Planning under Uncertainty using a Dynamical Systems Approach for Autonomous Vehicles

Alfredo José Bayuelo Sierra

Universidad Nacional de Colombia

Departamento de Sistemas e Industrial

Facultad de Ingeniería

Bogotá, Colombia

2021

Planning under Uncertainty using a Dynamical Systems Approach for Autonomous Vehicles

Alfredo José Bayuelo Sierra

In partial fulfillment of the requirements for the degree of:

Doctor en Ingeniería - Ingeniería de Sistemas y Computación

Advisor:

Ph.D., Luis Fernando Niño Vásquez

Co-advisor:

Ph.D., J. Leonardo Bobadilla

Research Field:

Motion Planning - Dynamic Non-Linear Systems - Robotics Algorithms

Research Group:

Laboratorio de Sistemas Inteligentes LISI

Universidad Nacional de Colombia

Departamento de Ingeniería de Sistemas e Industrial

Facultad de Ingeniería

Bogotá, Colombia

2022

”You don’t know where you are standing”.

Angela Rosa Miranda. Intuition of a basic localization problem.

Acknowledgements

I would like to thank my advisor Luis Fernando and all my professors for their dedication; they have shown me that professors and students make the school. Especially, I would like to express my sincere gratitude to Dr. Leonardo Bobadilla for introducing us to the Robotics community, for the advice, for all the time spent guiding, and for his group support. Also, I thank my family, especially my wife Mile. Finally, I would also like to thank Colciencias (Minciencias) for granting me the scholarship "757 Doctorados Nacionales".

Abstract

Title: Planning under Uncertainty using a Dynamical Systems Approach for Autonomous Vehicles

The problem of Motion Planning for Robotic Systems (in this work: robot or vehicles) has been well studied. Some significant outcomes have been accomplished, and good results demonstrated in practical situations in industry and other commercial uses. Nevertheless, high computational cost and several assumptions on the problems present open challenges and opportunities for research. This work presents strategies to help in the solution of the navigation and other related problems for four different scenarios: unknown vehicle model, unknown positions/orientation of the vehicle, unknown map to navigate and unknown intention of other vehicles in the same environment. First, realistic simulation is used to overcome the lack of a model, or the difficulties to calculate it. Simulated environments have taken advantage of the improvements in computer systems in the last decades; for example, computer games have progressively shown more realistic environments, these environments have already been used to train models by fooling the sensors of robots and making them to learn from gameplays, in this fashion, simulators are used here to help solving the navigation problem. It is also presented here a feedback-based motion planer for a simple bouncing robot, showing how it can navigate a complex world even if the current position is not know all the time. Of course the map must be known before hand to create such a plan, for the case where the map is not known *a priori*, a strategy for simultaneous localization and mapping is presented here to determine the world around and the position of the vehicle in such map. Finally, when considering simpler robots, it might be necessary to use multiple of them to succeed at a particular task, and they might also be in the presence of a third party robot, hence, a strategy is presented here

to communicate and avoid collisions while preserving privacy.

Keywords: Motion Planning, Sim-to-real, Dynamical Systems, SLAM, Aquatic Vehicles.

Resumen

Título: Planeación bajo incertidumbre usando una aproximación de los sistemas dinámicos para vehículos autónomos

La planeación del movimiento para sistemas robóticos (o simplemente robots, o vehículos) es un problema bastante bien estudiado. Resultados significativos se han obtenido en la literatura y se han llevado a la práctica en la industria y otros usos comerciales. Sin embargo, altos costos computacionales y simplificaciones hechas en la formulación de los problemas presentan retos abiertos y oportunidades de investigación. Este trabajo presenta estrategias para ayudar en la solución del problema de navegación, y otros relacionados, en cuatro escenarios: Cuando no se conoce el Modelo que describe el vehículo, No se conoce la posición ni orientación del vehículo, no se conoce el Mapa del lugar, Y cuando no se conoce la intención (aliado/adversario) de otros robots en el ambiente. Primero, se presenta una estrategia que usa ambientes simulados realísticos para superar la falta de modelo del vehículo o las dificulatades que conlleven su cálculo. Los ambientes simulados se han beneficiado de las mejoras en los sistemas computarizados de la última década; por ejempo, los juegos de computadora han progresivamente mostrado ambientes más y más realistas, y estos han sido ya usados para entrenar robots al mostrarle a los sensores del robot esta información como cierta, de tal forma que se logra que los robots aprendan de secuencias del juego, de esta misma forma, en este trabajo se usan los simuladores para ayudar a resolver el problema de la navegación. También se presenta un esquema de planeación basado en la retro alimentación para un sencillo robot que rebota, mostrando cómo dicho robot puede navegar ambientes complejos sin saber su posición en todo momento. Por supuesto el mapa debe ser conocido para crear tal esquema de planeación, cuando no se conoce el mapa, la estrategia conocida como Localización y Mapeo Simultaneos, puede

usarse para determinar el mapa alrededor y encontrarse en el mismo. Finalmente, cuando se consideran robots más simples, puede llegar a ser necesario usar más de un robot para cumplir una tarea, y puede que en el ambiente hayan robots adversarios, por lo tanto, se presenta una estrategia que permite comunicarse para evitar colisiones que mantiene la privacidad al mismo tiempo.

Palabras clave: Planeación de movimiento, Simuladores, Sistemas dinámicos, SLAM, vehículos acuáticos.

This Ph.D. Thesis was defended on March the 3rd 2022 at 7:00 a.m., the evaluation committee included the following members:

RYAN N. SMITH (PhD. Ocean & Resources Engineering)
VP of Technology Solutions
Oxbotica

MARCO MORALES AGUIRRE (Phd. Computer Science)
Teaching Associate Professor
University Of Illinois Urbana-Champaign

JESUS ALBERTO DELGADO RIVERA (Phd. Cybernetics)
Full Professor
Universidad Nacional de Colombia

Contents

Figures list

List of Algorithms

Nomenclature

c	Speed of light in a vacuum
h	Planck constant

1 Introduction

1.1. Motivation

We are in exciting times in the area of Autonomous Mobile Robots which are assigned progressively to more of our everyday duties. Applications are already making a major impact in critical areas of society such as healthcare, manufacturing, warehousing and self-driving vehicles [PCY+16a] (See Figure 1.1). There is a series of Mobile Robots domains that will be of fundamental importance in the near future such as remote operation, construction, surveillance, environmental monitoring and oceanic exploration. In order to be able to achieve the goals in these domains, mobile robotic systems, or simply robots, need to solve the task of navigation along with other related tasks such as patrolling, coverage, and persistent monitoring. The solutions to these tasks differ depending on the environment where the robots execute them. As an example, the complexity is minimal for a piston robot with a single degree of freedom, on the other hand, a vehicle inside a fluid with different currents has more challenges to travel the same distance as the former robot.

