique is typically used as a way to reason about the flow at other locations. We describe how it is used in a few different applications below.

Bright et al. [27] uses POD to help identify the type of flow that is occurring over a cylinder by analysing the pressure around it. First, the pressure around the cylinder is recorded

across multiple time steps for a particular Reynolds number. Then modes are extracted between these time steps, then truncated. This process is repeated for different Reynolds numbers, and the truncated modes across the different Reynolds numbers are collected all in a single library. The Reynolds number of flow can then be later identified by matching pressure measurements only on the surface of the cylinder to the library. The interesting aspect of this application is that the POD is used across different time instances, and is able to extract the patterns associated with each Reynolds number.

POD can also be used to help with data interpolation. Particle image velocimetry (PIV) is a technique that measures the flow velocities across an area. A particularly useful type of PIV is time-resolved PIV (TR-PIV), which takes measurements fast enough to identify flow structures. However TR-PIV systems are expensive. In contrast, time-resolved probes have good temporal resolution, but they have limited spatial coverage. To address this problem, Tu et al. [29] first extracts the spatial information from each PIV measurement using POD. Correlations are drawn between the extracted modes and the probe data, and is used in a stochastic estimation process to form a model that ultimately allows the reconstruction of data between the PIV measurements that resemble TR-PIV data.

In an application more practical for robotics, Salam and Hsieh [30] uses POD to track the evolution of a physical process, such as temperature, over a workspace using a team of robots. The workspace is identified to have s regions, each with multiple spatial points where data needs to be measured. However only $q < s$ robots are available. First POD is performed on historical data of the process which can be simulated or from prior experiments, then truncated. The robots are then assigned the q regions that best estimates the data at the other regions at each time step. The interesting use of POD here is that the modes are adjusted at every time step, allowing the robots to be reassigned to new regions for better results.

We described a few examples of POD used in estimation problems. It was used to identify distinct patterns of data from a set of a representative examples. However noise in the representative data can lead to *overfitting*, i.e. unnecessarily modelling patterns that are caused by noise.

2.1.3.3 Sparse representation

Here, we discuss *sparse representation*, a method designed to be more robust to noise. A sparse library can be created by trying to reduce the number of elements required to reconstruct each of the example data by tolerating some error. A related idea is compressed sensing, which also aims to reduce the number of elements required to explain noisy measurements. Both correspond to minimising the 0-"norm" of the weights with a tolerable amount of reconstruction error. However, their computation is computationally intractable, so it is typically approximated with the Manhattan norm (1-norm) instead.

An example of compressed sensing is used by Bright et al. [27] in a previously discussed study of flow around a cylinder. Instead of finding using a least squares fit of the library on the measurements on the surface of the cylinder, they solve the 1-norm minimisation problem to determine the weights. This allowed the authors to reconstruct the full flow field from the corresponding pressure field, and estimate its Reynolds number using a strategy reminiscent of voting.

Callaham et al. [31] compares the reconstruction error between different ways to construct the library, and different ways to estimate the full flow field from a few measurements. Their sparse library is obtained using k-SVD which iteratively applies the SVD perform 1-norm minimisation. The results suggest that using sparse representation leads to less reconstruction error than using POD. The trade-off for lower reconstruction error is higher computational demands since k-SVD is an iterative algorithm in contrast to SVD which has a closed form solution. Their results seem to also suggest that 1-norm minimisation is also better for reconstruction error if the library is based on the training data itself, and not if the library are POD modes.

2.1.3.4 Dynamic mode decomposition

An interesting application of POD is dynamic mode decomposition (DMD) [28, 32], which identifies linear dynamics in the weights to describe the dynamics of non-linear time-variant processes. The non-linear aspects of the system are described as DMD modes, whilst the time-variant aspect is only described as the dynamics of the weights. DMD is

an interesting tool as it approximates the *Koopman operator*, a linear infinite-dimensional operator that is equivalent to the finite-dimensional non-linear dynamics of a non-linear system. Whilst time-variant flow fields are outside the scope of this thesis, we find the capabilities of this technique fascinating, and believe further research in this direction would be worthwhile.

Salam and Hsieh [33] uses DMD to tackle a problem very similar to the interpolation problem that Tu et al. [29] addressed. They consider marine robots that can collect high-spatial, low-temporal resolution data at specific locations and aerial robots that can gather low-spatial, high-temporal resolution data over larger areas. The measurements from these different sensing modalities are fused through iterative updating of the spatial and temporal aspects of the DMD. In the online process, the aerial vehicles are assigned sensing locations prioritising coverage, whereas the marine vehicles are assigned locations that minimize the approximation error.

2.1.4 Kernel-based modelling

In Sec. 2.1.3, we mostly discussed the use of library elements extracted from representative data. Here we consider a different machine learning technique that revolves around a function that can actually be used as continuous variants of library elements.

A *kernel function* $k(\mathbf{x}, \mathbf{x}')$ can be loosely described as a function that quantifies the correlation of data between two given points, usually a query point $\mathbf{x}$ and the location of a latent variable $\mathbf{x}'$. A single kernel can act like a library element if it is associated with a known latent variable at $\mathbf{x}'$. The discrete library element can be obtained by constructing a column vector of the product between the latent variable and $k(\mathbf{x}, \mathbf{x}')$ at different query points. A model constructed this way allows data to be queried across the entire domain, as opposed to the models of ROMs which usually require interpolation techniques to query data at positions other than the ones associated with the training data. We say a kernel is *centred* at a point when the position of the latent variable is defined there.

This idea is used in support vector machines (SVMs), a machine learning algorithm that separates data for classification or regression analysis. SVM uses particular choices of

kernels to map data into a latent space. In classification problems, the different groups of data are identified using the linear separability of the data in the latent space.

There are many types of kernels, each with different patterns of correlation around the centred point and varying levels of effectiveness to express data. Particular aspects of the shape are controlled by hyperparameters. For 1D data in one-dimensional space, a commonly used kernel is the Gaussian radial basis function (GRBF), or Gaussian kernel [34]

$$k(x, x') = \exp\left(-\frac{(x - x')^2}{2l^2}\right), \tag{2.3}$$

where x and x' are the independent variables of the data, and l is a kernel-specific hyperparameter called a lengthscale representing a characteristic neighbourhood of influence. This kernel is useful for describing dependent variables that are similar when the independent variables are similar, i.e. data that is smooth. Note that this is called squared exponential (SE) sometimes and also has slightly different definitions across literature [35, 36].

For 1D data in D-dimensional space, a kernel can be constructed by multiplying kernels defined for each independent variable. Automatic relevance determination (ARD) is a special case of this, where the effect of each independent variable is scaled by a hyperparameter. For example, one study [36] defines SE-ARD through the use of the exponent product rule as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_{\text{ker}}^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x'_d)^2}{l_d^2}\right), \tag{2.4}$$

where the variable σ_{ker} is an additional scaling hyperparameter, and the subscript d refers to the component of the independent variables in that dimension and its corresponding lengthscale. This kernel is also useful for describing smooth data, and allows us to specify when data should be considered similar based on the different lengthscales of each independent variable.

It is sometimes useful to consider the matrix formed by the pairwise kernel evaluations between two sets of points X_A and X_B. In this thesis, this matrix will be referred to as

the *kernel matrix*

$$\mathcal{K}(\mathsf{X}_A, \mathsf{X}_B) = \begin{pmatrix} k_{(1,1)} & k_{(1,2)} & \cdots & k_{(1,n_B)} \\ k_{(2,1)} & k_{(2,2)} & \cdots & k_{(2,n_B)} \\ \vdots & \vdots & \ddots & \vdots \\ k_{(n_A,1)} & k_{(n_A,2)} & \cdots & k_{(n_A,n_B)} \end{pmatrix} \in \mathbb{R}^{n_A \times n_B}, \tag{2.5}$$

where $k_{(i,j)} = k(\mathbf{x}_i, \mathbf{x}_j)$ is the value of the kernel between $\mathbf{x}_i \in \mathsf{X}_A$ and $\mathbf{x}_j \in \mathsf{X}_B$. This kernel matrix can be interpreted as a library of elements, each of which are kernels centred at the points in X_B. If $\mathsf{X}_A = \mathsf{X}_B$, then the kernel matrix is known as a Gram matrix.

A particularly interesting use of kernels is found in studies to estimate flow fields using only the experienced drift from vehicles. This problem is particularly interesting because the observed vehicle drift is a result of unknown flow velocities experienced by the vehicle that travelled along an unknown trajectory. Flow velocity measurements can only be determined by resolving the tight coupling between the trajectory and the flow velocities along it.

In the case for multiple autonomous underwater vehicles (AUVs), Chang et al. [37] proposes to use an iterative algorithm that slowly adjusts a grid of flow velocities, such that the trajectory computed from the previous flow field matches with the drift measurements they measured. The flow field estimate is modelled using five GRBFs centred at pre-determined locations. However, the discretised cells of uniform flow and the linear combination of fixed GRBFs are overly simplistic models as they either have too many parameters to estimate, or they do not have the ability represent the details of the flow field.

2.1.4.1 Gaussian process

Gaussian processes (GPs) describe normally distributed random variables that are functions over a continuous domain. The distribution of the GP is interpreted as the joint distribution of random variable measurements with a given kernel. As the GP collects measurements, its distribution converges to the true distribution. This representation can provide the mean and variances of random variables at arbitrary locations. GPs can be

used as an interpolation algorithm by providing the mean at any location and potentially taking into account all collected measurements, their variances, and the kernel.

Queries of flow velocities for a given position requires a matrix inversion of a kernel matrix between the set of all measurement locations with itself which is a very computationally expensive operation that becomes increasing expensive at a complexity of $\mathcal{O}\left(n_{\text{mea}}^3\right)$ where n_{mea} is the number of measurements. If the query time complexity is critical, the inversion operation can be displaced to the procedure of measurement integration, i.e. the inversion can be performed after integrating the measurement information. However, the reduced query time complexity of $\mathcal{O}\left(n_{\text{mea}}\right)$ which still becomes more computationally expensive as more measurements are added.

To tackle the flow estimation problem from drift experienced by a underwater glider, Lee et al. [38] resolves the cyclic dependency using an expectation maximisation (EM) algorithm, which alternates between refining the flow velocities along the trajectory and then refining the trajectory caused by the flow velocities. Lee et al. [38] also introduces a special kernel to enforce an incompressibility constraint on their estimated flow field by implicitly describing a *stream function*, a scalar function describing the flow flux between two points which is only well-defined for 2D incompressible flow fields. In this thesis, we refer to this special kernel as the *incompressible kernel*. It is interesting to note that this kernel can also be obtained by following the strategy suggested by Jidling et al. [39] that can also be used to derive other kernels to enforce other linear constraints.

2.1.4.2 Kernel observer

The kernel observer is proposed by Kingravi et al. [40] to model systems that evolve across time and space. This is achieved by identifying linear dynamics in the latent space representation of data. This is reminiscent of the DMD technique in that both ideas find linear dynamics for the weights of their elements. In this approach, GRBFs which are positioned and tuned based on data. Regularisation using the 2-norm is then performed over the resulting weights to obtain the transition matrix that describes the evolution of the latent variables over time.

Whitman and Chowdhary [41] builds upon the idea of kernel observer by seeking a single transition matrix to describe the linear dynamics for similar time-variant systems. The transition matrix is obtained by simply performing least squares regression between observed latent variables in one time step against the latent variables in the next time step across all systems. This is shown to be surprisingly effective at describing the flow around a cylinder across multiple Reynolds numbers, the same problem as Bright et al. [27] considered. This is particularly surprising since the flow across multiple Reynolds numbers correspond to different rates of vortex shedding that occurs after the cylinder.

2.1.5 Summary

Estimating a continuous flow field from measurements, i.e. finite discrete data, is a highly underconstrained problem. Oceanographers employ laws of conservation to reduce the space of possible solutions to realistic ones. Assumptions are also used to mitigate certain types of numerical errors propagating, in addition to further reducing the solution space. Forecasts are produced from ensemble methods, and also describe forecast uncertainty due to uncertainty in measurements and the system dynamics. However these methods are too computationally intensive to be computed using the hardware onboard robotic systems, let alone underwater gliders.

Another way to approach flow field estimation is to use ROM. The flow field can be discretised and modelled as a linear combination of elements, which can be extracted from representative data either from simulation and/or from historical data. This effectively reduces the complexity of the problem by limiting the degrees of freedom in representation. An underlying issue of these approaches is that the data is only defined at the positions defined by the representative data. Interpolation methods can be used to obtain data between these positions, however this can correspond to physically infeasible flow fields.

Kernel methods however offer a continuous representation of a flow field estimate. Flow fields can satisfy linear constraints by being composed of kernels that follow linear constraints. This can be helpful to enforce laws of physics or assumptions for an approximation. In particular, a flow field estimate constructed from incompressible kernels is guaranteed to follow the incompressibility approximation. GPs also offer a probabilistic

interpretation for the estimated flow field, which can be useful in the context of certain problems. A limiting factor is that it is prohibitively slow to query as the number of measurements increases. This is problematic as path planning algorithms many flow velocity queries to account for the advection effect on vehicles.

In Ch. 3, we draw upon several ideas presented in this section to estimate a flow field with little computational resources onboard an underwater glider. A library is constructed by first transforming the data from an ensemble method into latent space representation through the incompressible kernel, followed by POD and truncation. This corresponds to a linear flow field model that is both continuous and incompressible. Flow velocity measurements can then be integrated using the Kalman filter (KF) update by treating the weights of this library as the Kalman state, which is a computationally simple process.

2.2 Planning under flow field advection

The problem of planning involves finding a sequence of states or actions that describe transitions from a given initial state to a desired final state [42]. Sometimes it is sufficient to reach a state that is within some specified neighbourhood of the final state. In this section, we review how different algorithms can be applied to the problem of path planning for a vehicle under advection dynamics from either air or water flow to optimise different cost functions, e.g. travel time, energy cost, or customised formulations of risk.

The problem we consider in this section is related to the long-standing problem known as Zermelo's navigation problem. The problem involves finding the time-optimal sequence of headings to take a vehicle travelling at a fixed speed in a 2D time-variant flow field from one spot to another. Zermelo [7] proposes a numerical approach to solve this problem which also addresses the 3D variant of the problem. It is later shown that it is not necessary to consider other travelling speeds as long as the control can be continuously varied [43, 44].

We loosely categorise different path planning techniques found in literature into four groups, and will describe them in rough chronological order. Firstly in Sec. 2.2.1, we explore approaches that apply general optimisation techniques to solve the path planning

problem. This is in contrast to the other categories where algorithms are developed primarily for path planning. Secondly in Sec. 2.2.2, we discuss approaches that plan across discretised space, i.e. the vehicle travels over a finite set of locations. Thirdly in Sec. 2.2.3, we discuss approaches that plan across continuous space despite potentially using an underlying discrete grid. Lastly in Sec. 2.2.4, we discuss approaches that produce plans across continuous space by constructing discrete graphs through the guided sampling. The findings in this section are then summarised in Sec. 2.2.5.

2.2.1 Optimisation-based approaches

Optimisation techniques are useful because they are developed for generic problems, making them great general-purpose tools. Some approaches in this section reformulate the path planning problem so that they can be solved by optimisation solvers. Others identify sub-problems within their applications which can be solved first with simple optimisation techniques, reducing the overall complexity of the problem. Techniques such as simulated annealing [45] and evolutionary algorithms can give globally optimal solutions, whilst techniques such as damped least squares (i.e. the Levenberg-Marquardt algorithm) [46], sequential quadratic programming (SQP) [46], and shooting methods [47] tend to give locally optimal solutions unless the problem is shown to be convex.

Techy [48] addresses the path planning problem for a particular flow pattern. In their work, they describe a strategy for a vehicle to escape a 2D time-variant flow field that is rotationally symmetric about a point. Flow fields of this structure can include sinks, sources, vortices, or combinations thereof. By considering a vehicle in this flow field as a Hamiltonian system, it is shown that paths that are at least locally time-optimal have the property where the heading rate of change is exactly half of the local vorticity. An optimal heading sequence can then be found using numerical optimisation, which is demonstrated by the use of damped least squares to find the required initial heading angle and traversal duration. This work is too specific for the problem considered in this thesis, however we find theoretical discovery interesting.

Smith and Dunbabin [49] considers an interesting path planning problem for an underactuated vehicle in an uncertain 2.5D time-variant flow field. The vehicle of interest

is a Lagrangian float, which is a device that can only actuate in the vertical direction, leaving its horizontal motion purely controlled by the flow field. In this paper, the flow field is modelled as the interpolation of the flow at 3 depths and is assumed to have regular periodicity. A two stage path planning approach is proposed to minimise cost as a linear combination of the final distance to goal, time, energy, and presence in shallow regions. In the first stage, a coarse path is found by using simulated annealing and rough approximations of drift. The controls of this coarse path is then used to determine a more realistic path by simulating drift more realistically. The second stage involves refining this path using simulated annealing with lower a initial temperature, and local random search.

Isern-González et al. [50] demonstrate different ways to use optimisation techniques on two path planning problems with underwater gliders. The first problem seeks to minimise of the distance between the underwater glider and the goal given a limited time budget. The second problem seeks to minimise the execution time to reach a given area. The first problem is simply addressed by assuming that the underwater gliders will use fixed-duration legs, and then performing direct optimisation on the number of possible waypoints to reach the goal. For the second problem, an algorithm is proposed that iteratively adds waypoints followed by the use of SQP to optimise the positions of waypoints until the final waypoint is within the target area.

One contribution from the work of Kularatne et al. [51] is that smooth trajectories that are approximately optimal can be found by using a flow-parallel co-ordinate system. They show that approximate Riemannian metrics for time and energy can be derived on the proposed co-ordinate system. This allows the path planning problem in 2D time-invariant flow fields to be posed as the optimisation of the geodesic equation on the corresponding Riemannian manifold is easier to solve than the original problem. In their work, the geodesic equations are solved with the shooting method, which involves systematically integrating the geodesic equations from the start with different conditions until the goal position is reached.

Lucas et al. [52] proposes the use of evolutionary algorithms to assist underwater glider

operators to plan in 3D time-variant flow fields. In their work, they address the multi-objective problem of finding paths that minimise the final distance to the goal and maximising the closest distance to any obstacle. Without providing any weighting to these objectives, the solutions to this problem lie on a Pareto front, which are the set of solutions that are not definitively better than one another, but are better than all other possible solutions. The authors propose the use of a generic global optimisation algorithm that is designed to search for solutions on the Pareto front called non-dominated sorting genetic algorithm (NSGA-II) [53].

Global optimisation solvers are useful tools as they allow problems with various constraints to be solved. The advection aspect of the problem is naturally handled by providing a model to simulate the drifting effect. However as generic problem solvers they are unable to take advantage of the inherent structure of path planning problems, which makes them susceptible to getting stuck in local minima. In contrast, dedicated path planning algorithms in the upcoming sections tend to be comparatively faster.

2.2.2 Discretised-space approaches

In this section, we discuss recent work that discretises the workspace of the flow field to apply graph searching algorithms like A* for optimal paths. Usually the workspace is divided into rectilinear grids, however we also see other forms of discretisation here. Whilst authors tend to give specific names to their algorithms, we will sometimes refer to them as A* as their algorithms can be interpreted as the A* algorithm with a specific graph construction strategy.

Pereira et al. [54] propose two different approaches to plan the path of underwater gliders in a 2D time-variant flow field whilst minimising the risk of ship collision. First, the environment discretised into rectangular cells. The risk of collision at each cell is then computed by taking the proportional occupancy of the cells computed from historical automatic identification system (AIS) data over a selected time period, then applying Gaussian blur over the cells for a conservative estimate. To account for flow field prediction uncertainty and advection effects, transition probabilities are computed between cells by simulating multiple transition attempts using the given flow field with random Gaussian

permutations. Their first approach to solve this problem is their proposed algorithm called minimum-expected risk planner which is effectively solving a graph using A* with edge weights defined by the expected risk of collision when attempting that transition. In their experiments, a rectilinear grid is used with 8-connectivity. We believe that this approach has unintended consequences as the path produced assumes that the vehicle is able to make all the transitions successfully. Their other approach is the risk-aware Markov decision process (MDP) planner which addresses the problem using techniques such as value iteration or policy iteration. The MDP approach addresses the problem properly but is computationally burdensome, whilst the A* formulation is more optimistic with lower computational costs.

Lee et al. [55] proposes an algorithm called EEA* to find an energy-optimal path for a non-holonomic forward-travelling surface vehicle in a 2D time-invariant flow field. The algorithm uses A* on a graph constructed after considering environmental effects. The connectivity of the graph is chosen to account for the kinematic constraints of the problem. By representing three state values in the graph nodes, the algorithm can constrain vehicle motion to travel within a certain angle of the vehicle's heading by controlling the graph's connectivity. In their problem, advection does not constrain the vehicle's movement, and is only considered as part of their detailed energy cost formulation, which includes the energy to overcome drag, the energy to overcome advection, the additional energy to travel in shallow waters, and finally hotel load. The algorithm also includes a post-processing step which smooths the resulting plan.

Kularatne et al. [51] addresses the problem of planning in a 2D time-variant flow field by discretising time and space into a 3D rectilinear grid. They propose an algorithm called multi-timestep search which is similar to A*, however it focuses on the idea that nodes can have several different connections to their spatial neighbours by changing the time it arrives. This effectively corresponds to different vehicle propulsion speeds. Edge connections that correspond to infeasible propulsion speeds are simply pruned. From this, the authors propose functions to evaluate the time and energy costs to traverse these graph edges in time-invariant and -variant flow fields. The authors note that the spatial connectivity of the graphs should be limited to constrain the branching factor of

the search, and the time connectivity between nodes should be small enough to preserve locality assumptions in the cost functions.

Kularatne et al. [56] consider energy-optimal path planning in 2D time-variant flow fields with uncertainty. Their work parallel the work done by Pereira et al. [54] in that they also experiment with an A* approach as well as an MDP approach. One significant difference between the two pieces of work is the expected cost to traverse between two cells. For the A* approach, Pereira et al. [54] does not account for transition failures when computing the minimum expected risk path. In contrast, Kularatne et al. [56] accounts for transition failures in the expected energy cost as increased energy consumption on top of the energy required to overcome drag plus the hotel load. The authors suggest that the MDP formulation could be useful if the start, goal, and flow field uncertainties are not updated during execution. However they appear to be more inclined to recommend the A* approach since it requires much less computation time. Another novelty of their A* approach is the use of a triangular grid, which is not common in literature. The authors also highlight a shortcoming of their work, which does not consider the consequences when the flow speeds are stronger than the vehicle's maximum speed.

A problem with all the approaches so far is that the workspace is discretised uniformly. Kularatne et al. [57] considers adaptive discretisation to plan a path in a 2D time-variant flow field, allowing them to use coarse grid resolutions in areas of relatively uniform flow, and fine grid resolutions in areas where the flow changes more quickly. They propose an algorithm called multi-timestep search which involves iterative construction of a graph where each node contains spatiotemporal information. Its graph exploration strategy is reminiscent to the A* algorithm. The key novelty in the proposed algorithm is that the graph is incrementally constructed with new nodes during the search, such that the error from assuming constant flow field velocity is limited.

The main criticism for approaches that discretise space is that the overall solution in continuous space is usually not optimal, despite being optimal in the constructed graph. The maximum discretisation size is also constrained by the scale at which the flow field changes in space and time, which can lead to high computation time. A subtle problem with the resulting paths is that the path segments are generally constrained to the angles

of the graph edges. This can potentially exclude useful or even necessary routes to reach the goal.

2.2.3 Continuous-space approaches

In this section, we study path planning approaches that construct paths over the continuous space which tend to be smoother than the approaches in Sec. 2.2.2. Some of these approaches consider multiple paths from the start to the goal by tracking a frontier, which is the boundary of reachability by some time. This frontier is represented through the computation of a function which is a description of different things depending on the approach. After computing this function, a path is usually returned by some variation of a gradient descent algorithm. The idea of *resolution completeness* is relevant for these types of approaches, which is the idea that the algorithm will find the solution if it exists, given a fine enough resolution of the constructed grid.

An early application of the frontier idea is demonstrated by Soulignac et al. [58], who address a path planning problem in a 2D time-invariant flow discretised into Voronoi cells of uniform flow. They propose an algorithm called sliding wavefront expansion which finds a time-optimal path by joining the boundaries of the Voronoi cells. The idea behind the algorithm can be loosely described as Dijkstra's algorithm on the Voronoi cell walls with an additional step after expansion which optimises the intersection position on the predecessor's Voronoi cell wall. The authors also consider the limited reachable angles of the vehicle from advection in this algorithm in the optimisation as well as the search expansion. Soulignac [59] later proves that the algorithm is correct and globally optimal.

Rhoads et al. [60] proposes an extremal field method for time-optimal path planning in a 2D time-variant flow field. An optimal feedback control law can be derived by taking the gradient of a value function. In this case, the value function is chosen to be time-to-go, i.e. the time left to reach the goal, so that the optimal feedback control law corresponds to the controls used in the time-optimal path. In their approach, a set of trajectories called extremals are constructed in reverse by tracing a frontier back from the goal, initialising with different headings. At each time step, the frontier can be propagated backwards in time by following the opposite of the optimal control law. The intuition behind this

extremal field approach is that at each time step, the frontier propagates in the direction that corresponds to the least travel time. In the algorithm, the gradient of the value function is estimated numerically using information present at the neighbouring trajectories. When the frontier reaches the start, the time-optimal trajectory can then be obtained by following the optimal feedback control law.

Elston and Frew [61] and Girardet et al. [62] demonstrate the use of the ordered upwind method (OUM) to find 2D time-optimal paths for aircraft in a time-invariant flow field. In these types of approaches, the value function itself is estimated across the workspace on a discretised grid, and is computed using an algorithm similar to Dijkstra's algorithm. Then the optimal path can be obtained by following a gradient descent algorithm on the value function. Here, the authors define the value function as time-to-go as well for time-optimal paths. Elston and Frew [61] shows how to include precipitation avoidance in the planning process by inflating their value function. Girardet et al. [62] shows how to avoid obstacles by inflating the value function around obstacles with potential fields, and also plans on spherical co-ordinates to account for the geometry of the Earth.

The limitation of the OUM is that it assumes that the optimal frontier during construction never reverses in direction. This happens when the flow field is time-variant, and it becomes necessary to consider "retreating" parts of the frontier as feasible. Lolla et al. [43] shows how level set methods can be used in these cases. In their approach, a *viscosity solution* is estimated instead of the value function, referring to the solution of the Hamilton-Jacobi equation, a PDE that governs the time-optimal path. The viscosity solution in this case is a function of space and time, and whilst the exact meaning behind its value is not important, a suitable interpretation is the signed distance to the frontier. This means that the evolution of the frontier is embedded in the viscosity solution as the zero level set. The algorithm of the level set method involves solving a narrow band around the frontier across time. When the frontier passes over the goal, the time-optimal path can be obtained in reverse by utilising values of the viscosity solution. Time-optimal paths can be found in 2D and 3D time-variant flow fields using this approach. Static obstacles are also trivially handled by preventing the viscosity solution to be computed at the obstacles. Lolla et al. [63] shows how this idea can be extended to handle dynamic obstacles through the level

set PDE. They also propose a control schemes for multiple vehicles to maintain regular polygon formations.

Subramani and Lermusiaux [64] derive a methodology of computing energy-optimal paths through repeated applications of the level set method to find time-optimal paths in 3D time-variant flow fields. The important realisation is that, given a speed profile of the vehicle over time, the time-optimal path is also the energy-optimal path. I.e. we can compute the energy-optimal path for a fixed speed profile by finding the time-optimal path. This means that we can solve the time-optimal problem for various speed profiles and collect the paths in the Pareto front for time and energy costs. Another key idea is that if the speed profile is represented as a stochastic variable over time, then it can be propagated through a stochastic version of the level set PDE [43]. In practice, the speed profile is represented as a step function with pre-defined intervals to simplify the search. A dynamically orthogonal (DO) decomposition, which is related to the SVD, is used on this stochastic representation which increases accuracy and computational efficiency.

Another way of using the stochastic level set PDEs is demonstrated by [65] to find time-optimal paths in uncertain 3D time-variant flow fields. Instead of representing the vehicle speed stochastically, the flow field is represented stochastically, and the corresponding stochastic level set equations are solved to obtain frontiers for different possible flow fields. However choosing the correct path to use from the stochastic frontiers assumes the vehicle can know what the full flow field is during execution which is unlikely in practice. [66] addresses a similar problem, where paths are chosen to minimise the risk of following a path that is sub-optimal in time due to flow field uncertainties. In their work, risk is computed as a cost or "tolerance" function of path error. A few varying definitions of the cost function and the path error are suggested. After choosing a pair of these definitions and using the approach described by Subramani et al. [65] to generate n_r optimal paths from the n_r possible flow fields, the cost is computed for the paths in the other possible realisations of the flow field. Finally the path with the lowest average cost across the possible flow fields is chosen as the risk optimal path.

[67] shows how level set methods can be used for vehicles with different propulsion systems. This is in contrast to vehicles with isotropic speed used previously, i.e. vehicles that can

move at the same speed in all directions relative to the water. In particular, they study path planning for the Lagrangian float which can only move in the vertical direction, and the underwater glider which moves at some given speed vertically. For the underwater glider, the 3D level set PDE is split into two equations, one describing the horizontal evolution of the viscosity equation and one that describes its vertical evolution. Since the vertical direction is fully defined, the control problem only lies in the horizontal direction. Numerical schemes are used that take advantage of the resulting level set equations.

There has been a lot of impressive work for path planning using continuous spaces. Approaches that use extremal field methods, OUMs, and some other work not discussed here (e.g. fast marching methods) systematically compute the value function of the problem, whilst level set methods compute a viscosity solution, in which the reachability frontier is embedded. An common issue for these methods is that there is still an underlying discretisation that occurs across space, and to obtain a valid solution requires the appropriate choice of grid resolution. Furthermore, these approaches generally assume that the vehicle can be continuously controlled which is not always true. These approaches also tend to use "one-shot" algorithms, and are unable to provide better solutions unless the entire algorithm is restarted with new parameters. This also means they can't provide sub-optimal solutions early, which can be very useful in our problem.

2.2.4 Sampling-based approaches

In this section, we discuss path planning approaches that construct a search graph using random samples, environmental aspects, or a their combination. Approaches that are not typically classified as sampling-based approaches are also covered here since they do not constrain points of their path to some discretisation of the workspace.

Before the seminal paper [10] that established the optimal properties of rapidly exploring random tree (RRT), [68] used a combination of RRT and A* to generate optimal paths for underwater gliders in 2D time-invariant flow fields. In their work, RRTs are grown from the start and the goal to a fixed number of nodes with some destination bias and the parent selection bias from previous work [69]. Dynamics constraints are enforced by limiting subsequent nodes to only have a maximum of 20° turns. When nodes are added

that are also within some Euclidean distance of a node from the other tree, the node makes a connection to the other tree. At the end of this process, nodes are pruned from the leaves of these RRTs until they reach a node with another successor. If there was a connection between the trees, then we are left with a graph with connections between the start and the goal. A* is then run on this graph for time- or energy-optimal paths using their formulated cost and heuristic functions. The algorithm provides optimal paths with respect to the constructed graph, however the algorithm does not generate optimal paths with respect to the workspace due to the way the graph is connected.

Chakrabarty and Langelaan [70] proposes a tree-based approach to find a feasible path that accounts for the dynamics of an aerial glider in a 3D time-variant flow field. The aerial glider can only change its yaw and heading without any actuated thrust so it is desirable to utilise upward moving air to extend the flight duration to reach the goal. To account for these dynamics, motion primitives are used to describe the motion of the glider without wind effects. The motion primitives are pre-computed forward integrations of the glider dynamics using a pre-defined set of inputs for a set amount of time. A tree is then constructed using the combination of motion primitives and the advection effect of the flow field. The flow velocity is assumed to be constant during the period of time, so the displacement is taken to be the displacement from the motion primitive and the displacement of the flow velocity over the same amount of time. To guide the search, expansion is biased to nodes that have a higher ratio between energy and distance to goal, where energy is the sum of the glider's gravitational potential energy and kinetic energy.

Ko et al. [71] proposes vector field RRT (VF-RRT) which is a RRT-variant that uses a special steering function that extends nodes in the direction of the flow field. They introduce a concept called the *upstream criterion* to help to guide RRT growth. In multi-dimensional time-invariant irrotational flow fields, this produces energy-optimal paths. In the algorithm, the amount of RRT growth bias to the flow is probabilistic, but also depends on the overall "spreading" effectiveness of the tree. This spread is quantified by tracking how often new nodes are created nearby existing tree nodes. The algorithm is shown to generate paths in ~0.5 s that are comparable to energy-optimal paths from RRT* which took ~100 min, and also readily extends to VF-RRT*. However their work does not consider the case when the vehicle's maximum speed is less than the flow field's speed.

Furthermore, the authors suggest the use of the non-asymptotically optimal variant of their algorithm since it is much faster and performs relatively well. This technique appears to perform only relatively well in cases where the goal is downstream from the start. This can be useful in other applications where the vector field is something that is controllable, e.g. using potential fields to guide robot arm planning to a desired pose.

Lee et al. [8] uses the fast marching tree (FMT) algorithm to find energy-optimal paths for an underwater glider in a 3D time-invariant flow field. In contrast to the work by Chakrabarty and Langelaan [70], this approach accounts for vehicle dynamics within a graph edge to minimise the energy usage. A concept of trim states from the aeronautical community [72] is used to describe the steady state dynamics of the vehicle given a set of controls between two positions. This is not to be confused with the process of trimming AUVs (described in [73–76]), which "calibrates" the vehicle's buoyancy given particular control inputs. This paper also shows that underwater gliders, which are designed to be energy efficient, could be more energy efficient if the motion of the underwater glider is allowed to be composed of a more complex set of motions, instead of following the industry standard of sawtooth patterns [77–82]. By accounting for the flow field advection effects in the planning process, an underwater glider is shown to avoid opposing flow and be able to traverse further in assistive flow.

Yoo et al. [83] addresses a problem of environmental uncertainty when path planning for a vehicle subject to advection in a 2D time-invariant flow field. In their approach, a sequence of controls is found to minimise the destination error. Using an ensemble of flow fields from ensemble forecasting techniques, a set of "realities" are tracked in the Monte Carlo tree search (MCTS) framework, each of which follow the same vehicle control that is randomly generated but are each advected by the flow field defined by their corresponding ensemble member. The result of this implementation provides a sequence of controls that can be executed by the vehicle that takes it nearby the goal on any of the ensemble member flow fields. This can be interpreted as path planning robust to flow field uncertainties assuming that the ensemble is reflective of real flow.

2.2.5 Summary

In this section, we reviewed various approaches for path planning in flow fields. Path planning that solely uses optimisation techniques are able to easily incorporate system dynamics and constraints for solutions that are usually only locally optimal. By discretising the workspace, discrete space approaches are able to leverage graph search algorithms like A* which provide optimality guarantees, however the resulting paths are only optimal with respect to the constructed graph which usually limits the possible angles the paths can take. By considering a variable that describes the reachability frontier, continuous space methods are able to produce smoother paths that are also shown to be optimal, but this is also subject to discretisation of the variable. Level set approaches are a useful set of approaches for particular formulations with a lower computational complexity, however it is unclear how it could be extended to consider vehicles with dynamic constraints such as limited turning rates. Furthermore, the algorithm is tailored to particular cost functions so further research would be needed to consider others.

Sampling-based path planning shows a lot of promise for our particular problem. It constructs paths using segments from a process of random sampling, giving it a chance to consider many different directions. Some algorithms like RRT* are also anytime and have the property of asymptotic optimality. RRT has also been shown to be able to handle dynamic constraints [70] which is useful in some applications. In Sec. 2.3, we give an overview of sampling-based path planning algorithms and some of its theory.

2.3 Sampling-based path planning

In this section we provide an overview for popular sampling-based planners in literature. The list of planners we discuss are by no means complete, but should provide a sense of their capabilities. We also provide some definitions and establish some theory, which is useful for discussion in this thesis.

Sampling-based path planners generally use samples from the state space to construct a search graph from the start towards the destination. The samples from a sampling

sequence usually do not have regular structure and should cover the state space more and more densely, allowing the planner to construct paths with them.

We will start with a light introduction of *primitive functions*, which are functions that are commonly used across different algorithms here. We will then describe probabilistic roadmap (PRM) in Sec. 2.3.2, rapidly exploring random graph (RRG) in Sec. 2.3.3, RRT in Sec. 2.3.4 FMT* in Sec. 2.3.5, informed tree methods in Sec. 2.3.6, and stable sparse RRT methods in Sec. 2.3.7. This section is summarised in Sec. 2.3.8.

2.3.1 Primitive functions

There are commonly used functions called primitive functions that appear across different sampling-based algorithms. These functions are, in some sense, customisable so that the sampling-based path planner can be used in different applications. In this section, we briefly describe these primitive functions and provide an interpretation of their purpose.

2.3.1.1 Cost functions

The primitive function `Cost` returns the cost to traverse between two states. Sampling-based planning algorithms are able to minimise different types of path properties by changing this function. Some planners require the function to satisfy the triangle inequality to confer optimality properties.

Some planners take advantage of *Bellman's principle of optimality*, and consider the `CostToCome` function or $g(\texttt{node})$ which returns the current minimum cost from the start to node `node`. This means that the path cost is computed as the sum of the cost of sequential pairs. For path costs that are multiplicative, `Cost` can return the log of the path segment cost. This leads to multiplicative path costs that can be recovered by taking the exponent. The corresponding function between `node` and the destination will be denoted `CostToGo` and $h(\texttt{node})$.

Some planners leverage a heuristic function that compute estimates the path cost between a node and either the start or the destination. In this thesis, which we will denote heuristic

functions with a tilde. E.g. in A*, the heuristic function would be denoted as $\tilde{h}(\texttt{node})$. The heuristic path cost refers to $\tilde{g}(\texttt{node}) + \tilde{h}(\texttt{node})$, and the semi-heuristic path cost refers to $g(\texttt{node}) + \tilde{h}(\texttt{node})$. Nodes with low heuristic path costs will be referred to as *encouraging*, whilst nodes with low semi-heuristic path cost will be referred to as *promising*.

In most algorithms, it is desirable for heuristics to underestimate the true path cost, as it leads the search to optimal paths. In these kinds of algorithms, heuristic functions that underestimate are called *admissible*. It is common for heuristics to be some function of the Euclidean distance between the states due to simple evaluation.

2.3.1.2 `SampleFree`

Given an index (e.g. iteration number) this function returns a feasible state. The feasibility of a state usually refers to a state that is in what's known as *free* space, i.e. they are within the workspace and are not inside obstacles or forbidden regions. If the state also describes properties like speed, this could also imply that the state is feasible under the constraints of system dynamics.

States are usually stochastically sampled from a uniformly random distribution over the state space. The probabilistic nature of sampling is important as it allows algorithms to consider states that increasingly cover the state space to find the optimal path. A measure of this coverage is known as *dispersion* which describes the largest distance between a point in space and its closest sample, so a smaller value corresponds to a higher spatial coverage. Deterministic sampling sequences such as the Halton sequence [84] are shown to have dispersion decay rates higher than probabilistically sampling from a uniformly random distribution [11]. Palmieri et al. [85] describe an algorithm that generates a deterministic sampling sequence. It works by iteratively choosing samples inside the largest area of uncovered space which reduces dispersion by definition. The choice of sampling sequence impacts on the convergence rates and properties of the sampling-based planner. Semi-deterministic sampling, which is the process of randomly choosing points from a deterministic set of points, are shown to give some planners desirable properties [86].

In planning algorithms that use samples as reference points, goal biasing [69] is sometimes applied to the sampling process which interweaves the destination as a sample after a number of the original sampling sequence, which encourages growth of the search towards the destination. Local biasing has been proposed [87] which attempts to re-sample intermediate nodes of a given path giving extra opportunities for the algorithm to refine the quality of the path. Akgun and Stilman [87] also suggest a sample rejection scheme, which skips samples which are not encouraging. This idea is further studied by Ferguson and Stentz [88], Otte and Correll [89], and Gammell et al. [90], by considering samples that are only generated in areas of the state space that are encouraging. This prevents the computational burden of rejection sampling at later stages of the search, when samples are less often encouraging in comparison to the refined cost of the path.

2.3.1.3 `Near` and `Nearest`

The function `Nearest` returns the code from a set of nodes that is the "closest" to a given node. The function `Near` returns a subset of nodes from a set of nodes that are considered "close" to a given node. The exact meaning of close for these two functions is not clear from literature, however some [91–94] suggest that it should reflect the cost function of interest. If that is the case, it can be difficult to implement effective ones for certain cost functions. Some algorithms only require the notion of distance in state space, regardless of cost function. The `Near` can sometimes be implemented as returning the closest k neighbours, and sometimes be implemented as returning the neighbours within some specified distance.

A brute force implementation of these functions take $\mathcal{O}(n_{\texttt{node}})$ time, where $n_{\texttt{node}}$ is the number of nodes in the original set. Fortunately, there are algorithms that can provide approximations of these functions e.g. Karaman and Frazzoli [10] claim that balanced box decompositions (BBDs) [95] are sufficient with implementations taking $\mathcal{O}(\log n_{\texttt{node}})$ time to query.

2.3.1.4 `Steer`

To deal with systems that involve kinematics or dynamics, some planners require a `Steer` function, which determines the control and trajectory that joins two states which sometimes require two-point boundary-value problem (TP-BVP) solvers. Certain systems, e.g. those that use Dubins or Reeds-Shepp models, have fast solvers for their TP-BVP.

2.3.1.5 `Extend`

This primitive function is typically used to grow a search tree (e.g. RRT) at the specified node after being stimulated with a given state, usually one from a sampling sequence. Implementations of `Extend` that drive the system towards another state tend to work better in practice. Karaman and Frazzoli [10] suggests a geometric implementation which returns a state in the direction of the given state, limiting the new node to be within a steering distance of the tree node which is shown to be probabilistic completeness by Kleinbort et al. [13], however this variant may not be possible for some systems.

For systems subject to kinematic constraints, Kunz and Stilman [96] shows that using RRT with an `Extend` implementation that attempts multiple random inputs for a fixed amount of time and returns the closest resulting state leads to probabilistic incompleteness. However, Kleinbort et al. [13] shows that simply generating a random state by taking a random input for a random amount from the tree node makes the overall algorithm probabilistic completeness.

2.3.1.6 `CollisionFree`

This function is generally used to determine whether the transition from one state to another is possible as it may intersect with obstacles, or leave the workspace.

Whilst some algorithms uses `CollisionFree` as new edges are added to the graph, *lazy* algorithms choose to do so later avoiding computation for edges that are not useful to the optimal path.

2.3.2 Probabilistic roadmap

There are slightly different definitions of PRM (e.g. [9, 97]) so in this thesis, any implementation that is equivalent to Alg. 4 in [10] for a given implementation of the `Near` primitive function will be referred to as PRM.

In the PRM algorithm, a graph is constructed by first creating a set of nodes which consists of the initial state and a number of randomly sampled states. For each node, an edge is created to nearby nodes if it corresponds to a feasible transition. To query the roadmap, a query node is added to the graph and edges are created with nearby nodes if they correspond to feasible transitions. A graph search algorithm such as A* is then used on the resulting graph to return a path. Typical usage of A* can actually be viewed as PRM with grid-based sampling and a particular neighbourhood definition.

The asymptotic optimality of PRM was first established in [10] given `Near` implementations based on radius and k nearest. If PRM uses a `Near` function with radius

$$r(n_{\texttt{node}}) = \gamma_{\text{PRM}} \left(\frac{\log n_{\texttt{node}}}{n_{\texttt{node}}} \right)^{\frac{1}{n_d}}, \tag{2.6}$$

where

$$\gamma_{\text{PRM}} > 2 \left(1 + \frac{1}{n_d}\right)^{\frac{1}{n_d}} \left(\frac{\lambda(\mathsf{X}_{\text{free}})}{\nu_d} \right)^{\frac{1}{n_d}}, \tag{2.7}$$

$n_{\texttt{node}}$ is the number of nodes in the graph, n_d is the dimension, $\lambda(\mathsf{X}_{\text{free}})$ is the Lebesgue measure of the obstacle free space, and ν_d is the volume of a unit n_d-dimensional hypersphere, then

$$\Pr\left(\left\{ \lim_{n_{\texttt{node}} \to \infty} c_{\mathsf{X}} = c^* \right\} \right) = 1, \tag{2.8}$$

where c_{X} is the cost of the path returned for $n_{\texttt{node}}$ samples, and c^* is the optimal cost.

Similarly, if PRM uses a `Near` function with the number of nearest neighbours

$$k(n_{\texttt{node}}) = k_{\text{PRM}} \log n_{\texttt{node}}, \tag{2.9}$$

where

$$k_{\text{PRM}} > \exp(1) \left(1 + \frac{1}{n_d}\right)^{\frac{1}{n_d}}, \tag{2.10}$$

then (2.8) also holds. A benefit of using this `Near` function is that it is not necessary to compute $\lambda(\mathsf{X}_{\text{free}})$ the volume of the free space.

In [98], a smaller radius can be used by PRM for a slightly different definition of asymptotic optimality. If a smaller radius used as in (2.6) except

$$\gamma_{\text{PRM}} > 2\left(\frac{1}{n_d}\right)^{\frac{1}{n_d}}\left(\frac{\lambda(\mathsf{X}_{\text{free}})}{\nu_d}\right)^{\frac{1}{n_d}}, \tag{2.11}$$

then

$$\lim_{n_{\texttt{node}}\to\infty}\Pr\left(c_{\mathsf{X}} \leq (1+\varepsilon)c^*\right) = 1, \tag{2.12}$$

for all $\varepsilon > 0$.

PRM is shown to have asymptotic near optimality in [99] when an even smaller radius is used. If we use the radius in (2.6) with

$$\gamma_{\text{PRM}} > 2\left(\frac{1}{2d\nu_d}\right)^{\frac{1}{n_d}}, \tag{2.13}$$

then

$$\lim_{n_{\texttt{node}}\to\infty}\Pr\left(c_{\mathsf{X}} \leq \zeta(1+\varepsilon)c^* + o(1)\right) = 1, \tag{2.14}$$

for all $\varepsilon > 0$ and $\zeta \geq 1$.

A critical radius is found in [86] to have asymptotic near optimality for PRM in Euclidean space. A special primitive function `Near_query` is used to connect the start and the destination to the set of nodes. There exists $\beta_0 > 0$ such that if `Near_query` uses the radius

$$r(n_{\texttt{node}}) = \frac{\beta(\log n_{\texttt{node}})^{\frac{1}{n_d-1}}}{n_{\texttt{node}}^{\frac{1}{n_d}}}, \tag{2.15}$$

and `Near` uses a radius

$$r(n_{\texttt{node}}) > \gamma^*\left(\frac{1}{n_{\texttt{node}}}\right)^{\frac{1}{n_d}}, \tag{2.16}$$

where $\beta \geq \beta_0$, and for all $n_d \geq 2$, $0.4 \leq \gamma^* \leq 0.6$, some values of which are provided in [86], then

$$\lim_{n_{\texttt{node}}\to\infty}\Pr\left(c_{\mathsf{X}} \leq \zeta(1+\varepsilon)c^*\right) = 1. \tag{2.17}$$

An interesting outcome of this, is that nodes will only have $\Theta(1)$ number of neighbours, instead of $\mathcal{O}(\log n_{\texttt{node}})$.