Most times, when people think about vehicles, they consider three categories: terrestrial, aerial and aquatic vehicles; nowadays improved versions of these vehicles are also used for extraterrestrial exploration. Motion is common ground for all of them, and they

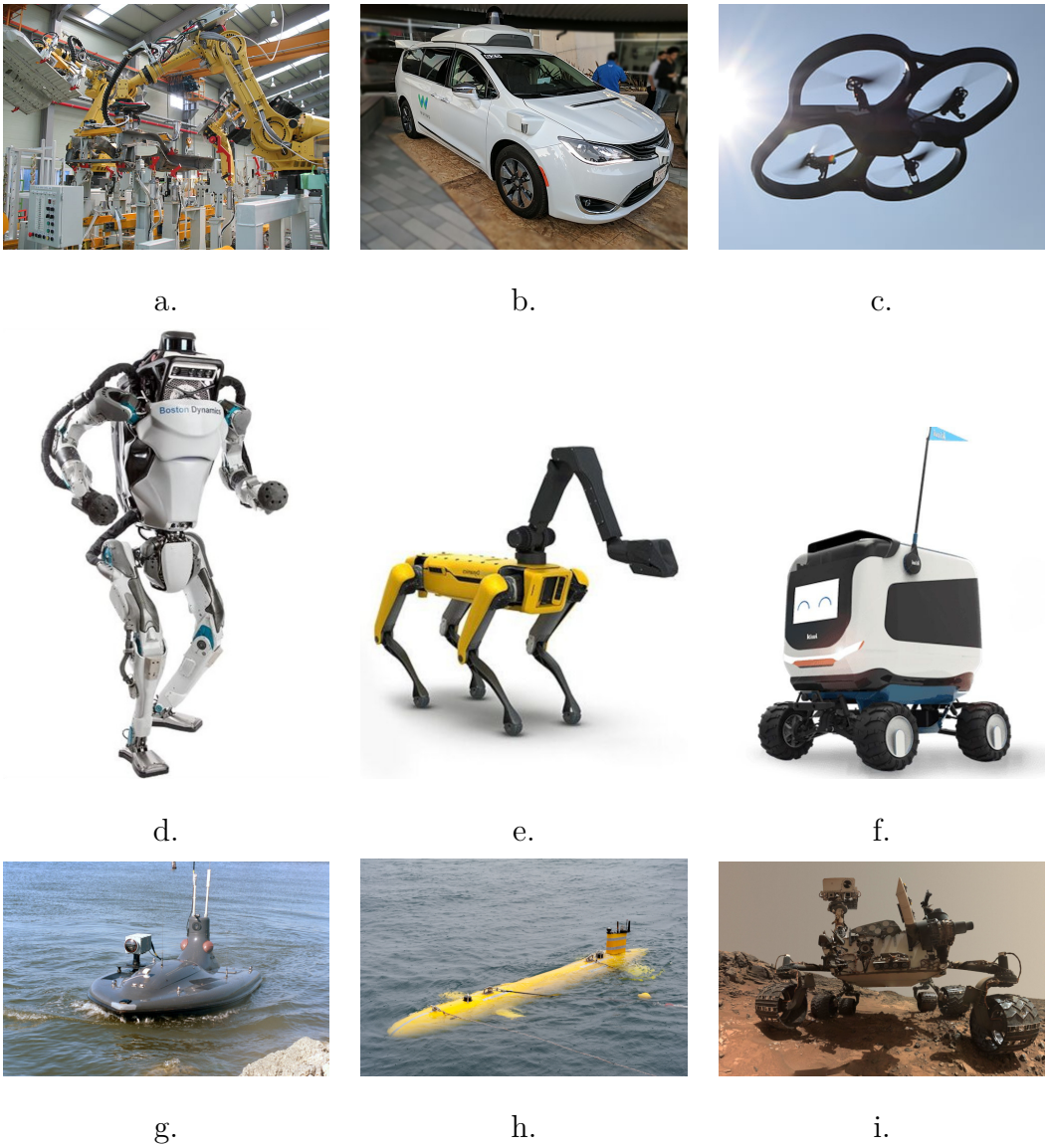


Figure 1-1: Examples of mobile robots: a. Manufacture industry arms [man]; b. Waymo Pacifica driverless minivan[way]; c. Surveillance and filming drone [Pix]; d. Boston Dynamics Humanoid robot Atlas [Dyna]; e. Boston Dynamics spot, dog-like robot for sensing and inspection [Dynb]; f. Kiwibot: An autonomous delivery robot [kiw]; g. Unmanned Surface Vehicle [usv]; h: Unmanned Underwater Vehicle MARUM-SEAL [uuv]; and i. Mars Rover, the Mars exploration robot [rov].

all require a plan for moving. Motion Planning is at a higher degree of abstraction and, it is also common for all of these kinds of vehicles; the difference lies in the details of the vehicle itself: the kinematics and dynamics make concrete the difference during the execution of the plan. Furthermore, if only kinematics is considered, an airplane can be moved along a road the same as a ground vehicle would; also, hover-crafts move on the surface of the water the same as they do on the ground. For that matter, dynamics would be the ultimate differentiator. For ground vehicles, dynamics basically describe the interaction between part of the robot: wheels, feet, crawler, etc. and the surface they are moving on; for aerial vehicles, the interactions not only rely on part of it but on the complete robot, and this is also the case for surface and underwater vehicles. Considering all this, it can be thought of the last kind of vehicles as a generalization of the former, and thus, the strategies developed for the latter can potentially be accommodated for the former. The work presented here takes this into consideration and devotes its efforts to underwater vehicles in the expectation of applying the strategies proposed to other kinds of vehicles. Complex environments requires complex sensors, at least this is the common use. Nevertheless, it is an interesting question as to what is the minimum amount of sensing needed to complete one task. It is well known that the more sensing is introduced, the more noise an variability, then, reducing the amount of sensors or the complexity of sensors would be beneficial for the plan execution. In this scenario it would be interesting to use multiple simple robots, i.e. with simple shapes, sensors, actuators, etc., to complete a particular task; that, however, introduces another layer of complexity which is the communication among the robots, and possibly the need to communicate securely in the presence of adversarial robots. In this work, it is also presented some considerations in this direction: a strategy to coordinate navigation keeping private sensitive information of the plan to execute. For all these scenarios trials is a key process since it allows to correct details

that might not be found during the planning process. Not only the controller, that keeps the robot following the plan, takes advantage of a set of trials, but also the plan itself can be "repaired" based on the trials and outcomes. Performing trials is most of the time impossible due to restrictions of one or many of the following: location, budget, time, device availability, among others. Even if none of these restrictions holds, trials using real robots can be catastrophic for the projects. A method that provides the ability to perform the required number of trials without any of the above concern is simulation, and it is seen more often how simulated environment trials are used to outperform autonomy in vehicle planning [Tes][Mus]. In this work, we also present a strategy that takes advantage of the simulated environments to find feasible plans of motion.

Altogether, this work presents strategies to the following incremental considerations: unknown model dynamics or kinematics, that can be addressed using realistic simulators, unknown robot's states, that can be addressed by feedback planning; unknown mapping, addressed by simultaneously localization and mapping; and unknown confidentiality in multi-robotic settings addressed by secure communications.

1.2. Challenges and Existing Approaches

Motion is a key element in the process of controlling a robot for a particular task. The *Motion Planning* problem involves the actual determination of movements and paths through a defined, possibly cluttered, space; allowing the robot to go from some starting conditions to another set of goal conditions, avoiding to collide into obstacles, or other robots, during the movement [LV11]. These conditions include the robot's set of parameters that identify it and its behavior at any point in time, also called the robot's *state*. The motion plan is executed through actuators (e.g., wheels, thrusters, propellers) that

allow the robot to change its state.

The strategies to solve the motion planning problem can be divided into two main approaches: Combinatorial Planning and Sampling-based Planning [LaV06a]. They differ in their completeness and in the amount of dimensions they can handle. Combinatorial planning algorithms construct a discrete representation of the problem that exactly contains all the possible solutions, hence they work in low state-dimensionality for simple robots, and they are complete, meaning that they always find a solution if at least one exists. Sampling based planning, on the other hand, only attempts to represent part of the problem, sufficiently enough to contain a solution. Thus, it can work with high state-dimensionality, complicated robot shapes and also with kinematics and dynamics constraints. Some Sampling-based method, such as [YF16a], [CK09], [AmCA14a], are defined as probabilistically complete, this means that the probability of finding an existing solution gets closer to 1 as the number of samples grows to infinity. In real-world applications, especially in high state-dimensionality and the presence of obstacles (i.e., part of the space that cannot be crossed), the combinatorial approach is not practical due to the high computational requirements. The obstacles mentioned impose global constraints, there are also local constraints that arise from the natural movements of the robot. Intuitively, an object cannot instantaneously stop or start moving, these constraints are introduced as *differential constraints* to account for *smoothness* in the movement.