Many different variants of PRM have been proposed such as fuzzy PRM [100], customisable-PRM [101], differential PRM [102, 103], lazy-PRM [104, 105], dancing PRM [106], soft-PRM, Bluetooth-PRM, and embedded-PRM [99].

2.3.3 Rapidly exploring random graphs

The RRG algorithm [10] incrementally generates a graph so it be interpreted as an incremental version of PRM. A graph is first created with only the starting node. Then for each iteration, a new node is created by first a randomly sampling a state, picking the closest node in the graph, then `Steer` from the closest node to the random state for the new node. If the transition from the closest node to the new node is a feasible transition, then it is added to the graph. In addition to that, an edge is created from the new node to nearby nodes if it also corresponds to a feasible transition. To query this roadmap, a query node is added as if it's the new node. Then a path is returned by using a graph search algorithm. RRG is anytime, so it means that if the iterations are interrupted, the roadmap can still be queried for a path.

RRG is shown to be asymptotically optimal if `Near` implementations are based on radius or k nearest as well. If RRG uses a `Near` function with radius

$$r(n_{\texttt{node}}) = \min\left(\gamma_{\text{RRG}}\left(\frac{\log n_{\texttt{node}}}{n_{\texttt{node}}}\right)^{\frac{1}{n_d}}, \eta\right) \tag{2.18}$$

where

$$\gamma_{\text{RRG}} > 2\left(1+\frac{1}{n_d}\right)^{\frac{1}{n_d}}\left(\frac{\lambda(\mathsf{X}_{\text{free}})}{\nu_d}\right)^{\frac{1}{n_d}}, \tag{2.19}$$

then (2.8) holds.

Similarly, if RRG uses a `Near` function with the number of nearest neighbours

$$k(n_{\texttt{node}}) = k_{\text{RRG}} \log n_{\texttt{node}} \tag{2.20}$$

where

$$k_{\text{RRG}} > \exp(1)\left(1 + \frac{1}{n_d}\right)^{\frac{1}{n_d}}, \tag{2.21}$$

then (2.8) also holds. Other than not needing to compute the volume of free space, this `Near` function also doesn't require a steering distance to be defined. This could be useful if `Steer` does not have a well-defined steering distance.

In [86], the analysis of the critical radius for PRM is shown to extend to RRG as well by showing that a RRG contains a PRM. This means that RRGs are asymptotic near optimality by the definition of (2.17), if the radius is defined by (2.15) and (2.16) in Euclidean spaces.

Two variants of RRG are RRT$^{\#}$ [107], and lazy-RRG [105].

2.3.4 Rapidly exploring random trees

The RRT algorithm [91] is an algorithm designed for planning with non-holonomic systems. It is similar to RRG since RRT influenced it; the difference is a tree is constructed instead which requires less edge connections. The tree is first created with only the starting node. Then for each iteration, a new node is created by first a randomly sampling a state, picking the closest node in the tree, then `Steer` from the closest node to the random state for the new node. If the transition from the closest node to the new node is a feasible transition, then it is added to the tree. To query this tree, a query node is added as if it's the new node. Then a path can be obtained by recursively following the parent node of the query node to the root node (the starting node). RRT is also anytime, so it can be queried after interrupting its iterations.

The algorithmic properties of RRT has been hard to establish since its behaviour is highly dependent on the definition of the function `Steer`. It has been established early that it is not asymptotically optimal [10], but whether it is probabilistically complete has been the focus of recent studies. If `Steer` is implemented as to give the resulting state from using the control from a set that moves the state closest to the random node for a fixed duration, then RRT is not probabilistic completeness [96]. If `Steer` is implemented as to give the resulting state from using a random control for a random duration, then RRT is

probabilistic completeness [13]. It is also shown in [13] that if `Steer` returns a node that is directly in line towards the random node, then RRT would be probabilistic completeness. The probabilistically complete property is particularly significant, as many algorithms derive algorithmic properties from this property including asymptotic optimality.

Many different variants of RRT have been proposed such as RRT-Connect [108], resolution complete RRT [109], heuristically guided RRT, iterative k-nearest RRT, best of k-nearest RRT [69], dynamic-domain RRT [110], dynamic RRT [111], multi-partite RRT [112], transition-based RRT [113, 114], multi-RRT [115], stable sparse RRT, its asymptotically optimal variant [116], discrete-RRT [117], its asymptotically optimal variant [118], deformable RRT [119], and generalised bi-directional RRT [120].

2.3.4.1 The asymptotically optimal upgrade

RRT* [10] is a version of RRT with the asymptotically optimal property. It is motivated by trying to construct an RRG as a tree without redundant edges for query. This is achieved by considering the path costs of the tree's local neighbourhood of the new node. After producing a new node, instead of connecting to the closest node on the tree, the node is connected to a neighbouring node that minimises the path cost from the neighbour. For all neighbouring nodes, the new node is set as the parent if the path cost through the new node is better. This "rewiring" step ensures that the edges that are part of the minimum-cost path from RRG are still in the tree.

RRT* is shown to be asymptotically optimal if `Near` implementations are based on radius or k number of nearest neighbours. If RRT uses a `Near` function with the number of nearest neighbours in (2.20) then (2.8) holds. The radius for asymptotic optimality initially described in [10] is corrected in [121]. If `Near` uses radius

$$r(n_{\texttt{node}}) = \gamma_{\text{RRT}} \left(\frac{\log n_{\texttt{node}}}{n_{\texttt{node}}} \right)^{\frac{1}{n_d+1}} \tag{2.22}$$

where

$$\gamma_{\text{RRT}} \geq (2+\theta) \left(\frac{\left(1+\frac{\varepsilon}{4}\right) c^*}{\theta(n_d+1)(1-\mu)} \right)^{\frac{1}{n_d+1}} \left(\frac{\lambda(\mathsf{X}_{\text{free}})}{\nu_d} \right)^{\frac{1}{n_d+1}}, \tag{2.23}$$

$\mu \in (0, 1)$, and $\theta \in (0, \frac{1}{4})$, then (2.12) holds. It is unclear how to choose the variables μ and θ in general. It is probabilistically better to pick smaller θ and larger μ, however this corresponds to larger radius which corresponds to higher computational demands. In practice, $(1 + \varepsilon/4)\, c^*$ can be chosen to be the cost of the best path so far.

Variants of RRT* have been proposed such as RRT*-Smart [12], bi-directional RRT [122], kinodynamic RRT* [123], RRT*_S [124], informed RRT* [90, 125], intelligent bi-directional RRT [126], RRT$^{\mathrm{X}}$ [127], transition-based RRT* [128], and sorted RRT* [129].

2.3.5 Fast marching tree

The FMT algorithm (FMT*) [98] is algorithmically similar to the fast marching method and Dijkstra's algorithm, but it finds a path by constructing a tree using sampled nodes. Three mutually exclusive sets of nodes are maintained. Nodes can only move in one direction between these sets from the unvisited set, to the open set, then finally to the closed set. At the start, the start node is placed in the open set which represents the frontier of the search. A number of sampled nodes and destination node(s) are added to the unvisited set. Then for the lowest cost node $\texttt{node}_{\mathrm{low}}$ in the open set, a set of unvisited neighbours are found. For each unvisited neighbour node $\texttt{node}_{\mathrm{nb}}$, an edge is created with a nearby open node that gives $\texttt{node}_{\mathrm{nb}}$ the lowest path cost from the start if the edge corresponds to a feasible state transition. This elaborate expansion process can be interpreted as finding the neighbourhood that needs to be added next to the tree (unvisited neighbours), then connecting them in the best way to the tree. The unvisited nodes that are connected to the tree are then moved to the open set. After all unvisited neighbour nodes are considered, then $\texttt{node}_{\mathrm{low}}$ is moved to the closed set. The expansion process is repeated until a destination node is the lowest code node in the open set, or when the open set is exhausted, which means no path is found. The path is then returned by recursively following the parent of the destination node.

FMT* is shown to be asymptotically optimal. If `Near` uses a radius defined by (2.6) with (2.11), then (2.12) holds. Furthermore, if `Near` uses the number of nearest neighbours

$$k(n_{\texttt{node}}) = k_{\mathrm{FMT}} \log n_{\texttt{node}}, \tag{2.24}$$

where

$$k_{\text{FMT}} > 3^d e \left(1 + \frac{1}{n_d}\right), \tag{2.25}$$

then (2.8) holds.

The critical radius for PRM in [86] also extends to FMT*. This means that FMT* is asymptotic near optimality by the definition of (2.17), if radii are defined by (2.15) and (2.16) in Euclidean spaces.

Three variants of FMT* are motion planning using lower bounds (MPLB) [130], bi-directional FMT* [131], and differential FMT* [102, 103].

2.3.6 Informed trees

The sampling-based algorithms described so far have been indiscriminately searching through the entire workspace for the entire duration of the search process. *Informed* approaches focuses search effort to areas which contain promising states. A state can be described as promising if the sum of cost estimates from the start to the state with the state to the destination is low. This idea is related to A*'s use of admissible heuristics, which promotes expansion only at nodes which are promising. Two informed tree searches that are not covered in this thesis are regionally accelerated batch informed trees [132], and dominance-informed region trees [133].

2.3.6.1 Informed RRT*

One of the earlier informed search approaches builds on RRT*. Informed RRT* [90, 125] makes three main adjustments to RRT*. Firstly, after the first solution is found, samples are only drawn from the *informed set*, an implicitly defined ellipsoidal region containing encouraging states. This means that the ellipsoid would have the start and the destination at its focal points. Secondly, nodes outside of the informed set are carefully pruned, retaining those with descendents in the informed set. Finally, the `Near` should be updated to utilise the volume of the informed set. The paper refers to the radius in [10] for

asymptotic optimality, however it would be prudent to use (2.22) and (2.23) from [121]. The number of nearest neighbours (2.20) following [10] can be used.

2.3.6.2 Batch informed tree

Batch informed tree (BIT*) [129, 134] can be contrasted with RRT* and FMT*. RRT* can be interpreted as an algorithm that searches every time a batches of one sample is added. FMT* can be interpreted as an algorithm that searches once for a single batch of samples. In contrast BIT*, searches each time after adding batches of multiple samples. Algorithmically, BIT* performs an A*-like search across the nodes within the informed set. This method of adding batches allows the algorithm to prioritise nodes that appear promising first. Searching through more promising nodes first means that better solutions could be found earlier. This help identify nodes that have less semi-heuristic path cost than the current best path cost, which can be discarded without processing. Note that these discarded nodes would have been considered in the informed RRT* algorithm. Fast-BIT* [135] is a variation that only considers $\tilde{h}(\texttt{node})$ before an initial solution is found. This speeds up the initial solution, after which the algorithm proceeds as BIT* does.

2.3.6.3 Advanced BIT*

A portion of the BIT* algorithm that was omitted has to do with the upkeep of node costs after sampling new batches of nodes. Advanced BIT* (ABIT*) [136] reduces the overhead by decreasing the amount of search effort spent in each batch iteration of the algorithm by choosing nodes to expand "optimistically", and ending batch iteration "pessimistically". The optimistic node expansion is based on a factor that inflates $\tilde{h}(\texttt{node})$. In A*, this leads to sub-optimal solutions, however it tends to find solutions faster. The pessimistic iteration termination inflates the heuristic cost through an edge when compared to the minimum path cost so far. This makes the algorithm neglect edges that only appear a little better than the minimum cost. Both of the effects of these factors are designed to decay every time a new batch of samples are taken which ensures that the algorithm is still asymptotically optimal.

2.3.6.4 Adaptive informed tree

A different extension of BIT* is the adaptive informed tree (AIT*) [137]. AIT* is an asymmetrical bi-directional search which uses BIT* in the forward direction and a variant of A* in the reverse direction. The forward search is responsible for constructing the minimal cost path, whilst the reverse search is responsible for constructing a heuristic $\tilde{h}(\texttt{node})$ that better represents the problem compared to the typical choice to use Euclidean distance. This is particularly useful when it is difficult to formulate an effective heuristic, especially in cases where there are many obstacles. The reverse search can still utilise heuristic $\tilde{g}(\texttt{node})$ so that it guides the computation of $\tilde{h}(\texttt{node})$. It is important to note that the reverse search does not perform collision checks, which makes it faster than the forward search.

2.3.7 Stable sparse RRT

Another way of finding an optimal path is to first sparsely cover the workspace with nodes first, then slowly allow the space to be filled with more nodes. A sparse data structure is beneficial as the time complexity of planning algorithms tend to grow super-linearly with respect to the number of nodes in the data structure.

Stable sparse RRT (SST) [116] is proposed for achieving asymptotic near optimality without a boundary-value problem (BVP) solver. Two mutually exclusive node sets are maintained with nodes only moving from the active set to the inactive set after which they can be removed entirely. An additional *witness* node set is used as a representation of workspace coverage. Witness nodes "supervise" a δ_{S} neighbourhood, and also points to the tree node in its neighbourhood with the lowest path cost.

The algorithm begins by setting the starting state as the root node of the tree, and making it the first witness node. For each iteration, a random state is sampled to select the tree node in the δ_{BN} neighbourhood with the lowest path cost (nearest is selected if nothing is nearby). A new node is then obtained by using a `Steer` function that uses a random control for a random duration at the selected node. If this new node better represents the local witness, then it is added to the active set and the witness node will now point to this

node (new node becomes a witness if there is witness within δ_{S}). The node that is now replaced by the new node is then moved to the inactive set. This inactive set is necessary to maintain the descendent nodes that are part of the active set. A branch of inactive nodes are recursively deleted if a leaf node moves to the inactive set.

By maintaining the active and the inactive sets of nodes, the algorithm always ensures that witnesses always represents feasible paths that end in their neighbourhood. In practice, some thought needs to go into the selection of δ_{BN} because choosing a large number means that node expansion tends to occur with lower cost nodes which will discourage the consideration of alternate paths. It is also important not to make δ_{S} too large too. Whilst a large radius corresponds to a sparser representation, it can prevent the addition of nodes which are necessary to traverse through small gaps in state space. The authors suggest using δ_{S} to represent neighbourhoods small enough to fit between obstacles and smaller than the radius of the goal region. The authors assert that δ_{BN} should be larger than δ_{S} to ensure proper growth of the tree.

By choosing parameters such that

$$\delta_{\mathrm{BN}} + 2\delta_{\mathrm{S}} < \delta, \tag{2.26}$$

where δ represents the distance from the path to obstacles, then the algorithm is shown to be asymptotically near-optimal, i.e. for all $C_\Delta > 0$, there exists a $K_x > 0$ such that

$$\Pr\left(\left\{\limsup_{n_{\mathrm{node}}\to\infty} c_{\mathsf{X}} \leq \left(1 + \frac{K_x \delta}{C_\Delta}\right) c^*\right\}\right) = 1. \tag{2.27}$$

A variant called SST* is also proposed which achieves asymptotic optimality by calling the main SST algorithm multiple times with decaying δ values. This is done by multiplying the δ_{S} and δ_{BN} parameters by a number between 0 and 1, and an increasing number of SST iterations is used to give the algorithm more time to adjust the entire tree. This gradually relaxes the mechanisms in the algorithm that enforces the sparse structure allowing more optimal paths to be formed.

An informed variant of SST is proposed in [138] to focus its search. A major difference in how informed SST (iSST) works is that the node selection process of expansion relies on

a constructed heuristic function similar to AIT* [137]. In this paper, a shortest-path tree is constructed without collision checks using a PRM graph. Node selection for expansion is probabilistically biased to nodes that appear promising which guides the search.

2.3.8 Summary

Single batch approaches like PRM, RRG and FMT* are not desirable for the problem we consider in this thesis as they are not anytime. Tree-based approaches that are shown to be asymptotically optimal are ideal. We do not expect stable sparse tree methods to converge faster than RRT methods since they are designed to avoid the use of a TP-BVP solver. On the other hand, informed tree methods are shown to be faster than RRT methods as they leverage heuristics to the destination. In Ch. 4, we find an efficient way to solve the TP-BVP between two positions in 2D incompressible flow fields. In Ch. 5, we propose a distance heuristic that better describes the local reachability of any point in strong 2D incompressible flow. The techniques proposed in these chapters can be used to improve applications of tree-based path planners for ocean navigation.

Chapter 3

Flow field estimation using spatial correlations

In this chapter, a new algorithmic framework is proposed to maintain an estimate of the flow field specifically designed to have a small computational footprint for the purposes of runtime execution onboard underwater gliders with limited computational capacities. The proposed algorithms avoid heavy computation by leveraging the result of rigorous considerations and computations involved in the generation of ensemble forecasts from organisations such as the Australian Bureau of Meteorology (BOM). This estimator is responsible for providing an estimate of the environment behaviour so that the system can account for the advecting effect of ocean currents in the path planning module.

Parts of this chapter have been published as two separate conference paper publications [139, 140].

3.1 Introduction

Estimates of ocean currents are critical for many applications of marine robots, particularly in supporting the mobility of slow autonomous vehicles such as underwater gliders. A vehicle that is comparatively slower than ocean currents means that the effects of the environment on the vehicle must be considered in the path planning process to ensure

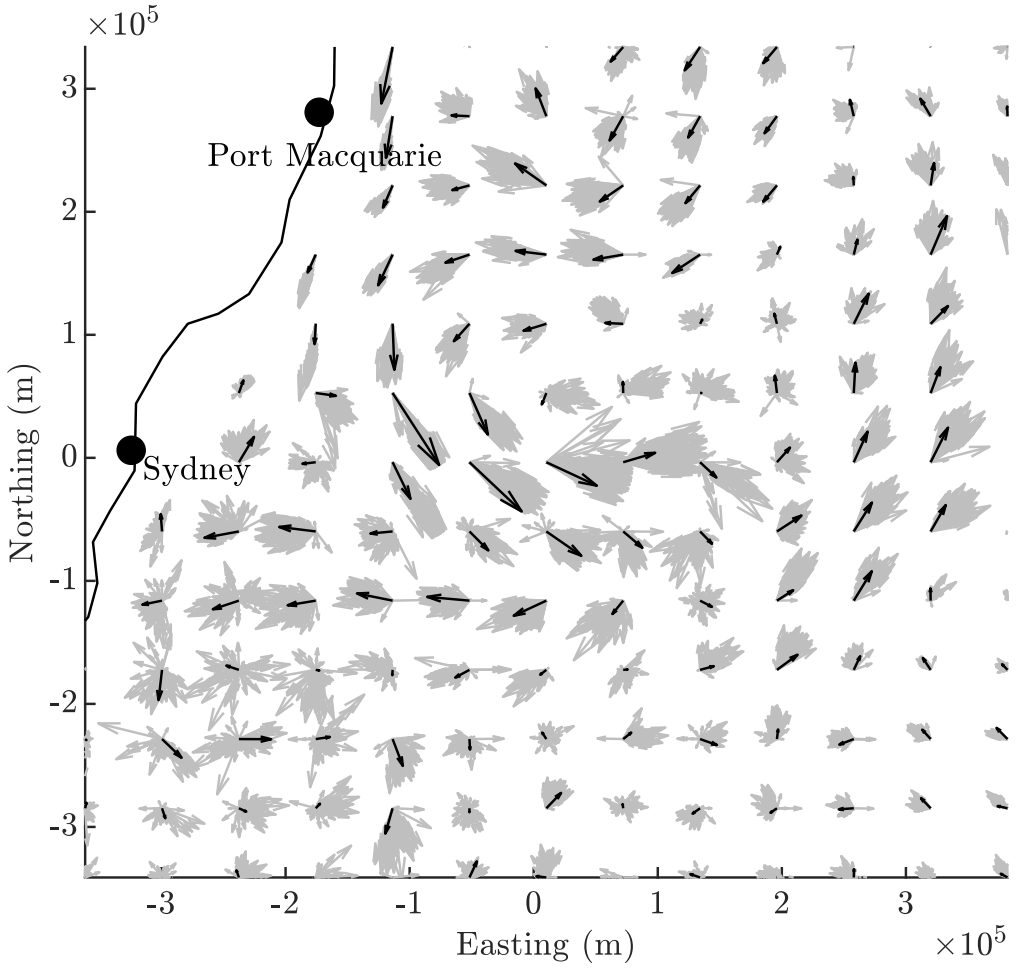


FIGURE 3.1: Subsampled ensemble flow field data off the coast of New South Wales from the Australian BOM. This ensemble forecast consists of $n_{\mathsf{E}} = 96$ flow fields, each predicting ocean surface conditions for 16$^{\text{th}}$ November 2018. One of the 96 flow fields is shown in black, whilst others are shown in grey.

compensating controls are used. While flow field estimation has long been a topic of study in oceanography, obtaining and digesting meteorological flow field estimates in a form that is useful for practical robotics applications is less well-understood. We are interested in a representation of the flow field that can be informed by forecast data and also exploits a robot's ability to take measurements of the environment during traversal to produce realistic flow field estimates.

Forecast data from organisations such as the Australian BOM are often produced using ensemble forecasting, where a set (or, *ensemble*) of predicted flow fields is generated from a range of initial conditions. While none of the ensemble members themselves are likely to be

exactly correct, the ensemble as a whole tends to contain instances of largely similar flow patterns, an example of which is in Fig. 3.1. However, the ensemble format is awkward for robotics applications because planning algorithms, even those that address probabilistic representations of the environment, typically assume a single probabilistic environment model as opposed to a set of predictions [56, 141]. The challenge in producing a single model is how to distil the information contained in the ensemble in a way that captures uncertainty and that preserves spatial correlations seen in the ensemble data.

In this chapter, novel 2D and 2.5D flow field estimation methods are proposed which produce a single probabilistic model. These methods are built upon the idea that reoccurring spatially correlated flow patterns can be found across ensemble members. These patterns with different levels of detail can be used as useful building blocks to produce different flow fields. In the algorithms, the patterns are first extracted using kernel methods and the singular value decomposition (SVD), then at runtime the contribution of each flow pattern is determined for the flow field estimate by considering measurements taken by the vehicle using a recursive Bayesian estimator. This effectively utilises a compressed model represented as a linear combination of the flow fields that inherit spatially correlated flow fields found in the ensemble data, and only produces estimates reminiscent of the flow fields present in the ensemble data. This compact representation is key to achieving computational efficiency.

For the algorithm considering the three dimensional case, an additional insight is considered that assumes that ocean flow sufficiently far from the coastline has negligible vertical velocity [142–144]. This allows the environment to be treated as a 2.5D flow field, or a collection of 2D horizontal flow fields. Instead of considering flow patterns that across each of the ensemble members, this also finds patterns between different depths as well, allowing a larger basis to be formed increasing the expressivity of the compressed model.

Both proposed methods have an interesting property allowing them to make global adjustments of the estimate from local measurements. They also have more desirable computational times than existing work, with the routines to update and to query the velocity of flow having constant computational complexity and taking less than 1 ms each in our implementation. The low computational demand of this estimation technique allows the

navigation system of the underwater glider to quickly adapt to changing environments, the predictions of which are known to diverge from reality in short time spans compared to operational periods of underwater gliders.

More specifically, the 2D approach is compared against the incompressible Gaussian process (GP) [38] and a variant of our approach which is algorithmically similar to the kernel observer [40]. Theoretical and empirical comparisons are made between these methods showing the differences in computational complexity of measurement integration and flow velocity queries. The proposed method is then tested using real forecast ensemble data from the Australian BOM. The results suggest that the proposed method could also be used in an active perception context where measurement locations are intentionally chosen to reduce model uncertainty.

The 2.5D approach is compared against a direct extension of the 2D approach using BOM data in an area of the Tasman Sea between Australia and New Zealand. Results are reported with and without measurement noise highlighting the accuracy of the method. In several cases, the root-mean-square (RMS) error of the achieved error of the proposed algorithm fell below the RMS of the lower bound of the direct 2.5D extension. The algorithms and results in this chapter is a promising step forward in understanding how forecasts of underwater current can be used to produce useful flow field estimates for planning. Although this work focuses on the case of time-invariant flows, it lays a foundation to begin to address the challenges of the time-variant case.

In the next section, the flow field estimation problem is formally defined. Section 3.3 presents an overview of our approaches in 2D and 2.5D to the underdetermined problem. The 2D algorithm and its results are described in Sec. 3.4. Similarly, Sec. 3.5 describes the 2.5D algorithm and its results. Finally, Sec. 3.6 summarises the chapter.

3.2 Problem formulation

We consider the problem of estimating an unknown time-invariant incompressible flow field given an *ensemble forecast* and a sequence of flow velocity measurements at different locations. First, let $\mathbb{X}$ be a bounded subset of $\mathbb{R}^2$ in the context of the 2D flow field

estimation problem and a bounded subset of $\mathbb{R}^3$ for the 2.5D variant. Flow velocities in the 2.5D problem will not have a vertical component as they are assumed to be negligible.

Recall that, a flow field is incompressible if its velocity field is *non-divergent*. The aim is then to estimate a flow field $\tilde{\mathbf{f}}(\mathbf{x}) : \mathbb{X} \to \mathbb{R}^2$ with the constraint

$$\nabla \cdot \tilde{\mathbf{f}}(\mathbf{x}) = 0. \tag{3.1}$$

An ensemble forecast E is defined as a set of n_{E} *ensemble members*. The ensemble members correspond to a common set of n_{X} positions $\mathsf{X}_{\text{ens}} \subset \mathbb{X}$, each describing a set of flow velocities or *flow vectors*

$$\mathsf{e} = \left\{ \hat{\mathbf{f}}(\mathbf{x}) \,:\, \mathbf{x} \in \mathsf{X}_{\text{ens}} \right\} \in \mathsf{E}, \tag{3.2}$$

such that the ensemble member consists of the discretisation of a single ocean current prediction $\hat{\mathbf{f}} : \mathbb{X} \to \mathbb{R}^2$. The ensemble members of the forecast are deterministically generated using slightly different initial conditions so that the entire ensemble forecast describes multiple predictions of the ocean state. Note that the ensemble members e do not consist of measurements of the true flow field, but are predictions of flow velocities based on the simulation of ocean models.

The sequence of flow vector measurements are taken at the sequence of positions $\mathsf{X}_{\text{mea}} = (\mathbf{x}_k)_{1 \leq k \leq n_{\text{mea}}}$ and are denoted as

$$\mathsf{Z} = \left(\mathbf{f}_{\text{mea}}(\mathbf{x}_k) \,:\, \mathbf{x}_k \in \mathsf{X}_{\text{mea}} \right), \tag{3.3}$$

where $\mathbf{f}_{\text{mea}} : \mathbb{X} \to \mathbb{R}^2$ is the measurement function. While the positions are fully known, the measurement function $\mathbf{f}_{\text{mea}}$ is subject to sensor noise, such that

$$\mathbf{f}_{\text{mea}}(\mathbf{x}) = \mathbf{f}_{\text{tru}}(\mathbf{x}) + \nu, \tag{3.4}$$

where $\mathbf{f}_{\text{tru}} : \mathbb{X} \to \mathbb{R}^2$ is the true continuous unknown flow field and $\nu \in \mathbb{R}^2$ is the sensor noise model.

Problem 1 (Flow field estimation with ensemble data and measurements)**.** *Given the ensemble forecast* E *and a sequence of measurements* Z*, find the incompressible estimate* $\tilde{\mathbf{f}}$

for the true continuous flow field $\mathbf{f}_{\text{tru}} : \mathbb{X} \to \mathbb{R}^2$*:*

$$\tilde{\mathbf{f}}^*(\mathbf{x}) = \underset{\tilde{\mathbf{f}}(\mathbf{x}) \in \mathsf{C}^\infty(\mathbf{x})}{\arg\min} \quad \sum_{k=1}^{n_{\text{mea}}} \left\| \mathbf{f}_{\text{mea}}(\mathbf{x}_k) - \tilde{\mathbf{f}}(\mathbf{x}_k) \right\|_2^2, \tag{3.5}$$
$$\text{subject to} \quad \nabla \cdot \tilde{\mathbf{f}}(\mathbf{x}) = 0, \quad \forall \mathbf{x} \in \mathbb{X},$$

where $\|\cdot\|_2$ *is the Euclidean norm (2-norm) of a vector, and* $\mathsf{C}^\infty(\mathbf{x})$ *is the set of smooth 2-vector valued functions on* $\mathbf{x}$*.*

3.3 Approach overview

As established in Sec. 2.1, the flow field estimation in Prob. 1 is severely underdetermined, as there are infinitely many smooth, non-divergent vector fields that can fit a finite number of measurements Z. We propose to constrain the representation of the flow field estimate $\tilde{\mathbf{f}}$ to only use the spatially correlated flow fields or *spatial correlations* present in the ensemble forecast. To do this, we propose to construct $\tilde{\mathbf{f}}$ as the product between a basis $\boldsymbol{H} : \mathbb{R}^2 \to \mathbb{R}^{2 \times n_{\text{W}}}$, and a weight vector $\mathbf{w} \in \mathbb{R}^{n_{\text{W}}}$:

$$\tilde{\mathbf{f}}(\mathbf{x}) = \boldsymbol{H}(\mathbf{x})\mathbf{w}, \tag{3.6}$$

for n_{W} degrees of freedom in representation. In this representation, the estimate $\tilde{\mathbf{f}}(\mathbf{x})$ can be interpreted as a linear combination of *basis flow fields*, each of which correspond to spatial correlations.

We propose a two-stage estimation framework consisting of offline and online components. In the offline component, the flow field estimate is modelled using a basis $\boldsymbol{H}(\mathbf{x})$ that is chosen based on the ensemble forecast E and an additional incompressibility prior. This is achieved by describing the ensemble members as continuous incompressible flow fields using a set of latent variables. The spatial correlations from E can then be extracted by taking the SVD of the sets of latent variables to construct $\boldsymbol{H}(\mathbf{x})$, preserving the flow patterns observed in the ensemble forecast. The offline component is described as having *regression* and *compression* steps. The online component uses a Bayesian filter to iteratively update $\mathbf{w}$ using measurements of the true flow field which are often sequential.

3.4 Estimation of time-invariant 2D flow fields

This section describes the proposed estimation algorithm for time-invariant flow fields for 2D flow in 2D space. First, the regression and compression steps of the offline component will be described in Sec. 3.4.1 and 3.4.2. Section 3.4.3 will then describe how the offline components are used in the online component. The algorithm is then analytically compared with other algorithms in Sec. 3.4.4. This is followed by empirical comparisons in Sec. 3.4.5. Finally, some findings for the application of the method in active perception contexts are shared in Sec. 3.4.6.

3.4.1 Flow field regression through kernel embedding

To obtain continuous flow field representations from the discrete data of the ensemble members, a process of regression using kernels is used. Existing work [38, 40, 41] have successfully used kernels to represent flow fields in the past. In particular, the *incompressible kernel* [38] has recently been shown to be an apt description of smoothness and incompressibility of physical 2D flow fields. The incompressible kernel $\boldsymbol{K}_{\text{incomp}} : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}^{2\times 2}$ can be expressed as

$$\boldsymbol{K}_{\text{incomp}}(\mathbf{x}, \mathbf{x}') = \mathscr{D}(\mathbf{x}) k(\mathbf{x}, \mathbf{x}') \mathscr{D}\left(\mathbf{x}'\right)^{\mathsf{T}}, \tag{3.7}$$

where a 2D position is denoted as $\mathbf{x} = [x_1, x_2]^{\mathsf{T}}$, the kernel describing the similarity of stream function values between two positions is denoted as $k : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, and $\mathscr{D}(\mathbf{x})$ is the differential operator

$$\mathscr{D}(\mathbf{x}) = \begin{bmatrix} \frac{\partial}{\partial x_2} & -\frac{\partial}{\partial x_1} \end{bmatrix}^{\mathsf{T}}. \tag{3.8}$$

An appropriate kernel function can be chosen as the inner kernel to better model the flow field, which can lead to better convergence properties. In this scenario, a reasonable choice in kernel would be the squared exponential-automatic relevance determination (SE-ARD) described in (2.4) as we expect the values of the stream function of two positions to be similar when the positions themselves are near one another.

We will then define the notation for the application of the incompressible kernel between two sets of positions in a similar fashion to (2.5). Let X_A and X_B be sets of n_A and n_B

positions. The incompressible kernel matrix is then:

$$\boldsymbol{\mathcal{K}}_{\text{incomp}}(\mathsf{X}_A, \mathsf{X}_B) = \begin{bmatrix} \boldsymbol{K}_{(1,1)} & \cdots & \boldsymbol{K}_{(1,n_B)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{K}_{(n_A,1)} & \cdots & \boldsymbol{K}_{(n_A,n_B)} \end{bmatrix} \in \mathbb{R}^{2n_A \times 2n_B}, \tag{3.9}$$

where $\boldsymbol{K}_{(i,j)} = \boldsymbol{K}_{\text{incomp}}(\mathbf{x}_i, \mathbf{x}_j)$ for $\mathbf{x}_i \in \mathsf{X}_A$ and $\mathbf{x}_j \in \mathsf{X}_B$.

The incompressible flow field can now be expressed with a *latent state vector* $\beta \in \mathbb{R}^{2n_\mathsf{X}}$ and the incompressible kernel between any spatial location and the ensemble positions with

$$\tilde{\mathbf{f}}(\mathbf{x}) = \boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\beta. \tag{3.10}$$

The flow field data is said to be "embedded" in the kernel as β. In our approach, the estimated flow field is refined by adjusting values in the latent state representation. Our choice of the kernel function $\boldsymbol{K}_{\text{incomp}}$ ensures that the resulting flow field is always incompressible. Note that representation through the kernel embedding allows us to represent a flow field estimate that is continuous across space, acting like a "smart" interpolator for flow velocity estimates between the positions in X_{ens}.

The ensemble is also expressed in this latent state representation. We can now perform regression to find a set of latent state vectors B for the ensemble, or more specifically

$$\mathsf{B} = \left\{ \arg\min_{\beta} \|\text{vec}\,(\mathsf{e}_i) - \boldsymbol{\mathcal{K}}_{\text{incomp}}(\mathsf{X}_{\text{ens}}, \mathsf{X}_{\text{ens}})\beta\|_2^2 \,:\, \mathsf{e}_i \in \mathsf{E} \right\}, \tag{3.11}$$

where $\text{vec}\,(\cdot)$ arranges the elements of the given set into a column vector in the same order as the positions in (3.9), i.e. $\text{vec}\,(\mathsf{e}_i) \in \mathbb{R}^{2n_\mathsf{X}}$.

3.4.2 Model compression by the SVD

In (3.10), the kernel matrix $\boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})$ could be considered a candidate $\boldsymbol{H}(\mathbf{x})$, which is how the authors of [40] model their flow field. However, it does not preserve the spatial correlations that appear in the ensemble, and the number of variables to estimate is $2n_\mathsf{X}$ which can be fairly large. Here, a $\boldsymbol{H}(\mathbf{x})$ is constructed that both preserves the

spatial correlations through the SVD, and reduces the number of variables to estimate, hence the name "compression".

To preserve the spatial correlations, we need to ensure that the vector of flow velocities at the ensemble positions lies in the span of the discretised flow fields, i.e.

$$\operatorname{vec}\left(\left\{\tilde{\mathbf{f}}(\mathbf{x}_i) \,:\, \mathbf{x}_i \in \mathsf{X}_{\text{ens}}\right\}\right) \in \operatorname{span}\left(\{\operatorname{vec}(\mathsf{e}_j) \,:\, \mathsf{e}_j \in \mathsf{E}\}\right) \subseteq \mathbb{R}^{2n_\mathsf{X}}. \tag{3.12}$$

By tuning the hyperparameters of the inner kernel k, the kernel matrix $\boldsymbol{\mathcal{K}}_{\text{incomp}}$ is full rank, so we can preserve the spatial correlations by finding a latent state vector β in (3.10) such that

$$\beta \in \operatorname{span}(\mathsf{B}). \tag{3.13}$$

One way to satisfy this constraint is to represent β as a linear combination of modes (i.e. left singular vectors) obtained from the SVD. First, let $\boldsymbol{B} \in \mathbb{R}^{2n_\mathsf{X} \times n_\mathsf{E}}$ be the horizontally concatenated matrix of the column vectors in B. The thin SVD of $\boldsymbol{B}$ is

$$\boldsymbol{B} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H, \tag{3.14}$$

with $n_\Sigma = \min(2n_\mathsf{X}, n_\mathsf{E})$ singular values, where $\boldsymbol{U} \in \mathbb{R}^{2n_\mathsf{X} \times n_\Sigma}$, $\boldsymbol{\Sigma} \in \mathbb{R}^{n_\Sigma \times n_\Sigma}$, and $\boldsymbol{V} \in \mathbb{R}^{n_\mathsf{E} \times n_\Sigma}$. Since the number of ensembles is typically much less than that of the variables in the latent state vector (i.e. $n_\mathsf{E} \ll 2n_\mathsf{X}$), the matrix of left singular vectors $\boldsymbol{U}$ is rectangular whilst the other two matrices are square.

The columns of $\boldsymbol{U}$ are the modes that describe the different spatial correlations found in $\boldsymbol{B}$ as normalised vectors. The basis $\boldsymbol{H}$ can now be defined as

$$\boldsymbol{H}(\mathbf{x}) = \boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\boldsymbol{U}, \tag{3.15}$$

along with the weight vector $\mathbf{w}$ having $n_\mathrm{W} = n_\Sigma$ elements.

From the other two matrices, we form the *weight matrix*

$$\boldsymbol{W} = \boldsymbol{\Sigma}\boldsymbol{V}^H \in \mathbb{R}^{n_\Sigma \times n_\mathsf{E}}. \tag{3.16}$$

The columns of the product are *weight vectors* which describes the amount of each mode that is required to reconstruct the corresponding ensemble member.

A benefit of using the SVD is that the singular values in $\mathbf{\Sigma}$ impose an implicit ordering on the modes, and describe the relative importance of them, i.e. the basis flow field corresponding to the largest singular value is the most important in the reconstruction of the ensemble. The significance of the basis flow fields with higher singular values can be confirmed by visualising them. They can be visualised by querying arbitrary positions on the columns of $\boldsymbol{H}(\mathbf{x})$, or

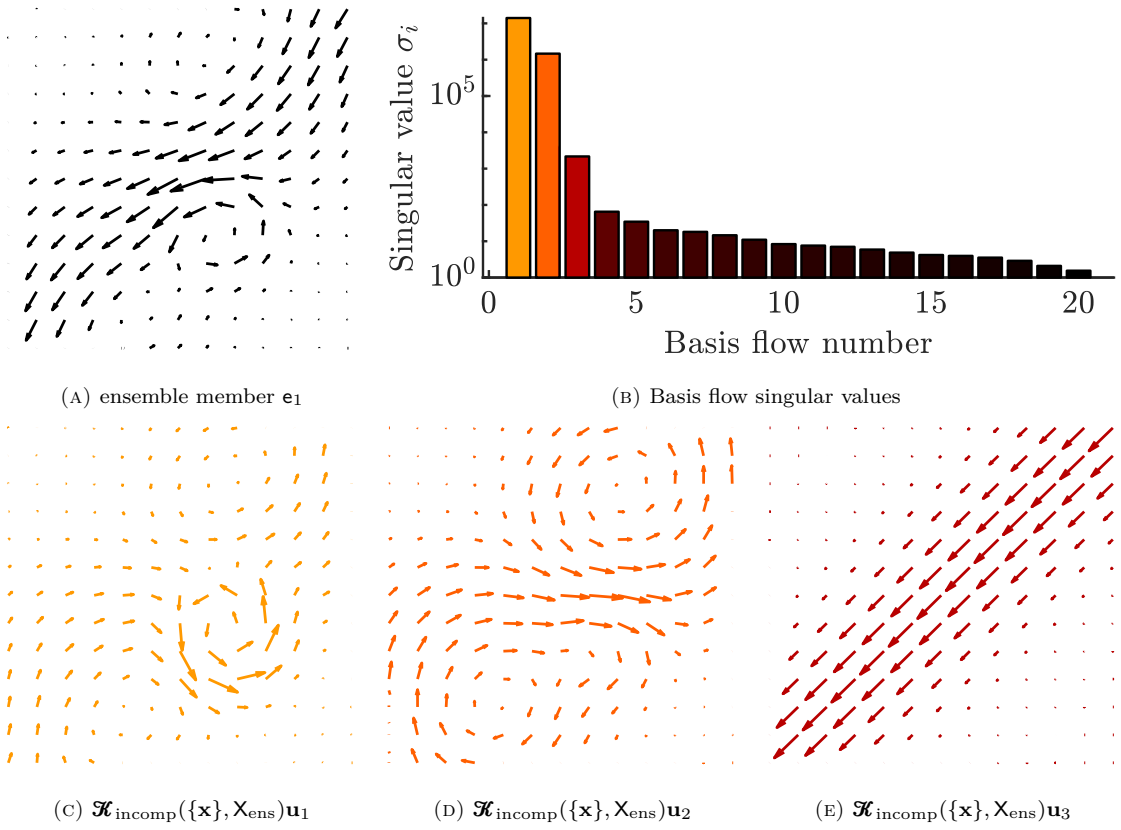
$$\mathscr{K}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\mathbf{u}_i, \tag{3.17}$$

where $\mathbf{u}_i$ is the i^{th} column of $\boldsymbol{U}$. Without loss of generality, the singular values are assumed to be in descending order.

Figure 3.2 shows an example of this that uses an ensemble consisting of flow fields similar to the one in Fig. 3.2a. From Fig. 3.2b, it can be seen that the three basis flow fields corresponding to the three largest singular values in Fig. 3.2c-e are significantly more important than the rest. They also largely describe the spatially correlated flow field structures seen in Fig. 3.2a.

We are primarily addressing the case where ensembles describe much more velocities than it has members, or more specifically $n_{\mathsf{E}} < 2n_{\mathsf{X}}$, for which this approach offers both the preservation of spatial correlations and a reduced number of variables to estimate. However this approach still offers preservation of spatial correlations in the case of $2n_{\mathsf{X}} \leq n_{\mathsf{E}}$.

In both cases, this approach also offers the option to *truncate* representation in a principled way, which further reduces the number of estimated variables in the first case, and provides some level of reduction in the second. To truncate the representation to n_{r} variables, only the columns and rows in each of the SVD matrices corresponding to the highest n_{W} singular values are kept. Intuitively, this means that the effect of basis flow fields in the ensemble that are not likely to occur or vary less are discarded. In cases where the number of modes is relatively high, significant truncation can be performed with surprisingly small loss in expressive power of $\boldsymbol{H}(\mathbf{x})$. Choosing a $n_{\text{W}} = n_{\text{r}} < n_{\Sigma}$ results in a reduction of the



(A) ensemble member e_1 (B) Basis flow singular values

(C) $\mathscr{K}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\mathbf{u}_1$ (D) $\mathscr{K}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\mathbf{u}_2$ (E) $\mathscr{K}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\mathbf{u}_3$

FIGURE 3.2: Model compression for ensemble flow fields. (b) shows the singular values in descending order. (c-e) shows the three most significant basis flow fields which can be used to reconstruct the flow field in (a).

span of the column vectors in $\boldsymbol{U}$, which directly impacts the expressivity, so it needs to be balanced with the compactness of representation.

Models that are produced by the combined steps of regression from Sec. 3.4.1 and compression from above describe flow fields as linear combinations of continuous spatially correlated flow patterns instead of trying to construct a flow field from local flow velocities. This representation allows us to make flow field adjustments in other parts of the flow field based on spatial correlations.

3.4.3 Online flow field estimation

We described the offline stage of our approach, representing a flow field as a weight vector $\mathbf{w}$ with each value corresponding to the basis flow fields from the ensemble data. Next, we

propose a process to incorporate information from noisy online measurements to improve estimation accuracy by recursively updating $\mathbf{w}$ given an initial estimate of the flow field.

Despite being time-invariant in this chapter, we model the true flow field as a degenerate discrete-time linear dynamical system, making the time-variant variant a straightforward extension. The process and measurement models of the true flow field are

$$\mathbf{w}_k = \boldsymbol{F}\mathbf{w}_{k-1} \tag{3.18}$$

$$\mathbf{f}_{\text{mea}}(\mathbf{x}_k) = \boldsymbol{H}(\mathbf{x}_k)\mathbf{w}_k + \tilde{\nu}, \tag{3.19}$$

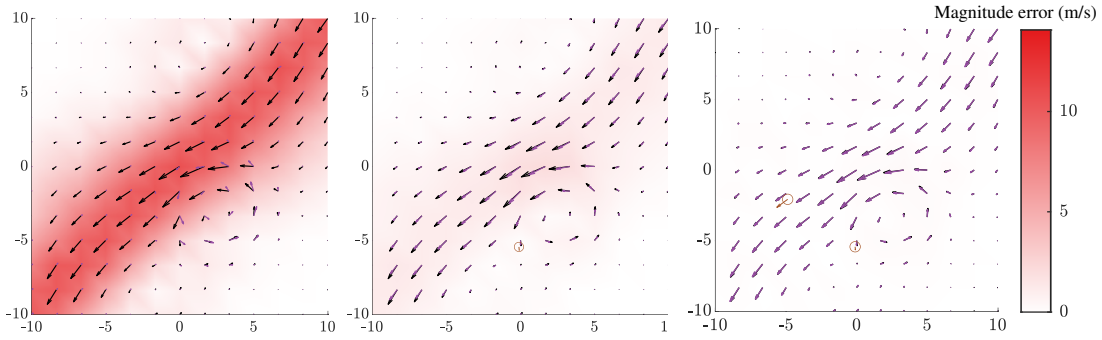
where $\boldsymbol{F} = \boldsymbol{I}_{n_{\text{W}}}$ is the state transition model for the static flow field is the identity matrix, $\mathbf{x}_k$ is the k^{th} measurement location, and $\tilde{\nu} \sim \mathcal{N}(\mathbf{0}_2, \widetilde{\text{cov}}(\mathbf{f}_{\text{mea}}))$ is the estimated sensor noise model with covariance $\widetilde{\text{cov}}(\mathbf{f}_{\text{mea}}) \in \mathbb{R}^{2\times 2}$. The Kalman filter [145] is employed as it is the optimal estimator for measurements with Gaussian noise [146]. Since the Kalman filter equations are well-known (e.g. [147]), they are omitted for brevity.

As a recursive estimator, the Kalman filter needs to be initialised with some estimate. Recall that the columns of the weight matrix $\boldsymbol{W}$ are the weight vectors that correspond to each ensemble flow field. If we consider the ensemble members as samples from a distribution of likely flow fields, we can form an initial estimate by taking the sample mean and sample covariance of the columns in $\boldsymbol{W}$. Initialising the Kalman filter in this way expediently assumes that the estimated flow field is a random variable described by the same probability distribution from which the ensemble members are sampled.

Figure 3.3 shows an example of the estimated flow fields after k measurements. The rapid convergence to the true flow field can be attributed to a good initial estimate from the ensemble data, and well-placed measurements.

3.4.4 Analysis

We consider the computational time complexity of the *initialise*, *update*, and *query* procedures. Our proposed algorithm will be compared with three related algorithms: the kernel observer (KO) [40], the incompressible GP [38], and a variant of our algorithm that uses least squares (LS) regression instead of a Kalman filter. The analysis will include



(A) $k = 0$ (B) $k = 1$ (C) $k = 2$

FIGURE 3.3: Estimated flow fields after k measurements. The estimated flow field (purple) quickly converges to the true flow field (black) after a few measurements. The red heatmap shows the magnitude flow field error across space. Measurements are shown as circled brown arrows.

the two cases that arise from the SVD, i.e. $2n_{\mathsf{X}} < n_{\mathsf{E}}$ and $n_{\mathsf{E}} < 2n_{\mathsf{X}}$. Note that all the compared approaches produce incompressible flow field estimates through the use of the incompressible kernel [38].