The traditional approach to solving the motion planning problem is to first compute a collision-free path, this path is subsequently smoothed to satisfy differential constraints, after this, a nominal trajectory is calculated to follow the path satisfying the robot's dynamics; and finally, a controller is designed to maintain the execution within the nominal trajectory [LaV11c]. This incremental approach has a clear disadvantage of strongly relying on any decision taken in previous steps, that is, every decision limits the scope of

the later steps.

To overcome this limitation, *planning under differential constraints* covers steps 1, 2 and 3 in a single consideration by incorporating kinematics and dynamics of the robot into the planning process, producing a path that satisfies the differential constrains. Nevertheless, At the end of the process a controller is still needed to maintain the robot in the path. The controller step is usually achieved by the use of sensors (e.g., GPS, compass, gyroscope, accelerometer). A combination of these sensor readings is used to estimate the current state and, based on this estimation, take corrections. Difficulties in this approach arise due to the uncertainty generated by unreliable motions of the robot, or due to poor state estimations obtained by imperfect sensors. The more sensors, the more uncertainty is added to the system. [LDFT16] and [DT15] present an example of a controller strategy. Their controller requires the exact locations of the obstacles and off-board sensing is used to track the robot's position. Additionally, a pre-calculated convex free region is provided to the controller, to alleviate the difficulties mentioned. In the latter document, it is also shown how large computational effort is needed to compute the mentioned convex free-region in the presence of less than ten obstacles.

Feedback-based Motion Planning deal with the limitations of the previously described approaches. Here, inferred states are fed to a policy to obtain an action. By performing actions and feeding back the new inferred state to the policy in an iterative fashion, the goal is achieved. Even though sensing or action errors are still a concern, the policy acts as a dynamic re-planner always redirecting the robot to the goal. Despite the benefits of this strategy, the feedback motion planning methods have to deal with the curse of dimensionality as they do not work well for larger dimensions and high-resolution sampling, because the plan is calculated over the entire space. [MT16] is an example of a feedback-based motion plan. The concept of a funnel, first introduced in [Mas85a], is used here to

compose trajectories based on current states. Although the selection and composition of funnels is performed on-line, the higher computational cost is caused by the computations of the funnel's library off-line.

One important feature that this approach offers is the possibility to accomplish a plan even if the environment or its conditions are changing; this feature makes feedback-based motion planning a more robust approach to the motion planning problem. However, it is worth to mention that in order to be able to account for the different scenarios that can be found in these dynamical environments, more computational effort has to be done in the construction of the library, and afterwards, in the selection of the appropriate funnels.

Other approaches similar to the funnel metaphor are described in [BR11], [AmCA14a], [AmAK+18] and [WSF+18]. There, the estimated position of the robot is not a single point in the state space; rather, it is distributed over a *belief space*. Gaussian distribution is typically chosen as the distribution of the position, so that the robot's position is described by the mean position and a variance around it.

The work in [HSF+13] presents some representative multi-objective optimization of a controller. The linear quadratic regulator (LQR) formulation is used for the controller, where a cost function accumulated over all the trajectory execution time is minimized. This work again shows the high-computational effort needed to accomplish the task. In [XQX+14], the authors use dynamical system analysis to find a trajectory in a parameters space. They use a gradient function which implies a significantly high computational cost for multi-obstacle scenarios. Aside from the well known disadvantage of local minima of such a strategy.

A different approach, which tries to overcome the need for large computational and sensing requirements, is described in [ABS17a]. There, Alam *et al.* used as little as just contact sensors and a clock to solve the navigation and coverage problem. They descri-

bed the environment using a discrete method from dynamical systems called *Generalized Cell-to-cell Mapping* (GCM). The properties shown by this representation allowed them to find minimal navigation plans and coverage policies. In [ABS18] they use the same tools to determine the position of the robot with limited sensing; this is a crucial subtask to solve the motion planning problem, most common approaches assume the initial position as given. [CS02] showed how this cell-to-cell mapping can be used not only to solve the motion planning problem but also to evaluate the solutions found. The key concept in this work is the use of Markov chains as the representation of the environment. This representation exposes some important properties such as attractors, basins of attraction, and transient regions. Also, Alam *et al.* use the same approach in [ARBS18a] to solve the coverage problem for oceanic drifters, specifically, they find the minimum number of devices needed to cover certain area, and the policy to move in long-term periods. Their drifters are moving in the ocean, and they represent the water flow pattern using data from the Regional Ocean Modeling System (ROMS) [SM05]. This flow model helps to construct a Markov chain over which the GCM technique is then used. Reis *et al* also investigate the motion planning problem in marine environments in [RFA+17a] and [RFA+17c]. The localization subtask is harder in this kind of environments, since GPS is denied under water, and visibility is at best limited. Bathymetry and others measurable water properties can be used to improve localization as shown in [AS18][MRCS17]. In [ARBS18c], a Hidden Markov Model was used, along with the GCM strategies constructed from ROMS data, to find the most probable path followed by an aquatic vehicle.

1.3. Proposed Approach

As presented above, the problem of Motion Planning has been well studied. Some important results have been accomplished that showed to perform satisfactorily in practical situations such as industry and other commercial uses. But still high computational cost, and a set of assumptions on the problems (previously discussed), present open challenges and opportunities for research. This work builds on the philosophy of augmenting the simplicity of each one of the task that must be faced to do Motion Planning. Taking this into account, the following strategies are presented in this work: a strategy that uses open software and home-computers to calculate robot's transitions and find feasible paths for the navigation problem; a building block to construct feedback plans for simple robots; a sampling scheme to create a representation of the environment where a vehicle is moving so that it can localize and update the map while reducing the amount of data needed; and a coordination strategy, such that, if simple robot are used to complete complex tasks, the answer would be in the number of simple robots needed to succeed, the last strategy is extremely useful when multiple groups of robots are sharing the same environment to solve different tasks.

1.3.1. Key Themes and Contributions

In this work, four strategies are presented to contribute to the Motion Planning problem under the same number of considerations:

- When the robot model is unknown or hard to calculate.
- When the state of the robot is unknown or uncertain.
- When the map where the robot is moving is unknown or uncertain.

- When there are other actors in the environment and it is not known whether they are allies or adversaries.

An Aquatic Unmanned Vehicle mission planning software pipeline using realistic simulators

The first contributions is made to the scenario where the robot model is not known or is hard to calculate, in this case realistic simulators can be used to calculate the outcome of a particular action. The simulator does not need to know the dynamics or kinematics of every particular robot, instead, it builds on the simple blocks of the physics of the objects in the simulation; the emerging outcome would account for the dynamics and kinematics implicitly. Chapter 2 presents a pipeline to find a feasible path using a variation of the well known RRT algorithm. The pipeline presented only uses open software, and shows how practical scenarios can be simulated using home computers.

Computing Feedback Plans from Dynamical System Composition

The second contribution is made to the scenario were the state of the robot is not known, feedback plans are useful since they do not create a single plan to reach a position, instead, they create a policy with action to perform all around the map. The strategy proposed, does not rely on a pre-descretization of the action space which is the common use. Considering this, such algorithm would outperform a descritization that does not include an action needed to complete a task. Chapter 3 presents a strategy to calculate feedback plans for a simple bouncing robot.