The initialise procedure refers to any offline processing that the estimation algorithm needs prior to any measurements, including the processing of the ensemble. This procedure is considered to be the least time-critical out of the three procedures. The update procedure refers to the assimilation of a single measurement at any position into the flow field estimate. The query procedure refers to the evaluation of $\tilde{\mathbf{f}}(\mathbf{x})$ for a single position $\mathbf{x}$. This procedure is the most critical as the flow field estimate serves as part of a robot's model of the world, and can be queried many times for other processes such as path planning.

Our implementation of the KO algorithm is based on the framework used in [40]. In their work, a time-variant flow field is given over discrete steps in time at a discrete set of locations. The data is then represented as a kernel embedding and a least-squares based regression is used to find a transition matrix for the evolution of the embedding from one time to the next. For comparison, we use their model to estimate a flow field state instead of the transition matrix given an ensemble and measurements. Our implementation of the KO estimates the embedding β and is initialised using $\boldsymbol{B}$ in contrast to our method which estimates $\mathbf{w}$ initialised with $\boldsymbol{W}$. This can be interpreted as being equivalent to our approach without the compression step as we considered at the start of Sec. 3.4.2.

TABLE 3.1: Computational complexity of flow estimation algorithms

	Initialise	Update	Query
KO	$\mathcal{O}\left(n_{\mathsf{X}}^3 + n_{\mathsf{X}}^2 n_{\mathsf{E}}\right)$	$\mathcal{O}\left(n_{\mathsf{X}}^3\right)$	$\mathcal{O}\left(n_{\mathsf{X}}\right)$
GP	$\mathcal{O}\left(n_{\mathsf{X}} n_{\mathsf{E}}\right)$	$\mathcal{O}\left(1\right)$	$\mathcal{O}\left((n_{\mathsf{X}} + n_{\text{mea}})^3\right)$
LS	$\mathcal{O}\left(n_{\mathsf{X}}^3 + n_{\mathsf{X}}^2 n_{\mathsf{E}}\right)$	$\mathcal{O}\left(n_{\Sigma}^3 + n_{\Sigma}^2 n_{\text{mea}} + n_{\mathsf{X}} n_{\Sigma} n_{\text{mea}}\right)$	$\mathcal{O}\left(n_{\mathsf{X}} n_{\Sigma}\right)$
Ours	$\mathcal{O}\left(n_{\mathsf{X}}^3 + n_{\mathsf{X}}^2 n_{\mathsf{E}}\right)$	$\mathcal{O}\left(n_{\Sigma}^3 + n_{\mathsf{X}} n_{\Sigma}\right)$	$\mathcal{O}\left(n_{\mathsf{X}} n_{\Sigma}\right)$

We compare with an implementation of the GP algorithm using the incompressible kernel from [38]. Recall from Sec. 2.1.4 that GPs produces estimates from all the gathered measurements, which improves modelling accuracy but leads to computational time increasing as more measurements are collected. Here we consider the initialise and update procedures to simply accumulate measurements, whilst the query procedure includes a matrix inversion and subsequent multiplication with the measurements. The ensemble data is incorporated as measurements at the n_{X} positions, each averaged over the n_{E} members.

The last related algorithm is a variant of our proposed algorithm that uses weighted LS (WLS) instead of the Kalman filter framework. This corresponds to simultaneously solving equations for the accumulated measurements that are inversely weighted by the variances. This means that this algorithm does not need to use an initial state estimate. Note that a WLS formulation including the initial state would give the same estimates as a Kalman filter for linear systems disregarding numerical differences.

Recall that n_{X} is the number of positions the ensemble uses, n_{E} is the number of ensemble members, n_{mea}, and $n_{\Sigma} = \min\left(2n_{\mathsf{X}}, n_{\mathsf{E}}\right)$ is the number of singular values from the SVD. The computational complexities for these algorithms for the three procedures are arranged in Tab. 3.1. Note that the number of estimated variables n_{W} for our method and its LS variant is upper bounded by n_{Σ}.

The time complexity to initialise for KO, LS, and our method is $\mathcal{O}\left(n_{\mathsf{X}}^3 + n_{\mathsf{X}}^2 n_{\mathsf{E}}\right)$ due to the dominating shared operation performing regression for the latent state vectors in B. The time complexity for GP is $\mathcal{O}\left(n_{\mathsf{X}}^3 + n_{\mathsf{X}} n_{\mathsf{E}}\right)$. The first term arises from the matrix inversion done ahead of time to make queries faster. The second term comes from taking the statistical summary of the ensemble over each of the n_{X} positions. Note that for the case where $n_{\mathsf{E}} \leq 2n_{\mathsf{X}}$, the computational complexity of this procedure for all approaches becomes $\mathcal{O}\left(n_{\mathsf{X}}^3\right)$.

The time complexity to update is dominated by some cubic operation that is present across all methods. If $2n_{\mathsf{X}} \leq n_{\mathsf{E}}$, then KO and our method is $\mathcal{O}\left(n_{\mathsf{X}}^3\right)$ to update the covariance of a state sharing the same size. However in the other case, our method has a lower complexity of $\mathcal{O}\left(n_{\mathsf{E}}^3 + n_{\mathsf{X}} n_{\mathsf{E}}\right)$. This is due to the compression of β to the low-dimensional representation $\mathbf{w}$, unlike the KO approach where compression is not used. The LS method has a similar time complexity as our method, however the time is also dependent on the accumulated number of measurements since it solves the least squares problem with all measurements each time. The GP method has a computational complexity that increases cubically with the number of measurements due to an inversion of a Gram matrix from (3.9) between the positions of the collected measurements and itself. Any procedure that has a complexity dependent on the number of measurements becomes eventually intractable over time without additional intervention. The intractability of the GP update procedure is even more pronounced due to the cubic relation.

Due to the implementation choice to minimise the time complexity of the query procedure, the overall complexities across methods are comparatively low. The KO method has the smallest time complexity of $\mathcal{O}\left(n_{\mathsf{X}}\right)$. The query procedure for our method and the LS variant is algorithmically identical and has a complexity of $\mathcal{O}\left(n_{\mathsf{X}} n_{\mathsf{E}}\right)$. The GP method has a time complexity of $\mathcal{O}\left(n_{\mathsf{X}} n_{\text{mea}}\right)$. Despite the prioritisation of procedures, the computational time for the query operation still increases over the number of measurements which is problematic in robotic applications.

The GP and LS flow field estimation methods are unsuitable for robotic applications since their time complexity of online procedures increases for every new measurement. The KO and our flow field estimation methods are constant in time complexity at runtime since n_{X} and n_{Σ} do not change at runtime. This is an important property, especially in long-term missions and closed-loop path planning where a robot continuously measures the flow field to update its estimate. For the update procedure, the time complexity of our method is at least as good as KO due to compression. For the query procedure, KO is computationally more efficient, however the magnitude of n_{X} and n_{E} are constrained by practical computational limits imposed by the initialise procedure, so KO and our method have similar query times in practice. Finally if truncation is permitted, the variable n_{Σ} is

effectively replaced by $n_{\mathrm{W}} = n_{\mathrm{r}}$ which will reduce the computational complexity for online procedures for our method.

3.4.5 Empirical results

In this section, we compare the performance of our method against existing approaches in numerical simulations. Our method is shown to reduce estimation error in a spatially correlated way across the environment while the existing approaches only make corrections in neighbourhood of the measurements. The simulations also provide validation to the computational time complexities established in Sec. 3.4.4 for the online procedures. Then, we test our method in realistic environments based on a real ensemble forecast provided by the BOM.

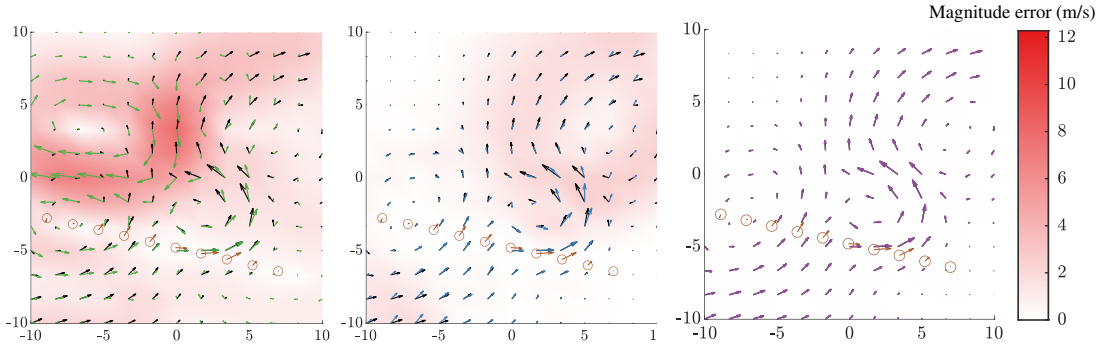
The errors of different measurement policies are shown to converge differently, showing the importance of measurement location selection. This is related to the field of active perception [148–150], where decisions are made to make future observations more effective. In both sets of simulations, we use the SE-ARD kernel

$$k(\mathbf{x}, \mathbf{x}') = \sigma_{\mathrm{ker}}^2 \exp\left(-\frac{1}{2}\sum_{d=1}^{D}\frac{(x_d - x_d')^2}{l_d^2}\right), \qquad \text{(2.4 revisited)}$$

and choose $l_1 = l_2$ since the flow field is expected to be rotationally similar about the vertical axis.

3.4.5.1 Estimation performance comparison with existing approaches

We compare the estimation quality of the compared approaches using a synthetic ensemble of $n_{\mathsf{E}} = 20$ flow fields over $n_{\mathsf{X}} = 169$ positions from ensemble members similar to the one in Fig. 3.2. The true flow field $\mathbf{f}_{\mathrm{tru}}$ is depicted as black arrows in Fig. 3.4. Each of the estimates for the compared approaches are visualised in the same figure as uniquely coloured arrows and the magnitude of the error between the estimates and the true flow field is visualised with a red heatmap. In this simulated scenario, the vehicle makes 10 measurements in a straight line 2 m apart starting from $\mathbf{x} = [6.98, -6.42]^{\intercal}$. The measurements



(A) Kernel observer (B) Incompressible GP (C) Our approach

FIGURE 3.4: Comparison of flow field estimation using equally spaced measurements (circled brown arrows) from the bottom right to the left. The magnitude error between the estimated and the true flow field (black) is shown as a red heatmap.

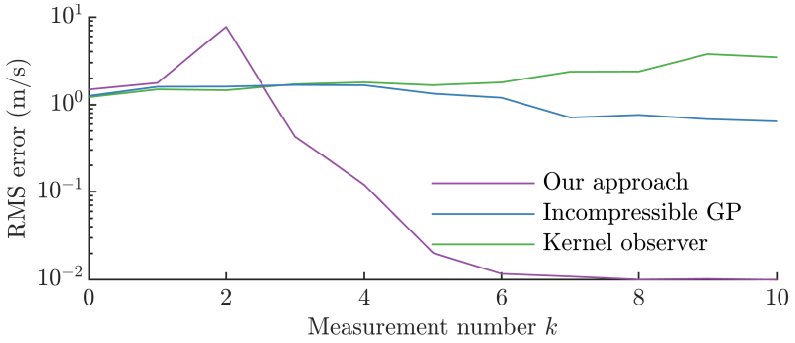


FIGURE 3.5: Root-mean-square (RMS) error from flow field estimation using different flow estimation techniques

have normally distributed noise with $\mathrm{cov}\,(\mathbf{f}_{\mathrm{mea}}) = 10^{-3}\boldsymbol{I}_2$ and are depicted as brown circles and arrows. The RMS error across these measurements are shown in Fig. 3.5.

It can be seen that the proposed approach leads to the least amount of error, followed by the GP with the incompressible kernel [38], then finally the KO framework. Interestingly, the error for all approaches rises for the initial measurements. In particular, KO rises for the entire scenario, GP rises a little then starts to drop, and finally the proposed approach rises significantly, but quickly decreases rapidly and converges. The increase in error at the beginning suggests that the initial estimate of the flow field is not similar to the true flow field.

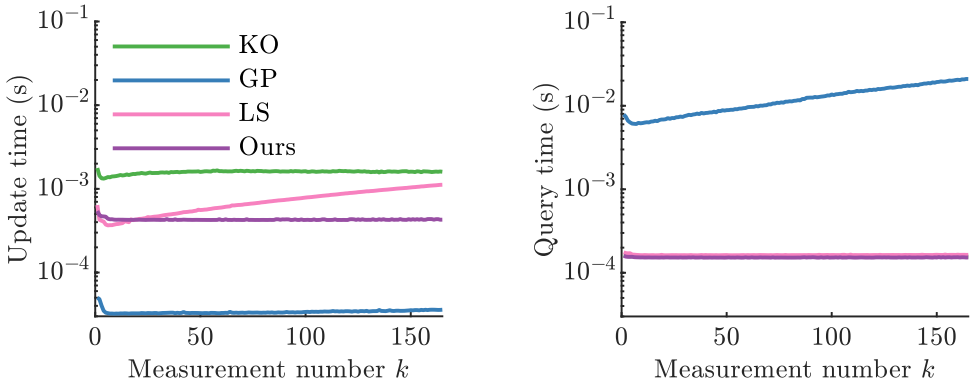
In this scenario, the KO framework needs to estimate the state β corresponding to $2n_{\mathsf{X}} = 338$ latent values. We suspect that the increasingly large amount of error is the estimator slowly trying to escape a region of local minimum as a result of low observability, i.e.

insufficient measurements taken in areas that inform about the values of the state variables, and the problem being underdetermined, i.e. there are only 20 values to try to estimate 338 values.

The GP approach performs better, as it synthesises a flow field estimate from all the data it has gathered as opposed to estimating a latent set of variables. This results in an estimate that is more consistent with the measurements in the local area compared to KO. However the updates from measurements only change the estimate in the local area determined by the choice of the kernel (2.4) and the hyperparameters used.

In contrast to KO and GP, the proposed algorithm leverages the spatially correlated information present in the ensemble to make global updates. For this scenario, the approach identifies three critical basis flow fields shown in Figs. 3.2c - 3.2e from the decomposition process in Sec. 3.4.2 that describe the ensemble well. As a consequence, we can choose $n_{\mathrm{W}} = n_{\mathrm{r}} = 3$, i.e. the number of basis flows is reduced from $n_{\mathsf{E}} = 20$ to 3 after truncation with virtually no degradation in estimation performance. Thanks to the smaller number of variables in its state estimate, it does not suffer from the underdetermined problem, and the representation of the estimate from spatially correlated flow fields improve the observability of the problem despite only measuring in a limited region of the area. A consequence of large corrections is that they can also increase the overall error quickly, however this scenario shows that this happens for a short period of time.

It is important to note that the GP has higher expressivity for flow field estimation since it constructs an estimate from past data. This means that it can theoretically produce more accurate flow field estimations compared to the proposed approach which confines the internal representation of kernels around a fixed set of points. However in practice, this depends on proper hyperparameter tuning, otherwise the estimate becomes numerically unstable over time. The proposed approach is more practical in application, as it is not as sensitive to sub-optimal hyperparameters.



(A) Iteration time per measurement

(B) Query time per position

FIGURE 3.6: Comparison of mean computational times between the kernel observer (KO) [40], incompressible Gaussian process (GP) [38], a least squares (LS) implementation of our approach, and our approach on the flow field shown in Fig. 3.4. Values are averaged over 1000 trials, the 99.73% confidence intervals of which are virtually imperceptible, and are omitted.

3.4.5.2 Computational time comparison with existing approaches

The computation times of the compared methods for the procedures we discussed in Sec. 3.4.4 are plotted in Fig. 3.6. This figure emphasises and verifies the computational time properties of the methods during online execution.

Figure 3.6a shows that the time it takes to perform an update is constant for KO and our method whilst GP and LS increases over time. The GP approach can be seen to increase faster over time compared to LS because n_{mea} is cubic in GP's procedure compared to linear in LS's procedure. It is also shown here that the time to update is shorter for our method than KO which can be a result of a smaller cubic term ($n_{\text{W}}^3 = 3^3 < n_{\text{X}}^3 = 169^3$).

Figure 3.6b shows that the time it takes to perform a query is constant for KO, LS, and our method while GP increases over time. The KO has a faster computational time compared to our method and the LS variant, however the difference appears to be negligible, and is not worthwhile compared to the difference in estimation quality observed earlier.

3.4.6 Measurement policies in real ensemble flow fields

We show how different measurement policies affect the estimation performance using our method. Three policies are compared: *uniform* where measurement positions are taken

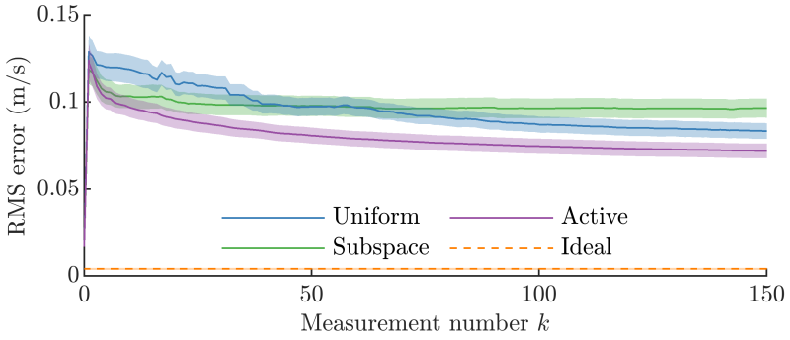


FIGURE 3.7: Root-mean-square (RMS) error from flow field estimation using different measurement policies using our approach. The flow field is based on the real ensemble data shown in Fig. 3.1. The 3σ (99.73%) confidence intervals for these errors are shown for each measurement policy.

in a uniformly random manner, *subspace* where measurements are only taken in a region with high uncertainty of the full workspace, and *active* where subsequent measurements are taken at positions with flow vector high uncertainty.

We use the top layer of forecasted currents in the Tasman Sea between Australia and New Zealand on 16$^{\text{th}}$ of November 2018 provided by the BOM. Notably, the forecast itself does not include data for the vertical velocity of the currents. A downsampled vector field of the ensemble is shown in Fig. 3.1. An azimuthal projection about the centre of the region was used to produce a Cartesian co-ordinate grid from the geographic co-ordinates of the data, i.e. latitude and longitude.

In Fig. 3.7, we show the RMS error with increasing number of past measurements. For reference, we have the *ideal* condition where measurements are taken exactly at ensemble positions X_{ens} such that the final error after all measurements is shown with dashed orange line. Since the true flow field is not known for this forecast data, the error for each policy is evaluated using *leave-one-out cross-validation (LOOCV)* [34], where we form performance statistics from trials: one for each ensemble member. In each trial, an ensemble member is set aside which is treated as the true flow field and the rest of the members are used to initialise the estimate. The performance metric used is the RMS of the residuals at X_{ens}, i.e. the RMS of

$$\text{vec}\left(\left\{\tilde{\mathbf{f}}(\mathbf{x}) - \mathbf{f}_{\text{tru}}(\mathbf{x}) \,:\, \mathbf{x} \in \mathsf{X}_{\text{ens}}\right\}\right) \in \mathbb{R}^{2n_{\mathsf{X}}}. \tag{3.20}$$

Any randomness in each measurement policy is controlled with a fixed random seed. The RMS error for each policy is shown in different coloured lines and their 3σ confidence interval is shown as a shaded band.

The results show that the active policy outperforms the other policies. The uniform policy initially underperforms compared to the subspace because the subspace policy is constrained to take measurements in a region with high uncertainty so it is able to reduce error quicker. However, as the number of measurements increases, the uniform policy starts to outperform the subspace policy because it can take measurements everywhere.

These results suggest that using our algorithm with a carefully chosen measurement policy can improve the performance of active perception algorithms. For example, suppose we need to find a set of unknown features in an ocean with unknown currents. An approach can find the location of the features as well as the velocities of the ocean flow which will enable faster traversal across the workspace since practical ocean vehicles are typically advected by ocean currents [76, 141, 151]. Our method is both theoretically and empirically validated for use in such applications.

3.5 Estimation of time-invariant 2.5D flow fields

In Sec. 3.4, we estimate an unknown 2D flow field as a linear combination of 2D basis flow fields derived from the ensemble forecast. If we were to apply this methodology directly to a 3D scenario, the derived basis flow field are "blocks" that describes the flow across 3D space. Here, we propose to use "layers" of basis flow fields. Multiple layers of flow fields across depth are estimated, each of which are linear combinations of the basis flow field layers derived from the 2D flow fields of all the depths of all the ensemble members. This representation is used to better capture the stratified nature of the ocean and to address some issues with the 2D approach. The full framework is illustrated in Fig 3.8.

The large portions of the ocean is stratified into horizontal layers based on the density of the fluid [14, 17]. The difference in densities hinders the vertical mixing of waters so most of the kinematic energy of the flow is directed into horizontal components. Indeed, measurements and best estimates place the vertical velocity of the ocean's current on the

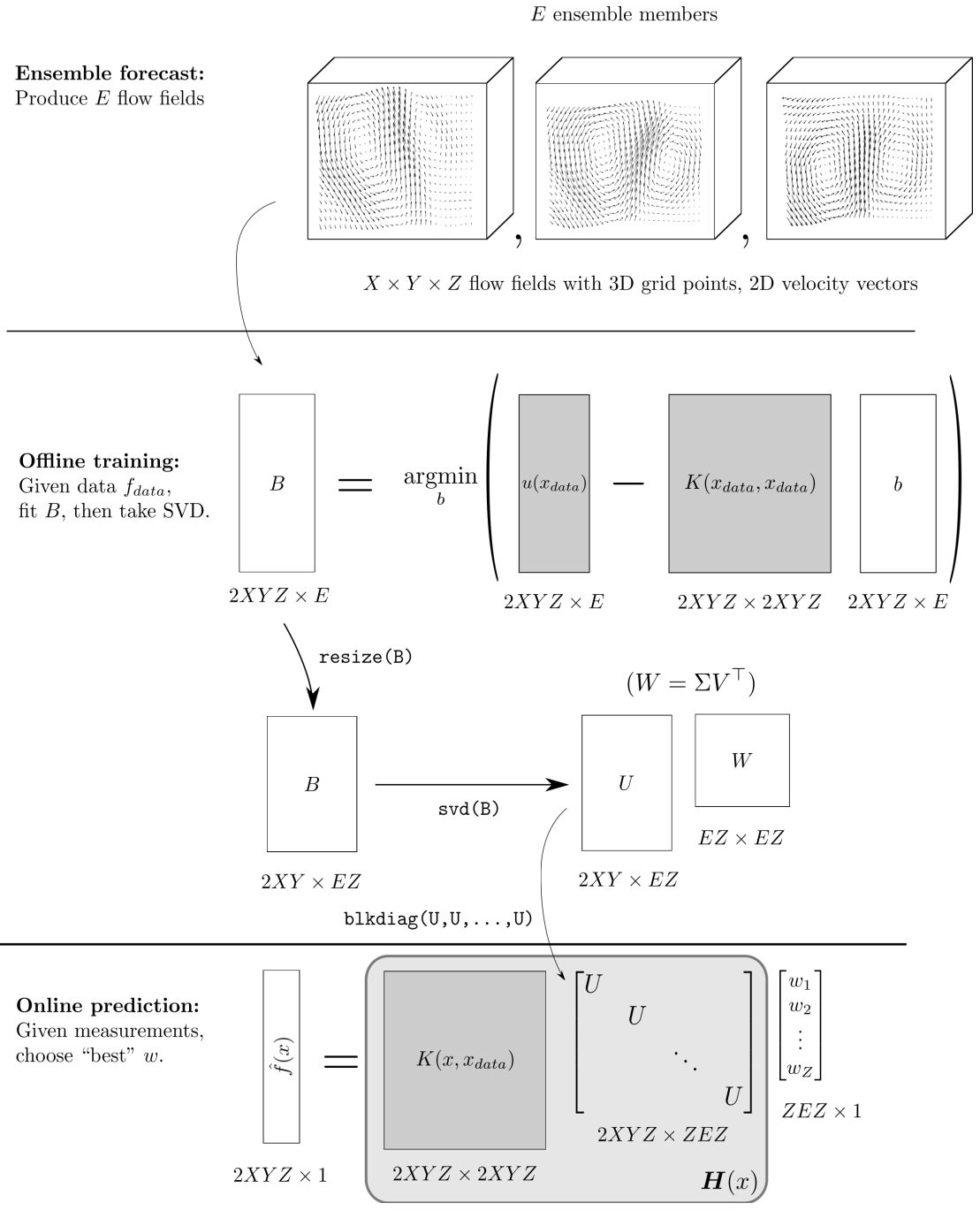


FIGURE 3.8: A sketch of the different components of $\tilde{\mathbf{f}}(\mathbf{x})$, showing the case of the thin SVD and where $n_\Sigma = n_\mathrm{V} n_\mathtt{E}$. A key algorithmic contribution is the `reshape`, which allows many more modes (i.e. left singular vectors) in $\boldsymbol{U}$, resulting in a large span of the basis $\boldsymbol{H}(\mathbf{x}) = \boldsymbol{K}(\mathbf{x}, \boldsymbol{X}_\mathrm{grid})\boldsymbol{U}$. Shaded grey boxes indicate mathematical objects that are fixed at each stage.

order of 10^{-5} m/s or less [142–144]. In contrast, the horizontal current in our dataset is on the order 10^{-1} to 10^{0} m/s. Since the flow is primarily horizontal, we can observe that the flow from one layer *vertically influences* nearby layers in depth due to shear forces.

A limitation of the direct extension is that it adopts an inherent shortcoming of the 2D approach. Both the 2D and direct 2.5D representation is subject to our assertion:

$$\operatorname{vec}\left(\left\{\tilde{\mathbf{f}}(\mathbf{x}_i) \,:\, \mathbf{x}_i \in \mathsf{X}_{\text{ens}}\right\}\right) \in \operatorname{span}\left(\{\operatorname{vec}(\mathsf{e}_j) \,:\, \mathsf{e}_j \in \mathsf{E}\}\right) \subseteq \mathbb{R}^{2n_{\mathsf{X}}}. \qquad \text{(3.12 revisited)}$$

This ensures that the flow field estimates are constructed from spatially correlated flow fields found in the ensemble. However the true flow field is unlikely to be an exact linear combination of the basis flow fields. The discrepancy emerges as a type of estimation error referred to as the *out-of-span* error.

The decoupling of the representation into layers allows us to construct a basis with the 2D flow fields from each depth and from each ensemble member. The resulting basis is enriched by increasing its expressivity in a way to include spatially correlated flow that are more likely to occur, as opposed to adding random basis vectors. This increased expressivity reduces the effects of the out-of-span error that occurs.

For this approach, we choose a slightly different formulation of the basis compared to (3.15). The basis for the 2.5D flow field is

$$\boldsymbol{H}(\mathbf{x}) = \boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\boldsymbol{\mathcal{U}}. \tag{3.21}$$

We also assume that the positions X_{ens} have a particular spatial structure such that all the positions can be described as all combinations of a set of horizontal 2D positions $\mathsf{X}_{\text{H}} \subset \mathbb{R}^2$ and a set of vertical displacements from the origin $\mathsf{X}_{\text{V}} \subset \mathbb{R}$. I.e. the total number of positions in the ensemble is

$$n_{\mathsf{X}} = n_{\text{H}} n_{\text{V}}, \tag{3.22}$$

where n_{H} and n_{V} are the number of elements in X_{H} and X_{V}. This is not an overly restrictive assumption since the ensemble positions are usually provided as grid points.

We will next describe how to construct $\boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})$ and $\boldsymbol{\mathcal{U}}$. In Sec. 3.5.1, we focus on capturing the vertical influence one layer has on others. In Sec. 3.5.2, we focus on the representation of 2.5D information from 2D patterns.

3.5.1 Incompressible kernel embedding with vertical influence

Recall from Sec. 3.4.1, the incompressible kernel [38]

$$\boldsymbol{K}_{\text{incomp}}(\mathbf{x}, \mathbf{x}') = \mathscr{D}(\mathbf{x}) k(\mathbf{x}, \mathbf{x}') \mathscr{D}\left(\mathbf{x}'\right)^{\mathsf{T}}, \tag{3.7 revisited}$$

which we used to describe 2D flow in 2D space. This kernel uses an inner kernel $k(\mathbf{x}, \mathbf{x}')$ to describe the similarity between the stream function values between two points. The differential operator $\mathscr{D}$ projects this to describe the similarity between the 2D flow velocity components of the two points.

We can use the inner kernel to describe the similarity of the 2D stream functions across different depths as well. This reflects the vertical influence each layer has on others through the dragging effect caused by shear forces. To achieve this, we can use the SE-ARD kernel (2.4) extended to 3D, carrying out the differential operator only on the horizontal components of space. This allows us to describe different lengthscales for the horizontal and vertical components. The resultant kernel $\boldsymbol{K}_{\text{incomp}} : \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^{2\times 2}$ would also allow us to express the flow velocity between the depths, giving interpolation-like functionality as before.

Using the corresponding incompressible kernel matrix $\boldsymbol{\mathcal{K}}_{\text{incomp}}$, any flow field represented in the form of

$$\tilde{\mathbf{f}}(\mathbf{x}) = \boldsymbol{\mathcal{K}}_{\text{incomp}}(\{\mathbf{x}\}, \mathsf{X}_{\text{ens}})\beta, \tag{3.10 revisited}$$

is still incompressible as the sum of flow field velocities are zero: horizontal components sum to zero (property of stream functions), and vertical components sum to zero (by construction). Furthermore, the embedding of the flow field as β captures the vertical influence that occurs between layers. We then find the set of latent state vectors B through regression with the ensemble as in (3.11).

3.5.2 Vertically decoupled model compression

The major contribution of this section is to exploit the fact that 2.5D flow fields can be seen as a collection of 2D horizontal flow fields that vary across depth, and that more modes can be found by comparing the flow fields between depths.

Recall from Sec. 3.4.2 that the SVD was used to decompose the latent state representation of each of the ensemble members into two parts: the modes $\boldsymbol{U}$ and the weight matrix $\boldsymbol{W}$ to reconstruct the latent state variables for each of the ensembles from the modes.

To decouple the patterns that are found through the SVD, we first re-interpret the values of β_i into smaller vectors of latent variables for each depth. We then form a matrix that concatenates the smaller vectors from all depths and all ensemble members and we put it through the SVD for 2D patterns derived from each depth of each ensemble member. This results in up to a multiple of n_V times the number of basis flow fields compared to using the direct extension of Sec. 3.4 on the 2.5D flow fields in the ensemble directly, increasing the expressiveness of $\boldsymbol{H}(\mathbf{x})$.

More concisely, we form a matrix

$$\boldsymbol{A}_i = \begin{bmatrix} \beta_{i,1,1} & \cdots & \beta_{1,1,n_\mathrm{V}} \\ \vdots & \ddots & \vdots \\ \beta_{i,n_\mathrm{H},1} & \cdots & \beta_{1,n_\mathrm{H},n_\mathrm{V}} \end{bmatrix} \in \mathbb{R}^{2n_\mathrm{H} \times n_\mathrm{V}}, \tag{3.23}$$

where $\beta_{i,j,k}$ is a 2×1 vector containing the elements of β_i that correspond to the j^{th} horizontal position from X_H, and the k^{th} horizontal position from X_V. Then, let the matrix $\mathscr{A} \in \mathbb{R}^{2n_\mathrm{H} \times n_\mathrm{V} n_\mathsf{E}}$ be:

$$\mathscr{A} = \begin{bmatrix} \boldsymbol{A}_1 & \boldsymbol{A}_2 & \cdots & \boldsymbol{A}_{n_\mathsf{E}} \end{bmatrix}. \tag{3.24}$$

The thin SVD of $\mathscr{A}$ is

$$\mathscr{A} = \boldsymbol{U}\boldsymbol{\Sigma}\boldsymbol{V}^H, \tag{3.25}$$

with $n_\Sigma = \min(2n_\mathrm{H}, n_\mathrm{V} n_\mathsf{E})$ singular values. Note that $\boldsymbol{U}$ has $\min(2n_\mathrm{H}, n_\mathrm{V} n_\mathsf{E})$ columns instead of $\min(2n_\mathrm{H} n_\mathrm{V}, n_\mathsf{E})$. Since the computation of ensemble forecasts is computationally

expensive and computed over large grids, resulting in small n_{E} and large n_{X}, a typical outcome of the vertical-decoupling of $\boldsymbol{B}$ for the SVD is that it increases the number of columns in $\boldsymbol{U}$ which increases the expressivity of the basis.

As in our previous use of the SVD, we can truncate the columns of $\boldsymbol{U}$ associated with lower singular values. We will use overhead tildes ($\tilde{}$) to denote the rank $n_{\mathrm{r}} \leq n_{\Sigma}$ truncations of $\boldsymbol{U}$, $\boldsymbol{\Sigma}$, and $\boldsymbol{V}$, i.e. $\tilde{\boldsymbol{U}} \in \mathbb{R}^{2n_{\mathrm{H}} \times n_{\mathrm{r}}}$.

A linear combination of the left singular vectors in $\tilde{\boldsymbol{U}}$ only describe a flow patterns in 2D for a particular depth. To describe a 2.5D flow field, the matrix needs to be repeated for each depth. To achieve this, we can use a particular ordering of $\mathrm{vec}\,(\cdot)$ such that the resulting vector consists of n_{V} sections corresponding to different depths, and that the positions in different sections are represented in the same order. Then we can form the block matrix $\boldsymbol{\mathcal{U}} \in \mathbb{R}^{2n_{\mathsf{X}} \times n_{\mathrm{r}} n_{\mathrm{V}}}$ in (3.21), created with n_{V} copies of matrix $\tilde{\boldsymbol{U}}$ down the diagonal, or more compactly

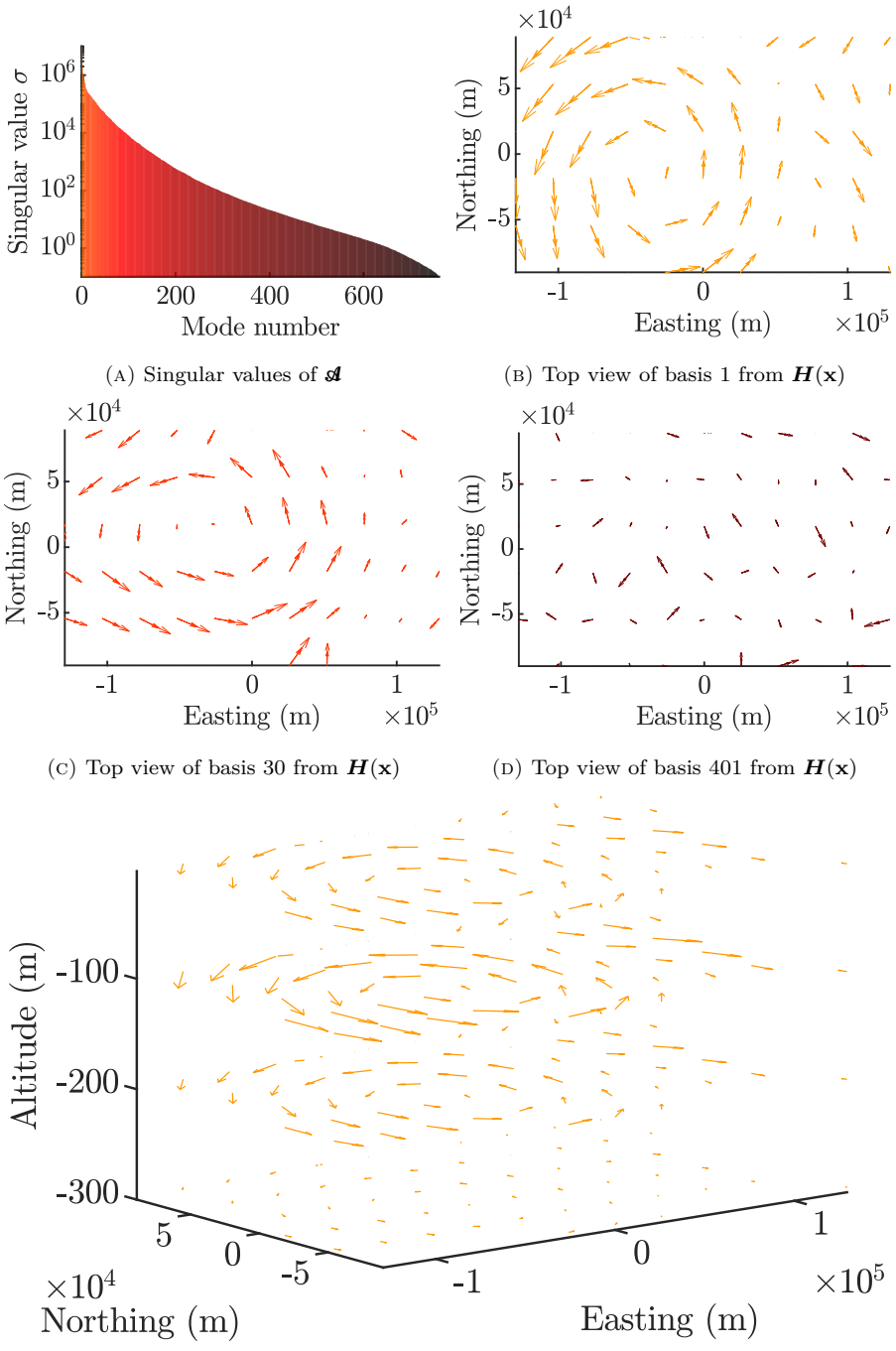
$$\boldsymbol{\mathcal{U}} = \boldsymbol{I}_{n_{\mathrm{V}}} \otimes \tilde{\boldsymbol{U}}, \tag{3.26}$$

where $\otimes$ is the Kronecker product. This formulation of $\boldsymbol{\mathcal{U}}$ allows the 2D flow fields at each depth to have its own set of weights, which may be distinct from other depths. This results in an increased expressivity by a factor of n_{V} compared to the naïve 2.5D extension of our approach in Sec. 3.4 which would only have n_{r} basis flow fields.

The weight vector $\mathbf{w}$ now uses $n_{\mathrm{W}} = n_{\mathrm{r}} n_{\mathrm{V}}$ variables to describe the magnitude of each of the 2.5D basis flow fields in the columns of $\boldsymbol{H}(\mathbf{x})$. Each column can be seen as a 2.5D basis flow field induced by some horizontal 2D flow field at the depth corresponding to the $\mathrm{vec}\,(\cdot)$ section of the column. Figure 3.9e shows a visualisation of the first column of $\boldsymbol{H}(\mathbf{x})$, i.e. $\tilde{\mathbf{f}}(\mathbf{x})$ when $\mathbf{w} = [1, 0, \ldots, 0]^{\intercal}$, showing the influence of a single basis flow field. Note that the flow field is strongest at $-100\,\mathrm{m}$ and decays in strength at depths further from this horizontal plane. The rest of Fig. 3.9 shows visualisations of several examples of the basis flow fields described by $\boldsymbol{H}(\mathbf{x})$, and Fig. 3.9a shows the relative contribution of each of these modes to the ensemble.

For this formulation, the weight matrix is obtained through an intermediate term $\tilde{\boldsymbol{W}}' \in \mathbb{R}^{n_{\mathrm{r}} \times n_{\mathrm{V}} n_{\mathsf{E}}}$

$$\tilde{\boldsymbol{W}}' = \tilde{\boldsymbol{\Sigma}} \tilde{\boldsymbol{V}}^{H}. \tag{3.27}$$



(A) Singular values of $\mathscr{A}$

(B) Top view of basis 1 from $\boldsymbol{H}(\mathbf{x})$

(C) Top view of basis 30 from $\boldsymbol{H}(\mathbf{x})$

(D) Top view of basis 401 from $\boldsymbol{H}(\mathbf{x})$

(E) 3D view of basis 1 from $\boldsymbol{H}(\mathbf{x})$

FIGURE 3.9: Example SVD of $\mathscr{A}$. (a) shows the singular values corresponding to the basis flow fields of $\boldsymbol{H}(\mathbf{x})$. Notice there are $n_\Sigma = 760$ modes, which is many more than $n_\mathsf{E} = 95$ (the number that would have been produced by a direct extension of the approach in Sec. 3.4. (b-e) shows some examples of basis flow fields, corresponding to singular values of 1.0×10^7, 1.16×10^5, and 2.09×10^1.

This intermediate matrix is actually the weight matrix for an ensemble created from the aggregated 2D flow fields from all the original ensemble members. Since we reshaped from 2.5D to 2D for the SVD, we can reshape the 2D weight matrix into the 2.5D weight matrix $\tilde{\boldsymbol{W}} \in \mathbb{R}^{n_\mathrm{r} n_\mathrm{V} \times n_\mathsf{E}}$. The reshape operation is outlined below:

$$\tilde{\boldsymbol{W}}' \Rightarrow \tilde{\boldsymbol{W}} \tag{3.28}$$

$$\begin{bmatrix} w_{1,1,1} & \cdots & w_{1,n_\mathrm{V},1} & \cdots & \cdots & w_{1,n_\mathrm{V},n_\mathsf{E}} \\ \vdots & \ddots & \ddots & \ddots & \ddots & \vdots \\ w_{n_\mathrm{r},1,1} & \cdots & w_{n_\mathrm{r},n_\mathrm{V},1} & \cdots & \cdots & w_{n_\mathrm{r},n_\mathrm{V},n_\mathsf{E}} \end{bmatrix} \Rightarrow \begin{bmatrix} w_{1,1,1} & \cdots & w_{1,1,n_\mathsf{E}} \\ \vdots & \ddots & \vdots \\ w_{n_\mathrm{r},1,1} & \ddots & w_{n_\mathrm{r},1,n_\mathsf{E}} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \\ w_{n_\mathrm{r},n_\mathrm{V},1} & \cdots & w_{n_\mathrm{r},n_\mathrm{V},n_\mathsf{E}} \end{bmatrix} \tag{3.29}$$

The order of this operation should not lead to the re-ordering of data in memory for software that uses a column-major representation of matrices.

3.5.3 Kalman filtering with a 2.5D flow field representation

We now address the remaining part of (3.6), estimating $\mathbf{w}^*$ corresponding to $\tilde{\mathbf{f}}^*$, based on online measurement data. Once again, the Kalman filter is not strictly necessary; however, it will be an invaluable tool when extending to the time-variant case.

Recall that the well-known Kalman filter recursively estimates the maximum likelihood state $\mathbf{w}$ of a linear dynamical system, given a sequence of measurements $\mathbf{f}_\mathrm{mea}(\mathbf{x}_k) \in \mathsf{Z}$ experiencing Gaussian sensor noise. In the case of time-invariant flow fields, the process model is simply the identity of size $n_\mathrm{W} = n_\mathrm{r} n_\mathrm{V}$, so the predict step is:

$$\mathbf{w}_{k+1} = \boldsymbol{I}_{n_\mathrm{W}} \mathbf{w}_k, \tag{3.30}$$

$$\mathbf{f}_\mathrm{mea}(\mathbf{x}_k) = \boldsymbol{H}(\mathbf{x}_k)\mathbf{w}_k + \tilde{\nu}. \tag{3.31}$$

The state and covariance matrix of the Kalman filter is initialised with the mean and variance of the weight matrix across the ensemble members. We see here that the reshaping

of the weight matrix into $\tilde{\boldsymbol{W}}$ means that we can simply take the mean and variance horizontally on the matrix obtained after reshaping.

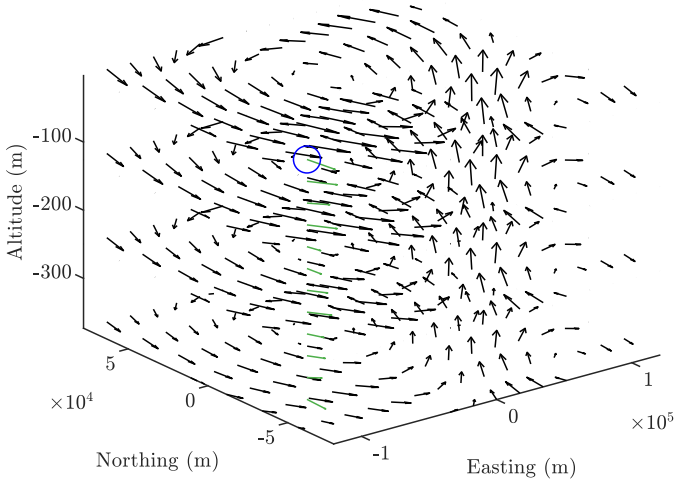
3.5.4 Simulation results and discussion

In this section, we compare the proposed algorithm with some alternate estimation methods by estimating an example flow field, and emphasise the significance of flow field estimations in the application to underwater glider path-planning.

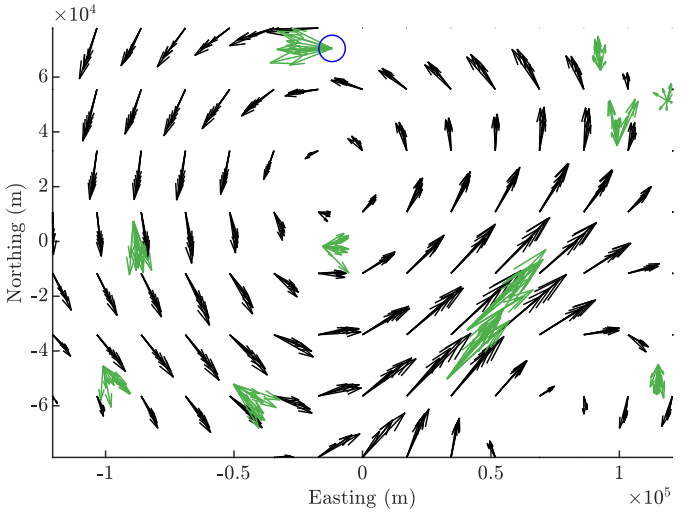
We use ensemble data from the BOM, the top horizontal layer of which was shown in Fig. 3.1 and was used in Sec. 3.4.6. Since the flow field data covers a vast volume of ocean; we consider only a small region from 39.2° to 40.8° S, 151° to 154° E, and from the depths of 2.5 m to 685 m. To reduce computation time, this grid was then downsampled by a factor of 5 in the depth direction, resulting in a grid of size $31 \times 17 \times 8$, with grid points ∼10 km apart horizontally, and ∼85 m apart vertically.

Since we do not have access to ensemble data and high-resolution sensor measurements for the same time and space, we simulate a measurement sensor that provides data based on a synthesised true flow field based on one member from the ensemble that is then removed from the data provided to the algorithms for initialisation. The ensemble member is chosen based on what we believe to be the most distinct visually compared to the other members. In our implementation, the true flow field $\mathbf{f}_{\text{tru}}(\mathbf{x})$ is evaluated using cubic interpolation of the discrete data in the chosen ensemble member. The remaining 95 ensemble members form E. Note that $\mathbf{f}_{\text{tru}}(\mathbf{x})$ is not physically realistic as it does not guarantee an incompressible flow field.

The simulated measurement sensor is based on an ADCPs, which are sensors that measure flow velocities along acoustic beams. We use a hypothetical ADCP that measures the flow velocities in a single water column directly underneath a surface vehicle at regular depth intervals with a vertical resolution of 32 m and a precision standard deviation of 9 cm/s matching specifications of the ADCP Pinnacle 45 [152] from Teledyne RD Instruments. It is worth noting that 9 cm/s standard deviation is quite large compared to the average



(A) 3D view of the environment with one acoustic Doppler current profiler (ADCP) ping



(B) Top view of the environment with multiple ADCP pings

FIGURE 3.10: Visualisation of measurements in the true flow field in black, downsampled by a factor of 2. A single ADCP "ping" results in 22 measurements (in green), only 12 of 450 are shown in (b) for readability. The location of a simulated ADCP-equipped surface vessel is shown in blue. Note the large noise on measurements.

speed of the current in the dataset, which was 22.8 cm/s (see the spread of the green arrows in Fig. 3.10).