Feedback Motion Planning on a self updating map

The third contribution is made to the scenario where the map is currently unknown and hence the position is also known. Building upon previous missions in the same area, and the readings of the sensors for an aquatic environment, a strategy is presented in Chapter 4 to construct/update the map and localize in it. This is known as the Simultaneous Localization and Mapping (SLAM) problem.

Coordinated multi-robot planning while preserving individual privacy

The fourth contribution is made to the scenario where multiple robots navigate the same environment, a major concern in this scenario is that the robots might collide to each other; the first though to solve this issue is to coordinate by sharing the position and destination, but if the other robots are adversarial, sensitive information such as the destination, must be kept private. Chapter 5 presents a strategy to coordinate navigation with other robots while preserving privacy. Encrypted data of the destinations is used to predict collisions, and thus, avoid them.

1.4. Thesis Organization

The remainder of this document is organized as follows: Chapter 2 (An Aquatic Unmanned Vehicle mission planning software pipeline using realistic simulators) presents a strategy to find a feasible path using a variation of the RRT algorithm and a simulator to determine the dynamics, kinematics and uncertainty in the movement of the robot. Chapter 3 (Computing Feedback Plans from Dynamical System Composition) presents a novel feedback planning strategy that does not rely on action space discretization. Chapter

4 (Toward Simultaneous Localization and Mapping in Aquatic Dynamic Environments) Presents a dynamical representation for Motion Planning in aquatic environment. Chapter 5 (Coordinated multi-robot planning while preserving individual privacy) presents a strategy to securely navigate in the presence of adversarial robots avoiding collisions by sharing path information while keeping private sensitive information. Finally, Conclusions and some future perspectives are drawn in Chapter 6.

2 Black-box Modeling for Motion Planning for underwater vehicles using realistic simulators

Part of this work was submitted as a scientific note at Boletín de Investigaciones Marinas y Costeras ISSN: 0122-9761 — ISSN: 2590-4671 (online).[1]

Researching marine environments is challenging because of its features. New technology such as AUV's (Aquatic Unmanned Vehicles) has gained a lot of attention since, they are reliable equipment that accounts for affordability and maneuverability. One consequence can be evidenced in the fact that more than 80 % of the oceans are still unmapped, unexplored, and unobserved. When sufficient trials can be conducted, and several outcomes can be studied, a robotic system would be more securely deployed and would accomplish the objective. Nevertheless, running field trails can be expensive in both computational and economic sense. We propose a pipeline that provides the option to do enough trials in simulated environments. Once the simulation trials are conducted and the path is deter-

[1]The author acknowledges Leonardo Bobadilla for his contribution in the developements of the ideas, simulations and guiding.

mined, the real-world deployment of the robotic system would be more secure and more likely to achieve its goals. The strategy is based on the well-known RRT algorithm (Rapidly exploring Random Trees). We considered a variation of this algorithm to take into account the kinematics, dynamics, and uncertainty of the vehicles' movement. The pipeline proposed here allows for running more secure experimentation for both the system itself and the researchers.

2.1. Introduction

Researching marine environments is challenging because of its features such as high dynamism, poor visibility, extreme temperatures, extreme pressures, and wildlife (flora and fauna) that can affect equipment. For these reasons, research in these environments has been entrusted to robotic systems: traditionally large equipment mounted on dedicated ships. Even though new technology such as AUV's (Aquatic Unmanned Vehicles) has lowered the budget and improved maneuverability, field experimentation requirements make it difficult and sometimes unaffordable, especially in developing countries, to do enough trials to get a mission completed. A consequence of these difficulties can be evidenced in the fact that more than 80 % of the oceans, which in turn comprehends around 70 % of the earth's surface, are still unmapped, unexplored, and unobserved [N+19]. This finally translates into a poor understanding of the processes that occur in these environments, which of course, have direct and indirect repercussions on human activities: health-related, economic, and environmental.

In this context, if sufficient trials can be conducted, and several outcomes can be studied, the robotic system would be more securely deployed and would accomplish the objective with highest probabilities. Nevertheless, running field trials can be expensive in

both computational and economic sense; on the one hand, determining the set of actions needed by the robotic system to accomplish a goal is a non-trivial problem in aquatic environments, nor even for short straight-line paths. Moreover, mathematically modeling a complex vehicle to plan the set of actions beforehand is a challenging problem on its own; the uncertainty associated can be hazardous for the system itself. On the other hand, failed trials can be catastrophic and stop the complete project by losing the assets and the budget to replace them. The pipeline proposed here provides the option to do enough trials in simulated environments. Once the simulation trials are conducted and the path is determined, the real-world deployment of the robotic system would be more secure and more likely to achieve its goals.

2.2. Related Work

Every robotic system needs a *plan* to perform the actions to accomplish its goals. Also, every robotic system deals with motion, whether part of it or the whole system moves. Designing the set of actions needed to accomplish a movement from one point to another is called *Motion Planning*. The classical approach divides the process into four steps [LaV11c, pp. 108-118]: first, a feasible path is found; second, the path is smoothed to account for constraints in the robot movement. For example, an autonomous robotic boat cannot do sharp turns, that is also the case for regular boats; third, the path is revisited to account for dynamic constraints; that is, the boat cannot immediately accelerate or decelerate; and finally, a controller is designed to keep the robot in the path. Each one of these steps has several challenges for research. Some strategies such as Planning Under Differential Constraints aim to address several of these steps in a single consideration [SJP15][JP12][LL06]. In this sense, we present a strategy that accounts for feasible

paths, while considering kinematics and dynamics.

Complementary to this, computers have dramatically increased their capabilities even faster than expected by Moore's Laws [M+65]. By 2021 the components of the computational processors are as small as 2 nm [McC21], which translates into smaller and faster computers, also more efficient in energy consumption. These technologies have opened the gates to a new generation of autonomous robots, particularly for the aquatic environments, where relative small platforms are being used to conduct exploration exhibiting endurance of days and weeks. Also allowing for on-board advanced calculation despite the size of the embedded controllers; these, of course, impacts the complexity of the controller mentioned on step four, and so it is also true for re-planning strategies.

One domain that has greatly benefited from these technologies is gaming consoles. the higher computational strength allows for making calculations and generating realistic environments as a player moves around a digital world. Furthermore, the use of artificial intelligence techniques have been used to make these world generation models look photo-realistic [RAK21]. These kind of Simulations have been used to train robotic systems such as self-driving cars [GMF19][Mus], warehousing operations [VDP+18], householding [VPSP17] and other domains. The same technology is also used to train humans in immersive simulations for safe learning industrial equipment manipulation [CLY+20]. Following these ideas, in this work a simulated environment is used to explore the surroundings, as the real vehicle would, to find a feasible path in a form of a policy. The dynamics for the vehicle are emulated using the operating system of the actual robots and the outcome is accounted by the simulator, that is somehow an instance of the strategy hardware-in-the-loop [Led99]; this is very convenient since the policy obtained can be transferred into the real robots, the last step is known as sim-to-real [RVR+17][JKR+19][ZQW20], i.e., a transference of the data learned during the process into the real robot. There is an

extra step in between hardware-in-the-loop and sim-to-real which is Virtual Reality for Robots (VRR) [SNL20], after obtaining the policies the sensors of the actual robot can be fooled to believe it is moving in the real world environment, but the data spooled to the sensors is still being generated by the simulator by keeping a "ghost" of the robot and calculating what the sensor would "see", the generated sensed data is sent through the operating system.