The kernels used in the different approaches are all tuned the same way. The lengthscale hyperparameters were tuned by hand, with $l_1 = l_2 = 10^4$ m, $l_3 = 100$ m. The hyperparameter σ_{ker} is heuristically picked to be $\sigma_{\text{ker}} = \mu/l_1 = 1718.9\,\text{m}^2/\text{s}$, where μ is the average flow vector magnitude at the surface, where the flow is usually fastest.

The proposed method, denoted as the "layered" approach, is compared with the direct extension of the approach in Sec. 3.4 and a baseline approach. The direct extension uses the SE-ARD (2.4) extended to 3D to form the columns of the latent matrix $\boldsymbol{B}$. The SVD is then performed on $\boldsymbol{B}$ instead of $\boldsymbol{A}$ for a maximum of $n_{\mathsf{E}} = 95$ modes describing 2.5D patterns. It is denoted as the "block" approach referring to the spatial coverage of each derived mode. Finally, the baseline approach involves choosing the ensemble member such that the least squares error is minimised between cubic interpolation of the vectorised data at the measurement locations and the vectorised noisy measurements. It is denoted as the "nearest" approach for choosing the nearest ensemble member using the 2-norm.

3.5.4.1 Flow field estimation error comparison

Two simulation experiments were performed to highlight particular differences between the methods. The experiments are repeated for our method using different levels of truncation. The first experiment shows the potential performance of the proposed by using favourable conditions, i.e. measurements have no measurement noise and are taken exactly at the ensemble positions X_{ens} ignoring ADCP measurement patterns. As there is no measurement noise, the Kalman filter noise covariance was set to some low values: $\widetilde{\text{cov}}(\mathbf{f}_{\text{mea}}) = 0.01^2\boldsymbol{I}_2$. The second experiment involve ADCP measurements with Gaussian noise of standard deviation 9 cm/s taken from ∼450 randomly uniformly distributed surface locations for a total of 10^4 flow velocity measurements. The Kalman filter noise covariance is estimated as $\widetilde{\text{cov}}(\mathbf{f}_{\text{mea}}) = 0.12^2\boldsymbol{I}_2$ to account for both the sensor noise model and the additional noise incurred from modelling error.

Table 3.2 and 3.3 show the results for both experiments. The primary performance metric is the RMS error between the estimated and true flow fields evaluated at the ensemble positions X_{ens} regardless of where the measurements were taken. For the block- and layered-based approaches, a lower bound of the RMS error can be computed by performing a best fit of the flow velocities at X_{ens} for the best weight vector given their constructed basis $\boldsymbol{H}$, i.e.

$$\mathbf{w}^* = \underset{\mathbf{w}\in\mathbb{R}^{n_{\mathsf{W}}}}{\arg\min} \sum_{\mathbf{x}\in\mathsf{X}_{\text{ens}}} \|\mathbf{f}_{\text{tru}}(\mathbf{x}) - \boldsymbol{H}(\mathbf{x})\mathbf{w}\|_2^2 \,. \tag{3.32}$$

TABLE 3.2: Error performance of different truncations of our method (layered basis) compared to other methods in ideal conditions

	No. modes	Rel. error (%)	RMS error (cm/s)	RMS error of bound (cm/s)
Block basis	95	8.30	2.33	2.33
Layered basis	760	1.42	0.39	0.077
	400	1.78	0.50	0.34
	300	2.46	0.69	0.57
	200	4.18	1.17	1.14
	95	11.00	3.08	3.07
Nearest e_i ($i = 82$)	—	45.6	12.80	—

TABLE 3.3: Error performance of different truncations of our method (layered basis) compared to other methods with measurement noise and measurement location randomisation

	No. modes	Rel. error (%)	RMS error (cm/s)	RMS error of bound (cm/s)
Block basis	95	11.11	3.11	2.33
Layered basis	760	9.81	2.75	0.077
	400	9.91	2.78	0.34
	300	10.64	2.98	0.57
	200	11.96	3.35	1.14
	95	17.15	4.81	3.07
Nearest e_i ($i = 82$)	—	45.6	12.80	—

Note that $\mathbf{f}_{\text{tru}}(\mathbf{x})$ at X_{ens} is unknown in real applications. The RMS error of the flow velocities estimated by $\mathbf{w}^*$ serves as a measure of the error due to the inadequate expressivity of the basis. I.e. the RMS error quantifies the degree at which

$$\text{vec}\left(\{\mathbf{f}_{\text{tru}}(\mathbf{x}_i) \,:\, \mathbf{x}_i \in \mathsf{X}_{\text{ens}}\}\right) \notin \text{span}\left(\{\text{vec}\left(\{\mathbf{h}_j(\mathbf{x}_i) \,:\, \mathbf{x}_i \in \mathsf{X}_{\text{ens}}\}\right) \,:\, j \in [1, n_{\text{W}}]\}\right), \quad (3.33)$$

where $\mathbf{h}_j$ is the j^{th} column of $\boldsymbol{H}$.

It can be seen from Tab. 3.2 that the proposed method have consistently lower error bounds than the naïve method. This is evidence that the span from the proposed method's basis is higher than the span from the direct extension or the span from the ensemble in (3.12), which reduces the out-of-span modelling error.

Other than having a lower error bound, the proposed method also has achieved a lower RMS error when 200 or more modes were used. This illustrates that an improved estimation performance can be achieved in the proposed method by increasing the expressiveness of $\boldsymbol{H}(\mathbf{x})$.

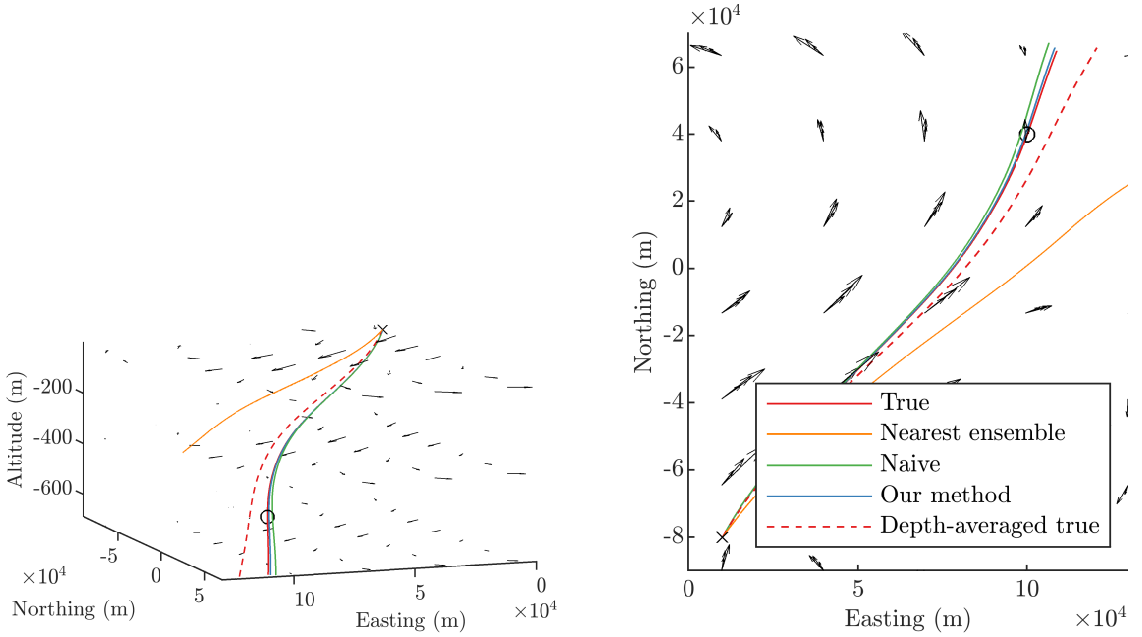
Figure 3.9a clarifies the reason why so many modes can be discarded during truncation while still retaining good representational ability of $\boldsymbol{H}(\mathbf{x})$. First, note that the vertical axis is logarithmically scaled. It can be seen that the last few hundred modes have a much smaller contribution compared to the first few. For example, the basis flow field pattern shown in Fig. 3.9d doesn't appear to be particularly necessary for the reconstruction of a flow field because it appears to be mostly noise, and this is reinforced numerically as its contribution in Fig. 3.9a is approximately 10^{-4} the dominant mode. Hence the 401^{st} mode can be omitted without much loss in representational power.

However, there does come a point where too many modes are truncated. We see in both Tab. 3.2 and 3.3 that the estimation performance deteriorates as the number of modes drops to 200 or below. This could be a result of truncating more modes than we could afford, reducing expressivity of the estimator. Still, both tables show that a significant number of modes can be discarded while suffering only a small loss in estimation performance which is reinforced by the results from using 760 and 400 modes.

3.5.4.2 Significance of flow field estimation for trajectory prediction

Next we highlight the importance of accurate flow field estimation for path planning applications. Paths generated using poor state estimation is effectively useless since the true system will not be able to follow the path with the corresponding controls. Here we show additional simulations of an underwater glider following a fixed velocity generated different flow field estimation methods.

The proposed technique are compared with three other different flow field estimation methods: cubic interpolation of the nearest ensemble to the true flow field (minimum 2-norm), the block-based approach, and the depth-averaged true flow field. The estimation of the basis-based methods was generated using measurements with noise without truncation.



(A) Overview (B) Top view

FIGURE 3.11: Simulated glider trajectories based on controls computed from different flow field estimates

The simulation environment is located roughly in the right half of the region considered in the previous simulations. For this demonstration, we consider an underwater glider diving with a fixed speed of 0.3 m/s released from the surface in aiming to pass through a point at a depth of 500 m roughly 150 km away. Due to these large distances, we make the holonomic assumption, i.e. the vehicle is assumed that it can change instantaneously change velocity as the amount of space it needs to reach the steady-state dynamics with the desired velocity is comparatively small. The fixed velocity is chosen from different flow estimation methods by minimising the distance between the intended target and the associated trajectory generated through forward Euler integration of 100 m steps. Resulting trajectories from this control selection process is generated by executing the chosen controls on the true flow field using the same forward integration scheme.

The simulated underwater glider trajectories after applying the fixed velocities from the different flow field estimates in the true flow field are shown in Fig. 3.11 with their closest approach shown in Tab. 3.4. It is clear that planning with the estimate of the proposed method is the most ideal, followed by the block basis method. This simulation shows that

TABLE 3.4: The closest approach for simulated glider trajectories based on controls computed from different flow field estimates

	Closest approach (m)
True	16
Nearest e_i $(i = 82)$	29858
Depth-averaged true	6192
Block basis	1808
Layered basis	505

a small difference of about 2% in relative error between the basis-based methods shown in Tab. 3.3 can result in a significant difference in error of 1303 m.

Figure 3.11 also reinforces the idea that the existing methods depth-averaging the flow field or choosing the nearest ensemble can underperform. This statement specifically applies to the practice of depth-averaging the flow field, as this simulation takes the depth-average of the true flow field, and not an estimated one. We also see that using the nearest ensemble member in this demonstration performs worst of all despite proving a 2.5D flow field, suggesting that for the purpose glider path-planning, using the nearest ensemble is insufficient. Both these approaches perform significantly worse than the estimates using a layer-based or block-based basis which make them unsuitable for predicting trajectories required in the path planning process.

3.6 Summary

In this chapter we considered problem of using flow velocity measurements to estimate a flow field. We proposed anytime flow field estimation algorithms for 2D and 2.5D flow fields that has fast query computational times. It takes advantage of the heavy computation done prior to execution that produces multiple flow field estimates as an ensemble. The data is encoded into a different representation through the incompressible kernel [38], giving the described flow field the property of always being incompressible. Patterns in this representation are extracted by the SVD, effectively extracting flow field patterns that are used to reconstruct a flow field estimate. Finally a Kalman filter is used to determine the contributions of each flow field pattern based on the measurements obtained online. The 2.5D variant of our algorithm leverages the stratified nature of the ocean, increasing

the expressivity of its model by a factor of the number of horizontal layers present in the ensemble data. The proposed approaches are able to make adjustments in the estimate that are distant to the measurement locations using the spatial-correlations that are present in the data.

Our approach is shown to be computationally tractable for navigational purposes as its computation time remains constant for measurement updates and flow velocity queries during online execution. It is also shown to strike a good balance between computational time and estimation quality, being slightly slower than the KO algorithm for queries, but also having faster estimation error convergence. The increased number of modes in the 2.5D variant is shown to be more suitable to combat the significantly increased variety of flow field patterns that can occur with an additional spatial dimension than a direct extension of the 2D variant.

Chapter 4

Streamline-based trajectories for point-to-point traversal

In this chapter, a novel computationally efficient `Steer` primitive function is proposed to determine the system control necessary to drive an underwater glider between two points whilst accounting for the displacing effect of ocean currents. Streamline-based control theory is developed to formulate constraints that reduce the search space of the problem by one dimension. We compare against the unconstrained implementation of the same function in a simple sampling-based path planner demonstrating higher-quality paths obtained with similar computation times. We argue that the trade-off between computer resource allocation and path quality implies that a particular path solution quality can be obtained using less computation time. We also explore other consequences of streamline-based control theory that is useful in practice.

Parts of this chapter have been published as two separate conference paper publications [153, 154].

4.1 Introduction

Path planning is an aspect of navigation to determine a sequence of positions to follow that allows a destination to be reached. Optimal path planning is a variant in which a path is

computed that can minimise some cost such as time, distance, or energy. This is critical for large scale industries that use ships like global logistics, and for underwater operations such as environmental monitoring [80], oil and gas exploration [155], and defence [156].

Many path planning algorithms take advantage of the ability to update controls continuously to achieve more optimal paths, however in this thesis, we are particularly interested in waypoint-based path planning for underwater gliders, where control adjustments are only made at the waypoints along the path. Despite being sub-optimal compared to paths that use continuously adjusting control, waypoint-based paths are logistically, and practically easier to use. The simple representation is easier for humans to convey, explain, execute, supervise during execution, and adjust when necessary. Furthermore, underwater gliders would benefit from major control adjustments to improve mission longevity. For these applications we define *persistent controls* to be controls that are held constant over some time. Graph-based path planning algorithms are particularly suited for waypoint-based path planning, many of which are already well-established and commonly used.

Some graph-based path planning algorithms require what is known as a steering function, however a vehicle's motion is heavily influenced by prevailing ocean currents during the execution of persistent controls making it difficult to determine the correct persistent control to another position. Vehicles executing a persistent control can take complex and curved trajectories over long control durations due to large inter-waypoint distances, or low vehicle forward propulsion e.g. from underwater gliders. This type of disturbance must be taken into account during the planning process. To illustrate these effects, a simple 2D underwater glider example is shown from the side in Fig. 4.1 in which a glider (starting at the diamond) is tasked with diving in a strong flow field to reach its goal (circle). The trajectories using persistent controls differing only in control are shown as pink lines. The trajectory that correctly reaches the target is shown in green with arrows indicating the direction of control. If a persistent control is chosen without considering the flow field, then the vehicle will take the red path and ultimately fall short of the target.

A way to account for ocean advection is to simulate the resulting trajectories of multiple persistent controls to find the one that best steers the vehicle to its target. However, this is a computationally intensive process because each persistent control considered in

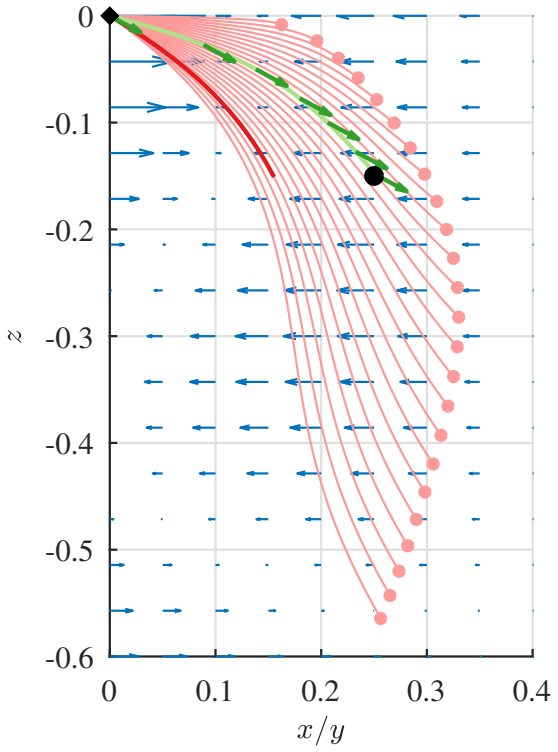


FIGURE 4.1: Side view of underwater glider trajectories in a flow field (blue arrows). Trajectories of persistent controls with different pitch but same duration are shown in pink. The trajectory using a control that reaches the target (circle) is shown in green. The trajectory of a control chosen without taking into account the currents is shown in red.

a numerical search over the different degrees of freedom requires forward integration to determine reachability.

In this chapter, we can leverage the concept of *streamlines* from fluid dynamics, which is a powerful tool for analysis since it describes the path of particles in time-invariant flow fields. Our contribution comes from the realisation that the environment's streamlines can be augmented with the effects from the vehicle's control allowing us to consider warped streamlines that connects the vehicle's position and a target position for feasible controls. From this, we derive a constraint to the search for feasible controls, reducing the dimensionality of the problem without excluding solutions that are feasible. We also propose an extension of this idea to underwater gliders with buoyancy-based propulsion in 3D environments by leveraging the idea that ocean flow far from the coastline has negligible vertical velocity [142–144].

We demonstrate how these constraints can be used in the context of time-optimal path planning in simulated environments, using synthetic flow fields and ocean current predictions from the Australian Bureau of Meteorology (BOM). It is shown through a sampling-based approach that the constraint allows the search algorithm to focus on a significantly smaller subset of the search space, effectively reducing the required computational time and improving the solution quality demonstrated by paths that tend to take advantage of more favourable currents leading to shorter transit durations.

A brief introduction of relevant concepts is provided in the next section to form a foundation for the ideas presented in this chapter. The core problem of this chapter is brought into perspective in Sec. 4.3. Our main contribution to the problem for marine surface vehicles is presented in Sec. 4.4 and the extended idea for underwater gliders is presented in Sec. 4.5. Finally, a summary for the chapter is provided in Sec. 4.6.

4.2 Background

In this section, we establish the 2D and 3D vehicle models that will be used in this chapter and provide an introduction to *stream functions* which is the mathematical representation of streamlines. We will also highlight properties that are crucial to the development of the main contributions in this chapter.

4.2.1 Vehicle model

The position of the vehicle $\mathbf{x}$ is considered to be its state and lies within a bounded space $\mathbb{X}$, i.e. $\mathbf{x} \in \mathbb{X} \subset \mathbb{R}^2$. The vehicle motion is subject to advection by a time-invariant flow field $\mathbf{f} : \mathbb{X} \to \mathbb{R}^2$ and is modelled by the continuous-time transition model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{v}, \tag{4.1}$$

where $\mathbf{v}$ is the vehicle's controlled velocity *relative* to the flow which is constrained by vehicle actuation limits to a control velocity space or simply *control space* $\mathbb{U}$, i.e. $\mathbf{v} \in \mathbb{U} \subset \mathbb{R}^2$. We use an analogous model for the 3D variant of the problem.

A persistent control is represented as a velocity-duration pair $\mathsf{a} = (\mathbf{v}, \tau) \in \mathbb{A}$. Let $\mathbf{f}_{\mathrm{x}}(\mathbf{x}, \mathsf{a})$ be the resultant state of the system after applying a persistent control a on the state $\mathbf{x}$ under dynamics (4.1), and $f_{\mathrm{c}}(\mathbf{x}, \mathsf{a})$ be the corresponding cost.

Modelling the vehicle as a holonomic system is common in literature [37, 49–52, 56, 57, 64, 75, 141, 157–162]. In this chapter, we justify the modelling of the vehicle as holonomic systems since vehicles like ships and underwater gliders take relatively little space and time to perform re-orienting manoeuvres compared to the scope of the problem. The chosen controls are maintained either by an underlying controller for ships, or by steady-state dynamics that arise from the passively stable design of underwater gliders.

In 2D settings, we model the dynamics of the vehicle as a system that is only constrained by a maximum speed $V_{\max}$ through water, i.e.

$$\mathbf{v} \in \mathbb{U} = \left\{ \mathbf{v} \,:\, \|\mathbf{v}\|_2 \leq V_{\max}, \mathbf{v} \in \mathbb{R}^2 \right\}. \tag{4.2}$$

4.2.1.1 Trim-based control of an underwater glider

For underwater gliders, we constrain their controls to the 3D steady-state kinematics resulting from possible actuator setpoints. The steady-state behaviour that arises from the passively stable design of the vehicle is determined by actuator setpoints after disturbances or control variations and are referred to as *trim state*.

Since the vehicle is in equilibrium when in a trim state, it also has a constant velocity. To determine the possible steady-state velocities, we then adopt the underwater glider model from [8]. In this model, the vehicle speed can be described as a function of the glide angle, the upwards-positive angle between the vehicle's velocity and the horizontal plane. In a simplified model, the vehicle's speed

$$V_{\mathrm{g}}(\beta) = \sqrt{\frac{m(\beta) \cdot g}{D(\beta) \cdot \sin\beta - L(\beta) \cdot \cos\beta}}, \tag{4.3}$$

which includes other functions of the glide angle related to the lift and the drag forces of the underwater glider. It also includes the function $m(\cdot)$ which relates to the mass within the vehicle hull, and takes two possible values due to ballasting depending on whether the

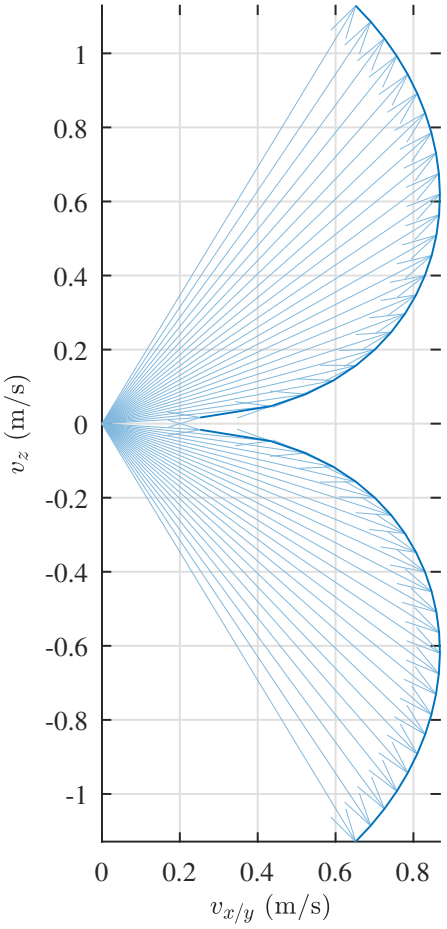
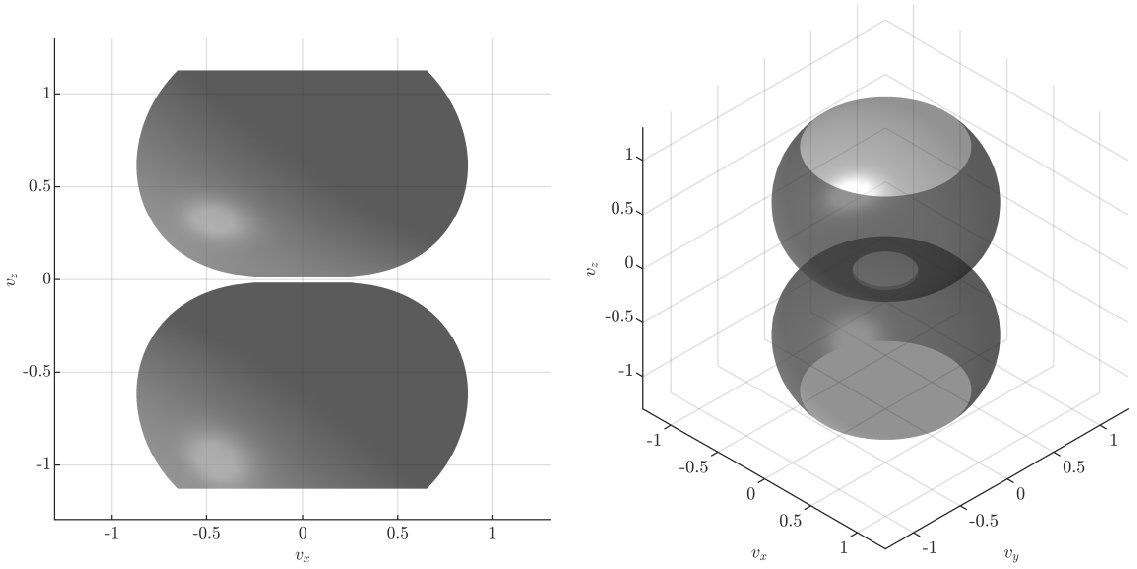


FIGURE 4.2: Feasible forward velocities using the trim state model for underwater gliders. The velocities are disjoint due to the omission of those that cause system instability.

desired glide angle is positive. These functions are fully defined in Appendix A. In the model, the magnitude of the glide angle is bounded for system stability, i.e.

$$\beta \in [-\beta_{\mathrm{U}}, -\beta_{\mathrm{L}}] \cup [\beta_{\mathrm{L}}, \beta_{\mathrm{U}}] \tag{4.4}$$

where $\beta_{\mathrm{L}} = \frac{8\pi}{180}$ and $\beta_{\mathrm{U}} = \frac{60\pi}{180}$. The forward speed profile with a maximum horizontal speed of 0.867 m/s is plotted in Fig. 4.2.



(A) Front view (B) Isometric view

FIGURE 4.3: The control velocity space based on an underwater glider steady-state model represented by disjointed, partially transparent grey surfaces in $\mathbb{R}^3$.

The full 3D model of the underwater glider's steady-state velocity can be concisely represented as a function of glide angle β and yaw γ defined as

$$\mathbf{v} = \begin{bmatrix} V_{\mathrm{g}}(\beta)\cos\beta\cos\gamma \\ V_{\mathrm{g}}(\beta)\cos\beta\sin\gamma \\ V_{\mathrm{g}}(\beta)\sin\beta \end{bmatrix}, \tag{4.5}$$

which assumes that the underwater glider is upright during transit. The set of all executable velocities, or the control space $\mathbb{U}$, for the underwater glider is shown in Fig. 4.3 as two unconnected grey surfaces due to disjoint pitch values in (4.4). The upper surface corresponds to ascending velocities as a result from an empty ballast tank, whilst the lower surface corresponds to descending velocities from a full ballast tank. Note that steady-state velocities inside or outside of the surfaces are not attainable based on the underwater glider's kinematic model. An interesting consequence of this model is that the vehicle must switch between ascending and descending manoeuvres to produce effective horizontal motion and to stay within the vertical span of the workspace $\mathbb{X}$.

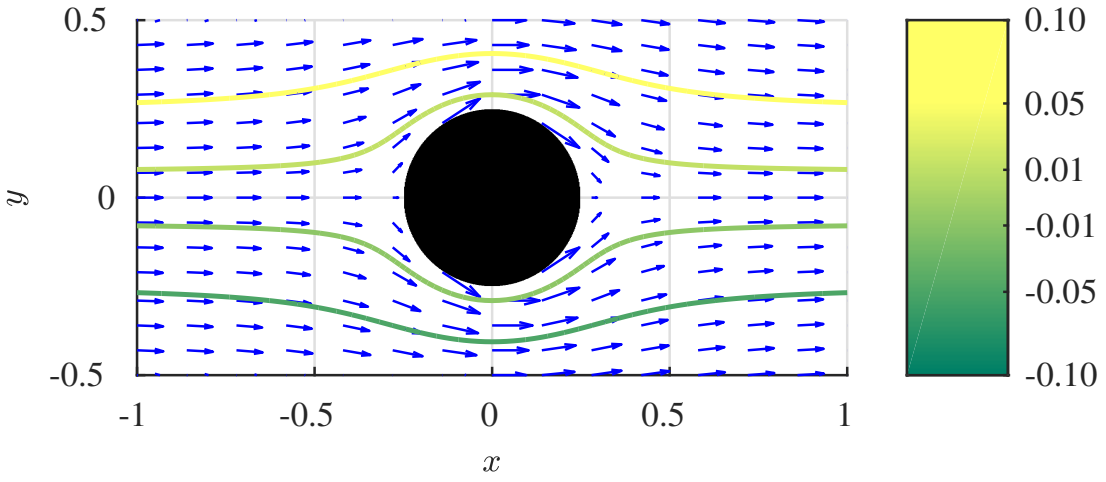


FIGURE 4.4: Incompressible flow visualised by streamlines with different flow flux relative to the reference point $[-1, 0]^\intercal$

4.2.2 Stream functions

An approximation that is used in fluid analysis is the *incompressibility* of water, effectively assuming its volume will not change when it is subject to various forces. The flow velocities of incompressible fluids [163] can be represented as a vector field with no divergence, i.e.

$$\nabla \cdot \mathbf{f} = 0, \tag{4.6}$$

where $\nabla\cdot$ is the divergence operator.

In the special case of 2D time-invariant incompressible flow fields, scalar functions called *stream functions* $\psi : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$ can be defined, which describes the net flow or *flow flux* that crosses any curve connecting the two points [164], and also uniquely defines the flow field. The amount of flow flux ψ_{PQ} between two points P and Q can be computed with the path-independent path integral

$$(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}) = \int_{\mathbf{x}_\mathrm{P}}^{\mathbf{x}_\mathrm{Q}} f_1(\mathbf{x})\, dx_2 - f_2(\mathbf{x})\, dx_1, \tag{4.7}$$

for the flow field $\mathbf{f}(\mathbf{x}) = [f_1, f_2]^\intercal(\mathbf{x})$ and $\mathbf{x} = [x_1, x_2]^\intercal$. Stream functions are useful representations because their level sets after fixing one of their arguments are streamlines which are useful to visualise the directions of flow.

An example of some streamlines are shown in Fig. 4.4. Four streamlines can be seen which have different flow flux relative to the arbitrary reference point at $[-1, 0]^\intercal$ showing the flow of fluid around a circular obstacle.

For brevity, when the second argument of flow flux is omitted, it implies that some arbitrary common reference point $\mathbf{x}_\mathrm{P}$ is used, i.e. $\psi(\mathbf{x}) = \psi(\mathbf{x}, \mathbf{x}_\mathrm{P})$. E.g.

$$\psi(\mathbf{x}_\mathrm{A}, \mathbf{x}_\mathrm{B}) = \psi(\mathbf{x}_\mathrm{A}) - \psi(\mathbf{x}_\mathrm{B}). \tag{4.8}$$

4.2.2.1 Streamlines as paths

In this thesis, a distinct streamline is defined to be the *contiguous* set of points of $\mathbf{x}'$ such that $\psi(\mathbf{x}, \mathbf{x}') = 0$ for any reference point $\mathbf{x}$. A useful property of the streamline is that any particle or idle vehicle (i.e. $\mathbf{v} = \mathbf{0}$) on a streamline is carried along the same streamline. This can be verified by considering the how $\mathbf{x}$ changes based on (4.1) in the time derivative of (4.7), i.e.

$$\frac{d}{dt}\psi(\mathbf{x}) = -f_2 f_1 + f_1 f_2 = 0. \tag{4.9}$$

In other words, if a point $\mathbf{x}_\mathrm{Q}$ is downstream of a point $\mathbf{x}_\mathrm{P}$, then $\psi(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}) = 0$. This allows us to consider the inverse:

Remark 1 (Streamline reachability condition)**.** Given an incompressible flow field $\mathbf{f}$ and its stream function ψ, it is impossible for a drifting element at some point $\mathbf{x}_\mathrm{P}$ to reach point $\mathbf{x}_\mathrm{Q}$ if

$$\psi(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}) \neq 0. \tag{4.10}$$

It is important to clarify that zero flow flux is a *necessary*, but not *sufficient* condition for reachability, since the point could be upstream or still be on a different streamline.

4.2.2.2 Superposing stream functions

Vector fields can be *superposed* with others by taking the of sum their contributions across the domain. It can be easily shown that the result from superposing two incompressible

flow fields is also incompressible. Furthermore, the stream function of the resultant flow field is equal to the sum of stream functions of the component flow fields.

Remark 2 (Additive property of stream function)**.** Given two incompressible flow fields $\mathbf{f}_{\mathrm{A}}$, $\mathbf{f}_{\mathrm{B}}$, and their stream functions ψ_{A} and ψ_{B}, the stream function of the resultant flow field can be obtained as the summation of the ψ_{A} and ψ_{B}. I.e. for $\mathbf{f}_{\mathrm{A}}$ and $\mathbf{f}_{\mathrm{B}}$ satisfying (4.6), if

$$\mathbf{f}_{\mathrm{A+B}} = \mathbf{f}_{\mathrm{A}} + \mathbf{f}_{\mathrm{B}}, \tag{4.11}$$

then the stream function for $\mathbf{f}_{\mathrm{A+B}}$ is

$$\psi_{\mathrm{A+B}} = \psi_{\mathrm{A}} + \psi_{\mathrm{B}}. \tag{4.12}$$

This property is vital in the derivation of a *virtual* stream function which describes the trajectory resulting from the combined effects of the flow field and vehicle actuation. The property is also useful in the construction of complex flow fields from simpler flow fields.

4.3 Problem formulation

Path planning for a vehicle that uses persistent controls between a given pair of starting and goal positions can be represented as a *waypoint selection problem*, i.e. find a sequence of *waypoints* at each of which the vehicle will change its actuation setpoints to reach the next position. Suppose the vehicle is constrained by to the control space $\mathbb{U}$ travelling in a time-invariant incompressible flow field $\mathbf{f}$. The waypoint selection problem is to find a sequence of waypoints between the starting position $\mathbf{x}_{\mathrm{init}} \in \mathbb{X}$ and the goal position $\mathbf{x}_{\mathrm{goal}} \in \mathbb{X}$ whilst minimising the overall cost to traverse. I.e. find

$$\begin{aligned} \mathsf{W}^* = \underset{(\mathbf{x}_1,\ldots,\mathbf{x}_K)\in\mathbb{X}^K}{\arg\min} & \sum_{k=0}^{K} f_{\mathrm{c}}\left(\mathbf{x}_k, \mathsf{a}_k\right) \\ \text{subject to } \mathbf{x}_{k+1} &= \mathbf{f}_{\mathrm{x}}(\mathbf{x}_k, \mathsf{a}_k) \\ \mathbf{x}_0 &= \mathbf{x}_{\mathrm{init}} \\ \mathbf{x}_{K+1} &= \mathbf{x}_{\mathrm{goal}}. \end{aligned} \tag{4.13}$$

Many sampling-based path planning algorithms are suitable to address this problem and have desirable properties such as probabilistic completeness and asymptotic optimality such as dominance-informed region trees [133] and stable sparse rapidly exploring random trees (SSTs) [116]. A subset of such algorithms leverage the availability of a two-point boundary-value problem (TP-BVP) solver for their steering function which provide the control action that the system can take to transition from one state to another. These algorithms employ TP-BVP solvers generally to prolong the relevance of computational resources previously committed during planning. For example, probabilistic roadmap (PRM) [10] constructs a roadmap that can be queried multiple times for different start and goal states, rapidly exploring random tree (RRT)* [10] adjusts the connectivity of the neighbourhood of newly added nodes in search tree during construction to extend the usefulness of descendant nodes, and RRT$^{\mathrm{X}}$ [127] rewires the constructed search tree to respond to newly discovered obstacles during the execution of the path it provides.

We are interested in finding an efficient TP-BVP solver for the steering function to make this class of sampling-based path planning algorithms more practical for use in computationally limited systems. We consider steering functions that provide a persistent control a that minimises a given cost function f_{c} that takes a vehicle from $\mathbf{x}_{\mathrm{P}} \in \mathbb{X}$ to $\mathbf{x}_{\mathrm{Q}} \in \mathbb{X}$ under the effects of advection from $\mathbf{f}(\mathbf{x})$ if it exists. I.e. find

$$\begin{aligned} \mathsf{a}^* = \underset{\mathsf{a}\in\mathbb{A}}{\arg\min}\ & f_{\mathrm{c}}\left(\mathbf{x}_{\mathrm{P}}, \mathsf{a}\right) \\ \text{subject to } & \exists \mathsf{a} \in \mathbb{A} \text{ s.t. } \mathbf{x}_{\mathrm{Q}} = \mathbf{f}_{\mathrm{x}}(\mathbf{x}_{\mathrm{P}}, \mathsf{a}). \end{aligned} \tag{4.14}$$

This is in contrast to steering functions that provide controls that vary between the two endpoints which would lead to overall solutions that are complex to convey between humans, and impractical to use for underwater gliders.

There are multiple numerical approaches that can solve (4.14), however direct applications of them to our problem involves a search over three degrees of freedom for 2D path planning, and four degrees of freedom for 3D path planning. The computation required is magnified by the necessity to perform forward integrations to check if a candidate persistent control takes the vehicle to its intended target during the numerical search. This

is a computationally intensive process that also needs to occur for every pair of positions when solving (4.13), which motivates us to address the core problem in this chapter.

Problem 2 (Search space reduction for optimal persistent controls)**.** *We are interested in reducing the search space in* $\mathbb{A}$ *when solving* (4.14) *using numerical computation techniques, or sampling-based approaches to reduce the computational time required.*

4.4 Streamline-based steering in 2D

In this section, we present a constraint to the search for a persistent control that steers a vehicle modelled by (4.2) following the system dynamics (4.1) from one waypoint to another in a 2D time-invariant incompressible flow field. We consider a *virtual* flow field which accounts for the effects of the environmental flow field and the effects of the vehicle's actuation. The constraint eliminates one dimension of the search and excludes only control candidates that are provably infeasible which allows the path planning algorithm to focus efforts on more promising alternatives improving solution quality and/or reducing the needed computation time.

4.4.1 Streamline-based search for the persistent control

4.4.1.1 Constraining velocity with the stream function

Recall that the system (4.1) is controlled by the vehicle's actuation which produces a velocity $\mathbf{v} = [v_1, v_2]^\intercal$ relative to the flow field. We will first describe the effects of the actuation as a flow field. Actuation that is constant over an indefinite amount of time can be expressed as an additional flow field $\mathbf{f}_{\text{veh}}$ acting on an idle vehicle, i.e.

$$\mathbf{f}_{\text{veh}}\left(\mathbf{x} \,|\, \mathbf{v}\right) = \mathbf{v}, \tag{4.15}$$

which can be referred to as the *flow due to control*. This flow field is incompressible since it satisfies (4.6), and also has the stream function

$$\psi_{\text{veh}}\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}} \,|\, \mathbf{v}\right) = v_1 \Delta x_2 - v_2 \Delta x_1, \tag{4.16}$$

where

$$\begin{bmatrix} \Delta x_1 \\ \Delta x_2 \end{bmatrix} = \Delta \mathbf{x} = \mathbf{x}_\mathrm{Q} - \mathbf{x}_\mathrm{P}. \tag{4.17}$$

The system dynamics can then be purely described by superposing the environment's flow field and the flow due to control, the result of which will called the *virtual flow field*

$$\mathbf{f}_\mathrm{vir}\left(\mathbf{x} \,|\, \mathbf{v}\right) = \mathbf{f}\left(\mathbf{x}\right) + \mathbf{f}_\mathrm{veh}\left(\mathbf{x} \,|\, \mathbf{v}\right). \tag{4.18}$$

We can now interpret the system to be an unactuated drifting particle in the virtual flow field, but is subject to a flow field that we can warp by changing the control $\mathbf{v}$.

From Remark 2, the virtual flow field's stream function is

$$\begin{aligned} \psi_\mathrm{vir}\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q} \,|\, \mathbf{v}\right) &= \psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right) + \psi_\mathrm{veh}\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q} \,|\, \mathbf{v}\right) \\ &= \psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right) + v_1 \Delta x_2 - v_2 \Delta x_1. \end{aligned} \tag{4.19}$$

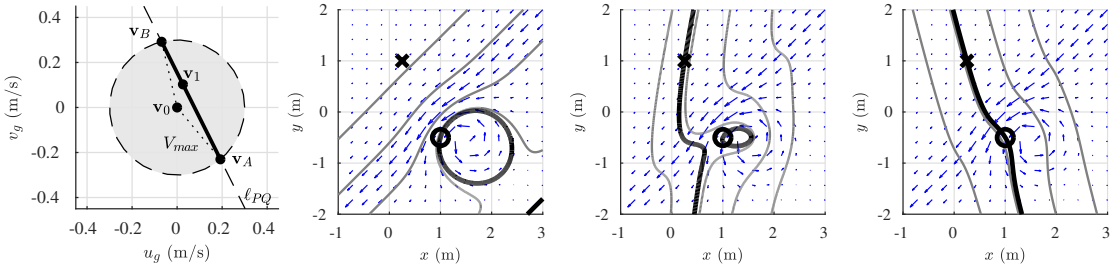
We know that the drifting particle at some position P cannot reach some other target position Q unless $\psi_\mathrm{vir}\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q} \,|\, \mathbf{v}\right) = 0$ from Remark 1, i.e.

$$\psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right) + v_1 \Delta x_2 - v_2 \Delta x_1 = 0. \tag{4.20}$$

For each pair of positions P and Q, this imposes a constraint on the possible values of the control velocity $\mathbf{v}$ to a line in the control space, which is denoted by ℓ_PQ. We will refer to this as the *streamline constraint*, and ℓ_PQ as the *control line*. In other words, any persistent control that steers the vehicle from P to Q must satisfy the streamline constraint.

The trajectory of the particle or vehicle is usually curved, despite the constant control velocity because they are subject to the environment's flow velocities that vary across space. In practice, the term $\psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right)$ is trivial to compute using (4.7), since an integral can be taken on the line segment between $\mathbf{x}_\mathrm{P}$ and $\mathbf{x}_\mathrm{Q}$.

A consequence of this theory is that no solution can exist if the control line does not intersect the vehicle's control space. In application, this allows us to quickly reject reachability



(A) Control space (B) $\mathbf{v} = \mathbf{v}_0$ (C) $\mathbf{v} = \mathbf{v}_1$ (D) $\mathbf{v} = \mathbf{v}_B$

FIGURE 4.5: Samples of velocity controls and their corresponding virtual streamlines, which are based on superposing the environment's flow field with the flow field that reflects vehicle actuation. (a) shows the controls the vehicle can execute as a grey disc, the control line ℓ_{PQ} which is our proposed search constraint, and some control samples. (b-d) show the virtual streamlines as grey lines for each of the control samples. The vehicle's location is marked as a circle and the target is marked as a cross. The resulting trajectory from executing the control and the environment's flow field (blue arrows) is captured by the level set of the virtual stream function. Choosing a control with proposed constraint ensures that the target is on the level set of the virtual stream function, but the level set may describe multiple streamlines.

queries between two positions since we can determine whether the intersection is empty using a flow flux query without forward integration.

An interesting property about the control line is that it is not affected by the direction of travel, i.e. the same control line applies to traversal from P to Q as it does from Q to P.

4.4.1.2 Generating constrained velocities

A more convenient way to constrain the velocity search with the streamline constraint is to reformulate the constraint to a the parametric form, then search for the parameter. The parametric form of the constraint using κ as the parameter is as follows:

$$\begin{aligned} \mathbf{v}(\kappa) &= \mathbf{v}_{\text{base}} + \kappa \hat{\mathbf{v}}_{\text{forward}} \\ \mathbf{v}_{\text{base}} &= \begin{bmatrix} \Delta x_2 \\ -\Delta x_1 \end{bmatrix} \cdot \frac{\psi\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)}{\|\Delta \mathbf{x}\|_2^2} \\ \hat{\mathbf{v}}_{\text{forward}} &= \frac{\Delta \mathbf{x}}{\|\Delta \mathbf{x}\|_2}, \end{aligned} \tag{4.21}$$

where $\mathbf{v}_{\text{base}}$ is the control velocity with lowest speed that still satisfies the streamline constraint, and $\hat{\mathbf{v}}_{\text{forward}}$ is a unit velocity vector towards the target. The vector $\hat{\mathbf{v}}_{\text{forward}}$ is

directed to target from the vehicle's position, so that the parameter κ can be interpreted as effective driving speed towards the target. On the other hand, the vector $\mathbf{v}_{\text{base}}$ is perpendicular to $\hat{\mathbf{v}}_{\text{forward}}$, and can be interpreted as a correction term to cancel out the flow flux from the environment's flow field between the two positions. The magnitude of this vector

$$V_{\text{LSB}}\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right) = \left\|\mathbf{v}_{\text{base}}\right\|_2 = \frac{\left|\psi\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)\right|}{\left\|\Delta\mathbf{x}\right\|_2}, \tag{4.22}$$

turns out to be quite useful as it provides the *lower speed bound (LSB)* of vehicle propulsion speed for a solution to exist between the points $\mathbf{x}_{\text{P}}$ and $\mathbf{x}_{\text{Q}}$.

Figure 4.5 shows various control velocities chosen from $\mathbb{U}$, and how they affect the vehicle's virtual streamlines. In Fig. 4.5a, the control space determined by the maximum speed constraint $\left\|\mathbf{v}\right\|_2 \leq V_{\max}$ is shown with the control line ℓ_{PQ} along with some example velocities $\mathbf{v}_0$, $\mathbf{v}_{\text{base}}$, and $\mathbf{v}_{\text{A}}$. The velocity $\mathbf{v}_{\text{A}}$ is chosen such that it is a velocity that lies on the intersection of $\mathbb{U}$ and control line ℓ_{PQ} that also causes the vehicle to travel towards the target the most. Since $\mathbf{v} = \mathbf{v}_0$ is not on ℓ_{PQ}, the virtual flow flux between the start and target points is non-zero implying that the vehicle is guaranteed to not reach the target since no virtual streamline can possibly connect the start to the target. If $\mathbf{v} = \mathbf{v}_{\text{base}}$, then the flow flux between the start and the target is zero so both positions are on the same level set of the virtual stream function, however no *continuous* streamline connects the two positions, so a vehicle at one position cannot reach the other. If $\mathbf{v} = \mathbf{v}_{\text{A}}$, one streamline connects the start and the target and the target is downstream of the start in the virtual flow field so the target is therefore reachable.

After parametrising the streamline constraint, the search boundaries of κ is described by

$$\kappa \in \left[-\sqrt{V_{\max}^2 - \frac{\psi\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)^2}{\left\|\Delta\mathbf{x}\right\|_2^2}}, \sqrt{V_{\max}^2 - \frac{\psi\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)^2}{\left\|\Delta\mathbf{x}\right\|_2^2}}\right]. \tag{4.23}$$

Feasible controls only exist when the intersection between the control line and the velocity space is non-empty, i.e. when κ can take one or more values, or when

$$V_{\max} \geq V_{\text{LSB}}\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right). \tag{4.24}$$

Whilst we primarily consider cases where the environmental flow field noticeably affects the trajectory of the vehicle, the streamline constraint also applies for cases where the environmental flow field negligibly affects the trajectory, i.e. when the vehicle can travel quickly, or when the environmental flow is weak.

Remark 3 (Asymptotic behaviour of the control line)**.** Consider the case where $\kappa \to \infty$ whilst $\psi(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}) \to 0$. We see that the direction of the vehicle tends towards the target because $\mathbf{v} \to \kappa \mathbf{v}_\mathrm{forward}$ based on the streamline constraint. This matches the intuition that when such a vehicle is primarily driven by its own actuation, it would practically lead to a straight trajectory and can therefore drive directly towards the target.

4.4.1.3 Role of forward integration

There are two main reasons that necessitate forward integration. The first reason is that there is no known way to analytically determine whether a candidate control velocity would take a vehicle from the starting position to the target as both the resulting virtual streamline must connect the two positions, and the vehicle must travel in the direction of the target. The other reason is to determine the duration of the persistent control.

A simple numerical integration scheme that can be used is the Euler method, taking relatively small time steps with fixed step size. This discretises the continuous system (4.1) is with a time step ΔT to

$$\mathbf{x}_{i+1} = \mathbf{x}_i + \mathbf{f}_\mathrm{vir}(\mathbf{x}_i \,|\, \mathbf{v})\, \Delta T, \tag{4.25}$$

which can be used to generate a sequence of trajectory positions from the initial conditions consisting of the starting position and the control velocity. We consider the trajectory to be the polyline where the vertices are the generated trajectory points.

A target is considered to be reachable from the initial conditions if it is within some *reaching threshold* distance of the trajectory. When reachable, the duration of the persistent control to the target is computed as the sum of full integration steps plus a partial integration step to reach the point on the polyline which is closest to the target. The partial

integration step is computed as

$$\Delta T_{\text{par}} = \frac{\mathbf{f}_{\text{vir}}\left(\mathbf{x}_i \mid \mathbf{v}\right) \cdot \left(\mathbf{x}_{\text{Q}} - \mathbf{x}_i\right)}{\left\|\mathbf{f}_{\text{vir}}\left(\mathbf{x}_i \mid \mathbf{v}\right)\right\|_2^2}, \tag{4.26}$$

where line segment described by $\mathbf{x}_i$ and $\mathbf{x}_{i+1}$ is the closest to $\mathbf{x}_{\text{Q}}$.

Numerical integration error can be reduced by using linear multistep methods such as the Adams-Bashforth method [165] instead of using the Euler method. These types of methods incur lower numerical error by leveraging previous computed values from previous iterations. This allows larger integration steps without building as much error, and therefore leads to lower computational demands.

The process of forward integration also gives us the benefit of considering cost functions that are dependent on the time and positions throughout the trajectory without much overhead. One example of such cost function is risk described in [54] where it quantifies how much the vehicle passes through busy regions.

4.4.1.4 Application of the constraint

Any algorithm that can be used to solve (4.14) can instead be used to search through a strict subset of $\mathbb{A}$ after applying the streamline constraint to find the control that minimises the cost to traverse between two waypoints. This includes many iterative numerical optimisation algorithms. Furthermore, by being able to constrain the search to a single variable, we can also consider univariate optimisation algorithms. We also find that discretising the interval by sampling and choosing the best control from a fixed number of sampled velocities works well in practice. The sampled controls can be fine-tuned after obtaining the full sequence of persistent controls after solving (4.13).

4.4.2 Discussion

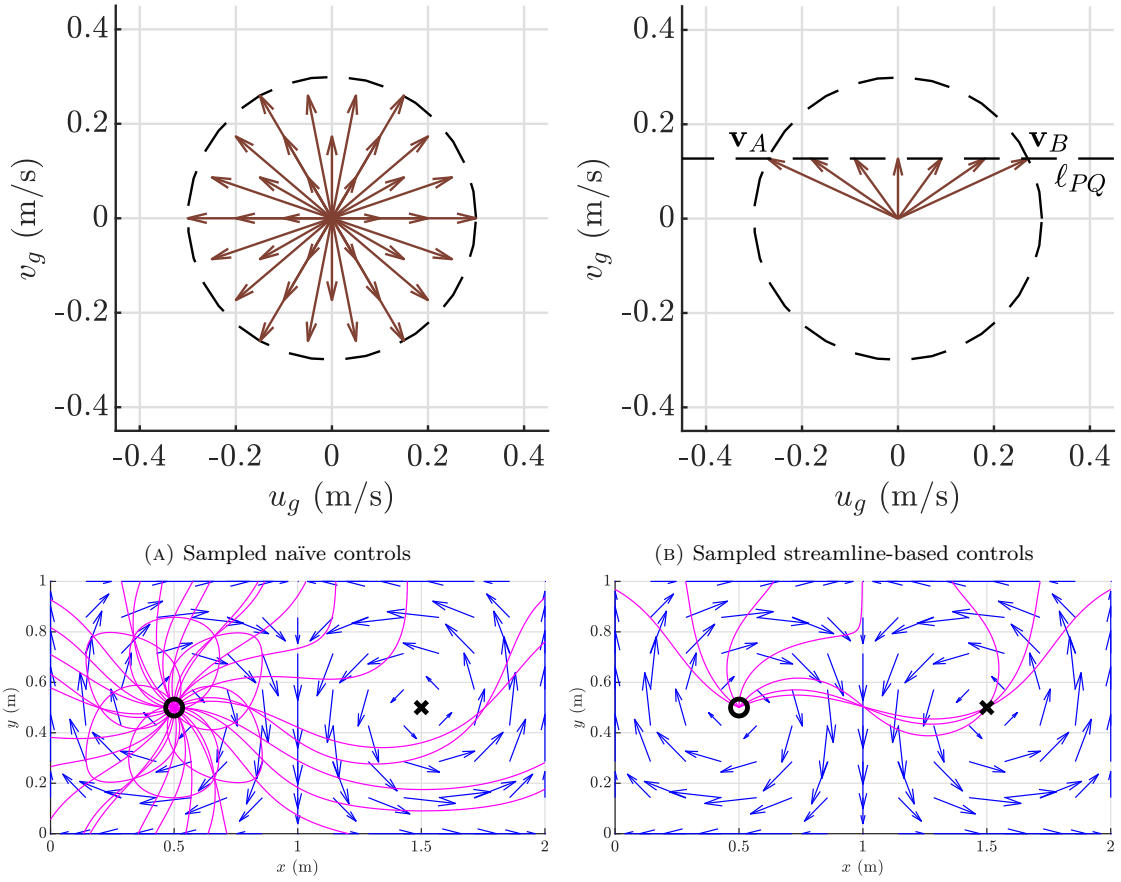
In this section, we focus on the merits of applying the streamline constraint when searching for the persistent control to solve (4.14) compared to without. We will refer to approaches leveraging the streamline constraint with the adjective *streamline-based* and virtual flow

flux-unaware approaches with *naïve*. Techniques that generate paths based on continuous control variation without considering the environmental influence of ocean advection like path generation based on a pure pursuit controller proposed by Kuwata et al. [166] is not suitable for reasons explained in Sec. 4.3, as well as inefficient use of vehicle propulsion as it can travel unnecessarily upstream by trying to follow a particular path.

Firstly, the main difference between streamline-based searches and the naïve searches is that there is one less degree of freedom in the search for feasible controls. Our approach reduces the dimension of the search space from (4.2) in a way that does not exclude feasible solutions, and excludes most of the infeasible solutions. In contrast, naïve approaches would have to find solutions aligned to the 1D manifold within the 2D search space which is cumbersome in practice. Consequently, we expect higher-quality edge connections from the streamline-based approach since it can focus all the computational effort into a smaller space where all the solutions must exist which increases the chance of finding better ones.

Next, we highlight issues that can arise from sampling-based approaches that would not be present if constraint is applied. If a sampling-based approach is used, it is desirable to take samples in such a way that have a lower dispersion, i.e have better coverage of the search space. The sampling pattern with the lowest Euclidean norm (2-norm) dispersion in 2D is an *equilateral triangle tiling pattern* [42]. However this can be problematic for the naïve approach because lower sampling numbers can lead to the resulting the grid to not cover the control line. This is illustrated in Fig. 4.6 which shows sampled controls and their corresponding trajectories for both the naïve and the streamline-based approach. This issue is avoided by construction if the search is performed using the parametric formulation in (4.23).

Finally, we discuss the possibility of reducing the search space further in time-optimal path planning applications. Bang-bang control is sometimes applied to these problems by switching between control extremes [42]. In the thesis [44], the holonomic vehicle is controlled with continuous adjustments, and it is shown that the maximum vehicle speed should be used to construct the time-optimal trajectory. The level set method used in their work utilises a reachability front, or a set of points that describe the extent of reachability for a particular time. By propagating this reachability front forward in time,



(A) Sampled naïve controls

(B) Sampled streamline-based controls

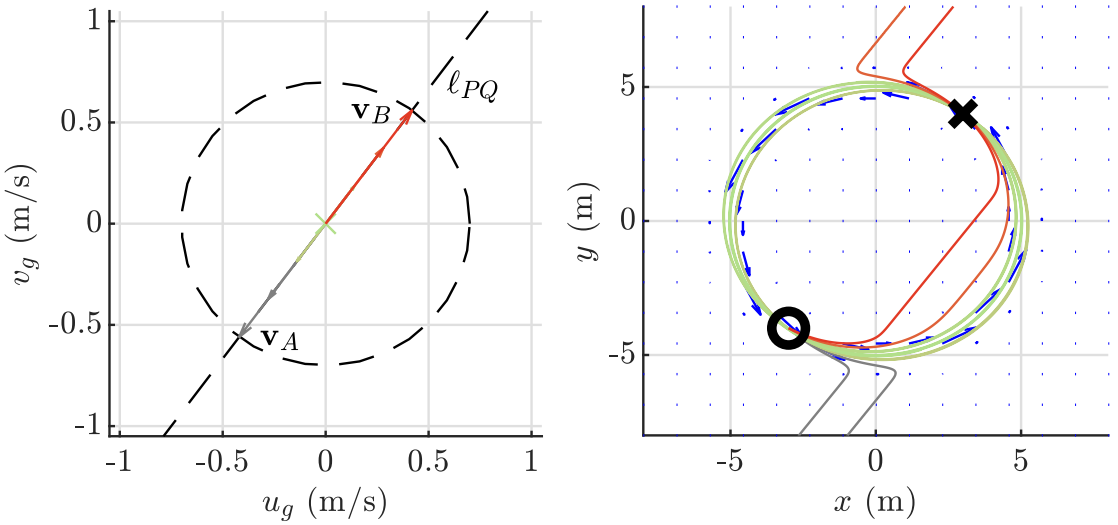
(C) Naïve approach trajectories

(D) Streamline-based approach trajectories

FIGURE 4.6: Comparison between the naïve (a,c) and streamline-based (b,d) steering functions that use a sampling strategy to find a suitable control vector. The control space view (a,b) shows the control vectors (arrows) bounded by vehicle actuation limits (dashed circle) that are used to generate trajectories. The state space view (c,d) shows the generated trajectories (lines) from the initial position (circle) to the target position (cross).

a time-optimal path can be obtained when the set of points first passes the destination. The reachability front is represented as a continuous curve that is propagated forward by considering the highest perpendicular speed outwards the vehicle can achieve taking into account the flow velocity and the vehicle's own velocity. This allows the vehicle to always use the maximum speed, therefore the time-optimal trajectory always involves the vehicle travelling at maximum speed.

However, this does not apply when solving the sub-problem (4.14) since we consider the



(A) Control space (B) Resulting trajectories

FIGURE 4.7: Example scenario where the time-optimal persistent control to manoeuvre from the start (circle) to the target (cross) does not use the vehicle's maximum speed

case where velocity is not continuously updated. We now present a counterexample, showing that travelling at the maximum speed does not necessarily yield time-optimal trajectories in our problem. Figure 4.7 shows an example how the vehicle with limited speed can travel to its target in the shorted amount of time without using its maximum speed. In this scenario, the vehicle is on a curved stream of flow that is much faster than the vehicle's maximum speed $5 \gg 0.7\,\mathrm{m/s}$. If the vehicle tries to travel towards the target at its maximum speed, then it would leave the assistive stream's influence and would have to rely on its own actuation to get to the target. However if the vehicle uses no actuation, it can reach the target in a shorter period of time despite the longer travelled distance. This means that we cannot guarantee a time-optimal trajectory between subsequent waypoints by only considering the maximum speed. Despite this, it would be very computationally advantageous make this maximum speed assumption in practice since the search space would be reduced by one more dimension to 0D, i.e. a constant number of candidates.

It is interesting to note that whilst we cannot change controls in (4.14), the overall waypoint selection problem can consider them at each added waypoint. As more waypoints are considered, the solution of the overall problem converges to the solution from the level set approaches [43, 44, 167, 168], and therefore would make the maximum speed assumption more valid.

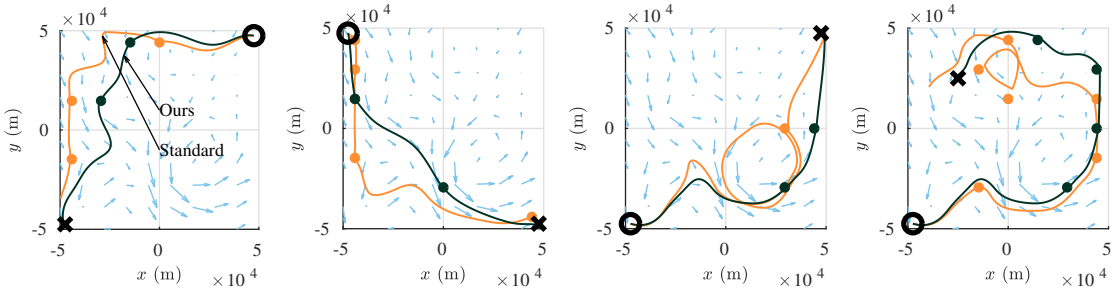
4.4.3 Case studies

In this section, we show how the proposed search constraint improves the performance of path planning enough to enable effective planning over large scales in challenging flow fields featuring higher flow speeds than the vehicle's maximum speed of $|V_{\max}| = 0.3\,\mathrm{m/s}$. This will be demonstrated by showing the outcomes in two scenarios using the streamline-based constraint and without using the constraint, i.e. searching over $\mathbb{U}$. The first scenario is a fictional flow field, and the second scenario is a flow field based on a forecast from the Australian BOM.

We believe that the choices in steering function and the sampling-based path planner to be interchangeable. Since we are focusing on Prob. 2 in this chapter, we choose to use a directed variant of PRM [10] to address the waypoint selection problem described by (4.13) for its well-known properties and algorithmic simplicity. This variant of the algorithm is outlined as follows. First a graph is constructed only with a set of waypoints as nodes consisting of the initial position $\mathbf{x}_0$, the goal position $\mathbf{x}_{\mathrm{goal}}$, and $n_{\texttt{node}} - 2$ sampled positions from $\mathbb{X}$. For each of these nodes, we consider constructing an edge to a nearby node by solving (4.14). If the nearby node is reachable, i.e. if the steering function finds an optimal persistent control that takes the vehicle within some distance threshold of the nearby node, then the cost of traversal between the two waypoints is used as the weight of an edge constructed between the two nodes. After edge construction from all nodes, Dijkstra's algorithm [169] is performed on the resulting graph to determine the sequence of persistent controls corresponding to the graph-optimal path from $\mathbf{x}_0$ to $\mathbf{x}_{\mathrm{goal}}$.

The PRM nodes are kept the same for both methods, and are selected from a rectangular grid. Despite existing work [10, 86, 98, 99] that recommends using a threshold based on Euclidean distance to determine which nodes are considered to be near, we consider all other nodes to be near to demonstrate capabilities to establish long-distance edge connections. A more practical choice of the distance function is further explored in Ch. 5 which instead encourages long-distance connections to be formed without considering all other nodes.

The steering function used for these demonstrations is sampling-based. For both compared approaches, the same number of controls are sampled from their search spaces so that the



(A) Case 1 (B) Case 2 (C) Case 3 (D) Case 4

FIGURE 4.8: Time-optimal trajectories in a synthetic flow field obtained through PRM for a vehicle (circle) to reach its goal (cross). The intermediate trajectory waypoints (dots) are chosen by the PRM algorithm to circumnavigate unfavourable currents. Manoeuvres between waypoints are either found using our proposed streamline-based steering function (dark green) or a naïve steering function (orange) constrained by the maximum speed of the vehicle.

computation time for both approaches are similar. For the streamline-based approach, the sampled controls are equispaced along the interval described by (4.23). For the naïve approach, the sampled controls from the disc described by (4.2) are based on an equilateral triangular grid since it has minimum dispersion in two dimensions [42]. The forward integration scheme uses $n_{\mathrm{h}} = 2000$ steps of size $\Delta T = 750\,\mathrm{s}$ and the reaching threshold to determine control feasibility is 625 m.

For these simulations, we will focus on producing time-optimal paths using the following cost function

$$f_{\mathrm{c}}\left(\mathbf{x}, \mathsf{a}\right) = \tau = i\Delta T + \Delta T_{\mathrm{par}}. \tag{4.27}$$

The steering function will provide the persistent control that takes the vehicle from one waypoint to another in the shortest time if any of the sampled controls reach their target, and PRM would return a time-optimal sequence of persistent controls that takes the vehicle to the destination. We will analyse the trajectories which are generated by forward integrating the sequences of persistent controls in succession.

4.4.3.1 Simulations in a synthetic flow field

In this scenario, the synthetic flow field is an incompressible flow field with a maximum flow speed of 1 m/s which is constructed from many simple incompressible flow fields, e.g. uniform flow used in (4.15), Taylor-Green vortices [170] in Fig. 4.6, and the stream and

TABLE 4.1: Duration to traverse the time-optimal paths shown in Fig. 4.8

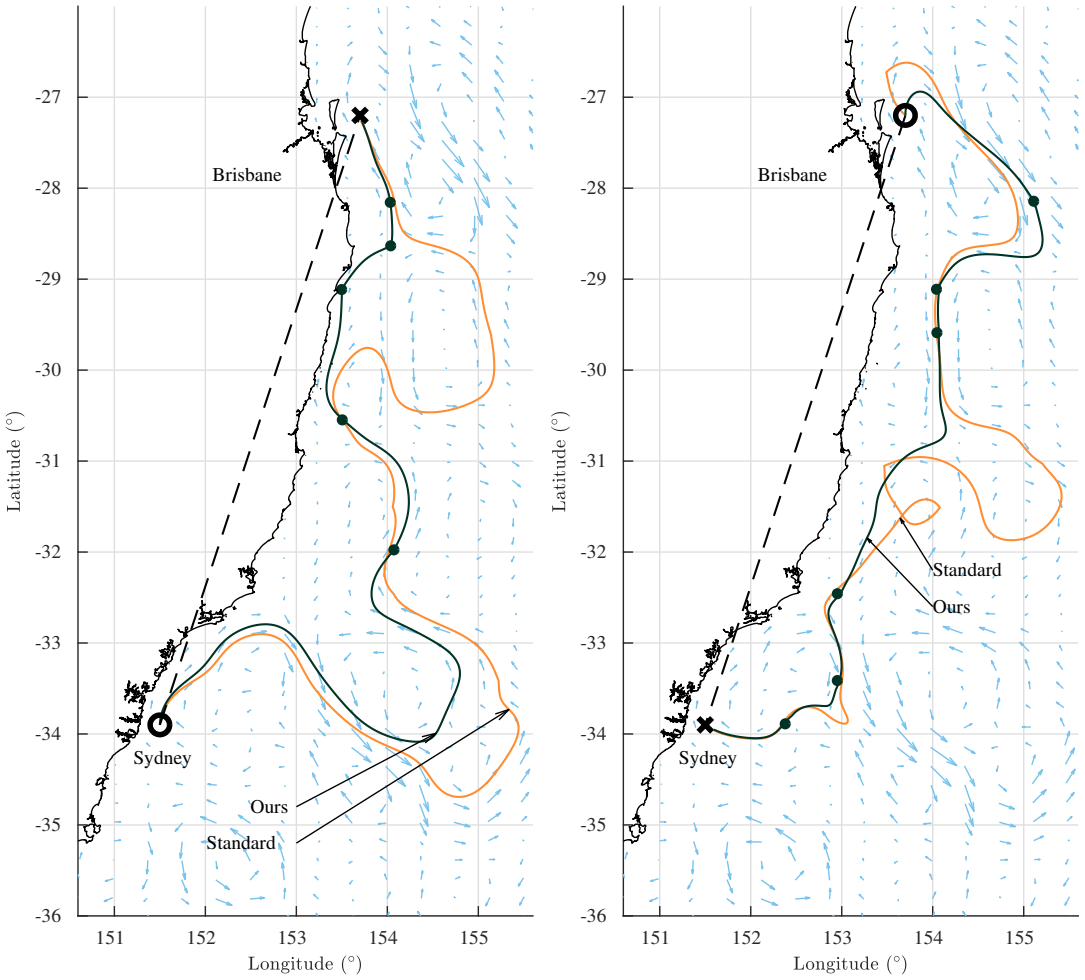
Case	Naïve (days)	**Streamline-based** (days)	Improvement (%)
1	8.1	3.9	51.9
2	3.5	2.2	37.1
3	7.9	3.4	57.0
4	10.4	5.4	48.1

gyre flows which are combined for Fig. 4.5. The PRM algorithm uses $n_{\texttt{node}} = 51$ nodes to build a graph and the steering function uses $n_{\text{c}} = 19$ equispaced control samples for this scenario. We examine the paths generated by the two approaches with three pairs of start and goal positions at opposite corners of the region, and a final pair to find a path to circumnavigate the current on the left going in the negative y-direction.

The trajectories for the four different settings are shown in Fig. 4.8. At a glance, the paths produced from the streamline-based approach are more direct than the naïve approach. We hypothesise that this is because the control samples for the naïve approach rarely align with the control line (4.20), and good quality solutions are unlikely to be among them. This could force the PRM algorithm to take the unintuitive routes like the loops in Fig. 4.8c and Fig. 4.8d. The large final position error for the naïve approach in the 4-th case could be explained by poorly sampled controls. The low-quality control samples would lead to higher overall trajectory error due to accumulated error from heavy reliance on the size of the reaching threshold to form the edges between each waypoint.

The durations for all four trips using each of the approaches are shown in Tab. 4.1. We see that our method consistently produces significantly better solutions than the naïve approach. There is less of an improvement in case 2 because the flow field between the start and goal is already very assistive thus making it very easy for the naïve approach to produce desirable outcomes.

We also observed that all of the controls chosen by the streamline-based steering function uses the vehicle's maximum speed, which suggests that constraining the search to using the vehicle's maximum speed is useful in practice for time-optimal path planning despite the existence of optimal counter-examples as discussed in Sec. 4.4.2. This means that the same trajectory could be generated in roughly one nineteenth of the computation time.



(A) From Sydney to Brisbane

(B) From Brisbane to Sydney

FIGURE 4.9: Large-scale time-optimal path planning on forecasted ocean currents using our streamline-based approach (dark green) compared to the naïve approach (orange). The vehicle (circle) uses a single control to manoeuvre between waypoints (dots) selected by PRM to traverse all the way to the goal (cross).

4.4.3.2 Simulations in the Eastern Australian Current

In this scenario, we use a flow field based on a forecast dataset of the currents along the east coast of Australia from the BOM. The Eastern Australian Current (EAC) is known to have a strong southward current and many eddies with flow speeds up to 2.1 m/s, which is seven times the vehicle's maximum speed. Being able to effectively plan on such an environment would provide a significant improvement in performance by taking advantage of the currents.

TABLE 4.2: Duration to traverse time-optimal paths from Sydney (SYD) and Brisbane (BNE) shown in Fig. 4.9

Case	Naïve (days)	**Streamline-based** (days)	Improvement (%)
SYD to BNE	22.8	17.0	25.4
BNE to SYD	29.4	14.6	50.3

The flow field we use, shown in Fig. 4.9, is based on the top layer of the predicted flow field for 5$^{\text{th}}$ September 2018. Since geodetic projection and direct linear interpolation on the data will introduce divergence to the flow field, we first fit a stream function $\psi(\mathbf{x}, \mathbf{x}_{\text{ref}})$ to the projected flow field using an arbitrary reference point $\mathbf{x}_{\text{ref}}$, then re-compute a flow field $\mathbf{f}(\mathbf{x})$ from the stream function which is incompressible by construction.

We demonstrate path planning capabilities across large distances by finding time-optimal paths shown in Fig. 4.9 between Sydney and Brisbane in either directions. For this scenario, $n_{\texttt{node}} = 212$ PRM nodes are used, 2 from the start and goal positions and 210 were chosen from the ocean area in a rectilinear grid pattern. The sampling-based steering function used $n_{\text{c}} = 19$ equispaced controls to determine reachability and the edge traversal cost.

For the path from Sydney to Brisbane, both approaches use the assistive currents in the south-east region that require a relatively large deviation from the direct path. Note that the choice of routes is a result of the PRM algorithm trying to minimise the overall time cost of the path. This ability is only enabled by a performant steering function that is able to establish many edge connections. The difficulty in finding an optimal path is finding a balance between taking a route with more assistive currents and taking more direct routes. An example of this idea is shown at the end of the paths from the two compared approaches in Fig. 4.9a. The path from using our streamline-based constraint takes a more direct path, whilst the naïve approach takes a more assistive but indirect path. For the other direction shown in Fig. 4.9b, the path from the naïve method resorts to using an unintuitive sequence of waypoints again. The significant difference in path quality is because our method is more likely to find control samples that are valid, which increases the chance of finding a connection with less cost. This highlights the importance of having a steering function that provide edge connections when they exist so that path planning algorithms like PRM does not have to resort to inefficient paths.

The duration of the paths in both directions are shown in Tab. 4.2. The application of the streamline-based constraint leads to better paths than the naïve method for both directions. Despite the EAC's strong southerly current, the naïve method produced a slower path from Brisbane to Sydney compared to the other direction. This did not happen with the streamline-based approach, and required only an additional 2.4 days to travel from Sydney to Brisbane than the other direction. Note that if the vehicle was travelling directly between the two positions marked with dashes in Fig. 4.9 in still water, the journey would take 29.9 days. This highlights that the streamline-based constraint is allowing the algorithm to produce paths that take advantage of the currents, roughly halving the travel time.

An important aspect to highlight is the infrequent number of control adjustments for the path between the cities that are roughly 775 km apart. On average, adjustments are only made every 2 to 3 days.

Whilst both approaches took around 23 hrs to produce a path, we performed an additional computation with the streamline-based constraint that uses a similar control sampling density as the naïve approach ($n_c = 5$) which only took around 6.1 hrs. We also noticed that the two paths from the streamline-based approach with two different control sampling numbers were identical. Further investigation revealed that it is because the controls that were chosen always used the vehicle's maximum speed in the direction of the target. This suggests that in practice, it is reasonable to apply the maximum speed assumption for time-optimal path planning for lower computation times. In this case, applying this assumption would lead to an estimated computation time of 73 min.

Since the computation for the time-optimal paths are much shorter than the time it takes to execute them, it implies that replanning can be done in practice when there are changes in the flow fields. However, it is more useful to be able to obtain a solution on demand, yet also to be able to obtain better solutions through further computation at execution time. This would be particularly important in time-critical applications, so that progress can be made towards the goal, without needing to commit time only for computation. In Ch. 5, we demonstrate the algorithm in an anytime algorithm called RRT*, and we also propose a distance heuristic which can be used to encourage long-distance connections without

adversely affecting the computational complexity compared to forming a fully connected graph. An RRT* approach also enable the incorporations of additional constraints between waypoints such as some minimum duration or minimum distance covered between changes in control.

4.5 Streamline-based steering in 2.5D

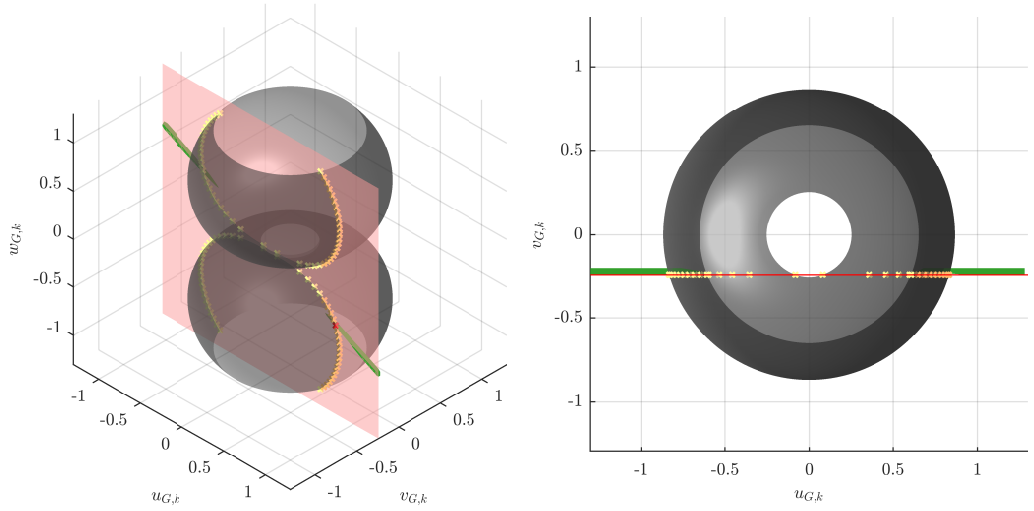
In this section, we present an extension to the streamline-based constraint for the 2D steering function proposed in Sec. 4.4 to plan a sequence of persistent controls for an underwater glider in a 2.5D time-invariant incompressible flow field. The steering function in the 3D setting provides a persistent control that correspond to a single rise or fall component of the typical underwater glider's sawtooth trajectory. The successive execution of a sequence of such ascents and descents generate a long-distance motion plan. To constrain the search for such a control, we leverage the fact that ocean flow sufficiently far from the coastline has negligible vertical velocity [142–144] and that the flow structures are generally preserved across depth to make an assumption that enables us to extend the control line into a plane for 2.5D flow fields, once again constraining the search for feasible persistent controls. The plane's intersection with the control velocity space $\mathbb{U}$ in Fig. 4.3 leads to a 1D set of candidate velocities from which we can search for the optimal control to connect two waypoints.

4.5.1 Streamline-based steering in 2.5D

4.5.1.1 Approximation of a control plane for 2.5D incompressible flow fields

In this section, we propose a way to constrain the search for a persistent controls in the steering function to traverse between two positions, $\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q} \in \mathbb{X} \subset \mathbb{R}^3$ under the advection from a flow field $\mathbf{f} : \mathbb{X} \to \mathbb{R}^2$. We make two assumptions about the environment: 1) the vertical component of the flow field's velocity is zero, and 2) the flow field is incompressible.

For each steering function query between $\mathbf{x}_\mathrm{P}$ and $\mathbf{x}_\mathrm{Q}$, we approximate the flow field between their depths to be *vertically identical* (same in the vertical direction), to the flow field at



(A) Isometric view (B) Top view

FIGURE 4.10: An example intersection of control space (4.5) (grey surfaces) derived from the steady-state underwater glider dynamics and the control plane (pink) based on the streamline constraint between two positions $\mathbf{x}_\mathrm{P}$ and $\mathbf{x}_\mathrm{Q}$ shown over an underwater glider's velocity. Our approach constrains the search space to the set of control candidates (crosses) along the intersection of the surfaces and the plane. Controls that take the vehicle to the correct horizontal position but not the correct vertical position are marked as orange crosses. The control that takes the vehicle to the correct horizontal position and also the correct vertical position are marked as a red cross. Controls that either take the underwater glider from $\mathbf{x}_\mathrm{P}$ to $\mathbf{x}_\mathrm{Q}$ or from $\mathbf{x}_\mathrm{Q}$ to $\mathbf{x}_\mathrm{P}$ are represented as a green volume aligned to the control plane.

the depth centred between both depths, i.e. the flow field is represented as:

$$\hat{\mathbf{f}}(\mathbf{x} \mid \mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}) = \mathbf{f}\left(\left[x_1, x_2, \frac{x_{\mathrm{P},3} + x_{\mathrm{Q},3}}{2}\right]^\intercal\right), \tag{4.28}$$

where $x_{\mathrm{P},3}$ and $x_{\mathrm{Q},3}$ are the vertical positions of $\mathbf{x}_\mathrm{P}$ and $\mathbf{x}_\mathrm{Q}$, and $\mathbf{x} = [x_1, x_2, x_3]^\intercal$.

If a flow field is vertically identical with zero vertical flow velocity, then a persistent control that connects $\mathbf{x}_\mathrm{P}$ to $\mathbf{x}_\mathrm{Q}$ must satisfy the streamline constraint (4.20) after projecting the control, waypoints, and flow field to the horizontal plane. The projected control line corresponds to a *control plane* in this setting, an example of which is shown in Fig. 4.10.

Remark 4 (Control plane definition)**.** Given two 3D positions $\mathbf{x}_\mathrm{P}$ and $\mathbf{x}_\mathrm{Q}$, a control plane is a set of control velocities that ensures that the virtual flow flux of $\mathbf{x}_\mathrm{Q}$ relative to $\mathbf{x}_\mathrm{P}$ is zero in a vertically identical flow field.

In a similar fashion to the 2D case, the streamline constraint in vertically identical flow fields with no vertical flow will only exclude controls that cannot possibly take the vehicle to the target. There are also still controls that satisfy the constraint that do not take the vehicle to the target in the horizontal projection, e.g. Fig. 4.5c and Fig. 4.6d, but here the depth the vehicle could be incorrect as it passes through the horizontal position of the target. As a consequence, forward integration is still necessary to determine target reachability of controls.

4.5.1.2 Representation of constrained search space

The intersection between underwater glider controls (4.5) and the control plane can be divided into ascending controls and descending controls, both of which satisfy glider dynamics and the streamline constraint. The intersection can be described as the set

$$\begin{aligned}
\mathsf{V}_{2.5} &= \left\{ \begin{bmatrix} \mathbf{v}_{\text{base}} \pm \kappa(\beta)\hat{\mathbf{v}}_{\text{forward}} \\ V_{\text{g}}(\beta)\sin\beta \end{bmatrix} \,:\, \beta \in [-\beta_{\text{U}}, -\beta_{\text{L}}] \cup [\beta_{\text{L}}, \beta_{\text{U}}] \right\} \\
\kappa(\beta) &= \sqrt{(V_{\text{g}}(\beta)\cos\beta)^2 - \frac{\hat{\psi}\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)^2}{\|\Delta\mathbf{x}\|_2^2}} \\
\mathbf{v}_{\text{base}} &= \begin{bmatrix} \Delta x_2 \\ -\Delta x_1 \end{bmatrix} \cdot \frac{\hat{\psi}\left(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}\right)}{\|\Delta\mathbf{x}\|_2^2} \\
\hat{\mathbf{v}}_{\text{forward}} &= \frac{\Delta\mathbf{x}}{\|\Delta\mathbf{x}\|_2}
\end{aligned} \quad , \tag{4.29}$$

where $\hat{\psi}$ is the stream function of the vertically identical flow field $\hat{\mathbf{f}}$, and $\Delta\mathbf{x}$ is the horizontal distance between $\mathbf{x}_{\text{P}}$ and $\mathbf{x}_{\text{Q}}$.

However half of the entire set is infeasible for a pair of query waypoints since ascension and descension is purely determined by control, and it is clear which would be needed based on the waypoint depths, i.e. $\beta \in [\beta_{\text{L}}, \beta_{\text{U}}]$ if $x_{\text{Q},3} > x_{\text{P},3}$; otherwise $\beta \in [-\beta_{\text{U}}, -\beta_{\text{L}}]$.

A number of sampled controls at the intersection are shown as an example in Fig. 4.10. The controls are coloured differently depending on whether they reach the target, only reach the target's horizontal position, or they do not reach at all.

To verify the algorithm, we sampled controls densely within the 3D space shown in Fig. 4.10, then performed forward integration from the vehicle's position to check reachability with some tolerance. The control samples were repeated from the target position, to try to reach the initial position since the reverse steering problem shares the same control plane. We found one distinct set of samples that reached for each steering problem direction. The sets are visualised as green volumes in Fig. 4.10, We can see that the controls that reach their target closely align with the control plane and intersect with the control that is found through our method. The result illustrates that the proposed constraint based on an approximation leads to solutions that can be very close to the exact numerical solution. It is interesting to note that a 3D control line potentially exists, which would align to the two green volumes in this case for 2.5D flow fields in a similar fashion to the scenario described in Rem. 3.

The intersection between the control space and the control plane is the empty set if $\kappa(\beta)$ in (4.29) is imaginary. Similar to the 2D case, feasible controls can only exist if

$$\max_{\beta} \left(V_{\mathrm{g}}(\beta)\cos\beta\right) \geq \frac{\left|\hat{\psi}\left(\mathbf{x}_{\mathrm{P}}, \mathbf{x}_{\mathrm{Q}}\right)\right|}{\|\Delta\mathbf{x}\|_2}, \tag{4.30}$$

allowing us to avoid unnecessary computation when this is not true for a particular $\mathbf{x}_{\mathrm{P}}$ and $\mathbf{x}_{\mathrm{Q}}$ pair.

4.5.2 Discussion

Similar to the discussion from Sec. 4.4.2, the 2.5D variant of the constraint on the control space $\mathbb{U}$ reduces the search space from 2D into 1D. Vehicles that can be modelled with a control space that has volume such as autonomous underwater vehicles (AUVs) and even underwater gliders with thrusters [171] could apply this constraint to reduce the search space from 3D into 2D instead. In this section, we discuss one other theoretical aspect of the streamline-based constraint.

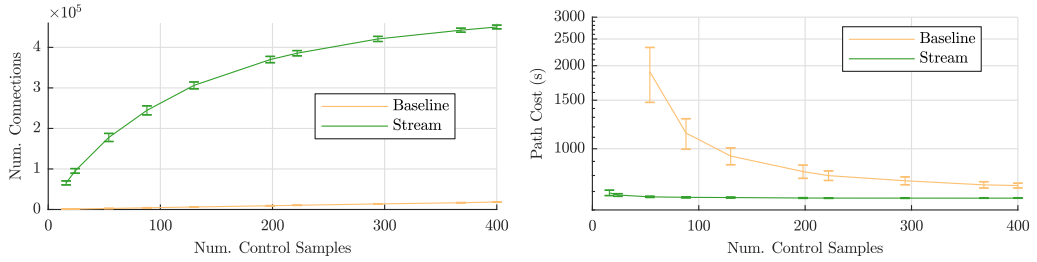
As a compromise for the reduced search space, the streamline-based constraint loses completeness when applied for numerical searches to solve (4.14). The vertically identical approximation of the flow field can misalign from the underlying solution space for large

changes of flow patterns across the depths of the two query waypoints, which causes the algorithm to exclude feasible controls. The algorithm is still be sound since the forward integration process confirms whether the target is truly reachable. However, the degree of incompleteness depends on the difference in depths of the waypoint pair, since the approximation is made for every pair. If the two query positions are more vertically close, then the approximation would be more accurate since the flow field would vary less. This shortcoming is alleviated in sampling-based planner approaches for the waypoint selection problem because since the distance between two query positions decrease as the number of waypoint samples increases. Therefore a sampling-based planner that uses a steering function applying the streamline-based constraint is probabilistically complete as the number of waypoint samples increases.

4.5.3 Empirical results

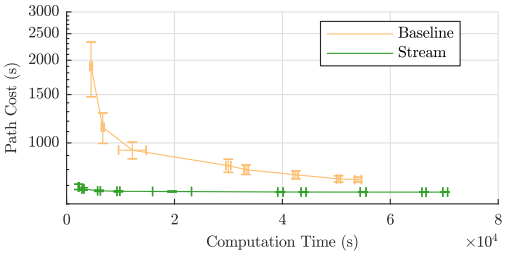
In this section, we test the streamline-based constraint for the 2.5D steering function against the naïve approach in the context of a full path planning scenario to reinforce the merits of using the proposed approach. For the naïve approach, the steering function will be constrained to search in either the top or the bottom surfaces of $\mathbb{U}$ depending on whether the query corresponds to an ascending or a descending manoeuvre. We employ the same implementation described in Sec. 4.4.3 where we used PRM to find time-optimal paths consisting of waypoints, a sampling-based steering function solver to find persistent controls connecting two subsequent waypoints, and the forward integration scheme using Euler method. We use 125 steps in the forward integration process using step sizes of $\Delta t = 5\,\mathrm{s}$, and use a reaching threshold of $5\,\mathrm{m}$.

We first compare how performance varies with different numbers of sampled controls from the search spaces of each approach. We then demonstrate the difference in generated paths from the two algorithms in a synthetic flow field. Note that the EAC dataset was not used for comparison since it has relatively small changes in flow velocity in the vertical direction. The synthetic flow field is more likely to invalidate the control plane assumption since it has flow velocities that vary more drastically across the vertical direction.



(A) Number of edge connections in PRM graph with respect to the number of control samples

(B) Path quality with respect to the number of control samples



(C) Path quality with respect to PRM computation time

FIGURE 4.11: Performance comparison between the streamline-based (green) and naïve (orange) PRM in a 2.5D flow field with 1000 equispaced waypoint samples in 3D using a number of randomly sampled control velocities ranging from 16 to 400. The 99.7% confidence interval over 32 runs around the mean value is also shown.

Since sampling evenly in the control space $\mathbb{U}$ is not straightforward in the naïve approach, the control velocity samples are derived from sampled values of pitch and yaw. For the proposed constraint, pitch samples are taken first to determine the vertical speed, and then these are augmented with the horizontal control velocity calculated from the intersection.

4.5.3.1 Performance in sampling-based steering functions

To compare the performance of our proposed approach and the baseline, we investigate the impact caused by using varying amounts of randomly sampled controls ranging from 16 to 400 for the steering function to find time-optimal paths. In this experiment, the same 1000 equispaced waypoints were used in the PRM algorithm for the same the start and goal positions in the same environment. The trends from this experiment are shown in Fig. 4.11, which plots the mean performance statistics with 99.7% confidence intervals for the mean over 32 runs.

The first graph in Fig. 4.11a shows the average number of edge connections that were established using the steering functions. The difference in connectivity between the two approaches is stark. The streamline-based approach leads to a more connected graph even using only 16 control samples compared to the naïve approach with 400. This most likely because the proposed approach is able to drastically reduce the search space which focuses the computational efforts to areas at which feasible controls are likely to be. As for the 2D approach, we believe that this is an important contributing factor to the difference in path quality as shown in Fig. 4.11b. This is reinforced by the fact that the quality of the solutions for lower control sampling numbers for the naïve approach has high variance. It is also important to note that the naïve approach was not able to find any solution using any of the settings with less than 54 control samples in any of the runs.

Despite having some overhead in the computations of flow flux, Fig. 4.11c shows that the planning with the streamline-based steering function is still able to converge to a solution with very little number of control samples and in a very short period of time. As a result, the proposed method is able to produce a more optimal solution in a fraction of the time it takes for the naïve approach to produce its best solution. This is a useful property to exploit in practical applications as the rapid convergence allows our planner to be used in a *plan-as-you-go* manner, adapting to changes in the flow field.

4.5.3.2 Simulations in a synthetic 2.5D flow field

To demonstrate the types of paths for underwater gliders that are created from the streamline-based and naïve approach, we show four paths are generated using $n_{\texttt{node}} = 1026$ PRM nodes using either 16 or 54 control samples for either approaches. Different number of control samples are used to show the difference in resulting path qualities.

The paths are computed from north-east region to the south-west region of a synthetic flow field is based on the flow pattern from the case study in Sec. 4.4.3. The flow field is rescaled such that the spatial extent is smaller and that the maximum flow speed is more than double the maximum horizontal speed of the underwater glider. For large changes in flow velocities in the vertical direction, the flow field velocities are completely reversed in direction over the vertical span.

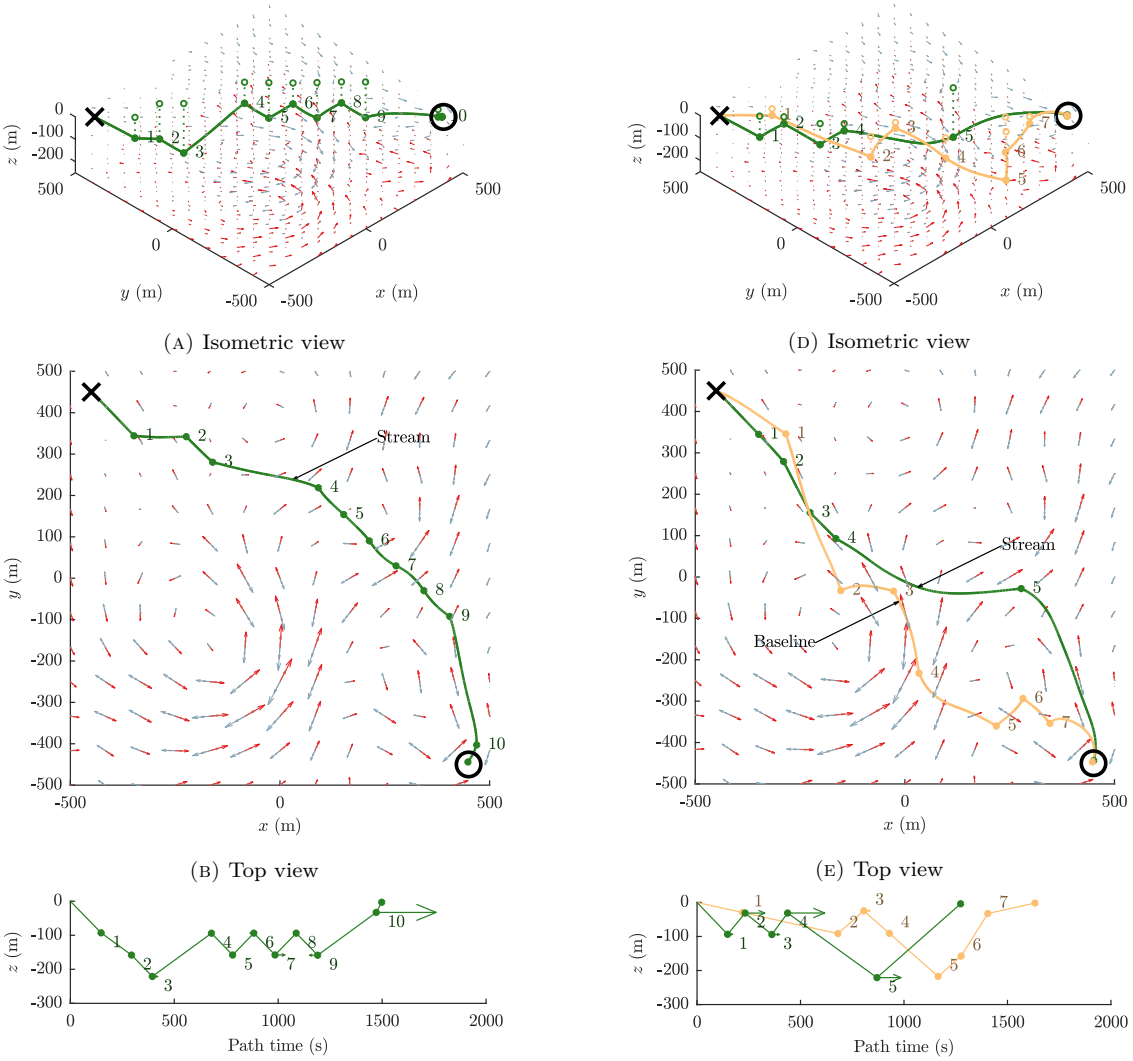
TABLE 4.3: Number of edge connections for the paths shown in Fig. 4.12

# of control samples	# of edge connections	
	Naïve	Streamline-based
16	849	12423
54	3738	29734

For this demonstration, controls were sampled in a consistent way between both approaches. Control velocities from equispaced pitch and yaw were considered in the naïve approach. Control velocities from equispaced pitch were considered in the streamline-based approach, with yaw components determined by the intersection between the control space and the control plane.