2.3. Problem Formulation

Given a marine environment $\mathcal{W} \in \mathbb{R}^3$. Let $\mathcal{O} \in \mathcal{W}$ represent the set of locations that are inaccessible for a vehicle $\mathcal{A} \in \mathcal{W}$. The robot's position and pose is described by a configuration $q \in \mathcal{C}, q = (x, y, z, h)$, where $\mathcal{C}$ is the set of all possible configurations, x, y, z are coordinates and h is a unit quaternion representing the orientation in space. Let $\mathcal{A}(q)$ represent the robot transformed to configuration q, then $\mathcal{C}_{obs} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset\}$ is the obstacle region and $\mathcal{C}_{free} = \mathcal{C} \setminus \mathcal{C}_{obs}$ is the free region, i.e., the configurations where the robot does not collide into any obstacle.

Problem I: Getting feasible trajectories from BlackBox marine simulator
Given a starting configuration q_S and a set of goal configurations Q_G, determine if there exists a set of configurations $\tilde{X} = \{q_0, q_1, ..., q_n\}$ such that each configuration q_{i-1} can be reached from q_i, and $q_0 = q_S, q_n \in Q_G$. The set of configurations $\tilde{X}$ is called a trajectory, and when $q_0 = q_S, q_n \in Q_G$ it is called a feasible trajectory.

Problem II: Trajectory Risk Analysis
Given a set of feasible trajectories $\hat{X} = \{\tilde{X}_1, \tilde{X}_2, ..., \tilde{X}_n\}$ determine which is the least hazardous trajectory to reach the goal region.

2.4. Methods

Algorithm 1: SIMULATEDRRT(S, U, T, n, q_S, Q_G)

Input: S,U,T,n, q_S, Q_G – a simulator, a set of actions, the duration of the action, the number of steps for the algorithm, the starting position, the goal region

Output: R – a graph (V,E) that might contain a feasible path to the Goal region.

```
1  R.V ← q_S        R.E ← ∅
2  for step in n do
3      if random() < 10 then
4          q_r ← random_point(Q_G);
5      else
6          q_r ←random_point(S);
7      q_closest ←R.closestNode(q_r);
8      for u in U do
9          q ← S.execute(u, q_closest);
10         if q then
11             R.V.add(q);
12             R.E.add(q_closest,q);
13 return R
```

It is proposed here a strategy based on the well-known RRT algorithm (Rapidly exploring Random Trees) [LaV98]. A variation of this algorithm was considered to take into account the kinematics, dynamics, and uncertainty. The strategy presented also allows for complex landscapes with complex obstacle regions. It is A BlackBox strategy that lets a

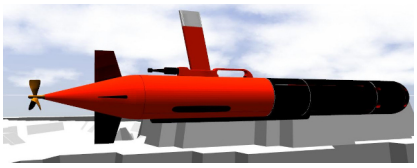


Figure 2-1: Light Autonomous Underwater Vehicle (LAUV).

physics simulator take the heavy lifting of calculating the dynamics of the vehicle, perform the actions and deal with the uncertainty associated as described in [LaV06a]. The robot considered in this document is a torpedo-shaped Light Autonomous Underwater Vehicle (LAUV) [dSTMdS07][SMC+12], shown in figure **2-1**. It has a single propeller, and a set of four fins disposed around the vehicle with 90° of separation. With this configuration, the vehicle can roll, pitch, and yaw by combining the rotation of the fins. Also, the propeller can move clockwise and counterclockwise to move forward or backward. At each time, the vehicle has a position in the space (x,y,z) and orientation (roll, pitch, yaw) that combined together define the state of the vehicle uniquely, and it is known as configuration.

The RRT variation used is described next: Starting from the desired initial configuration q_S, the root of the tree starts to grow toward the branches. The algorithm selects a random position in the environment and finds the closest node in the current tree to that position. A set of k actions are performed from the mentioned closest node for a period of time T, and the reached configurations are added to the tree saving: the parent node, the action performed at the parent, and a set of way-points visited during T. The algorithm continues selecting random positions in the environment; a portion of the time (10 %) the goal position is selected to guide the search by adding a bias toward the goal.

The simulator used is Gazebo 9 [KH04]. This open-source software provides physics simulation and rendering capabilities. The underwater physics are provided by a Gazebo plugin available in the ROS project (Robot Operating System) [QGC+09]

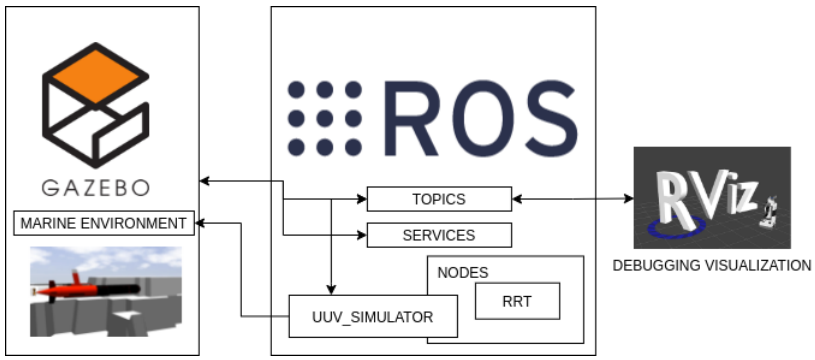


Figure 2-2: Simulation General Scheme: Gazebo calculates the physics and renders the simulation. ROS orchestrates the communication between the environment and the robots, also directs actions and messages through Topics and Services. RViz is utility software that allows visualizing all the data traversing the topics.

UUV_Simulator [MSV+16]. ROS is a set of libraries that allow both the simulation and deployment of robotic applications. The algorithm described previously was implemented as a ROS Node, which is the unit execution block in the ROS environment as shown in Figure **2-2**. The ROS Node is a script that can be written in C++ or Python; the RRT implementation described was coded in python. One of the advantages of these settings is that the simulator only knows the building blocks physics, that is: the physics of the propeller, the physics of the fins, the physics of movement for the shape of the robot in the water, and all this put together in an iterative simulation predicts realistically the outcome for the actions taken. This outcome is established without actually modeling the robot kinematics nor considering the particular dynamics of the robot; that is why this strategy is called BlackBox modeling: the simulator serves as a BlackBox and provides the outcomes.

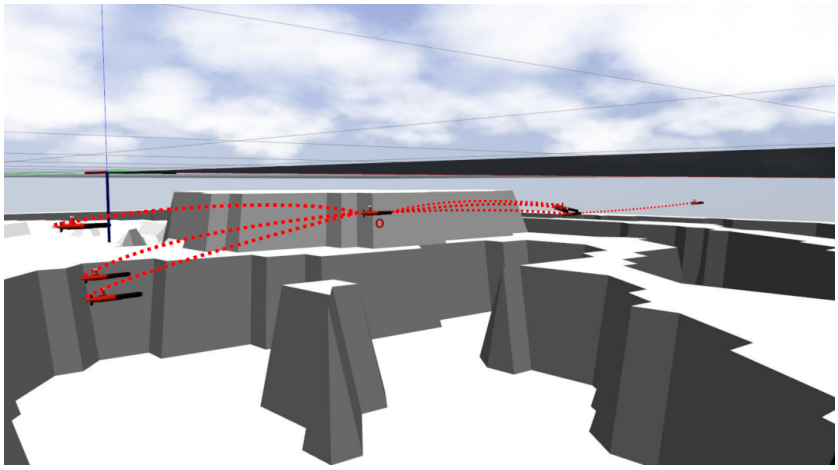


Figure 2-3: Representation of the RRT in simulation. The LAUV marked with letter **O** represents the current configuration to be expanded. The lines represent the different outcomes from performing the actions from configuration **O**

2.5. Experimental Results

Figure **2-3** shows the algorithm in action. Particularly, shows all the outcomes after executing the set of available actions. One big advantage of simulated environments is replicability: based on a selected seed, the exact same simulation can be repeated several times, if a scenario is found where a modification needs to be performed on the plan or on the settings of the vehicle, it can be fully tested before configuring the actual robots. Another benefit of this property is exploited in this work, once a particular action is executed for the defined period, the vehicle can be set back to the initial conditions, including the position and velocities of the vehicle, to perform another action, this behavior would be impossible in real life.

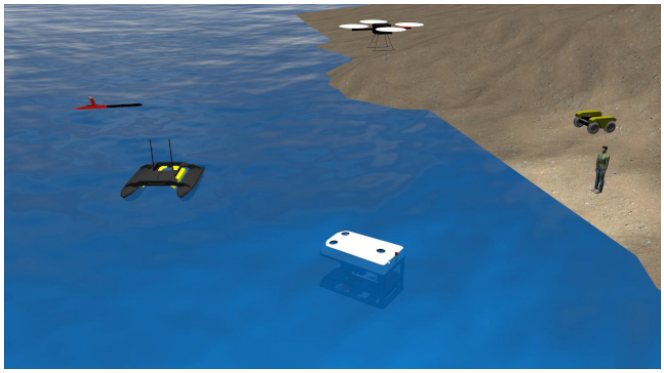


Figure 2-4: Simulated environment where multiple robots are deployed to cover different regions or to complement the coverage, other entities such as human operators can also be considered.

2.6. Conclusions and Future Work

The scheme of simulations presented in this work, also allows to select safer paths; since in the simulation the information of the world is complete, the distance to every obstacle can be calculated all the time. With this information, hazardous maneuvers that takes the vehicle too close to an obstacle can be avoided. This is also the case for compliance with regulations: if a mission needs to be executed in a region where some particular restrictions apply, the feasible path can be obtained in such a way that aims to prevents accidental noncompliance.

The safety can also be extended to other parties in the environment. As shown in Figure **2-4** multiple robots can be simulated, the communication among them can also be simulated and safe zones can be defined to maneuver while preserving safety for human operators and other vehicles.

Finally this simulated environments also apply to ground vehicles in the presence of delicate goods, as is the case for warehouses shown in Figure **2-5**. Real world disasters

Figure 2-5: Delicate obstacles: the warehouse settings is a special case in which the obstacle region is not only unsafe for the vehicle but for the business itself. Safety planning for autonomous vehicles is critical.

have been recorded in surveillance cameras showing how a minimal collision with the obstacle region can destroy the complete warehouse, safer paths are critical to prevent that kind of outcomes. Also, the obstacle region can be re-defined based on multiple deployments and navigations of the simulated warehouse. In such a manner that design is also possible through simulations. Figures **2-4** and **2-5** were constructed following the same strategy, for which the operating system is common for both simulations and real robots, this allows for completing the three steps: hardware-in-the-loop, virtual reality for robots and sim-to-real. The evident further step of this work is to transfer the obtained paths to real world robots.

3 Computing Feedback Plans from Dynamical System Composition

This work was presented at IEEE 15th International Conference on Automation Science and Engineering (CASE).[1]

Computing plans for systems with differential constraints is a fundamental component in numerous robotic applications. Most previous approaches are based on creating motion plans between an initial and a goal location. However, a more robust approach is to compute *feedback plans* over the entire configuration space to account for uncertainty in the robot's motions. In this paper, we therefore propose a new method that constructs a feedback plan by incrementally composing the long-term behavior of the robot's motions for a set of actions. Our method takes advantage of dynamical system analysis techniques and efficient combinatorial algorithms. We implement our method in simulations considering a robot under a simple bouncing behavior. A feedback plan for the robot to reach the goal region starting from any location of an environment is successfully constructed using the implementation of our method. Our method is also applicable to non-linear systems with uncertainty.

[1]The authors acknowledges Tauhidul Alam, Leonardo Bobadilla and Ryan N. Smith for their contribution in the developements of the ideas, simulations and guiding.

3.1. Introduction

Feedback motion planning under differential constraints is the problem of computing plans for global robot navigation from any location of the environment to a goal region. This is also a fundamental problem in control and robotics. In traditional path planning approaches, a collision-free path τ is computed from a known initial configuration to reach a goal region, then τ is smoothed to satisfy differential constraints. A trajectory for a robot considering its geometry, motion model and time is then designed that attempts to follow the smoothed τ [LaV11a].

However, there is no additional information as feedback in the traditional approach to tackle the unpredictability of future configurations. Furthermore, all the uncertainties and disturbances are predominantly not taken into account in the system model. To overcome this issue, it is crucial to compute a *feedback plan* when the initial configuration is not known or the plan execution is not predictable due to disturbances or errors in the system model. This feedback plan is constructed to ensure that the robot knows which action to take when an unexpected configuration appears during the plan execution [LaV06b].

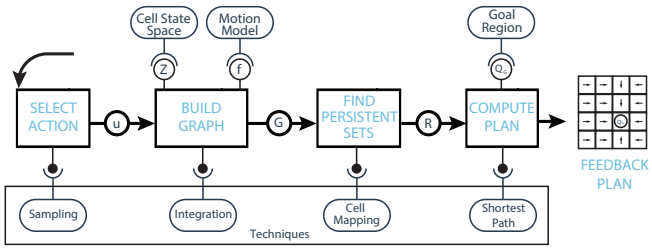


Figure 3-1: This pipeline summarizes our proposed feedback plan construction method. The modules inside the techniques box can be implemented using different algorithms that are efficient for particular problems. The pipeline is iterated as many times as required to improve the feedback plan.

The construction of a feedback plan is significantly more challenging than path planning. The feedback motion planning methods suffer the curse of dimensionality as they do not work well for larger dimensions and high-resolution sampling. Nonetheless, these methods have some advantages over path planning methods. One advantage is that a feedback plan helps to adjust the paths or trajectories of robots in the presence of modeling errors or any other form of uncertainty in future configurations. Another advantage is that a feedback plan can work as a dynamic replanner if the robot deviates from its path during the execution; this is possible because the plan is calculated over the entire free configuration space. Moreover, a feedback plan can be utilized both in open- and closed-loop control.

This paper aims to examine a numerical feedback planning method for a robot to reach a goal region of an environment. The complexities of the value iteration and the policy iteration based feedback planning methods [KLM96] rely on the discretization over the action space. However, our method does not discretize the action space. Thus, the complexity of our method does not depend on the cardinality of the action set. This improves the running time of our method which makes it suitable for online applications.

We also use a dynamical system technique called *generalized cell-to-cell mapping* (GCM) [Hsu13] to find the long-term behavior of the robot's motions for a set of available actions.

The main contribution of this work is to propose a new method for constructing a feedback motion plan by composing the long-term behavior of different actions of a robot. First, we select an action from the action space and calculate the long-term behavior of the robot's motion for the selected action using GCM. Then, we compute an initial feedback plan from the calculated long-term behavior. We repeat the same process and combine the calculated long-term behavior for different actions until a goal region can be reached from any configuration of the environment. At the end of the process, the feedback plan

is fully constructed. This feedback plan provides the robot's action at every configuration of the environment. Even though our method is tested with a linear system, it is also applicable to stochastic and nonlinear systems.

The rest of the paper is organized as follows: Section 3.2 describes related efforts in the literature that serves as a starting point to elaborate on our method. Section 3.3 introduces our environment and robot motion model, and formulates the problem of interest. Section 3.4 presents our proposed method developed to address the problem of interest. Section 3.5 presents the model used to test our approach and some simulation results. Finally, we show some conclusions and directions for future work in Chapter 6.