The resulting trajectories for both approaches and both levels of control samples are presented in Fig. 4.12. The figure is divided into two columns: the 16 control sample setting is on the left, the 54 control sample setting is on the right. It is also divided into three rows. The top row shows an isometric perspective of the trajectories with additional circle markers which are projections of the waypoints on surface to highlight their depth. The middle row shows the top-down view of the trajectories. The bottom row shows a depth profile of the trajectories with arrows that indicate the presence of assistive or adversarial currents. The trajectory waypoints are also indexed to identify trajectory waypoints between rows.

The path planner with the streamline-based constraint has made some interesting and intuitive choices in the generated trajectories. For the 16 control sample setting, the underwater glider initially dives deep to avoid weak opposing currents and then avoids perpendicular flow by travelling along a depth of 150 m between the 4-th and 9-th waypoints where the magnitude of the flow velocity is lower. We also see that the trajectory takes advantage of the assistive flow just before the goal at the 10-th waypoint in Fig. 4.12c. At the higher control sample setting of 54, the generated path uses more assistive currents which can be seen in Fig. 4.12f as more forward-pointing projected flow velocity arrows. For the first few waypoints, the trajectory utilises the assistive currents by staying in shallow regions. Between the 4-th and 5-th waypoints, the trajectory takes advantage of the shallow currents early in the transit but also the deep currents late in the transit. Finally the trajectory takes advantage of the assistive currents as it rises to the goal.



(A) Isometric view

(B) Top view

(C) Depth profile over execution time

(D) Isometric view

(E) Top view

(F) Depth profile over execution time

FIGURE 4.12: Time-optimal trajectories from $[-450, 450, 0]^\intercal$ (cross) to $[450, -450, 0]^\intercal$ (circle) generated by the proposed streamline-based (green) and the naïve (orange) approaches. Trajectories are computed through a PRM framework where the compared steering functions return the fastest manoeuvre between two positions after either sampling 16 (a-c) or 54 (d-f) control velocities. The flow field velocities are represented with arrows that range from light blue to darker red for increasing depth. The arrows in (c) and (f) indicate the projection of the flow velocity tangential to the trajectory. Note that no solution was found for the naïve approach with 16 control samples.

TABLE 4.4: Travel durations for the paths shown in Fig. 4.12

	Path cost (s)	
# of control samples	Naïve	Streamline-based
16	-	1498
54	1633	1274

For the 16 control sample setting, the naïve method was unable to produce a solution whereas the proposed streamline-based method was able to. The lack of a solution implies that the graph is disconnected between the start and the goal positions. From Tab. 4.3, we can surmise that the disjointed graph is caused by the naïve approach which does not form edge connections between waypoints as effectively. In contrast, the approach using the streamline-based constraint is able to establish many more edges, so much so that it produces more edge connections with 16 control samples than the naïve approach with 54 control samples. This is significant because many more edges established using the proposed approach compared to the naïve approach which took more than 3 times the computation time.

We also observe a correlation between the number of edge connections and the path quality. More edge connections seem to leading to lower cost paths, shown in Tab. 4.4 as paths with lower time required to execute. In the PRM algorithm, the higher levels of graph connectivity implies more available alternate routes. We believe that this allows Dijkstra's algorithm to select more optimal waypoint sequences to reach the goal position. As before, we see that the streamline-based method with 16 control samples produced a better solution than the naïve method with 54 solutions. This means that the proposed method can produce faster and better solutions than the naïve method.

In practice, these trajectories will not need to be re-computed during execution since due to changing flow fields since currents do not change much in short time frames such as those less than 1 hr. Furthermore, recall that we are using PRM with full graph connectivity which involves more computation. For example, if we used $k_{\text{PRM}} = 27$ nearest neighbours suggested in [10], we can compute a trajectory corresponding to a transit time of 1530 s in 399 s. In Ch. 5, we use a more practical path planning algorithm with RRT and the explore the use of a specialised distance heuristic to find more reachable neighbours instead of using Euclidean distance to gain higher convergence rates to the optimum solution.

4.6 Summary

In this chapter we proposed constraints to reduce the search space when choosing a persistent control to connect two positions for the purposes of planning paths for marine surface vehicles in 2D environments and underwater gliders in 3D environments. The fundamental idea is based on the idea of streamlines which effectively reduces the search space from 2D to 1D for both vehicles. The effectiveness of the proposed constraints is demonstrated in time-optimal path planning scenarios with PRM, a sampling-based path planner that used sampling-based steering functions. The proposed streamline-based constraints are shown to produce higher quality paths than the naïve alternative which considered controls over the full control space given the sample computational time budget. In practice, persistent controls can be directly executed with an underlying controller for the vehicle actuators. However, since each segment of the streamline-based trajectory is associated with a virtual streamline with a flow flux, a controller similar to the one proposed by Lagor et al. [172] can be used, reducing the tracking error during execution.

An interesting alternate application for the proposed control search constraint in 2.5D flow fields is path planning in 2D time-variant flow fields. The 2D time-variant problem can be viewed as a special case of the 2.5D problem if each of the 2D flow fields are represented as horizontal planes in the vertical axis of time. In this case, the vehicle is forced to travel at some vertical "speed", but is capable of choosing a horizontal control velocity subject to vehicle actuation limitations e.g. using (4.2). Further work to extend the method for 3D flow fields in a principled manner would improve the quality of the solutions in the above scenarios and potentially the efficiency of implementing algorithms. It would additionally allow us to consider applications where the vertical component of the flow field is not negligible, such as in shallow waters. A challenge is finding a suitable alternative for the stream function in (4.7) since it is only defined in 2D. A potential tool to use in 3D might be dual stream functions [173].

We propose that the streamline-based constraint from this chapter to be used in the steering function for the sampling-based path planner in the system shown in Fig. 1.2, which makes queries for flow field velocity estimates from the corresponding flow field estimator described in Ch. 3. At a baseline level, the constraint decreases the time it

takes to completely replan a path to the intended goal for the vehicle if the flow field estimate updates. For lower computational demands, a sampling-based path planner like RRT$^{\text{X}}$ [127] could potentially be used to prolong the relevance of prior computations. A benefit arising from using an estimator with an incompressible kernel [38] (including the methods proposed in Ch. 3) along with a planner using streamline-based steering functions is that a slightly modified kernel can be used in the estimator to enable flow flux to be directly queried as needed by the streamline-based constraint. This provides a more efficient means to compute the flow flux between points instead of performing the numerical computation of (4.7).

In the next chapter, we consider another improvement in a different primitive function for the planner module of the system that identifies nearby nodes to which we try to establish edge connections. The primitive function will use a proposed distance heuristic that better quantifies the reachability of other positions in incompressible flow fields. Contributions from this chapter and the next allow sampling-based path planners, including the anytime variants, searching for persistent controls to be more computationally tractable and improves its convergence rate to the optimal solution.

Chapter 5

Preceding point selection through flow-based reachability analysis

In Ch. 4, streamline-based control theory was proposed to address the two-point boundary-value problem (TP-BVP) to choose a persistent control to traverse between two points, exploiting an idea from fluid dynamics to achieve computational efficiency. In this chapter, we build upon the idea of streamline-based path planning by studying a different problem that affects path planning for underwater gliders under the influence of flow fields in order to improve the convergence rate of the overall algorithm. This problem is related to the reachability between points under the dynamics of strong incompressible flow fields. We first consider an alternate way to quantify the *reachability* of all arbitrary locations in incompressible flow fields, i.e. the difficulty of reaching other points in free space. The proposed measure is an analytical function related to the Euclidean distance and the concept of flow flux explored in Ch. 4. We consider the extension of these findings to the 2.5D case to be straightforward, and instead focus on a filtered approach to identify reachable neighbours, which leads to better performance. We evaluate the ideas presented in this chapter by testing them against best practices for kinodynamic problems in the framework rapidly exploring random tree (RRT)*, as well as other related techniques in an artificial flow field and in real predicted ocean data in the dominant region of the Eastern Australian Current (EAC) between Sydney and Brisbane. Results demonstrate faster convergence rates that arise from improving the chances the sampled node is added

to the tree by carefully choosing the parent tree node, outperforming other approaches in literature, and show promise for practical use with autonomous marine robots.

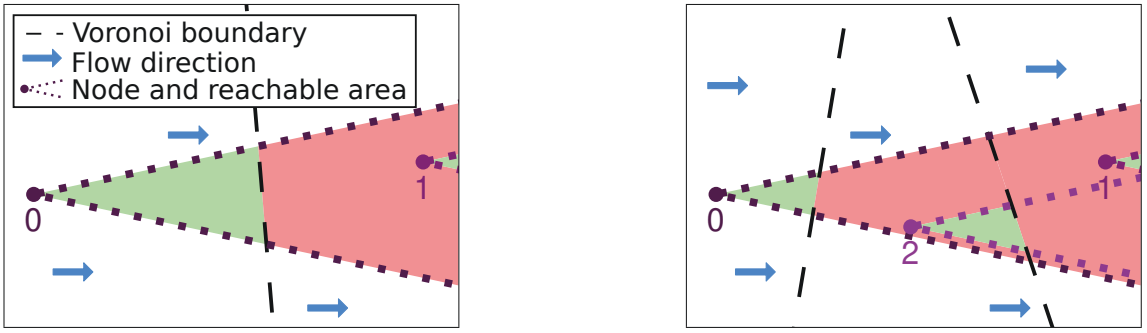
This chapter includes work that is published as [174], and additional work that could potentially be published as a journal article.

5.1 Introduction

The motion of vehicles affected by the advection effects of the environment's flow field can be described as a non-holonomic system. For these systems, the best practice to extend RRT-like search trees is to generate new states by propagating the system using a random control for a random duration. However, we suspect that the Voronoi bias [91] corresponding to this strategy to be weaker since the extension makes no guarantee of reaching the sampled state in larger Voronoi regions. In this chapter, we pursue a different strategy in order to effectively explore the search space by increasing the chances to extending the tree using the sampled point directly by leveraging the streamline-based control theory proposed in the previous chapter.

This leads to a subtle but critical issue that will be referred to as the *advection reachability problem*. Typically, the parent node is typically chosen based on nearest neighbour based on lowest Euclidean distance [68, 71]. However, under advection dynamics (4.1) with strong flow, this can lead to a poor choice of parent preventing connections that can worsen over each iteration of the overall algorithm.

An example of this phenomenon is visualised for a simple example in strong uniform flow in Fig. 5.1. The search tree contains two points labelled ⓪ and ① in Fig. 5.1a where ⓪ is the parent of ①. The same workspace is shown in Fig. 5.1b after point ② is added to the parent point ⓪. In these diagrams, regions are coloured either green, red, or white depending on the feasibility of the selected parent using the Euclidean distance if the next point is added there. The boundaries of these regions are implicitly defined by a combination of non-holonomic constraints (cones with dotted lines), and the selection policy based on Euclidean distance (Voronoi boundaries with dashed lines). If the next point is added in the green region, a feasible parent is chosen leading to the addition of



(A) When tree has 2 nodes

(B) When tree has 3 nodes

FIGURE 5.1: An illustration of the advection reachability problem, which occurs when assigning parents using lowest Euclidean distance in strong advection dynamics. The correct behaviour occurs if a sample is added in green and white areas: a feasible parent is chosen, and no feasible parent can be chosen, respectively. The incorrect behaviour occurs if a sample is added in the red areas: an infeasible parent is chosen despite the existence of a feasible alternative.

the point. If the next point is added in the red region, an infeasible parent is chosen which prevents that point from being connected to the tree despite the availability of a different feasible parent. If the next point is added in the white region, no feasible parent can chosen. Note that the amount of green space has reduced from Fig. 5.1a to Fig. 5.1b, and it is not hard to believe that this pattern continues even if the flow is not uniform, as long as the speed is greater than the vehicle's propulsion speed.

A fundamental issue with using Euclidean distance to identify the parent is that it does not account for the inherent difficulty of traversing in a flow field. To alleviate this problem, we can leverage the lower speed bound from the streamline-based control theory since a larger value implies higher required effort from the vehicle.

In this chapter, we explore two approaches to identify more suitable parents for the sampled node. The first approach is to formulate an alternative distance measure that better aligns with the reachability between two points under the spatially varying advection effect. Examples of alternate distance measures in other applications include the use of weighted sub-components of the state [91], and the ∞-norm for planning the motion of manipulator arms [175]. We construct measures that combine the 2-norm (Euclidean distance) with flow flux between two points and a measure of the offsetting flow between two points.

The second approach involves filtering out candidate parents before selecting based on the lowest Euclidean distance. We propose a multi-stage filtering procedure that progressively

removes candidates only when more suitable candidates can be identified. The stages of filtering are based on the lower speed bound, and the relative positions of sample from the parents compared to local flow vectors.

We integrate these ideas within the RRT framework and show that it generates earlier feasible solutions, and has clearly faster convergence rates than the current best practice for kinematically constrained problems, that can be interpreted to be a result of more intentional tree expansion to the sampled points. This approach is also applicable to planners that address time-variant flows [176] and to many RRT variants, such as goal-biasing RRT [87], bidirectional RRT [177], informed RRT [125], and even techniques that are designed for problems without steering functions such as GBRRT/GABRRT [120].

We present a detailed experimental evaluation of our methods in comparison to various implementations of kinodynamic RRT in a simple scenario. The experimental scenario features Taylor-Green vortices [170] which has been used previously in the literature [8, 51]. Results show that our methods outperform the other approaches including the best practice when using RRT for kinematically constrained problems. We also analyse how the more relevant approaches perform in a more realistic flow field using predictions from the Australian Bureau of Meteorology (BOM) with . Paths are found in both directions between Sydney and Brisbane along the coast under the influence of the EAC. In this scenario, we find that our methods are able to produce solutions earlier in adversarial conditions which can be vital in systems relying on the anytime property of the algorithm.

The main contribution of this chapter is a novel set of procedures to aid streamline-based planning, and their evaluation in a sampling-based planning framework. The significance of this contribution is an improved capability of autonomous marine robots by enabling on-board planning with embedded computation, by allowing for fast replanning to respond to unexpected situations, and by finding high-quality paths over large distances.

The core problem of this chapter discussed in Sec. 5.2. The proposed tools for effective streamline-based path planning are described in Sec. 5.3. The ideas presented in this chapter is contrasted and compared extensively with existing techniques in Sec. 5.4. Finally, a summary for the chapter is provided in Sec. 5.5.

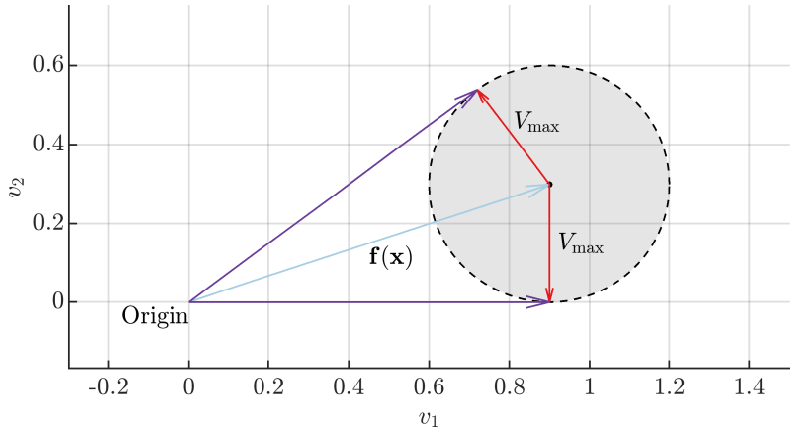


FIGURE 5.2: Velocity diagram when the instantaneous flow field speed (blue) is greater than the maximum vehicle propulsion speed (red). The possible net velocities based on the continuous model (4.1) is shown as a grey disc. At this instant, the vehicle's possible directions of motion are constrained to be between the two purple tangential velocities.

5.2 Problem formulation

5.2.1 A holonomic vehicle in a non-holonomic system

We assume the vehicle in a flow field has dynamics represented by the continuous-time transition model

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}) + \mathbf{v}. \tag{4.1 revisited}$$

where $\mathbf{x} = [x, y]^\intercal$ is vehicle position, $\mathbf{f}(\mathbf{x})$ is the time-independent flow field, and $\mathbf{v}$ is the propulsion velocity of relative to the local flow constrained by a maximum speed $V_{\max}$.

Despite modelling the vehicle as holonomic, i.e. it can instantaneously change its propulsion velocity, the overall system is non-holonomic under the advection of strong environment flow. In strong flow fields where the flow vector speeds can be higher than the vehicle's propulsion speed, i.e.

$$\exists \mathbf{x} \in \mathbb{X} \text{ s.t. } \|\mathbf{f}(\mathbf{x})\|_2 > \|\mathbf{v}\|_2, \tag{5.1}$$

where the directions of the vehicle motion are limited to certain headings.

The velocity diagram of the instantaneous modelled vehicle dynamics is shown in Fig. 5.2. The possible net velocities achievable by the vehicle when it is advected by the flow velocity $\mathbf{f}(\mathbf{x})$ in light blue is depicted as a partially transparent grey disc. The radius of the grey disc is defined by the maximum propulsion speed of the vehicle. The purple tangent velocities to the disc are net velocities that correspond to the limits of the vehicle's controllability, which are achievable by using the control velocities in red. The angle subtended by the two tangent velocities describe the vehicle's limited angles of travel referred to as the *accessibility cone* in existing work [58, 59], making the overall system non-holonomic.

5.2.2 Path planning in strong flow fields

We consider the waypoint selection problem (4.13) in a 2D time-invariant incompressible flow field for the vehicle using persistent controls to reach a given goal position. Sampling-based motion planners is a class of algorithms that addresses this problem by building a graph guided by sampling from the free configuration space. In particular, RRT builds a tree from which a path to a goal can be queried. Such algorithms rely on coverage of the configuration space to improve the solution availability of multi-query implementations, or to increase the probability of including configurations that are useful as a component of the optimal path.

Coverage can be quantified using the *dispersion* [11, 42, 178] of the graph nodes, which measures the radius of the largest ball not containing any of the nodes in the configuration space. In 2D, this can be quantified using the ℓ_2-dispersion which is visualised in Fig. 5.3 for a set of vertices $\mathsf{X}_{\text{graph}}$. The ℓ^2-dispersion is the radius δ of the largest empty circle in bounded search space $\mathbb{X} \subset \mathbb{R}^2$, i.e.

$$\delta(\mathsf{X}_{\text{graph}}) = \max_{\mathbf{x}_{\text{P}} \in \mathbb{X}} \left(\min_{\mathbf{x}_{\text{Q}} \in \mathsf{X}_{\text{graph}}} \|\mathbf{x}_{\text{Q}} - \mathbf{x}_{\text{P}}\|_2 \right). \tag{5.2}$$

Adding the sampled state directly to the search graph is more valuable for the overall algorithm because it would lead to a similar dispersion reduction rate of the chosen the sampling sequence, leading to solutions with lower path cost, and bounds to the cost [11]. To achieve this, two primitive functions are needed: 1) one that identifies the most suitable

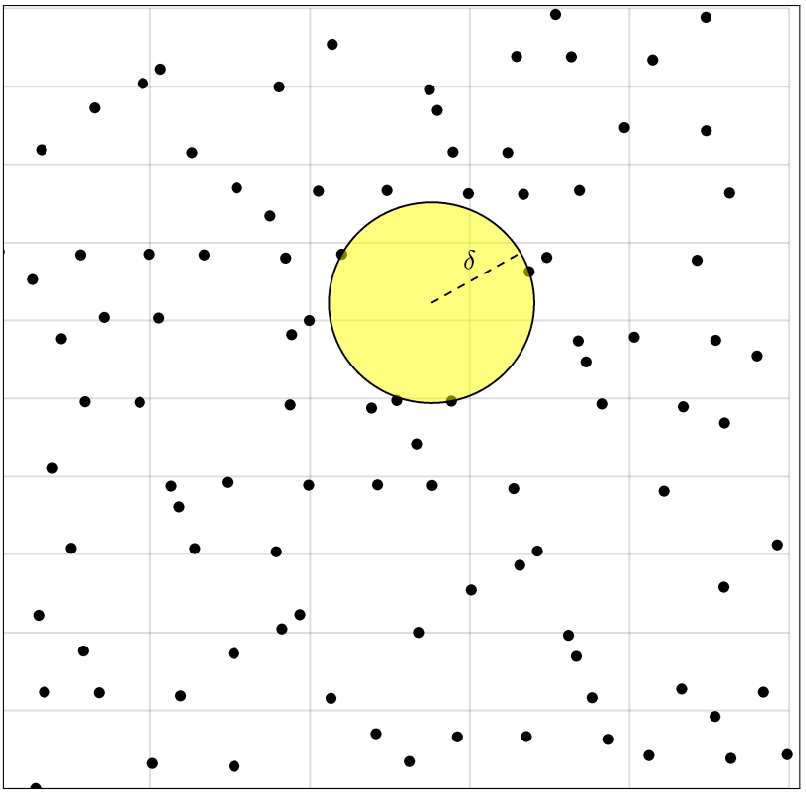


FIGURE 5.3: Visualisation of the definition of ℓ_2-dispersion δ for a set of points

tree node from which to connect to the sample, and 2) one that verifies that the system can reach the sampled node from the selected tree node in a computationally efficient manner.

5.3 Tree extension to sampled points in strong flow fields

In this section, we present our approaches for the two primitive functions `Nearest` and `Extend` in order to improve the chances of extending tree-based sampling search algorithms to the sampled points in order to improve search coverage in strong 2D incompressible flow. Under our philosophy, it is critical for the performance of the `Nearest` function to account for the way a new point is generated from `Extend`, so we will first present a fast `Extend` function based on our findings from Ch. 4 to primarily check the feasibility of tree extension directly to the sampled node in Sec. 5.3.1. Then in Sec. 5.3.2, we propose a `Nearest` function that uses a distance measure that integrates information from the underlying flow field to choose a suitable tree node from which to extend. In Sec. 5.3.3, we explore an alternate `Nearest` function to simply disregard unsuitable candidates based on

a process of multi-stage filtering before choosing the tree node based simply on Euclidean distance.

5.3.1 Adaptive optimistic steering

In the previous chapter, we considered a steering function to connect two points for the purpose of evaluating the cost of traversal between them. For this application, it is necessary to search for the optimal selection from a range of controls for the general cost function per queried pair of positions despite the reduced search space. However, for the purposes of determining whether it is possible to reach the sampled point from a given point, it is important to be only forward integrating one control to maintain a similar computation time as the standard kinodynamic extension technique [13].

Recall that the feasible controls to reach a given target $\mathbf{x}_\mathrm{Q}$ from point $\mathbf{x}_\mathrm{P}$ when constrained by a maximum speed V_max is given by

$$\begin{aligned} \mathbf{v}(\kappa) &= \mathbf{v}_\mathrm{base} + \kappa \hat{\mathbf{v}}_\mathrm{forward} \\ \mathbf{v}_\mathrm{base} &= \begin{bmatrix} \Delta x_2 \\ -\Delta x_1 \end{bmatrix} \cdot \frac{\psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right)}{\|\Delta \mathbf{x}\|_2^2} \\ \hat{\mathbf{v}}_\mathrm{forward} &= \frac{\Delta \mathbf{x}}{\|\Delta \mathbf{x}\|_2}, \end{aligned} \qquad \text{(4.21 revisited)}$$

and

$$\kappa \in \left[-\sqrt{V_\mathrm{max}^2 - \frac{\psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right)^2}{\|\Delta \mathbf{x}\|_2^2}}, \sqrt{V_\mathrm{max}^2 - \frac{\psi\left(\mathbf{x}_\mathrm{P}, \mathbf{x}_\mathrm{Q}\right)^2}{\|\Delta \mathbf{x}\|_2^2}} \right]. \qquad \text{(4.23 revisited)}$$

We choose the control that has the highest speed in the direction of the target, i.e. the upper bound of κ, which has the intuitive interpretation of using the most propulsion towards the target. This corresponds to $\mathbf{v}_\mathrm{B}$ in Fig. 4.6b.

In our steering-based `Extend` function called *adaptive optimistic steering (AOS)*, we also propose to use an adaptive integration horizon as a function of the spatial distance between the two points, so that the amount of computation adapts to an estimate of the travel time. Our previous work [174] proposed the use of forward integration based on a fixed distance. However, this prevents the use of higher-order numerical integration schemes

such as Adams-Bashforth method [165] instead of using the Euler method. Here, we achieve a similar effect by choosing a number of fixed duration time steps as a function of the spatial distance. More specifically,

$$\begin{aligned} T_{\text{est}}(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}) &= \frac{s \left\| \mathbf{x}_{\text{Q}} - \mathbf{x}_{\text{P}} \right\|_2}{V_{\max}} \\ n_{\text{step}} &= \left\lceil \frac{T_{\text{est}}(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}})}{\Delta t} \right\rceil, \end{aligned} \tag{5.3}$$

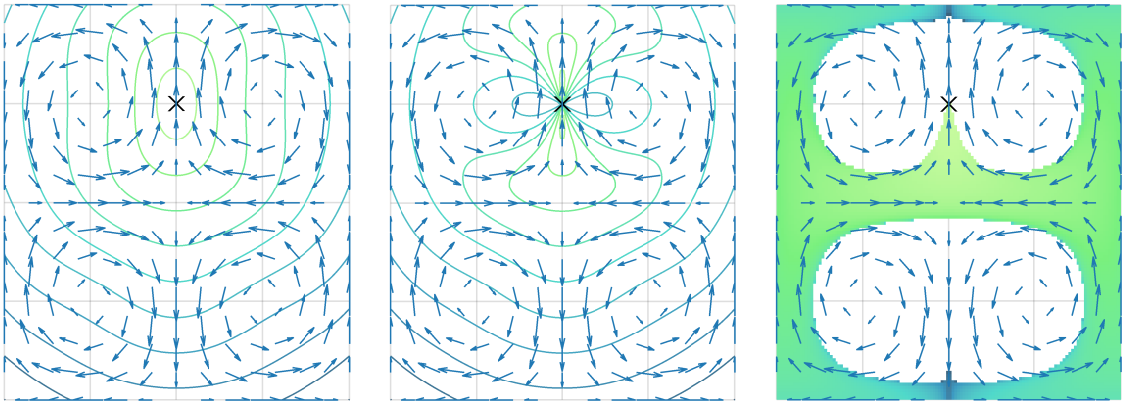
where $\lceil \cdot \rceil$ is the ceiling function, and s is a user-specified scaling factor. The given sample point is considered reachable if it is within some spatial threshold of any of the points from forward integration.

The approach varies the number of steps based on the query pair in RRT-based algorithms. This leads to committing more computational time to establish long edge connections in earlier stages of the algorithm, and will adapt to subsequent shorter query distances as the workspace is filled with points. When choosing a scaling factor s, it is important to consider that larger values will lead to more connections, however choosing a smaller value will lead to faster computation time. We find that choosing $s = \frac{\pi}{2}$ is a nice balance between the two ideas, which can be considered the time it takes for the vehicle to travel the semi-circle path between the two points at maximum speed without any flow field influence.

After integrating for the corresponding time horizon, the new point and the persistent control that is returned corresponds to the trajectory point that is closest to the sample point. It is not a huge impairment to the overall algorithm that this point is not spatially close to the sample point as adding this point will still lead to lower dispersion. In the case that the closest trajectory point is the first point, then the iteration is skipped.

5.3.2 Streamline-based distance measures

In this section, we consider alternate distance measures to identify the tree node that will most likely connect to the sampled node using the `Extend` function from Sec. 5.3.1. These measures are intended to quantify how reachable one point is from another, so the tree node is chosen based on the smallest measure to the sampled node.



(A) ψ-augmented 2-norm (B) V_{LSB}-augmented 2-norm (C) `Extend` time-to-go

FIGURE 5.4: Contours of distance measures to the point marked with a cross. The maximum flow strength is 4 times the vehicle's speed.

Typically Euclidean distance is used to quantify the reachability, however in strong flow fields it ignores the potentially powerful displacing effect of the environment on vehicle motion which limits the possible angles of motion. Before we compute the 2-norm, we propose to augment the spatial representation with a term dependent on the stream function, a well-defined function between any two points in 2D incompressible flow that describes the net flow flux between them. A scaling term is also added to the extra term in a fashion similar to previous work [91].

The first distance measure we propose is the *ψ-augmented 2-norm*, defined as

$$\texttt{dist}(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}}) = \sqrt{\|\mathbf{x}_{\text{Q}} - \mathbf{x}_{\text{P}}\|_2^2 + [\psi(\mathbf{x}_{\text{P}}, \mathbf{x}_{\text{Q}})/\alpha]^2}, \tag{5.4}$$

where α is a scaling term that can be interpreted as some characteristic velocity. This is equivalent to taking the 2-norm of all points $\mathbf{x} \in \mathbb{X}$ embedded in three dimensions (x_1, x_2, ψ) as a surface relative to some reference point. As a result, this distance measure is a metric satisfying the four necessary conditions [42]: non-negativity, reflexivity, symmetry, and the triangle inequality through the 2-norm. The ψ-augmented 2-norm penalises paths that cross streamlines; its level sets are therefore elongated along the flow direction. Contours of the ψ-augmented 2-norm are illustrated in Fig. 5.4a.

The second distance measure we propose is the V_{LSB}-*augmented* 2-*norm*, formulated using the lower speed bound (LSB) from (4.22) as:

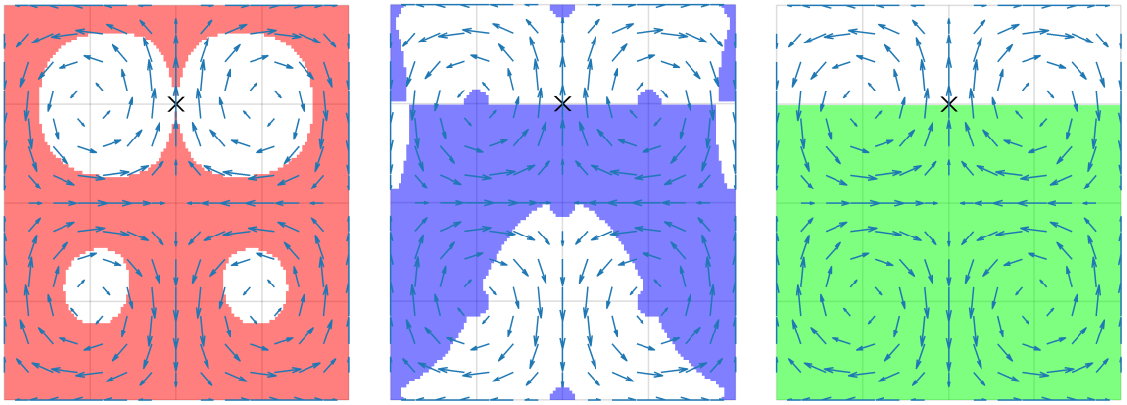
$$\texttt{dist}(\mathbf{x}_{\mathrm{P}}, \mathbf{x}_{\mathrm{Q}}) = \sqrt{\|\mathbf{x}_{\mathrm{Q}} - \mathbf{x}_{\mathrm{P}}\|_2^2 + [V_{\mathrm{LSB}}(\mathbf{x}_{\mathrm{P}}, \mathbf{x}_{\mathrm{Q}})\beta]^2}, \tag{5.5}$$

where β is a scaling term that can be interpreted as some characteristic time. The V_{LSB} term quantifies a lower bound on the speed needed for a feasible path between the two points, which is used as a way to quantify the difficulty of traversal in this context. This component is dominant in the proximity of sample point, but becomes dominated by the normal spatial components over larger spatial distances. Note that the V_{LSB}-augmented 2-norm does not satisfy the triangle inequality and is therefore not a metric.

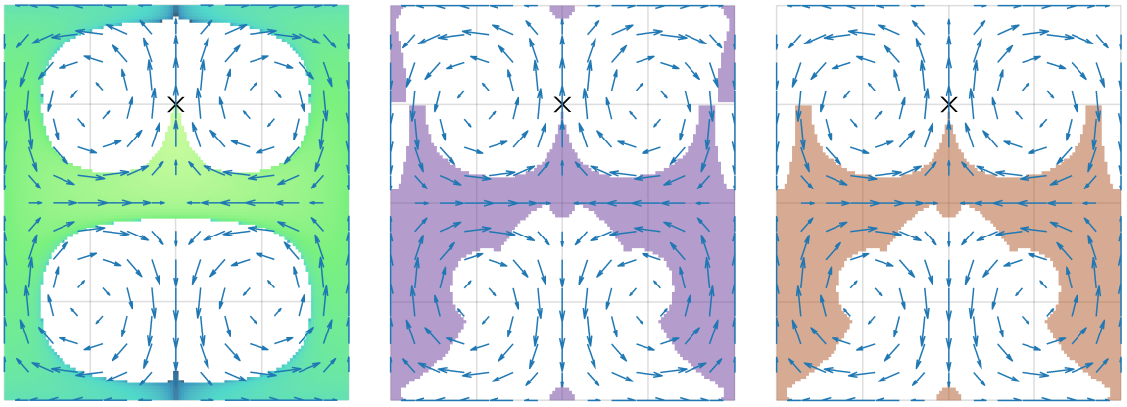
However, it is believed that the ideal distance measure is not a metric [91]. The ideal measure corresponds to the cost-to-go, which is generally asymmetric for kinematically constrained problems. In our case, the ideal measure should favour the selection of tree nodes that is upstream of the sampled point under strong flow. A visualisation of the cost based on the control selection strategy is provided in Fig. 5.4c for the `Extend` function described in Sec. 5.3.1, i.e. the cost to traverse from multiple locations to the same position using one optimistically-chosen persistent control. Despite its symmetry, the V_{LSB}-augmented 2-norm align better with the limits of reachability than the ψ-augmented 2-norm when comparing contours of the V_{LSB}-augmented 2-norm in Fig. 5.4b and the boundary of the region associated with low time cost from a single persistent control.

5.3.3 Flow-based multi-stage filter

In this section, we explore a different strategy to identify the tree node that is more likely to connect to the sampled node in the RRT framework. The idea is to progressively reduce the pool of candidates based on three filters based on system dynamics, then finally choose the candidate corresponding with the lowest Euclidean distance. We will first introduce the filters, then we will describe how they are used together as a multi-stage filter. Throughout this section we use the same scenario featured in Fig. 5.4 for the examples, and have reproduced Fig. 5.4c as Fig. 5.5d to facilitate comparison with the filtered regions.



(A) LSB filter (B) Initial flow filter (C) Final flow filter



(D) `Extend` time-to-go (E) Double filtered (F) Triple filtered

FIGURE 5.5: Comparison of regions from different stages of the proposed filter to reach the point marked with a cross. The maximum flow strength is 4 times the vehicle's speed.

The first filter shown in Fig. 5.5a called the *LSB filter*, is based on the streamline constraint

$$V_{\max} \geq V_{\mathrm{LSB}}\left(\mathbf{x}_{\mathrm{P}}, \mathbf{x}_{\mathrm{Q}}\right), \qquad \text{(4.24 revisited)}$$

from the previous chapter. When not satisfied, it has been shown that the vehicle would have insufficient speed to connect the points $\mathbf{x}_{\mathrm{P}}$ and $\mathbf{x}_{\mathrm{Q}}$, i.e. the uncoloured regions in Fig. 5.5a are guaranteed to not reach the target with one persistent control. Note that Fig. 5.5d includes some points that are not coloured in Fig. 5.5a, however they are attributed to numerical inaccuracies arising from integration error, and the spatial threshold at which a trajectory is considered to have reached the target.

The other filters in Figs. 5.5b and 5.5c are *local flow consistency filters*, based on the

Algorithm 5.1 Filter-based `Nearest` function

$\mathbf{x}_q$: The query point
C_0: A set of candidate points for consideration

function `Nearest`($\mathbf{x}_q$,C_0)
  $\mathsf{C}_1 = \{\mathbf{x} \in \mathsf{C}_0 : V_{\max} \leq V_{\text{LSB}}\}$
  if $|\mathsf{C}_1| == 0$ **then**
    return $\arg\min_{\mathbf{x}\in\mathsf{C}_0} \|\mathbf{x}_q - \mathbf{x}\|_2$ ▷ Fallback
  end if
  $\mathsf{C}_2 = \{\mathbf{x} \in \mathsf{C}_1 : \|\mathbf{f}(\mathbf{x})\|_2 < V_{\max} \vee \mathbf{f}(\mathbf{x}) \cdot (\mathbf{x}_q - \mathbf{x}) > 0\}$
  if $|\mathsf{C}_2| == 0$ **then**
    return $\arg\min_{\mathbf{x}\in\mathsf{C}_1} \|\mathbf{x}_q - \mathbf{x}\|_2$ ▷ Single filter only
  end if
  if $\|\mathbf{f}(\mathbf{x}_q)\|_2 < V_{\max}$ **then**
    $\mathsf{C}_3 = \mathsf{C}_2$
  else
    $\mathsf{C}_3 = \{\mathbf{x} \in \mathsf{C}_2 : \mathbf{f}(\mathbf{x}_q) \cdot (\mathbf{x}_q - \mathbf{x}) > 0\}$
    if $|\mathsf{C}_3| == 0$ **then**
      return $\arg\min_{\mathbf{x}\in\mathsf{C}_2} \|\mathbf{x}_q - \mathbf{x}\|_2$ ▷ Double filtered
    end if
  end if
  return $\arg\min_{\mathbf{x}\in\mathsf{C}_3} \|\mathbf{x}_q - \mathbf{x}\|_2$ ▷ Triple filtered
end function

direction of the target relative from the query point, and the direction of the flow velocity at either the query point, or the target point. More specifically, a point $\mathbf{x}_\text{P}$ passes the filter for the target $\mathbf{x}_\text{Q}$ if

$$V_{\max} > \|\mathbf{f}(\mathbf{x}_\text{P})\|_2 \vee \mathbf{f}(\mathbf{x}_\text{P}) \cdot (\mathbf{x}_\text{Q} - \mathbf{x}_\text{P}) > 0, \tag{5.6}$$

for the *initial flow filter*, and

$$V_{\max} > \|\mathbf{f}(\mathbf{x}_\text{Q})\|_2 \vee \mathbf{f}(\mathbf{x}_\text{Q}) \cdot (\mathbf{x}_\text{Q} - \mathbf{x}_\text{P}) > 0, \tag{5.7}$$

for the *final flow filter*. These filters disregard query positions that are inconsistent with the strong flow velocities of the environment which are unlikely to be positions that can reach the target position, especially for pairs of positions that are spatially near. Note that the circular regions that pass the initial flow filter in Fig. 5.5b are the result of the magnitude of the flow velocities being less than the vehicle's maximum speed.

To increase the likelihood of providing a tree node that the vehicle could be to reach the

sampled position, the proposed multi-stage filter progressively reduces the set of feasible candidates until the end of the sequence of filters, or until the application of the next filter would eliminate all candidates. This means that the order of the applied filters affect the performance. The LSB filter is applied first since any candidates it rejects are guaranteed to not be feasible theoretically. The initial and final flow filters are applied after as they are heuristic in nature, especially for large spatial distances. After filtering, the best candidate is selected as the one that has the lowest Euclidean distance to the query point. This is to make the assumptions in the two heuristic filters stronger, and also to reduce the computation time needed for AOS proposed in Sec. 5.3.1.

Pseudocode for the full multi-stage filter is presented in Alg. 5.1. The candidates after each filtering stage can be seen in Fig. 5.5a, 5.5e, and 5.5f. Whilst these regions do not closely confirm to the region in Fig. 5.5d, it removes a lot of areas known to be infeasible compared to the approaches based on alternate distance measures.

5.3.4 Implementation with fast neighbour query data structures

Experimentally, we find that data structures that have low computation time complexity ($\mathcal{O}(\log|\mathsf{X}_{\text{graph}}|)$) queries for nearest neighbour or k-nearest neighbours are not as fast as the exhaustive search for the scope of our computational parameters. This is primarily because the routines on tree-based data structures such as k-d trees [179], R* trees [180], and balanced-box decomposition (BBD) trees [95] are not as parallelisable as the exhaustive search. For the rest of this chapter, our implementation of the two proposed `Nearest` considers all the tree nodes to identify the most suitable parent for the sampled point.

If a lower computational complexity is required, we suggest searching over points in the neighbourhood of the query point identified by the 2-norm or the ψ-augmented 2-norm as described in our preliminary work [174]. The number of neighbours k can be based on $k_{\text{RRG}}\log(|\mathsf{X}_{\text{graph}}|)$, the number of neighbours considered for asymptotically optimal sampling-based planners [10].

Algorithm 5.2 Kinematic rapidly exploring random tree algorithm

```
1: function BuildRRT(planner)
2:     while NotTerminal(planner) do
3:         sample = SampleFree()
4:         AddSample(planner, sample)
5:     end while
6: end function
7:
8: function AddSample(planner, sample)
9:     nearest = Nearest(sample, GetNodes(planner))
10:    (new, ctrls) = Extend(nearest, sample)
11:    if CollisionFree(nearest, new) then
12:        ConnectSample(planner, nearest, new, ctrls)
13:    end if
14: end function
```

5.4 Experiments

In this section, we empirically compare our proposed approach against various techniques from literature, and variants thereof. We first provide the algorithmic implementation details used for the experiment, including the variations of implementation that were needed for our particular problem. Then we perform a thorough experiment in an environment with Taylor-Green vortices [170] to establish relative performance. Finally we study how the different approaches perform in a simulated environment based on a real oceanic flow prediction from the Australian BOM to demonstrate the implications of performance differences for a marine robotics application.

5.4.1 Algorithmic implementation

5.4.1.1 Implementation of overall algorithm

Our implementation is based on the kinodynamic RRT [91], extended as an RRT* algorithm for kinematically constrained systems, the pseudocode of which is provided in Alg. 5.2 and 5.3. We will briefly describe some implementation details below which were used for the experiments in this section.

Algorithm 5.3 `ConnectSample` for RRT*

```
1: function ConnectSample(planner, nearest, new, ctrls)
2:     parent = nearest
3:     costToNew = CostFromRoot(planner, nearest) + Cost(nearest, new, ctrls)
4:     near = Near(new, GetNodes(planner))
5:     for all n in near do
6:         if CollisionFree(n, new) then
7:             ctrls_n = Steer(n, new)
8:             if Valid(ctrls_n) then
9:                 cost = CostFromRoot(planner, n) + Cost(n, new, ctrls_n)
10:                if cost < costToNew then
11:                    parent = n
12:                    costToNew = cost
13:                end if
14:            end if
15:        end if
16:    end for
17:    Add(planner, parent, new, ctrls, costToNew)
18:    for all n in near do
19:        if CollisionFree(new, n) then
20:            ctrls_n = Steer(new, n)
21:            if Valid(ctrls_n) then
22:                cost = costToNew + Cost(new, n, ctrls_n)
23:                if cost < CostFromRoot(planner, n) then
24:                    Reparent(planner, n, new, ctrls_n, cost)
25:                end if
26:            end if
27:        end if
28:    end for
29: end function
```

`SampleFree` Uses rejection sampling from a deterministic sampling sequence. More specifically, the samples are obtained from a Halton sampling sequence with a random amount of rotation and offset for a set of samples with lower dispersion compared to uniform random sampling [11].

`CollisionFree` Simply checks whether the second argument is within the workspace. Other collision checking capabilities are handled by `SampleFree` and `Steer`.

`Near` Implemented by providing the k-nearest neighbours to the given node based on Euclidean distance, where k chosen such that the overall algorithm is asymptotically optimal [10].

TABLE 5.1: Summary of the different classes of approaches in experiments

Approach class	`Nearest`	`Extend`
NGEO	min 2-norm	Δt step
VF	min 2-norm	VF-RRT with Δt step
RCRD	min 2-norm	random velocity over random duration
EUCN	min 2-norm	adaptive optimistic steering
LSBN	min V_{LSB}-augmented 2-norm	adaptive optimistic steering
FILT	filtered 2-norm	adaptive optimistic steering

`Cost` Specified to return the time cost to execute the valid transit from one point to another so that the overall algorithm will return a time-optimal path.

`Steer` Returns the persistent control that takes the vehicle from one point to another. Only seven velocities are considered on the interval described by (4.21) and (4.23), six of which are randomly sampled, and the last corresponding to the optimistic velocity. When there are multiple solutions, the control that corresponds to the least time cost is chosen in accordance with `Cost`. The integration duration is chosen in a similar way to the `Extend` function described in Sec. 5.3.

Using Euclidean distance is still suitable for the `Near` function as its purpose in the algorithm is to consider alternate routes geometrically, rather than identifying the node that maximises reachability. This is reinforced by the proof presented in [10]. Future work could explore reducing the number of considered neighbours due to the reduced reachable areas based on local flow conditions.

The maximum propulsion speed of the vehicle $V_{\max}$ in this section is 0.3 m/s.

5.4.1.2 Implementation of compared approaches

The dependent variable for the experiments is the different implementations of the `Nearest` and `Extend` functions summarised in Tab. 5.1. We primarily compare our proposed methods against the state-of-the-art in path planning for general kinodynamic systems, but we also include other approaches for completeness whether or not they are probabilistically complete. We will use the abbreviations NGEO, VF, RCRD, EUCN, LSBN, and FILT to

refer to the different classes of approaches which are described below in the rough order of appearance in literature.

The NGeo approach is a naïve geometric approach that tries to emulate the growth of a geometric RRT [10]. The nearest node is selected using the lowest Euclidean distance to the sample, and a new node is created towards the sampled node using a feasible velocity for a duration up to the maximum allowable integration limit Δt. In the case where the sample is in a direction that is unreachable, the algorithm skips that iteration. The approach is known to be not probabilistically complete [96], however it provides as a good reference point for other approaches.

The VF approach is our adaptation of vector field RRT (VF-RRT) [71] for the non-holonomic system arising from strong flow fields. In the algorithm, an `Extend` function is proposed that aligns the position of the extended node more to the local flow field when the tree is effectively extending into new regions of the search space. There are a few parameters needed for this approach, the most significant of which is E_s which can be interpreted as the *exploration inefficiency tolerance factor*. We emphasise that the algorithm was not designed for strong flow fields, so when there is an attempt to add node where it is kinematically infeasible, the iteration is skipped.

The RCRD approach uses an `Extend` function that generates a new node by propagating the state of the tree node with a random control for a random duration. In our case, this corresponds to corresponds to forward integration of the system dynamics from the tree node after randomly selecting a velocity from the control space for a duration that is uniformly random sampled from the interval $[0, T_{\max}]$. This approach is the state-of-the-art for general kinodynamic problems, and has been shown to be probabilistically complete [13, 181]; however it makes no attempt at reaching the sampled point and can therefore spend more computational resources to explore the same search space.

The EucN approach considers the use of the `Extend` function proposed in Sec. 5.3, but chooses the nearest tree node using the minimum Euclidean distance. This approach highlights the consequences of the advection reachability problem illustrated in Fig. 5.1.

Finally, our the distance-based approach LSBN described in Sec. 5.3.2 and the filter-based approach FILT described in Sec. 5.3.3 also uses the `Extend` function proposed in Sec. 5.3.