3.2. Related Work

Sampling-based motion planners such as the Probabilistic Roadmaps (PRM) [SLOK96] and Rapidly-exploring Random Trees (RRT) [LaV98] along with their asymptotically optimal versions such as RRT* and PRM* [KF11] have been used ubiquitously in robotics. Our work is more closely related to variants of these methods that incorporate differential constraints [SJP15, WVDB13]. Sampling-based approaches find an open loop trajectory. However, a feedback plan will be more useful when uncertainty is present. As such, our effort is related to approaches [AMCA14b, hJKF15, YF16b] that use sampling-based ideas to compute feedback plans.

Connected to our idea, Mason introduced the metaphor of a funnel for a feedback policy which collapses a large set of initial conditions into a smaller set of conditions [Mas85b]. Burridge et al. [BRK99] extended this idea as a sequential composition of feedback policies or funnels from a set of initial conditions to a goal region.

Tedrake [TMTR10] presents a feedback motion planning algorithm which uses the

locally valid linear quadratic regulator (LQR) controllers into a nonlinear feedback policy that probabilistically covers a reachable subset of configuration space with a relatively sparse set of trajectories.

Majumdar and Tedrake presented a pre-computed library of funnels using the convex optimization that represents different maneuvers of the system within bounded disturbances or parametric model uncertainties [MT17]. However, the offline computation of the robust funnel library will make this approach inefficient when there are additional changes in the uncertainties at runtime as the whole funnel library is required to be re-computed in this scenario. A global vector field computation algorithm in configuration spaces for smooth feedback motion planning is presented in [ZLM09]. A potential field can be used in feedback control [RK92] to define a navigation function. Nonetheless, this potential field method suffers from local minima and does not scale well.

We built this work on our previous studies. Previously, we employed GCM in common path planning and coverage for a resource-constrained robot [ABS17b] in which the robot utilized only a clock and contact sensors to execute its simple bouncing behavior. In other studies, GCM was also applied in deployment [ARBS18b] and localization [ARBS18d] for minimally-actuated drifting vehicles. The long-term water flow patterns were analyzed from the water flow fields in order to deploy and localize these drifting vehicles. However, we did not take into account the robot's action for finding the long-term water flow patterns since the drifting vehicles are passive.

3.3. Preliminaries

This section defines the discrete planning model and formulates our considered problem.

3.3.1. Model Definition

We consider a two-dimensional environment as a *workspace* $\mathcal{W} = \mathbb{R}^2$ containing an *obstacle region* $\mathcal{O} \subset \mathcal{W}$. We assume that the workspace $\mathcal{W}$ is bounded. The *free space* in the environment is defined as $\mathcal{E} = \mathcal{W} \setminus \mathcal{O}$. A robot, defined as $\mathcal{A} \subset \mathcal{W}$, can move in the free space $\mathcal{E}$.

The *configuration space* of the robot $\mathcal{A}$ is denoted as $\mathcal{C} = \mathcal{E} \times S^1$ where S^1 represents the robot's orientations between 0 and 2π. Let $q \in \mathcal{C}$ denote the configuration of robot $\mathcal{A}$, in which $q = (x_t, y_t, \theta)$ where (x_t, y_t) represents its position and θ represents its orientation. Specifically, a configuration $q_I \in \mathcal{C}$ is designated as the robot's *initial configuration*. A set of configurations $Q_G \subset \mathcal{C}$ is designated as the robot's *goal configurations* or the *goal region*. The free configuration space is defined as:

$$\mathcal{C}_{free} = \{q \in \mathcal{C} \mid \mathcal{A}(q) \cap \mathcal{O} = \emptyset\}, \tag{3-1}$$

which is the set of all configurations at which the transformed robot $\mathcal{A}(q)$ does not intersect the obstacle region $\mathcal{O}$.

Let $U(q)$ denote the *action space* for each configuration q, which represents the set of all possible actions that could be executed from q. Therefore, the set U of all possible actions over all configurations is defined as:

$$U = \bigcup_{q \in \mathcal{C}} U(q). \tag{3-2}$$

Each action $u \in U(q)$ is an m-dimensional vector where m is the number of possible actions from q.

When u is applied from the current configuration q, a new configuration q' is produced as specified by a *configuration transition function* f. Let u_T be the *termination action* for

which, when u_T is applied from a configuration q then q is again obtained as a result. This function f can be utilized in expressing a *configuration transition equation*:

$$\dot{q} = f(q, u) = f_u(q). \tag{3-3}$$

The new configuration q' can be obtained by integrating the *configuration transition equation* in a period of time $\triangle t$:

$$q' = \int_0^{\triangle t} f_u(q)dt. \tag{3-4}$$

Eq. 3-4 represents the motion model of the robot. A *feedback plan* π is defined as a function $\pi : \mathcal{C}_{free} \rightarrow U$ which produces an action $u = \pi(q) \in U(q)$, for any configuration $q \in \mathcal{C}_{free}$, to reach the goal region Q_G. If the goal region is reached, then the termination action should be applied. This is specified as part of the feedback plan: $\pi(q) = u_T$, if $q \in Q_G$.

3.3.2. Problem Formulation

When a robot knows its initial configuration, then a plan can be computed by a sequence of actions from the known initial configuration. However, this approach is memory intensive since a replanner will be required whenever the robot deviates from the calculated plan. We assume here that the robot does not know its initial configuration *a priori*.

In this context, it is appropriate to find a feedback plan that maps every configuration to an action. A feedback plan is called a *solution* to the problem if it causes the goal region to be reached from every configuration in $\mathcal{C}_{free}$.

This motivates us to formulate our problem of interest as follows:

Problem 1. Constructing a feedback plan: *Given a workspace $\mathcal{W}$, a set of goal configurations Q_G, a set of possible actions U, and the motion model of a robot f, construct a feedback plan π for the robot to reach the goal region Q_G from any initial configuration.*

3.4. Methods

In this section, we describe our cell decomposition of the configuration space. We also provide a brief summary of the generalized cell-to-cell mapping method, and the pipeline of constructing a feedback plan.

3.4.1. Cell Decomposition

First of all, we discretize the $\mathcal{C}_{free}$ into a finite number of cells. This discretized configuration space is termed as the *cell state space*. We define the set of cells as $Z = \{1, 2, \ldots, z_i, \ldots z_N\}$ where N is the total number of cells and $z_i \in Z$ corresponds to a configuration q in $\mathcal{C}_{free}$. The movement of $\mathcal{A}$ over $\mathcal{C}_{free}$ can be represented as a series of cells in Z.

Let $e(k)$ be the cell representing the robot's configuration at a time $t = k\triangle t$ with $k = 0, 1, \ldots$ and $\triangle t$ been the time interval between two configuration inspections.
If the system evolution is deterministic, then a function $P : Z \to Z$ is defined such that:

$$e(k+1) = P(e(k)). \tag{3-5}$$

This system evolution P is called a *simple cell-to-cell mapping* (SCM) [Hsu13] in the cell state space.