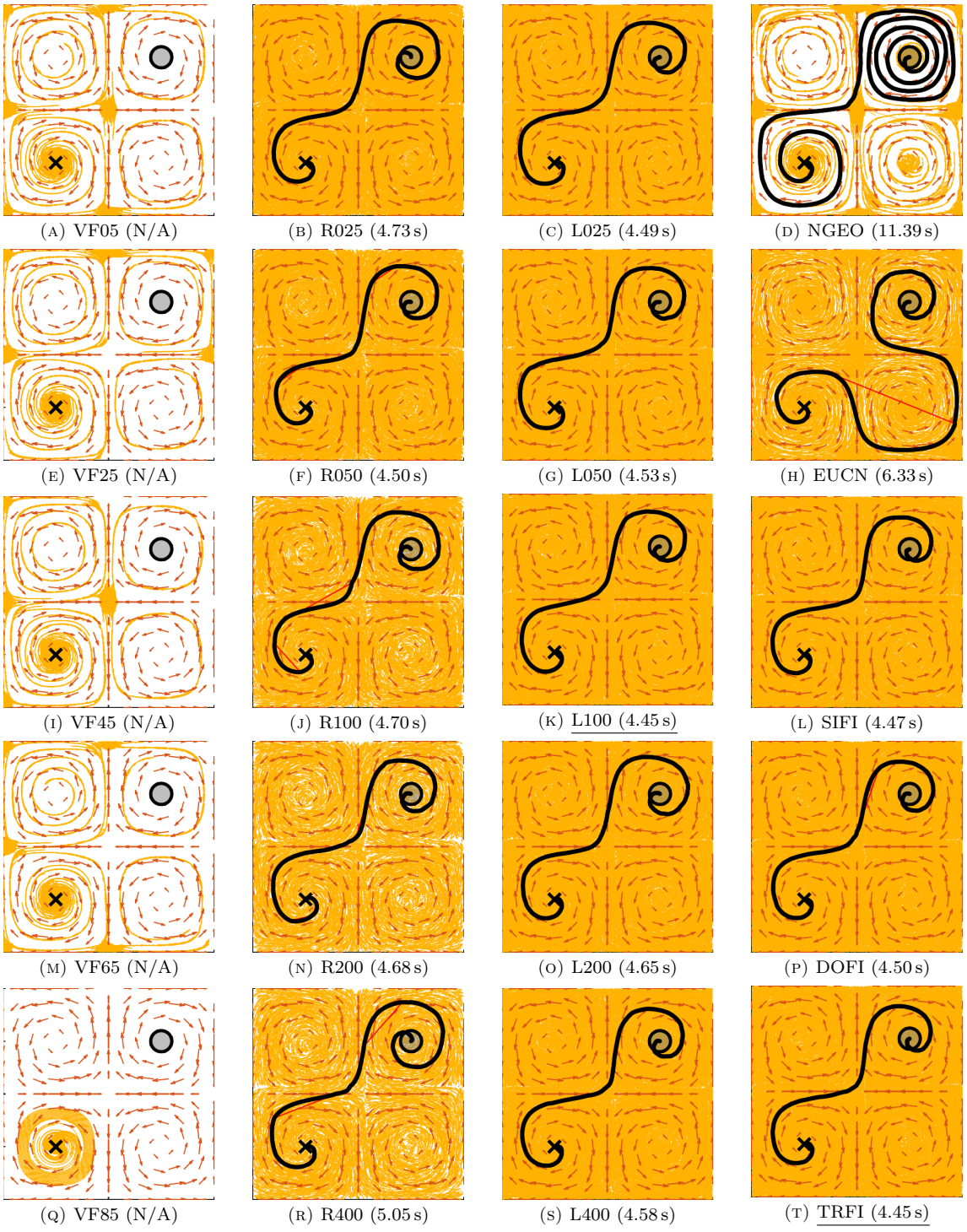
In our experiments, we include small variations for some of these classes of approaches. For VF, we use the same range of values for the exploration inefficiency tolerance factor E_s that were used the original paper. For RCRD, we use a range of values for the maximum random duration $T_{\max}$. For LSBN, we use a range of values for the characteristic time β. And finally, for FILT, we use different number of filters: single filter (SIFI) refers to just the use of LSB filter, double filter (DOFI) refers to LSB filter and initial flow filter, and triple filter (TRFI) refers to the full multi-stage filter. Note that EUCN can be considered the implementation of FILT without any filters.

5.4.2 Taylor-Green vortices

Extensive computations were performed on Taylor-Green vortices [170] which was used in the illustrative examples in Figs. 5.4 and 5.5. The environment consists of four vortices that rotate in opposite directions with a maximum speed that is four times the vehicle's maximum speed, except for the regions roughly indicated by the repeated pattern of circles in Fig. 5.5b. For this environment, the maximum tolerable integration time step is 0.01 s for the workspace size of $2 \times 2\,\mathrm{m}^2$.

For this scenario, the different variations of the compared approaches are listed in Tab. 5.2. The results over 100 runs each using a maximum of one hour on a single core of a 2.4 GHz AMD EPYC 7532 processor for each implementation are tracked to capture the overall behaviour as a mean. The repeated runs are necessary due to the randomness introduced by the sampling process of after randomly rotating and offsetting the low-dispersion deterministic sequence, the sampling process in `Steer` that is used in Alg. 5.3, and the random sampling for RCRD approaches.

An example final solution for each implementation is shown in Fig. 5.6 along with its cost. The final trajectories for many of these implementations align with our expectations due to their rough symmetry in a flow field with regular currents. The NGEO approach generates a graph very similar to the VF approach since they both attempt to grow the tree in



(A) VF05 (N/A) (B) R025 (4.73 s) (C) L025 (4.49 s) (D) NGEO (11.39 s)

(E) VF25 (N/A) (F) R050 (4.50 s) (G) L050 (4.53 s) (H) EUCN (6.33 s)

(I) VF45 (N/A) (J) R100 (4.70 s) (K) L100 (4.45 s) (L) SIFI (4.47 s)

(M) VF65 (N/A) (N) R200 (4.68 s) (O) L200 (4.65 s) (P) DOFI (4.50 s)

(Q) VF85 (N/A) (R) R400 (5.05 s) (S) L400 (4.58 s) (T) TRFI (4.45 s)

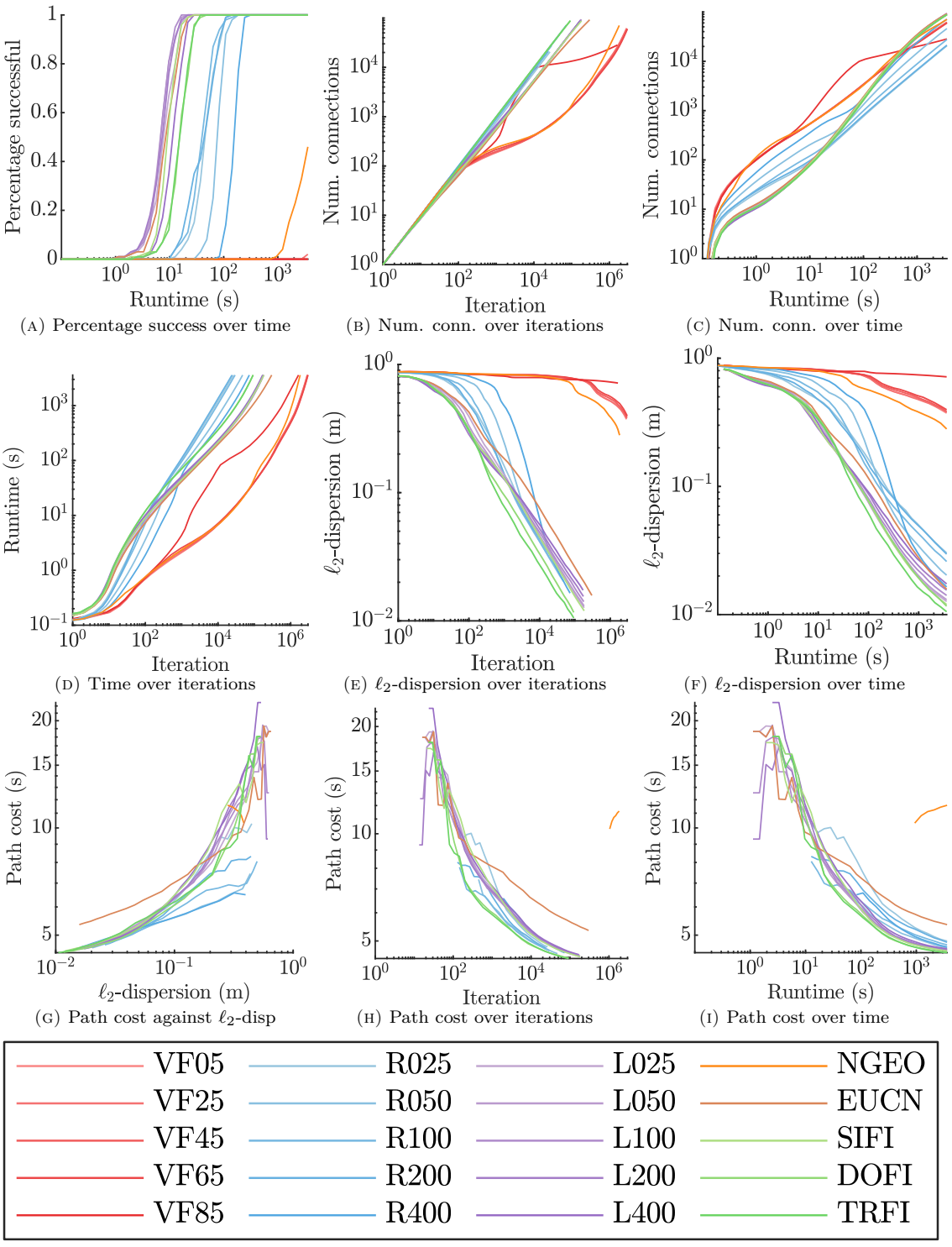
FIGURE 5.6: Example of the final path from different approaches and their execution cost for planning in Taylor-Green vortices. The path (black) is the best shortest-time solution after one hour of computation from the start (cross) to the goal (circular region). RRT edges are shown in orange and those that are part of the final path are shown in red.

TABLE 5.2: Details of compared approaches in Taylor-Green vortices

Label	Approach class	Additional comments
NGEO	NGEO	—
VF05	VF	$E_s = 0.05$
VF25	VF	$E_s = 0.25$
VF45	VF	$E_s = 0.45$
VF65	VF	$E_s = 0.65$
VF85	VF	$E_s = 0.85$
R025	RCRD	$T_{\max} = 0.25\,\mathrm{s}$
R050	RCRD	$T_{\max} = 0.50\,\mathrm{s}$
R100	RCRD	$T_{\max} = 1.00\,\mathrm{s}$
R200	RCRD	$T_{\max} = 2.00\,\mathrm{s}$
R400	RCRD	$T_{\max} = 4.00\,\mathrm{s}$
EUCN	EUCN	—
L025	LSBN	$\beta = 0.25$
L050	LSBN	$\beta = 0.50$
L100	LSBN	$\beta = 1.00$
L200	LSBN	$\beta = 2.00$
L400	LSBN	$\beta = 4.00$
SIFI	FILT	LSB filter
DOFI	FILT	LSB and init flow filters
TRFI	FILT	LSB, init flow, and goal flow filters

a geometric way. In this scenario, the NGEO appears to outperform the VF approach because it focuses more on exploring the space, rather than following the flow field which is not as productive in vortices. The EUCN approach is not as affected by this problem because AOS still extends the tree despite integrated trajectory points not being spatially close the sample. All methods using AOS benefit from the tree extension by construction in this way. All other methods produce similar paths with similar costs suggesting that they have converged to the optimal path. Among these other methods, it can also be observed that there is more space between the tree edges of RCRD implementations indicating poorer coverage of space, and is worse for higher choices of $T_{\max}$.

A set of graphs is shown in Fig. 5.7 which aggregates the result for each implementation by taking the mean across available values for the 100 random runs. From Fig. 5.7a, it can be seen that the ranking for early first available solution is roughly ordered as LSBN, EUCN, FILT, RCRD, NGEO, and VF for this scenario. The ranking is influenced by factors including effectiveness of extending the tree, the computational simplicity, and effective coverage of the workspace.



(A) Percentage success over time
(B) Num. conn. over iterations
(C) Num. conn. over time
(D) Time over iterations
(E) ℓ_2-dispersion over iterations
(F) ℓ_2-dispersion over time
(G) Path cost against ℓ_2-disp
(H) Path cost over iterations
(I) Path cost over time

FIGURE 5.7: Aggregated measurements tracked over the computation time for the 100 runs of each different approach in the Taylor-Green scenario. Note that each point on a path cost curve is computed across the number of available solutions for each implementation resulting in a curve that is not necessarily monotonically decreasing.

Figure 5.7d shows the computation time over the iterations of the algorithm. Implementations that use AOS takes longer than RCRD methods, however after about 60 s of computation, AOS approaches take less time to iterate. This *accelerating* property can be attributed to the approach adapting the amount of computation time committed to tree extension by adjusting the integration horizon, whereas the RCRD approach have a consistent expected amount of integration steps. The approaches NGeo and VF are faster than these techniques as expected as they only take a single discrete step per iteration.

Figures 5.7b and 5.7c show how new connections of each implementation are established over the algorithm's progress. In Fig. 5.7b, RCRD implementations as well as DOFI and TRFI are shown to have nearly 100% connection rate. For RCRD approaches, this is due to the fact that it extends the tree by construction. In contrast, AOS approaches relies on a suitable tree node chosen by the `Nearest` function. The AOS extension strategy also partially extends the tree by construction, however it fails when the closest point after forward integrations is still the starting point. Since DOFI and TRFI implementations identify a suitable tree node by considering the local flow direction, it is able to match the connection rate of RCRD implementations. When AOS acceleration is taken into account, the implementing methods can be seen in Fig. 5.7c to eventually establish connections faster than the RCRD methods.

Figures 5.7e and 5.7f show how the ℓ_2-dispersion of the tree nodes for each implementation over the algorithm's progress. It is interesting to see how different implementations of the `Nearest` function with EucN, LSBN, and Filt leads to a spread of dispersion in Fig. 5.7e. This highlights the importance of finding a feasible tree node that can connect to the sample node. The almost imperceptible difference in connection rate between TRFI and RCRD implementations seen in Fig. 5.7b can lead to larger differences in dispersion reduction seen in Fig. 5.7e. The RCRD implementations initially start with higher dispersion than implementations using AOS, however since they are able to extend the tree more often per iteration than SIFI, EucN, and LSBN approaches, they are able to reduce dispersion at a higher rate. However, AOS acceleration makes most of these techniques to reduce dispersion faster than the RCRD approaches in Fig. 5.7f. Another reason for higher dispersion in RCRD tree nodes is due to the addition of nodes that are unevenly

spread across the workspace since it is influenced by a limited integration horizon, and the growth of the tree.

Finally Figs. 5.7h and 5.7i show the path cost of the optimal path that the implementations provide over the algorithm's progress. In comparison with the dispersion graphs, we see that reductions in dispersion correlate to reductions in path cost. An exception to this pattern is the EucN method, which appears to be to reduce dispersion but it does not appear to lead to a reduced path cost. This outlier is more distinct in Fig. 5.7g, which shows how the dispersion of the tree nodes correlate to the best path cost obtainable from the algorithm. A potential explanation for this outlier could be due to a combination of an inconsistent coverage of space and the rewiring mechanism in Alg. 5.3 which is designed to improve the query path cost. The EucN method has difficulty adding nodes near the samples in certain places than others due to the advection reachability problem, and since the rewiring process only occurs with neighbouring nodes, the tree is optimised in a non-uniform manner, leading to sub-optimal tree structure.

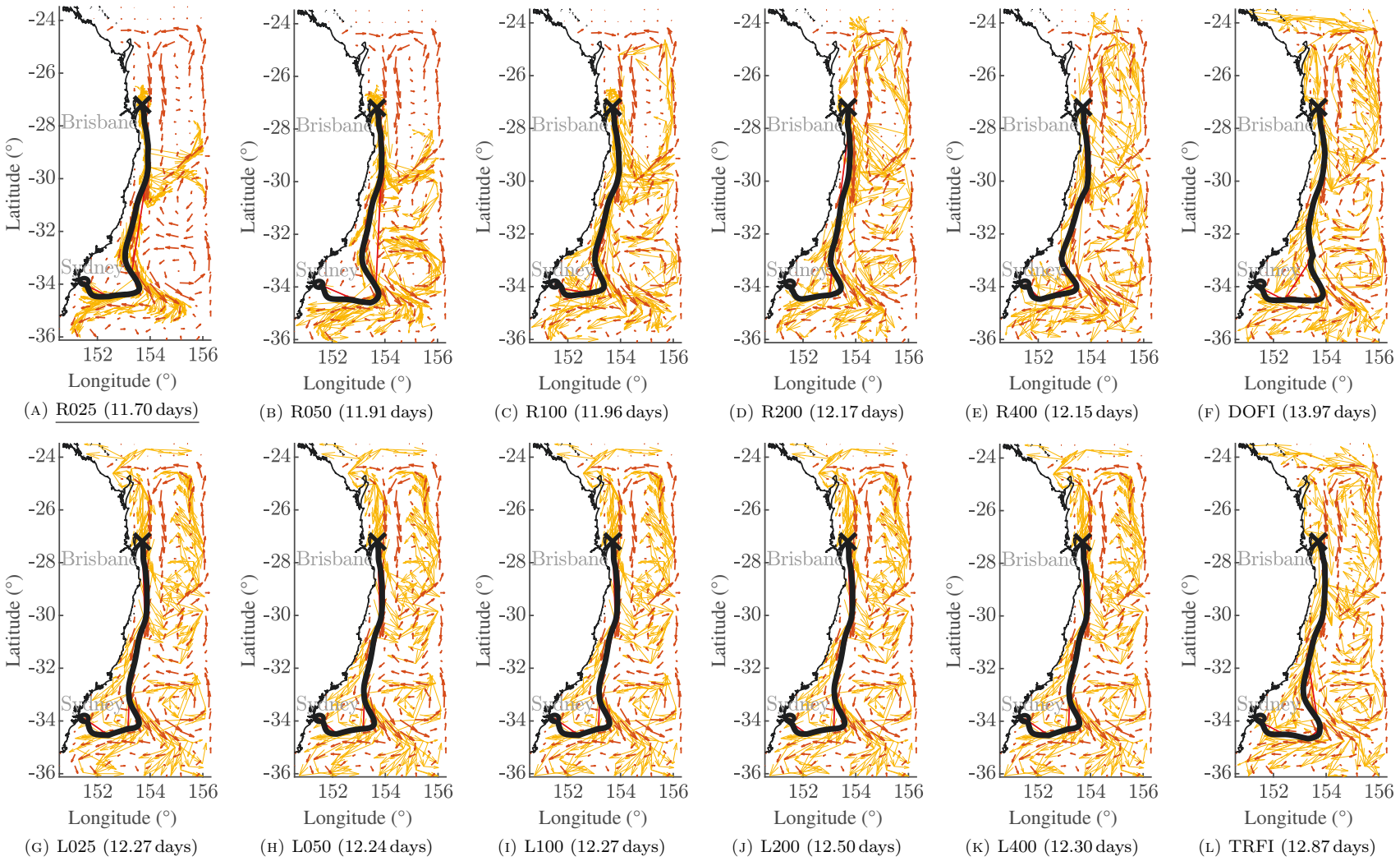
From the results of this experiment we see VF approaches performing worse than NGeo, an approach already known to perform poorly since it is not probabilistically complete [96]. The VF approach biases the growth of the tree to the flow field, but this strategy's effectiveness is very dependent on how downstream the goal is from the initial position. In this scenario, the goal is not downstream nor upstream of the start which makes it difficult for the tree to grow in the correct way. The EucN approach appears to produce suitable search trees in Fig. 5.6h, but it appears to perform inconsistently in comparison to others in Figs. 5.7g to 5.7i making it even less desirable for use in real applications. This experiment indicates that the RCRD, LSBN, and Filt approaches are the most promising. It highlights a need to choose a suitable value for $T_{\max}$ for the RCRD approach. A lower max duration leads to better path cost convergence, however it corresponds to slower coverage of the space. On the other hand, a higher max duration leads to faster coverage of space, but leads to worse path cost convergence. For the LSBN approach, a higher selection of β leads to better dispersion reduction up until the 20 s mark, but then worse after that. This suggests that it is important to consider the LSB when the points are spaced far apart, but less important when points are closer together. A lower selection of β appears to produce slightly better results though. As a whole, LSBN approaches

produce earlier solutions in this scenario, most likely due to a combination of effective tree node selection and simpler computation to select the parenting tree node. The FILT approaches also produce early solutions and also converges the fastest. It is interesting to note that DOFI ends with higher dispersion than TRFI, but has similar path cost. This indicates that the TRFI approach is more effective at covering the workspace with nodes, however those additional nodes weren't necessary in the formation of the best path. An inherent benefit of FILT-based approaches is that it is able to perform well without the need to specify or tune a parameter such as β for the LSBN approach, or $T_{\max}$ for the RCRD approach.

5.4.3 East Australian Current

We now compare the performance of LSBN and the FILT approaches DOFI and TRFI against the RCRD approach in a more realistic flow field featuring the strong southward current of the EAC based on predictions from the Australian BOM for the top layer of forecasted currents on 16^{th} of November 2018. The continuous flow field $\mathbf{f}(\mathbf{x})$ and its stream function $\psi(\mathbf{x})$ is computed by Gaussian regression using the incompressible kernel [38] using lengthscales $l_1 = l_2 = 32.5 \times 10^3\,\text{m}$ and σ_{ker} heuristically picked to be $\sigma_{\text{ker}} = \mu/l_1$, where $\mu = 0.43\,\text{m/s}$ is the average flow vector magnitude. The simulated environment features varying flow speeds reaching up to a maximum of $1.49\,\text{m/s}$ in contrast to the vehicle's speed of $0.3\,\text{m/s}$. For this environment, the maximum tolerable integration time step is $1\,\text{hr}$.

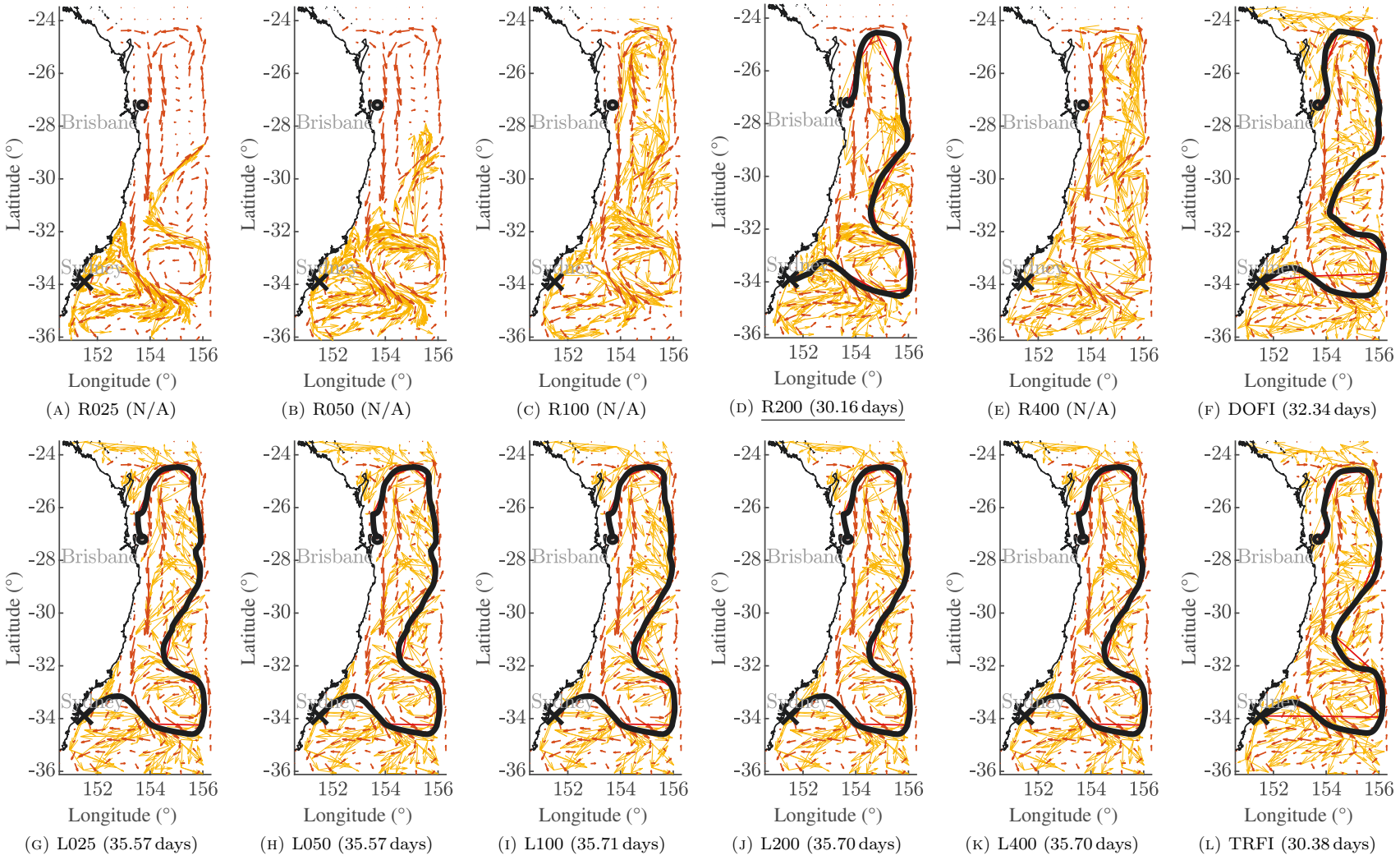
The experiment will also be more computationally limited than in the previous experiment to a $60\,\text{s}$ computational time budget reflecting realistic computational constraints of onboard processing. In this limited period of computation time, AOS methods are slower to iterate than RCRD methods. Paths are generated in both directions between Sydney and Brisbane to show how the approaches handle assistive and adversarial currents. The same parameters as listed in Tab. 5.2 are used, except for RCRD methods where $T_{\max}$ is multiplied by 3600, i.e. scaling the max duration to hours instead of seconds.



(A) R025 (11.70 days) (B) R050 (11.91 days) (C) R100 (11.96 days) (D) R200 (12.17 days) (E) R400 (12.15 days) (F) DOFI (13.97 days)

(G) L025 (12.27 days) (H) L050 (12.24 days) (I) L100 (12.27 days) (J) L200 (12.50 days) (K) L400 (12.30 days) (L) TRFI (12.87 days)

FIGURE 5.8: Example of the final path from different implementations and their execution cost for planning one minute from Brisbane to Sydney. The RRT edges are shown in orange and those that part of the final path are shown in red.



(A) R025 (N/A) (B) R050 (N/A) (C) R100 (N/A) (D) R200 (30.16 days) (E) R400 (N/A) (F) DOFI (32.34 days)

(G) L025 (35.57 days) (H) L050 (35.57 days) (I) L100 (35.71 days) (J) L200 (35.70 days) (K) L400 (35.70 days) (L) TRFI (30.38 days)

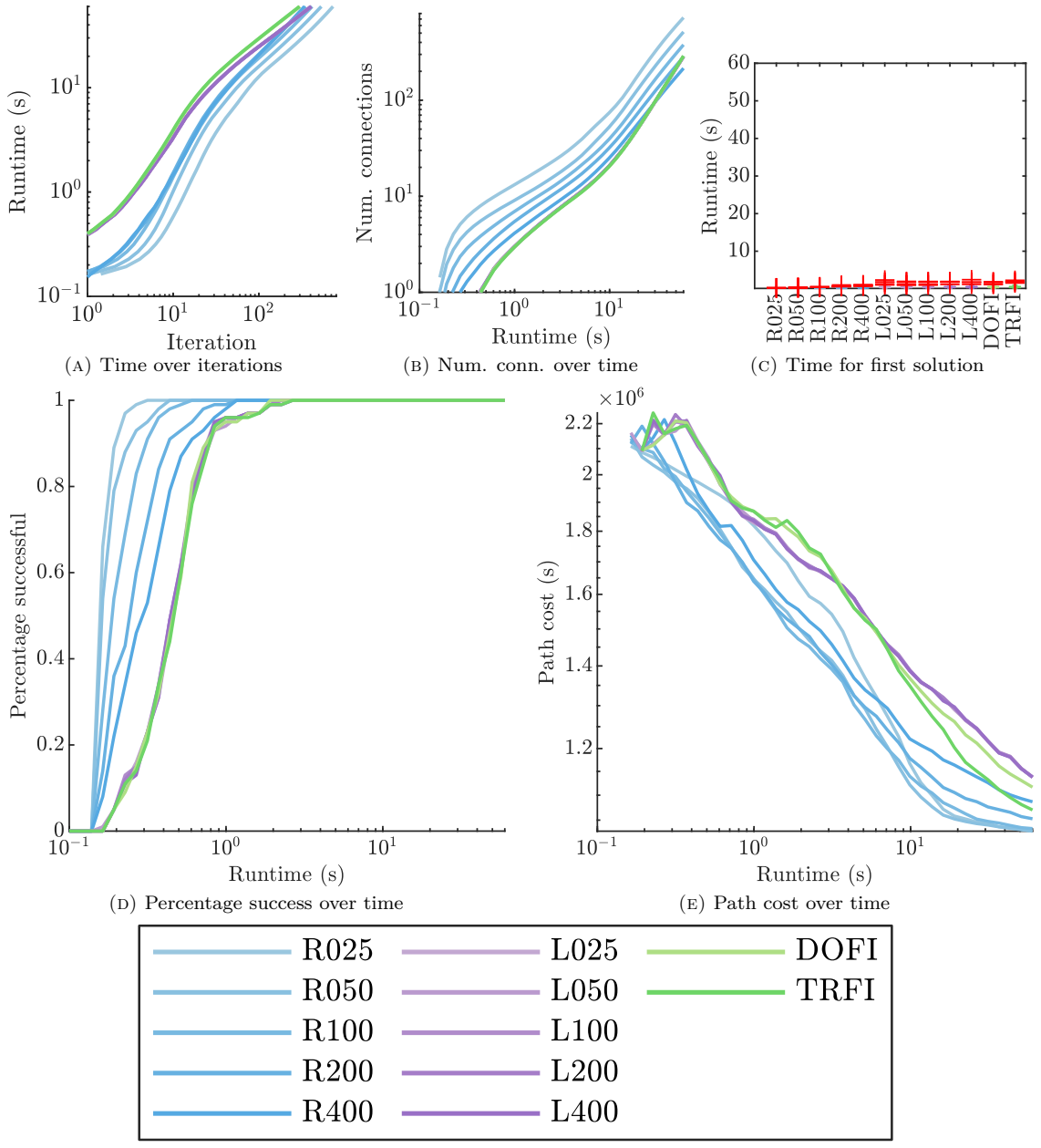
FIGURE 5.9: Example of the final path from different implementations and their execution cost for planning one minute from Sydney to Brisbane. The RRT edges are shown in orange and those that part of the final path are shown in red.

An example final solution for each implementation is shown in Fig. 5.8 for the southward trip and Fig. 5.9 for the northward trip. Despite only a minute of computation, the generated paths from the different implementations are similar. It can be seen in Figs. 5.8a and 5.9a how RCRD methods grow their tree by following the natural system dynamics. Lower values of the maximum integration horizon $T_{\max}$ tends to focus newer nodes in areas already covered. In the southward trip, this allows R025 to produce the best path since there were many opportunities to make small adjustments for shorter travel time. However for the northward trip, this prevents R025 from finding a path at all, since its tree has not grown to the goal yet. In contrast, higher $T_{\max}$ implementations tends to create newer nodes further away from the tree giving it better exploration qualities at the expense of path convergence rate. The fact that a solution is found for R200, but not for R400 in this set of RCRD implementation paths in Fig. 5.9 is also a good reminder that these techniques grow their trees more randomly and can be inconsistent at producing paths at the same time. Since the implementations using AOS are more systematic in terms of adding nodes, the trees they construct cover the space more consistently, however the corresponding paths are not as optimal. It can also be seen that for this set of experiments the LSBN implementations created very similar paths and trees which suggests that a wider range of values might be needed for larger effects. Out of the implementations that used AOS, TRFI appears to generate the best paths in terms of its cost.

Figures 5.10 and 5.11 show aggregated results across 100 runs, each of which are performed on a 2.2 GHz Intel Xeon Gold 6238R processor. For these results, the graphs related to ℓ_2-dispersion were note generated due to the complex boundary of Australia's landmass.

Figures 5.10a and 5.11a confirm that in the limited scope of computation time, AOS methods are slower to iterate than RCRD methods. Figures 5.10b and 5.11b also follow a similar pattern to 5.7c. Despite creating more connections, RCRD implementations does not cover the space well with new nodes as shown in Figs. 5.8 and 5.9, reinforcing the idea that the RCRD does not reduce dispersion effectively.

As discussed before, low $T_{\max}$ RCRD implementations constructs a tree that better follows the natural system dynamics which is advantageous when the goal is downstream of the start. In Fig. 5.10d, R025 is not only able to find solutions earlier, the time to find first



(A) Time over iterations

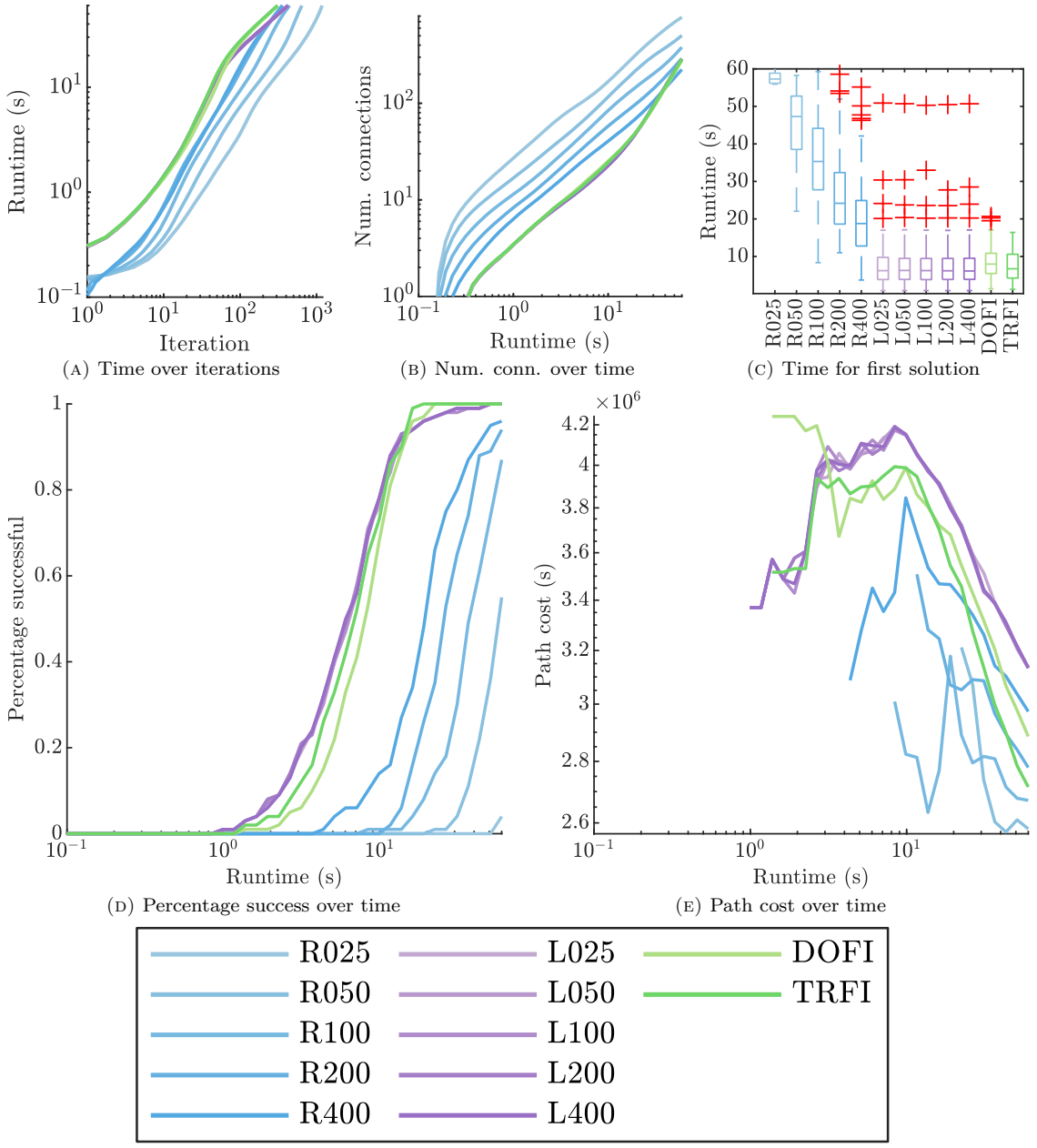
(B) Num. conn. over time

(C) Time for first solution

(D) Percentage success over time

(E) Path cost over time

FIGURE 5.10: Aggregated measurements tracked over the computation time for the 100 runs of each different approach for the Brisbane to Sydney trip. Note that each point on a path cost curve is computed across the number of available solutions for each implementation resulting in a curve that is not necessarily monotonically decreasing.



(A) Time over iterations

(B) Num. conn. over time

(C) Time for first solution

(D) Percentage success over time

(E) Path cost over time

FIGURE 5.11: Aggregated measurements tracked over the computation time for the 100 runs of each different approach for the Sydney to Brisbane trip. Note that each point on a path cost curve is computed across the number of available solutions for each implementation resulting in a curve that is not necessarily monotonically decreasing.

solution is also more consistent. But Fig. 5.11d shows that the same implementation has trouble finding a solution in the allotted time. The implementations using AOS are more consistent in terms of the time to first solution, with LSBN implementations being slightly faster on average on the northward trip. In practice, the TRFI implementation is more consistent in terms of time to first solution. This is emphasised in Figs. 5.10c and 5.11c which visualises time to first solution using a box plot with a linear scale. These figures also highlight that the TRFI implementation already has solutions for both sets of results by the 20 second mark.

Finally in Figs. 5.10e and 5.11e, RCRD methods are shown to produce path costs when solutions are available that tend to be lower than the proposed methods. However, in applications where the anytime property is important, a shorter time to first solution is more important. Furthermore, it should be emphasised that all of the curves in Fig. 5.11e does not account for the path costs in some runs as some have not found solution yet. This means that the true average path cost can be higher. These graphs also indicate the benefits of the third goal flow filter, allowing the TRFI implementation to converge faster than DOFI.

This experiment demonstrated that the proposed filter-based approach, TRFI, outperforms the RCRD method in scenarios with adverse currents. Although TRFI is slower than RCRD in scenarios with assistive currents, the difference is minimal. One advantage of using AOS-based implementations is their superior coverage of the search space with tree nodes, which benefits multi-query applications. For instance, AOS implementations in the EAC environment can more readily provide solutions to various destinations from their starting points compared to RCRD implementations. Additionally, if the RRT was constructed in reverse, with the goal position as the root node, it would allow for querying time-optimal paths to destinations from multiple locations.

5.5 Summary

In this chapter, we identified the advection reachability problem which hindered the intentional growth of search trees to sampled points. We strived to preserve samples from

any chosen sampling strategy in order to reap the benefits of certain sampling sequences described by existing research. To achieve this, we first presented an implementation of the `Extend` function using AOS. This `Extend` function is designed to find a possible persistent control that can be used to connect two points with a conservative amount of computation. Its adaptive integration horizon allows it to perform faster at later stages of the overall algorithm which improves convergence rates to the optimal path. We then explored two different approaches for the `Nearest` primitive function, to address the advection reachability problem. In the first approach, we proposed an alternate distance measure which incorporates a property of the flow field to quantify the reachability between two points. However this approach can incorrectly quantify downstream tree nodes as more reachable to the sample node than more suitable upstream ones, a problem that also applies for true distance metrics. In the second approach, we explore the idea that the implementations of the two primitive functions `Nearest` and `Extend` should be tightly associated. We believe that implementations of `Nearest` should focus on identifying the tree node from which is most likely possible to connect to the sample using the chosen `Extend` function. A multi-stage filter is proposed that progressively eliminates tree node candidates to improve the likelihood of identifying this tree node.

From our experiments, we find that the use of AOS lead to higher dispersion reduction which generally corresponded to faster path convergence rates and earlier initial solutions. Another advantage of this kind of approach is that the tree will more likely include the sample points themselves. E.g. in our experiments, we used a deterministic sampling sequence which has a low dispersion property. In the scenarios where the proposed approaches are expected to converged slower or have later initial solutions, the sampling sequence can be adjusted to inherit different properties like goal bias [69], local bias [87], beacon sampling [12], and informed sampling [90]. In contrast, the RCRD baseline method would not be able to inherit these benefits as much since the tree extension process is not intended to grow towards a specified target.

The findings in this chapter are promising and show potential for practical use in marine robotics, and even suggest further work exploring the connection between the primitive functions `Nearest` and `Extend` in general kinodynamic systems. Additionally, there is potential for further improvement in performance by considering a `Nearest` primitive

function that draw ideas from both proposed approaches. This combined approach largely follows the algorithmic structure of the flow-based multi-stage filter described in Alg. 5.1, however instead of using Euclidean distance to choose from the final pool of candidates, the V_{LSB}-augmented 2-norm is used. Whilst Euclidean distance is appropriate for the flow-based multi-stage filter to reinforce the locality assumptions of the filters, the V_{LSB}-augmented 2-norm would add a preference for candidates that are more aligned to the flow field which should be more numerically stable compared to those at the edge of reachability. However, choosing the candidate in this way requires additional mathematical operations to be evaluated, so further research is required to determine whether the computational overhead is a worthwhile tradeoff for the numerical stability, and to gain insight on how the β term should be chosen.

Chapter 6

Conclusion

In this thesis, we identified operational constraints and limitations for underwater gliders that lead to heavy reliance on remote supervision. Underwater gliders have comparatively low propulsion speeds compared to the speed of ocean currents which means that they can get stuck in situations where their effective directions of movement are limited. This makes them heavily reliant on remote supervision to navigate around challenging flow patterns, particularly as the industry standard typically assumes a uniform flow environment. In Sec. 6.1, we address some fundamental issues that arise from these circumstances as contributions, in Sec. 6.2 we discuss different avenues of future work that can follow the work in this thesis, and in Sec. 6.3 we end the thesis with an outlook to the future of underwater gliders.

6.1 Main contributions

In Ch. 3 we presented a new algorithm that provides a continuous flow field estimation by iteratively refining the contributions of basis flow fields identified from ensemble forecast data. This leverages the output of systems dedicated to more accurate predictions, avoiding the need to rigorously account for various physics-based phenomena, and the large amounts of measured data across space and time. The use of kernel methods to re-describe the ensemble data as latent variables ensures the representation retains incompressibility

of the flow field, an approximation also used by oceanographers. By vertically decoupling the latent representation of the ensemble data and extracting the spatially correlated flow patterns for 2.5D flow fields, we were able to reduce the out-of-span error that comes with this type of flow field estimation. This approach is shown to be more computationally efficient than other techniques and also has an interesting by-product of making global corrections from local measurements.

In Ch. 4 we lay the groundwork for streamline-based control theory, which a identifies a relation between the reachability of another point and the constant propulsion velocities. This reduces the search space for the persistent control that minimises the cost function of interest for the implementation of `Steer` and consequently `Cost`, which are necessary implementations for some sampling-based path planners. An extension of this idea is also presented for 2.5D flow fields, reducing the 2D search space of an underwater glider to 1D again. Despite relinquishing the guarantees of completeness, we argue that the overall algorithm remains to be probabilistically complete if used in a probabilistically complete sampling-based path planning algorithm. This is supported by experiments comparing against the implementation that considers the full available velocity space.

In Ch. 5 we address the advection reachability problem that arises from travelling in strong flow fields our proposed implementations of the primitive function `Nearest`. In this function, we seek a suitable the existing search graph node from which to extend towards the sampled state. Two implementations of this primitive function `Nearest` are proposed. The first explores the use of a distance measure that is augmented by an additional quantity called the lower speed bound (LSB), which is another way to characterise the difficulty to traverse between two points. This quantity comes from streamline-based control theory describing the speed below which a vehicle provably cannot use to reach the other point. The second approach is a filter-based implementation that progressively eliminates candidates based on conditions that hold stronger the closer the two compared points. We find that the filter-based approach identifies more suitable nodes to extend from, leading to faster convergence to the optimal path. Furthermore, by following the philosophy to retain the generated samples in the search graph, we show that both of these approaches inherit valuable qualities from the originating sampling sequence. In our experiments, both proposed methods are compared against the best practice using the kinodynamic

rapidly exploring random tree (RRT) framework, and produced search graphs that better cover the search space leading to more consistent computation times to first solution.

6.2 Future work

In this thesis, we explored algorithmic advancements for flow field estimation and path planning in 2D time-invariant flow fields and demonstrated how these methods can be extended to 2.5D flow fields. Here, the potential future directions search are discussed. More specifically, Sec. 6.2.1 outlines the details of an integrated system based on our findings and discusses its potential impact. Additionally, Sec. 6.2.2 provides insights into further developments and extensions of the ideas presented in this thesis.

6.2.1 Experimentation of integrated system

In this thesis, we proposed a time-invariant flow field estimation algorithm in Ch. 3, and addressed path planning issues in strong time-invariant flow fields in Chs. 4 and 5. Immediate future work to this is testing a system that integrates these ideas in a model predictive control (MPC)-like fashion. While such a system would not account for the time-variant nature of real ocean currents, its robustness would be of significant interest. This section provides details on the implementation of this system, illustrated in Fig. 6.1, along with alternative variants, and discusses the potential impact of a successful integration.

6.2.1.1 Autonomous 2D navigation with underwater gliders

We first consider a system that treats the ocean as a 2D flow field. The merits of this approximation is that the estimation and problem is drastically simplified. The operational workspace for underwater gliders is also very stretched out, with the typical ratio between horizontal and vertical dimensions being comparable to the ratio of the area and the thickness of a piece of paper.

We first describe the infrastructure to support the autonomous navigation of underwater gliders that is always ready to establish communications whenever the underwater glider

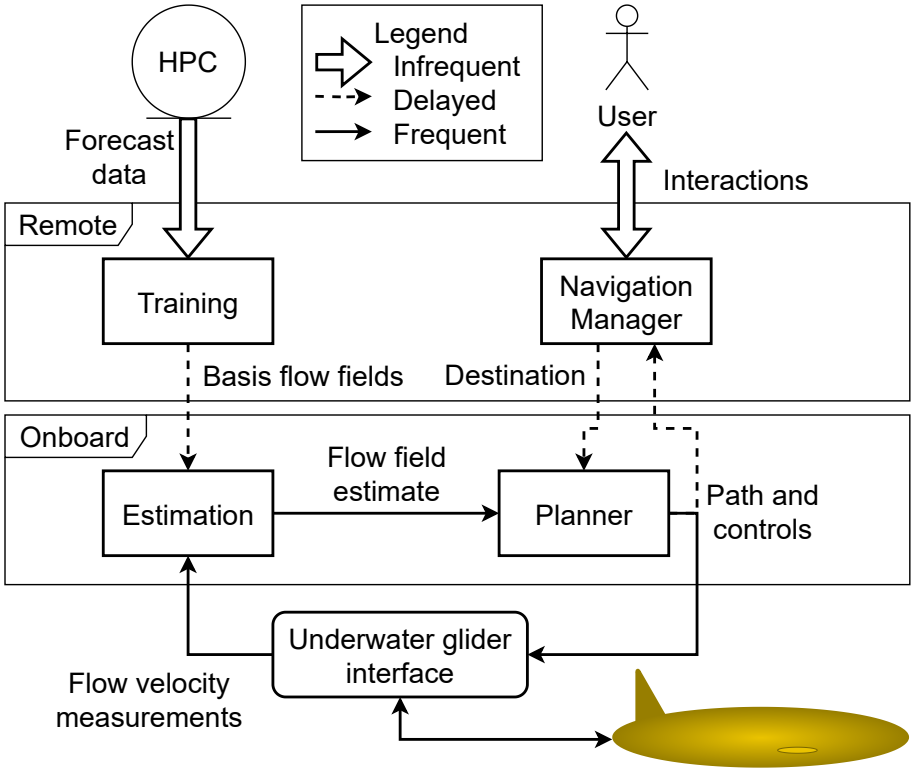


FIGURE 6.1: System diagram of an integrated navigation system for underwater gliders

surfaces. At the top right of the diagram, the user oversees the full mission and is able to set multiple waypoints for the underwater glider. At the top left, a high performance computing (HPC) solution produces ensemble forecasts of ocean behaviour roughly 6 hrs apart, e.g. from the Australian Bureau of Meteorology (BOM). The offline/training step described in Ch. 3 is then applied on the ensemble data at a particular depth to obtain basis flow fields $\boldsymbol{H}(\mathbf{x})$, weight vector initialisation $\mathbf{w}_0$, and the corresponding covariance matrix by taking the column-wise covariance of the weight matrix $\boldsymbol{W}$. We recommend that the chosen depth should be below the *Ekman layer*, the top boundary layer of the ocean with strong currents which ends at about 45 to 300 m. The operational depths for the underwater glider should also be below the Ekman layer, where the flow field is mostly horizontal flow and is more consistent across depth.

Next we describe the system onboard the underwater glider corresponding to the bottom half of the diagram. We assume that the underwater glider has something like an acoustic Doppler current profiler (ADCP) to measure its local flow velocity. Additional measurements of flow velocity can also be derived from drift using expectation-maximisation using

reference positions before and after a dive [38]. The measurements are then used to update the flow field estimate using the online step from Ch. 3, i.e. a Kalman filter (KF) update step. Inflating the covariance of the weight vectors may allow the algorithm to accommodate for the different sources of modelling inaccuracies such as the time-invariant approximation of the flow field. An advantage of using our flow field estimation framework is that the planner can query the flow flux relative to a fixed reference point by using a slightly different kernel. Instead of the incompressible kernel $\boldsymbol{K}_{\text{incomp}}$, the *flow flux kernel*

$$\boldsymbol{K}_{\text{flux}}(\mathbf{x}, \mathbf{x}') = k(\mathbf{x}, \mathbf{x}')\mathcal{D} \left(\mathbf{x}'\right)^{\mathsf{T}}, \tag{6.1}$$

should be used instead.

The planner should use an algorithm designed for replanning in dynamic environments since our flow field estimate updates over time. A potential candidate is RRT$^{\text{X}}$ [127] with some minor adjustments. Its implementation should include the primitive functions described in this thesis: the cost can be evaluated between two costs by using the steering function in Sec. 4.4.1, the search tree can be extended by using adaptive optimistic steering (AOS) described in Sec. 5.3, and finally the tree node to extend from can be selected by using the multi-stage filter technique described in Sec. 5.3.3. The search tree maintained by the planner acts as a *policy*, in the sense that it is able to provide controls and paths to the goal regardless of where the vehicle is. This is useful in our system as it is able to handle various sources of error such as underwater localisation error from dead reckoning, drifting from strong surface currents, flow field estimation errors.

Whenever the underwater glider is able to communicate with the remote system, information is updated on the vehicle, and some data is sent back to the user. If a new destination is sent, then a new search tree using the new position as the root. For the onboard estimation module, the basis flow fields and weight vectors can be completely replaced with the new quantities. It may be worthwhile fusing the previous weight vector with the updated one as the previous weight vector still holds some valuable information.

6.2.1.2 Additional considerations for 2.5D environments

We also proposed ideas for 3D environments using a 2.5D approximation of the environment's flow field. After small adjustments to the ideas presented in Ch. 5, they could be integrated along with the ideas in Sec. 3.5 and Sec. 4.5 for a system to explore the benefits of a different underwater glider control scheme in which each diving or rising manoeuvre is determined by the navigation system. This control scheme enables the generation of optimal trajectories with patterns different that are different to the standard zigzag or sawtooth pattern [8]. However, a consequence of this control scheme is that the horizontal spacing between the planner nodes is effectively limited by the horizontal range of a single manoeuvre which limits the horizontal size of the workspace in practice.

Planning in larger horizontal workspaces is feasible if we constrain underwater glider trajectories to be sawtooth-like. In this formulation, persistent controls describe the net horizontal velocity and the duration only. Since the distance between subsequent nodes are expected to be larger, we do not believe that the steering strategy proposed in Ch. 4 to be sufficient. A sampling-based algorithm that constructs an optimal tree without an effective two-point boundary-value problem (TP-BVP) solver may be necessary such as stable sparse RRT (SST) [116].

6.2.1.3 Impact of robust underwater glider navigation

Robust navigation simplifies the logistical challenges of underwater glider operations, allowing them to be treated as autonomous vehicles rather than mere tools. Additionally, ongoing research [182, 183] aimed at accurately predicting energy consumption patterns based on planned trajectories could enable these vehicles to autonomously navigate back to deployment and recovery stations, similar to how Roomba robot vacuums return to their docking stations. This advancement holds significant potential for improving the accessibility and utilisation of underwater gliders in extensive ocean monitoring initiatives.

By seamlessly integrating underwater gliders into fleets of robots, this advancement transcends mere autonomy. It enables mutual localisation, a collaborative approach that not

only improves navigation effectiveness but also enhances the quality and accuracy of collected data. This collective intelligence allows the vehicles to operate in harmony, boosting their capabilities and expanding the scope of their scientific contributions.

6.2.2 Insight for further algorithmic advancements

In this section, we point to possible links between our work and existing work that can potentially advance our theoretical and algorithmic contributions even further. We focus on advancements for ensemble-based flow field estimation in Sec. 6.2.2.1, and advancements for streamline-based path planning in Sec. 6.2.2.2.

6.2.2.1 Advancements of ensemble-based flow field estimation

Firstly, Ch. 3 provides a clear framework to estimating flow field dynamics. A prominent class of tools for this task involves approximating the Koopman operator [184], an infinite-dimensional operator that maps a vector field from one timestep to the next. One of the most recognised methods for this purpose is dynamic mode decomposition (DMD) [32], which employs tools like the singular value decomposition (SVD), similar to those used in Ch. 3. Non-linear extensions of DMD, such as Extended DMD [185] and Kernel DMD [186], could further enhance accuracy for this highly non-linear dynamic system.

Additionally, given the amount of meteorological data available, deep-learning based methods for estimating the Koopman operator may offer improved performance. Physics-informed neural networks [187] and physics-informed DMD [188] may also allow the use of models such as incompressibility to improve sample efficiency, further improving identification performance.

Finally, the Perron-Frobenius operator, which is the dual of the Koopman operator, propagates probability distributions under the dynamics of a flow field; this is another promising tool that may be able to capture the uncertainty in an ensemble forecast while taking into account the dynamics of the flow fields in the forecast. This may be useful in modelling the uncertainty of a glider moving in strong flows while it is experiencing poor communications.

6.2.2.2 Advancements of streamline-based path planning

In Ch. 5, we addressed the advection reachability problem for the primitive function `Nearest`, which facilitated the addition of nodes similar to the ones from the sampling sequences. Similar ideas can also be applied on the primitive function `Near`, which identifies a limited number of nearby graph nodes. The standard implementation identifies a number of nearby nodes by Euclidean distance, however the way these nodes are using in the RRT* framework suggests that two slightly different primitive functions should be used instead, one that identifies a number of candidate preceding nodes, and one that identifies a number of candidate succeeding nodes. It can also potentially be shown that a smaller number of nodes to be used for these primitive functions will still preserve the asymptotically optimal property of the algorithm and can improve the convergence rate to the optimal path.

In Ch. 4, we considered stream functions to establish the foundations for streamline-based control theory. This theory enabled us to efficiently find a persistent control that allows a vehicle to transition between two points in 2D incompressible flow fields, and a simple extension of the idea was also demonstrated for a special case of the 3D environment. We believe further work can lead to the development of a more principled extension for general 3D incompressible flow fields.

Through some preliminary numerical experiments, we observe that the persistent controls that traverse between two points in 3D incompressible flow fields appear to lie on a 3D line, whether the flow field is continuous or not. Developments could be made by considering *dual stream functions* $\psi : \mathbb{R}^3 \to \mathbb{R}$ and $\chi : \mathbb{R}^3 \to \mathbb{R}$ of a flow field $\mathbf{f} : \mathbb{R}^3 \to \mathbb{R}^3$

$$\mathbf{f}(\mathbf{x}) = \nabla\psi \times \nabla\chi, \tag{6.2}$$

where the $\nabla(\cdot)$ is the gradient of the argument. The level set intersection of a pair of these 3D stream functions correspond to streamlines, in contrast to the level sets of stream functions in 2D. Unfortunately, dual stream functions are not well-defined for general flow fields. Whilst there is existing work that establishes how some dual stream functions can be formulated for some basic flow patterns [189], the main challenge is that the dual

stream function of superposed flow fields is not always equal to the summation of their dual stream functions. A potential solution is to find a way to define the dual stream functions of basic flow patterns such that their summation corresponds to superposed flow. Resolving this issue holds the potential to unlock applications in various disciplines grappling with incompressible flow fields, including electromagnetic fields in physics and gravitational fields in astronomy.

6.3 Outlook

The vastness of the ocean has always posed a formidable challenge to researchers and explorers alike. However, the emergence of underwater gliders presents a beacon of hope in this endeavour. Their remarkable energy efficiency and capability to cover expansive distances mark them as invaluable assets in the pursuit of understanding our planet's largest and least explored realm. Looking ahead, one can envision a future where fleets of autonomous underwater gliders tirelessly patrol the oceans, delivering precise and targeted measurements essential for unravelling the secrets of meteorological phenomena. These tireless sentinels, equipped with advanced sensing capabilities, promise to revolutionise our ability to monitor and predict oceanic patterns on a scale previously unimaginable.

Beyond the realm of scientific inquiry, the potential applications of autonomous underwater gliders extend to the sphere of national security. In the future, we may witness these technological marvels taking up posts along offshore borders, enhancing efforts to safeguard our nations. Their stealthy and persistent presence could prove instrumental in bolstering maritime security and responding swiftly to potential threats. As technology advances and their capabilities continue to grow, underwater gliders are poised to play a pivotal role in safeguarding our marine borders, further emphasising their significance in the broader context of oceanic exploration and security.

Bibliography

[1] Fengying Dang, Sanjida Nasreen, and Feitian Zhang. DMD-based background flow sensing for AUVs in flow pattern changing environments. *IEEE Robotics and Automation Letters*, 6(3):5207–5214, 2021.

[2] Walid Remmas, Ahmed Chemori, and Maarja Kruusmaa. Diver tracking in open waters: A low-cost approach based on visual and acoustic sensor fusion. *Journal of Field Robotics*, 38(3):494–508, 2021.

[3] Supun Randeni, Toby Schneider, Ee Shan C. Bhatt, Oscar A. Víquez, and Henrik Schmidt. A high-resolution AUV navigation framework with integrated communication and tracking for under-ice deployments. *Journal of Field Robotics*, 40(2): 346–367, 2023.

[4] Pierre Del Moral. Nonlinear filtering: Interacting particle resolution. *Comptes Rendus de l'Academie des Sciences-Serie I-Mathematique*, 325(6):653–658, 1997.

[5] Jun S. Liu and Rong Chen. Sequential Monte Carlo methods for dynamic systems. *Journal of the American Statistical Association*, 93(443):1032–1044, 1998.

[6] World Meteorological Organization. Guidelines on Ensemble Prediction Systems and Forecasting, 2012. URL `https://library.wmo.int/doc_num.php?explnum_id=7773`.

[7] E. Zermelo. Über das Navigationsproblem bei ruhender oder veränderlicher Windverteilung. *Zeitschrift für Angewandte Mathematik und Mechanik*, 11(2):114–124, 1931.

[8] James Ju Heon Lee, Chanyeol Yoo, Raewyn Hall, Stuart Anstee, and Robert Fitch. Energy-optimal kinodynamic planning for underwater gliders in flow fields. In *Proceedings of the Australian Conference on Robotics and Automation*, 2017.

[9] Lydia E. Kavraki, Petr Švestka, Jean-Claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566 – 580, 1996.

[10] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *International Journal of Robotics Research*, 30(7):846–894, 2011.

[11] Lucas Janson, Brian Ichter, and Marco Pavone. Deterministic sampling-based motion planning: Optimality, complexity, and performance. *International Journal of Robotics Research*, 37(1):46–61, 2018.

[12] Jauwairia Nasir, Fahad Islam, Usman Malik, Yasar Ayaz, Osman Hasan, Mushtaq Khan, and Mannan Saeed Muhammad. RRT*-SMART: A rapid convergence implementation of RRT*. *International Journal of Advanced Robotic Systems*, 10:299, 2013.

[13] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E. Bekris, and Dan Halperin. Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation. *IEEE Robotics and Automation Letters*, 4(2):277–283, 2019.

[14] Stephen M. Griffies. *Fundamentals of Ocean Climate Models.* Princeton University Press, Princeton, 2004.

[15] Gurvan Madec. *NEMO Ocean Engine.* Note du Pôle de modélisation, Institut Pierre-Simon Laplace (IPSL), France, No 27, 2008.

[16] George S.K. Wong and Shi-ming Zhu. Speed of sound in seawater as a function of salinity, temperature, and pressure. *The Journal of the Acoustical Society of America*, 97(3):1732–1736, 1995.

[17] Robert Henry Stewart. *Introduction to Physical Oceanography.* Texas A&M University, 2008.

[18] Bengt Andersson, Ronnie Andersson, Love Håkansson, Mikael Mortensen, Rahman Sudiyo, and Berend van Wachem. Turbulent-flow modelling. In *Computational Fluid Dynamics for Engineers*, pages 86–145. Cambridge University Press, Cambridge, 2011.

[19] Alexander F. Shchepetkin and James C. McWilliams. The regional oceanic modeling system (ROMS): A split-explicit, free-surface, topography-following-coordinate oceanic model. *Ocean Modelling*, 9:347–404, 2005.

[20] Geir Evensen. Sequential data assimilation with a nonlinear quasi-geostrophic model using Monte Carlo methods to forecast error statistics. *Journal of Geophysical Research*, 99(C5):10143–10162, 1994.

[21] Peter L. Houtekamer and Herschel L. Mitchell. Data assimilation using an ensemble Kalman filter technique. *Monthly Weather Review*, 127(3):796–811, 1998.

[22] Geir Evensen. The ensemble Kalman filter: Theoretical formulation and practical implementation. *Ocean Dynamics*, 53:343–367, 2003.

[23] Nasir Ahmed, T. Natarajan, and Kamisetty R. Rao. Discrete cosine transform. *IEEE Transactions on Computers*, C-23(1):90–93, 1974.

[24] The International Telegraph and Telephone Consultative Committee. Digital compression and coding of continuous-tone still images - requirements and guidelines, 1992.

[25] Zhe Bai, Thakshila Wimalajeewa, Zachary Berger, Guannan Wang, Mark Glauser, and Pramod K. Varshney. Low-dimensional approach for reconstruction of airfoil data via compressive sensing. *AIAA Journal*, 53(4):920–933, 2015.

[26] Carl Eckart and Gale Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[27] Ido Bright, Guang Lin, and José Nathan Kutz. Compressive sensing based machine learning strategy for characterizing the flow around a cylinder with limited pressure measurements. *Physics of Fluids*, 25:127102, 2013.

[28] Jonathan H. Tu, Clarence W. Rowley, Dirk M. Luchtenburg, Steven L. Brunton, and José Nathan Kutz. On dynamic mode decomposition: Theory and applications. *Journal of Computational Dynamics*, 1(2):391–421, 2014.

[29] Jonathan H. Tu, John Griffin, Adam Hart, Clarence W. Rowley, Louis N. Cattafesta, and Lawrence S. Ukeiley. Integration of non-time-resolved PIV and time-resolved velocity point sensors for dynamic estimation of velocity fields. *Experiments in Fluids*, 54:1429, 2013.

[30] Tahiya Salam and M. Ani Hsieh. Adaptive sampling and reduced-order modeling of dynamic processes by robot teams. *IEEE Robotics and Automation Letters*, 4(2): 477–484, 2019.

[31] Jared L. Callaham, Kazuki Maeda, and Steven L. Brunton. Robust flow reconstruction from limited measurements via sparse representation. *Physical Review Fluids*, 4(10):31, 2019.

[32] José Nathan Kutz, Steven L. Brunton, Bingni W. Brunton, and Joshua L. Proctor. *Dynamic Mode Decomposition: Data-Driven Modeling of Complex Systems*. SIAM, Philadelphia, 2016.

[33] Tahiya Salam and M. Ani Hsieh. Heterogeneous robot teams for modeling and prediction of multiscale environmental processes. *Autonomous Robots*, 47:353–376, 2023.

[34] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer-Verlag, New York, 2006.

[35] Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, Cambridge, 2006.

[36] David Duvenaud. *Automatic Model Construction with Gaussian Processes*. Ph.d. dissertation, University of Cambridge, jun 2014.

[37] Dongsik Chang, Wencen Wu, Catherine R. Edwards, and Fumin Zhang. Motion tomography: Mapping flow fields using autonomous underwater vehicles. *International Journal of Robotics Research*, 36(3):320–336, 2017.

[38] Ki Myung Brian Lee, Chanyeol Yoo, Ben Hollings, Stuart Anstee, Shoudong Huang, and Robert Fitch. Online estimation of ocean current from sparse GPS data for underwater vehicles. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3443–3449, 2019.

[39] Carl Jidling, Niklas Wahlström, Adrian Wills, and Thomas B. Schön. Linearly constrained Gaussian processes. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 1215–1224, 2017.

[40] Hassan A. Kingravi, Harshal Maske, and Girish Chowdhary. Kernel observers: Systems-theoretic modeling and inference of spatiotemporally evolving processes. In *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, pages 3997–4005, 2016.

[41] Joshua Whitman and Girish Chowdhary. Learning dynamics across similar spatiotemporally-evolving physical systems. In *Proceedings of the Annual Conference on Robot Learning*, pages 472–481, 2017.

[42] Steven M. LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, 2006.

[43] Tapovan Lolla, Pierre F. J. Lermusiaux, Mattheus P. Ueckermann, and Patrick J. Haley Jr. Time-optimal path planning in dynamic flows using level set equations: Theory and schemes. *Ocean Dynamics*, 64:1373–1397, 2014.

[44] Sri Venkata Tapovan Lolla. *Path Planning and Adaptive Sampling in the Coastal Ocean*. Ph.d. dissertation, Massachusetts Institute of Technology, Cambridge, MA, feb 2016.

[45] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[46] Jorge Nocedal and Stephen J. Wright. *Numerical Optimization*. Springer, New York, 2nd edition, 2006.

[47] O. von Stryk and R. Bulirsch. Direct and indirect methods for trajectory optimization. *Annals of Operations Research*, 37:357–373, 1992.

[48] Laszlo Techy. Optimal navigation in a planar time-varying point-symmetric flow-field. In *Proceedings of the IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, pages 7325–7330, 2011.

[49] Ryan N. Smith and Matthew Dunbabin. Controlled drift: An investigation into the controllability of underwater vehicles with minimal actuation. In *Proceedings of the Australian Conference on Robotics and Automation*, 2011.

[50] Josep Isern-González, Daniel Hernández-Sosa, Enrique Fernández-Perdomo, Jorge Cabrera-Gámez, Antonio C. Domínguez-Brito, and Víctor Prieto-Marañón. Path planning for underwater gliders using iterative optimization. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1538–1543, 2011.

[51] Dhanushka Kularatne, Subhrajit Bhattacharya, and M. Ani Hsieh. Going with the flow: A graph based approach to optimal path planning in general flows. *Autonomous Robots*, 42:1369–1387, 2018.

[52] Carlos Lucas, Daniel Hernández-Sosa, David Greiner, Aleš Zamuda, and Rui Caldeira. An approach to multi-objective path planning optimization for underwater gliders. *Sensors*, 19(24):5506, 2019.

[53] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[54] Arvind A. Pereira, Jonathan Binney, Geoffrey A. Hollinger, and Gaurav S. Sukhatme. Risk-aware path planning for autonomous underwater vehicles using predictive ocean models. *Journal of Field Robotics*, 30(5):741–762, 2013.

[55] Taehwan Lee, Hanguen Kim, Hyun Chung, Yuseok Bang, and Hyun Myung. Energy efficient path planning for a marine surface vehicle considering heading angle. *Ocean Engineering*, 107:118–131, 2015.

[56] Dhanushka Kularatne, Hadi Hajieghrary, and M. Ani Hsieh. Optimal path planning in time-varying flows with forecasting uncertainties. In *Proceedings of the IEEE*

International Conference on Robotics and Automation (ICRA), pages 4857–4864, 2018.

[57] Dhanushka Kularatne, Subhrajit Bhattacharya, and M. Ani Hsieh. Optimal path planning in time-varying flows using adaptive discretization. *IEEE Robotics and Automation Letters*, 3(1):458–465, 2018.

[58] Michaël Soulignac, Patrick Taillibert, and Michel Rueher. Adapting the wavefront expansion in presence of strong currents. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1352–1358, 2008.

[59] Michaël Soulignac. Feasible and optimal path planning in strong current fields. *IEEE Transactions on Robotics*, 27(1):89–98, 2011.

[60] Blane Rhoads, Igor Mezić, and Andrew Poje. Minimum time feedback control of autonomous underwater vehicles. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 5828–5834, 2010.

[61] Jack Elston and Eric Frew. Unmanned aircraft guidance for penetration of pre-tornadic storms. *Journal of Guidance, Control, and Dynamics*, 33(1):99–107, 2010.

[62] Brunilde Girardet, Laurent Lapasset, Daniel Delahaye, and Christophe Rabut. Wind-optimal path planning: Application to aircraft trajectories. In *Proceedings of the International Conference on Control, Automation, Robotics and Vision*, pages 1403–1408, 2014.

[63] Tapovan Lolla, Patrick J. Haley Jr., and Pierre F. J. Lermusiaux. Path planning in multi-scale ocean flows: Coordination and dynamic obstacles. *Ocean Modelling*, 94: 46–66, 2015.

[64] Deepak N. Subramani and Pierre F. J. Lermusiaux. Energy-optimal path planning by stochastic dynamically orthogonal level-set optimization. *Ocean Modelling*, 100: 57–77, 2016.

[65] Deepak N. Subramani, Quantum Jichi Wei, and Pierre F. J. Lermusiaux. Stochastic time-optimal path-planning in uncertain, strong, and dynamic flows. *Computer Methods in Applied Mechanics and Engineering*, 333:218–237, 2018.

[66] Deepak N. Subramani and Pierre F. J. Lermusiaux. Risk-optimal path planning in stochastic dynamic environments. *Computer Methods in Applied Mechanics and Engineering*, 353:391–415, 2019.

[67] Chinmay S. Kulkarni and Pierre F. J. Lermusiaux. Three-dimensional time-optimal path planning in the ocean. *Ocean Modelling*, 152:101644, 2020.

[68] Dushyant Rao and Stefan B. Williams. Large-scale path planning for underwater gliders in ocean currents. In *Proceedings of the Australian Conference on Robotics and Automation*, 2009.

[69] Chris Urmson and Reid Simmons. Approaches for heuristically biasing RRT growth. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 2, pages 1178–1183, 2003.

[70] Anjan Chakrabarty and Jack Langelaan. UAV flight path planning in time varying complex wind-fields. In *Proceedings of the American Control Conference*, pages 2568–2574, 2013.

[71] Inyoung Ko, Beobkyoon Kim, and Frank Chongwoo Park. Randomized path planning on vector fields. *International Journal of Robotics Research*, 33(13):1664–1682, 2014.

[72] Emilio Frazzoli, Munther A. Dahleh, and Eric Feron. Maneuver-based motion planning for nonlinear systems with symmetries. *IEEE Transactions on Robotics*, 21(6): 1077–1091, 2005.

[73] Elgar Desa, R. Madhan, and P. Maurya. Potential of autonomous underwater vehicles as new generation ocean data platforms. *Current Science*, 90(9):1202–1209, 2006.

[74] Jesse Stuart Geisbert. *Hydrodynamic Modeling for Autonomous Underwater Vehicles Using Computational and Semi-Empirical Methods.* Master's thesis, Virginia Polytechnic Institute and State University, Blacksburg, VA, may 2007.

[75] Bartolome Garau, M. Bonet, Alberto Alvarez, S. Ruiz, and A. Pascual. Path planning for autonomous underwater vehicles in realistic oceanic current fields: Application to gliders in the Western Mediterranean sea. *Journal of Maritime Research*, 6(2):5–21, 2009.

[76] Brian Claus, Ralf Bachmayer, and C. D. Williams. Development of an auxiliary propulsion module for an autonomous underwater glider. *Proceedings of the Institution of Mechanical Engineers, Part M*, 224(4):255–266, 2010.

[77] Charles C. Eriksen, T. James Osse, Russell D. Light, Timothy Wen, Thomas W. Lehman, Peter L. Sabin, John W. Ballard, and Andrew M. Chiodi. Seaglider: A long-range autonomous underwater vehicle for oceanographic research. *IEEE Journal of Oceanic Engineering*, 26(4):424–436, 2001.

[78] Jeff Sherman, Russ E. Davis, W. B. Owens, and J. Valdes. The autonomous underwater glider "Spray". *IEEE Journal of Oceanic Engineering*, 26(4):437–446, 2001.

[79] Douglas C. Webb, Paul J. Simonetti, and Clayton P. Jones. SLOCUM: An underwater glider propelled by environmental energy. *IEEE Journal of Oceanic Engineering*, 26(4):447–452, 2001.

[80] Daniel L. Rudnick, Russ E. Davis, Charles C. Eriksen, David M. Fratantoni, and Mary Jane Perry. Underwater gliders for ocean research. *Marine Technology Society Journal*, 38(2):73–84, 2004.

[81] Oscar Schofield, Josh Kohut, David Aragon, Liz Creed, Josh Graver, Chip Haldeman, John Kerfoot, Hugh Roarty, Clayton P. Jones, Doug Webb, and Scott Glenn. Slocum gliders: Robust and ready. *Journal of Field Robotics*, 24(6):473–485, 2007.

[82] Daniel L. Rudnick. Ocean research enabled by underwater gliders. *Annual Review of Marine Science*, 8:519–541, 2016.

[83] Chanyeol Yoo, James Ju Heon Lee, Stuart Anstee, and Robert Fitch. Path planning in uncertain ocean currents using ensemble forecasts. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 8323–8329, 2021.

[84] John H. Halton. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numerische Mathematik*, 2:84–90, 1960.

[85] Luigi Palmieri, Leonard Bruns, Michael Meurer, and Kai O. Arras. Dispertio: Optimal sampling for safe deterministic sampling-based motion planning. *IEEE Robotics and Automation Letters*, 5(2):362–368, 2020.

[86] Kiril Solovey and Michal Kleinbort. The critical radius in sampling-based motion planning. *International Journal of Robotics Research*, 39(2-3):266–285, 2020.

[87] Baris Akgun and Mike Stilman. Sampling heuristics for optimal motion planning in high dimensions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2640–2645, 2011.

[88] Dave Ferguson and Anthony Stentz. Anytime RRTs. *Proc. of IEEE/RSJ IROS*, pages 5369–5375, 2006.

[89] Michael Otte and Nikolaus Correll. C-FOREST: Parallel shortest path planning with superlinear speedup. *IEEE Transactions on Robotics*, 29(3):798–806, 2013.

[90] Jonathan D. Gammell, Timothy D. Barfoot, and Siddhartha S. Srinivasa. Informed sampling for asymptotically optimal path planning. *IEEE Transactions on Robotics*, 34(4):966–984, 2018.

[91] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *International Journal of Robotics Research*, 20(5):378–400, 2001.

[92] Alexander Shkolnik, Matthew Walter, and Russ Tedrake. Reachability-guided sampling for planning under differential constraints. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2859–2865, 2009.

[93] Mukunda Bharatheesha, Wouter Caarls, Wouter Jan Wolfslag, and Martijn Wisse. Distance metric approximation for state-space RRTs using supervised learning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 252–257, 2014.

[94] Luigi Palmieri and Kai O. Arras. Distance metric learning for RRT-based motion planning with constant-time inference. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 637–643, 2015.

[95] Sunil Arya, David M. Mount, Nathan S. Netanyahu, Ruth Silverman, and Angela Y. Wu. An optimal algorithm for approximate nearest neighbor searching in fixed dimensions. *Journal of the ACM*, 45(6):891–923, 1998.

[96] Tobias Kunz and Mike Stilman. Kinodynamic RRTs with fixed time step and best-input extension are not probabilistically complete. In *Algorithmic Foundations of Robotics XI*, pages 233–244. Springer, Cham, 2015.

[97] Lydia E. Kavraki, Mihail N. Kolountzakis, and Jean-Claude Latombe. Analysis of probabilistic roadmaps for path planning. *IEEE Transactions on Robotics and Automation*, 14(1):166–171, 1998.

[98] Lucas Janson, Edward Schmerling, Ashley Clark, and Marco Pavone. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *International Journal of Robotics Research*, 34(7):883–921, 2015.

[99] Kiril Solovey, Oren Salzman, and Dan Halperin. New perspective on sampling-based motion planning via random geometric graphs. *International Journal of Robotics Research*, 37(10):1117–1133, 2018.

[100] Christian L. Nielsen and Lydia E. Kavraki. A two level fuzzy PRM for manipulation planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, volume 3, pages 1716–1721, 2000.

[101] Guang Song, Shawna Thomas, and Nancy M. Amato. A general framework for PRM motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 3, pages 4445–4450, 2003.

[102] Edward Schmerling, Lucas Janson, and Marco Pavone. Optimal sampling-based motion planning under differential constraints: The driftless case. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2368–2375, 2015.

[103] Edward Schmerling, Lucas Janson, and Marco Pavone. Optimal sampling-based motion planning under differential constraints: The drift case with linear affine dynamics. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 2574–2581, 2015.

[104] Robert Bohlin and Lydia E. Kavraki. Path planning using lazy PRM. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 521–528, 2000.

[105] Kris Hauser. Lazy collision checking in asymptotically-optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2951–2957, 2015.

[106] Donghyuk Kim, Youngsun Kwon, and Sung-Eui Yoon. Dancing PRM*: Simultaneous planning of sampling and optimization with configuration free space approximation. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 7071–7078, 2018.

[107] Oktay Arslan and Panagiotis Tsiotras. Use of relaxation methods in sampling-based algorithms for optimal motion planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2421–2428, 2013.

[108] James J. Kuffner and Steven M. LaValle. RRT-Connect: An efficient approach to single-query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 2, pages 995–1001, 2000.

[109] Peng Cheng and Steven M. LaValle. Resolution complete rapidly-exploring random trees. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, volume 1, pages 267–272, 2002.

[110] Anna Yershova, Léonard Jaillet, Thierry Siméon, and Steven M. LaValle. Dynamic-domain RRTs: Efficient exploration by controlling the sampling domain. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3856–3861, 2005.

[111] Dave Ferguson, Nidhi Kalra, and Anthony Stentz. Replanning with RRTs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1243–1248, 2006.

[112] Matt Zucker, James Kuffner, and Michael Branicky. Multipartite RRTs for rapid replanning in dynamic environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1603–1609, 2007.

[113] Léonard Jaillet, Juan Cortés, and Thierry Siméon. Sampling-based path planning on configuration-space costmaps. *IEEE Transactions on Robotics*, 26(4):635–646, 2010.

[114] Didier Devaurs, Thierry Siméon, and Juan Cortés. Enhancing the transition-based RRT to deal with complex cost spaces. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4120–4125, 2013.

[115] Wei Wang, Yan Li, Xin Xu, and Simon X. Yang. An adaptive roadmap guided multi-RRTs strategy for single query path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2871–2876, 2010.

[116] Yanbo Li, Zakary Littlefield, and Kostas E. Bekris. Asymptotically optimal sampling-based kinodynamic planning. *International Journal of Robotics Research*, 35(5):528–564, 2016.

[117] Kiril Solovey, Oren Salzman, and Dan Halperin. Finding a needle in an exponential haystack: Discrete RRT for exploration of implicit roadmaps in multi-robot motion planning. *International Journal of Robotics Research*, 35(5):501–513, 2016.

[118] Rahul Shome, Kiril Solovey, Andrew Dobson, Dan Halperin, and Kostas E. Bekris. dRRT*: Scalable and informed asymptotically-optimal multi-robot motion planning. *Autonomous Robots*, 44:443–467, 2020.

[119] Florian Hauer and Panagiotis Tsiotras. Deformable rapidly-exploring random trees. In *Proceedings of Robotics: Science and Systems*, 2017.

[120] Sharan Nayak and Michael W. Otte. Bidirectional sampling-based motion planning without two-point boundary value solution. *IEEE Transactions on Robotics*, 38(6): 3636–3654, 2022.

[121] Kiril Solovey, Lucas Janson, Edward Schmerling, Emilio Frazzoli, and Marco Pavone. Revisiting the asymptotic optimality of RRT*. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2189–2195, 2020.

[122] Matthew Jordan and Alejandro Perez. Optimal Bidirectional Rapidly-Exploring Random Trees. Technical report, Computer Science and Artificial Intelligence Laboratory, 2013.

[123] Dustin J. Webb and Jur Van Den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 5054–5061, 2013.

[124] Yun-xiao Shan, Bi-jun Li, Jian-Zhou, and Yue-Zhang. An approach to speed up RRT*. In *Proceedings of the IEEE Symposium on Intelligent Vehicles*, pages 594–598, 2014.

[125] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Informed RRT*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2997–3004, 2014.

[126] Ahmed Hussain Qureshi and Yasar Ayaz. Intelligent bidirectional rapidly-exploring random trees for optimal motion planning in complex cluttered environments. *Robotics and Autonomous Systems*, 68:1–11, 2015.

[127] Michael Otte and Emilio Frazzoli. RRTX: Asymptotically optimal single-query sampling-based motion planning with quick replanning. *International Journal of Robotics Research*, 35(7):797–822, 2016.

[128] Didier Devaurs, Thierry Siméon, and Juan Cortés. Optimal path planning in complex cost spaces with sampling-based algorithms. *IEEE Transactions on Automation Science and Engineering*, 13(2):415–424, 2016.

[129] Jonathan D. Gammell, Timothy D. Barfoot, and Siddhartha S. Srinivasa. Batch informed trees (BIT*): Informed asymptotically optimal anytime search. *International Journal of Robotics Research*, 39(5):543–567, 2020.

[130] Oren Salzman and Dan Halperin. Asymptotically-optimal motion planning using lower bounds on cost. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4167–4172, 2015.

[131] Joseph A. Starek, Javier V. Gomez, Edward Schmerling, Lucas Janson, Luis Moreno, and Marco Pavone. An asymptotically-optimal sampling-based algorithm for bi-directional motion planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 2072–2078, 2015.

[132] Sanjiban Choudhury, Jonathan D. Gammell, Timothy D. Barfoot, Siddhartha S. Srinivasa, and Sebastian Scherer. Regionally accelerated batch informed trees (RA-BIT*): A framework to integrate local information into optimal path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4207–4214, 2016.

[133] Zakary Littlefield and Kostas E. Bekris. Efficient and asymptotically optimal kinodynamic motion planning via dominance-informed regions. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7410–7415, 2018.

[134] Jonathan D. Gammell, Siddhartha S. Srinivasa, and Timothy D. Barfoot. Batch informed trees (BIT*): Sampling-based optimal planning via the heuristically guided search of implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3067–3074, 2015.

[135] Alexander C. Holston, Deok-Hwa Kim, and Jong-Hwan Kim. Fast-BIT: Modified heuristic for sampling-based optimal planning with a faster first solution and convergence in implicit random geometric graphs. In *Proceedings of the IEEE International Conference on Robotics and Biomimetics (ROBIO)*, pages 1892–1899, 2018.

[136] Marlin P. Strub and Jonathan D. Gammell. Advanced BIT* (ABIT*): Sampling-based planning with advanced graph-search techniques. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 130–136, 2020.

[137] Marlin P. Strub and Jonathan D. Gammell. Adaptively informed trees (AIT*): Fast asymptotically optimal path planning through adaptive heuristics. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3191–3198, 2020.

[138] Zakary Littlefield and Kostas E. Bekris. Informed asymptotically near-optimal planning for field robots with dynamics. In *Field and Service Robots*, pages 449–463. Springer, Cham, 2018.

[139] K. Y. Cadmus To, Felix H. Kong, Ki Myung Brian Lee, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Estimation of spatially correlated ocean currents from ensemble forecasts and online measurements. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 2301–2307, 2021.

[140] Felix H. Kong, K. Y. Cadmus To, Gary Brassington, Stuart Anstee, and Robert Fitch. 3D ensemble-based online oceanic flow field estimation for underwater glider path planning. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4358–4365, 2021.

[141] Tong Wang, Olivier P. Le Maître, Ibrahim Hoteit, and Omar M. Knio. Path planning in uncertain flow fields using ensemble method. *Ocean Dynamics*, 66:1231–1251, 2016.

[142] Walter H. Munk. Abyssal recipes. *Deep Sea Research and Oceanographic Abstracts*, 13(4):707–730, 1966.

[143] Moninya Roughan and Jason H. Middleton. A comparison of observed upwelling mechanisms off the east coast of Australia. *Continental Shelf Research*, 22(17): 2551–2572, 2002.

[144] Xinfeng Liang, Michael Spall, and Carl Wunsch. Global ocean vertical velocity from a dynamically consistent ocean state estimate. *Journal of Geophysical Research: Oceans*, 122(10):8208–8224, 2017.

[145] Rudolf Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[146] Jeffrey Humpherys, Preston Redd, and Jeremy West. A fresh look at the Kalman filter. *SIAM Review*, 54(4):801–823, 2012.

[147] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT Press, 2005.

[148] Fouad Sukkar, Graeme Best, Chanyeol Yoo, and Robert Fitch. Multi-robot region-of-interest reconstruction with Dec-MCTS. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 9101–9107, 2019.

[149] Ki Myung Brian Lee, James Ju Heon Lee, Chanyeol Yoo, Ben Hollings, and Robert Fitch. Active perception for plume source localisation with underwater gliders. In *Proceedings of the Australian Conference on Robotics and Automation*, 2018.

[150] Ruzena Bajcsy, Yiannis Aloimonos, and John K. Tsotsos. Revisiting active perception. *Autonomous Robots*, 42:177–196, 2018.

[151] Tamer Inanc, Shawn C. Shadden, and Jerrold E. Marsden. Optimal trajectory generation in ocean flows. In *Proceedings of the American Control Conference*, pages 674–679, 2005.

[152] Teledyne RD Instruments. Pinnacle 45 Datasheet, 2018. URL `https://www.teledynemarine.com/en-us/products/SiteAssets/RDInstruments/RDI_Pinnacle_45_datasheet.pdf`.

[153] K. Y. Cadmus To, Ki Myung Brian Lee, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Streamlines for motion planning in underwater currents. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4619–4625, 2019.

[154] K. Y. Cadmus To, James Ju Heon Lee, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Streamline-based control of underwater gliders in 3D environments. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 8303–8310, 2019.

[155] Louise M. Russell-Cargill, Bradley S. Craddock, Ross B. Dinsdale, Jacqueline G. Doran, Ben N. Hunt, and Ben Hollings. Using autonomous underwater gliders for geochemical exploration surveys. *The APPEA Journal*, 58(1):367–380, 2018.

[156] Hordur Johannsson, Michael Kaess, Brendan Englot, Franz Hover, and John J. Leonard. Imaging sonar-aided navigation for autonomous underwater harbor surveillance. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4396–4403, 2010.

[157] Alberto Alvarez, Andrea Caiti, and Reiner Onken. Evolutionary path planning for autonomous underwater vehicles in a variable ocean. *IEEE Journal of Oceanic Engineering*, 29(2):418–429, 2004.

[158] Bartolome Garau, Alberto Alvarez, and Gabriel Oliver. Path planning of autonomous underwater vehicles in current fields with complex spatial variability: An A* approach. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 194–198, 2005.

[159] Dov Kruger, Rustam Stolkin, Aaron Blum, and Joseph Briganti. Optimal AUV path planning for extended missions in complex, fast-flowing estuarine environments. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 4265–4270, 2007.

[160] Jonas Witt and Matthew Dunbabin. Go with the flow: Optimal path planning in coastal environments. In *Proceedings of the Australian Conference on Robotics and Automation*, 2008.

[161] Van T. Huynh, Matthew Dunbabin, and Ryan N. Smith. Predictive motion planning for AUVs subject to strong time-varying currents and forecasting uncertainties. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1144–1151, 2015.

[162] Daniele Di Vito, Daniela De Palma, Enrico Simetti, Giovanni Indiveri, and Gianluca Antonelli. Experimental validation of the modeling and control of a multibody underwater vehicle manipulator system for sea mining exploration. *Journal of Field Robotics*, 38(2):171–191, 2021.

[163] Philip J. Pritchard. *Fox and McDonald's Introduction to Fluid Mechanics*. Wiley, 8th edition, 2011.

[164] George Keith Batchelor. *An Introduction to Fluid Dynamics*. Cambridge University Press, Cambridge, 2000.

[165] David F. Griffiths and Desmond J. Higham. *Numerical methods for ordinary differential equations*. Springer London, 2010.

[166] Yoshiaki Kuwata, Justin Teo, Gaston Fiore, Sertac Karaman, Emilio Frazzoli, and Jonathan P. How. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology*, 17(5):1105–1118, 2009.

[167] Tapovan Lolla, Mattheus P. Ueckermann, Konuralp Yiğit, Patrick J. Haley Jr., and Pierre F. J. Lermusiaux. Path planning in time dependent flow fields using level set methods. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 166–173, 2012.

[168] Tapovan Lolla, Patrick J. Haley Jr., and Pierre F. J. Lermusiaux. Time-optimal path planning in dynamic flows using level set equations: Realistic applications. *Ocean Dynamics*, 64:1399–1417, 2014.

[169] Edsger Wybe Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[170] Geoffrey Ingram Taylor and Albert Edward Green. Mechanism of the production of small eddies from large ones. *Proceedings of the Royal Society A*, 141:499–521, 1935.

[171] Clayton Jones, Ben Allsup, and Christopher DeCollibus. Slocum glider: Expanding our understanding of the oceans. In *Proceedings of OCEANS*, pages 1–10, 2014.

[172] Francis D. Lagor, Kayo Ide, and Derek A. Paley. Touring invariant-set boundaries of a two-vortex system using streamline control. In *Proceedings of the IEEE Conference on Decision and Control (CDC)*, pages 2217–2222, 2015.

[173] Chia Shun Yih. Fonctions de courant dans les écoulements à trois dimensions. *Houille Blanche*, 6368(3):439–450, 1957.

[174] K. Y. Cadmus To, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Distance and steering heuristics for streamline-based flow field planning. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1867–1873, 2020.

[175] Fouad Sukkar. *Fast, Reliable and Efficient Database Search Motion Planner (FREDS-MP) for Repetitive Manipulator Tasks*. Master's thesis, University of Technology Sydney, Sydney, NSW, 2017.

[176] James Ju Heon Lee, Chanyeol Yoo, Stuart Anstee, and Robert Fitch. Hierarchical planning in time-dependent flow fields for marine robots. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 885–891, 2020.

[177] Jennifer Barry, Leslie Pack Kaelbling, and Tomás Lozano-Pérez. A hierarchical approach to manipulation with diverse actions. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 1799–1806, 2013.

[178] Harald Niederreiter. *Random Number Generation and Quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, 1992.

[179] Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, 1975.

[180] Norbert Beckmann, Hans-Peter Kriegel, Ralf Schneider, and Bernhard Seeger. The R*-tree: An efficient and robust access method for points and rectangles. In *Proceedings of the International Conference on Management of Data*, pages 322–331, 1990.

[181] Michal Kleinbort, Kiril Solovey, Zakary Littlefield, Kostas E. Bekris, and Dan Halperin. Corrections to "Probabilistic completeness of RRT for geometric and kinodynamic planning with forward propagation". *IEEE Robotics and Automation Letters*, 8(2):1149–1150, 2023.

[182] Yaojian Zhou, Jiancheng Yu, and Xiaohui Wang. Path planning method of underwater glider based on energy consumption model in current environment. In *Intelligent Robotics and Applications*, pages 142–152. Springer, Cham, 2014.

[183] Yang Song, Shuxin Wang, Xudong Xie, Yanhui Wang, Shaoqiong Yang, and Wei Ma. Energy consumption prediction method based on LSSVM-PSO model for autonomous underwater gliders. *Ocean Engineering*, 230:108982, 2021.

[184] B. O. Koopman. Hamiltonian systems and transformation in Hilbert space. *Proceedings of the National Academy of Sciences*, 17(5):315–318, 1931.

[185] Qianxiao Li, Felix Dietrich, Erik M. Bollt, and Ioannis G. Kevrekidis. Extended dynamic mode decomposition with dictionary learning: A data-driven adaptive spectral decomposition of the Koopman operator. *Chaos*, 27:103111, 2017.

[186] Matthew O. Williams, Clarence W. Rowley, and Ioannis G. Kevrekidis. A kernel-based method for data-driven Koopman spectral analysis. *Journal of Computational Dynamics*, 2(2):247–265, 2016.

[187] M. Raissi, P. Perdikaris, and G. E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[188] Peter J. Baddoo, Benjamin Herrmann, Beverley J. McKeon, J. Nathan Kutz, and Steven L. Brunton. Physics-informed dynamic mode decomposition. *Proceedings of the Royal Society A*, 479:20220576, 2023.

[189] Zhenquan Li and Gordon Mallinson. Dual stream function visualization of flows fields dependent on two variables. *Computing and Visualization in Science*, 9(1): 33–41, 2006.

Appendices

A Underwater glider kinematic model

This appendix provides the remaining components that fully define the kinematic model of the underwater glider's trim states mentioned in Sec. 4.2.1.1. Recall that the speed of the vehicle is

$$V_{\mathrm{g}}(\beta) = \sqrt{\frac{m(\beta) \cdot g}{D(\beta) \cdot \sin\beta - L(\beta) \cdot \cos\beta}}. \tag{4.3 revisited}$$

The other functions of the glide angle are defined in the following way:

$$m(\beta) = m_{\mathrm{g}} - m_{\mathrm{b}}(\beta), \tag{A.3}$$

$$L(\beta) = K_{L_0} + K_L \cdot \alpha(\beta), \tag{A.4}$$

$$D(\beta) = K_{D_0} + K_D \cdot \alpha(\beta)^2, \tag{A.5}$$

$$m_{\mathrm{b}}(\beta) = \begin{cases} m_{\mathrm{b}_{\max}}, & \beta < 0 \\ 0, & \text{otherwise} \end{cases}, \tag{A.6}$$

$$\alpha(\beta) = \frac{K_L}{2K_D}\left(\sqrt{1 - \frac{4K_D}{K_L^2}\left(\cot\beta\right)\left(K_{L_0} + K_{D_0}\cot\beta\right)} - 1\right)\tan\beta. \tag{A.7}$$

The constants that were used in this thesis are defined as:

$$\begin{aligned} g &= 9.81\,\mathrm{m/s^2}, \\ m_{\mathrm{g}} &= 1\,\mathrm{kg}, \\ m_{\mathrm{b}_{\max}} &= 2\,\mathrm{kg}, \\ K_{L_0} &= 0\,\mathrm{N}, \\ K_L &= 306\,\mathrm{N}, \\ K_{D_0} &= 5\,\mathrm{N}, \\ K_D &= 20\,\mathrm{N}. \end{aligned}$$