3.4.2. Generalized Cell-to-Cell Mapping

If the evolution is non-deterministic and $p(k)$ is the probability distribution of the system configurations at the time $k\triangle t$, then the evolution of the system is given by:

$$p(k+1) = \mathcal{P}p(k) = \mathcal{P}^{(k+1)}p(0), \tag{3-6}$$

where $\mathcal{P}$ is the one-step *transition probability matrix*, $\mathcal{P}^{(k)}$ denotes the k-step transition probability matrix and $p(0)$ is the initial probability distribution, also let p_{ij} be the $(i,j)th$ element of $\mathcal{P}$, denoting the transition probability from cell i to cell j; and let $p_{ij}^{(k)}$ be the $(i,j)th$ element of $\mathcal{P}^{(k)}$, denoting the k-step transition probability from i to j, i.e., the transition probability to reach cell j after k steps are executed starting from cell i.
This representation of probabilistic transitions corresponds to the *generalized cell-to-cell mapping* method [SH90]. The transition probability matrix $\mathcal{P}$ can be generated from a weighted directed graph $G = (V, E)$, also termed as a weighted digraph. The set of vertices V is equivalent to the set of cells Z, such that $\forall v \in V, \exists z \in Z$, and $\forall e \in E$ that represent a relation between vertex v_i and vertex v_j then $p_{ij} > 0$.

A *strongly connected component* defined as $SCC \subseteq V$ is a subset of vertices such that $\forall v_i, v_j \in SCC \Rightarrow p_{ij}^{(k)} > 0$ for some k. A *persistent set*, SCC_l, is a SCC such that $p_{ij} = 0$ for every $v_i \in SCC_l$ and $v_j \notin SCC_l$. A SCC that is not a persistent set is called a *transient set*. In addition, the *transitive closure* of a graph G is the set of reachable vertices for each vertex in G. It is important to note that any two vertices in a persistent set have the same transitive closure and the size is equal to the size of the SCC that contains them.

3.4.3. Feedback Plan Construction

The overall pipeline of our feedback plan construction method is illustrated in Fig. **3-1**. It first selects an action from the action space. For the selected action, a graph G is built from the cell state space through the robot motion model.

The G is further analyzed to determine the persistent sets and the transient sets. A composition of these persistent sets is used herein to construct a feedback plan π by finding common paths between them. The transient sets are also used to connect between the persistent sets. Note that every different action produces different persistent and transient sets. Thus, a robot can navigate among different regions of the workspace by changing the actions and escaping the persistent sets for an action.

Algorithm 2 constructs a feedback plan π. In line 1, the plan π is initialized for every configuration. Algorithm 2 runs until every configuration in π has an action assigned. In line 2, the function CELL(q) returns the cell that maps to the configuration q. In line 3, an action u is sampled from the action space U. Lines 4 and 5 call Algorithms 3 and 4 respectively to construct a graph $G = (V, E)$ and analyze its transient and persistent sets. In line 6, the plan π is updated using the Dijkstra's shortest path algorithm [CLRS01]. We define $R = (V, E_1)$ as a new graph extracted from the persistent and transient sets of G in which $E_1 \subset E$. The complexity of the shortest path algorithm is $O(|E| + |V|log|V|)$. This complexity dominates over the complexities of Algorithms 3 and 4.

Algorithm 3 builds a graph G for a particular action u. From every cell in Z, the configuration transition function f is integrated S times (steps). Every transition found in the robot's movement is recorded into G. In line 1, G is initialized with vertices Z and no edges. In line 3, the function CONFIGURATION(z_i) returns the configuration q mapped

Algorithm 2: ConstructPlan(Z, Q_G, U, f, S, $\triangle t$)

Input: Z, Q_G, U, f, S, $\triangle t$ – the cell state space, the goal region, the action space, a configuration transition function, the number of steps of the robot's motion, the time interval between two configuration inspections

Output: π – a feedback plan.

```
1 π ← ∅
2 while ∃q, s.t. π[CELL(q)] = ∅ do
3     u ← SAMPLE(U)
4     G ← BUILDGRAPH(Z, u, f, S, △t)
5     R ← GCM(G, |Z|)
6     π ← SHORTESTPATH(R, Q_G)
7 return π
```

by the cell z_i. In line 6, a new configuration q' is obtained by integrating f using the action u, and $\triangle t$ time. In line 9, $G.E_{kj}$ holds the weight of the edge between vertices k and j. If this edge does not exist in $G.E$ then it is first added. Here the transitions between pairs of vertices are accounted as they are encountered.

The complexity of Algorithm 3 is $O(|Z| \cdot S \cdot I)$ where I is the complexity of integrating f (line 6). Since S and I are constants in our models, the complexity is $O(|Z|)$.

Algorithm 4 constructs a new graph R based on the persistent and the transient sets of G. For this, it finds the transient closure TC using the strongly connected components SCC of G. Then, it finds those SCC whose elements have the same transitive closure. In line 1, R is initialized with Z as vertices and no edges. Lines 2 and 3 calculate the strongly connected components and transitive closure using the Tarjan's algorithm [Tar72], the complexity of this algorithm is $O(|V| + |E|)$. In this type of graphs, i.e., those describing

Algorithm 3: BUILDGRAPH(Z, u, $f, S, \triangle t$)

Input: Z, u, $f, S, \triangle t$ – the cell state space, an action, a configuration transition function, the number of steps of the robot's motion, the time interval between two configuration inspections

Output: G – the graph (V,E) resulting from the motion model using u.

```
1  G.V ← Z        G.E ← ∅
2  for z_i in Z do
3      q ← CONFIGURATION(z_i)
4      k ← z_i
5      for t ← 1 to S do
6          q' ← INTEGRATE(f, u, q, △t)
7          j ← CELL(q')
8          if k ≠ j then
9              G.E_kj ++
10             k = j
11         q ← q'
12 return G
```

a grid in 2D, each vertex has at most 8 edges, this means that $|E| \leq 8 * |V|$. Therefore, the complexity reduces to $O(|V| + 8|V|) = O(|V|) = O(|Z|)$. Lines 6-14 determines if a SCC is persistent or transient. The complexity is $O(|Z|)$, since this is checked once for every cell.

In lines 16 and 19, $G.E_i$ means the edges starting from i in G. In lines 17 and 20, $R.E_{1ki}$ holds the weight of the edge between vertices k and i, if this edge is not in $R.E_1$ then it is created. The weight is set proportional to the number of cells in the TC. Recall that this is equal to $|SCC|$ for persistent sets. The more cells in a SCC, the more likely this set is to be used in the final plan. Considering that the graph is built from an action previously selected, our algorithm prefers to use the actions that produce fewer subgraphs, i.e., largest SCCs. In line 20, the weight is penalized so that the shortest path in the line 6 of Algorithm 2 prefers to use the edges inside the persistent sets. Note in lines 17 and 20, for an edge $i \rightarrow k$ in the graph, another edge $k \leftarrow i$ is created in R, this is intentional to run the shortest path algorithm backward from the target set. The sets *persistent* and *transients* sum up to $|Z|$, thus, the overall the complexity is $O(|Z|)$.

3.5. Experimental Results

Our proposed method was tested using a *simple bouncing robot*. The bouncing robot follows a simple strategy: the robot will continue its straight motion unless a wall is hit, in this case, it *bounces* and rotates by a specific angle and continues its straight motion. The action space defined for the bouncing robot is $U = [0, 2\pi]$ which represents the bouncing angle.

Fig. **3-2** shows the bouncing behavior for an angle $\approx 15°$. Actions are in radians which are approximated to degrees here. In this example, a larger map is shown for better

Algorithm 4: GCM(G, $|Z|$)

Input: G,$|Z|$ – a graph, the number of cells in the mapping

Output: R – a graph (V,E_1) where $R.E_{1ij}$ is the weight of the edge between vertices i and j.

```
1  R.V ← Z        R.E_1 ← ∅
2  SCC = stronglyConectedComp(G)
3  TC = transitiveClosure(G, SCC)
4  persistents = NIL
5  transients = NIL
6  for component in SCC do
7      is_persistent = True
8      for j in component do
9          if len(component) ≠ len(TC[j]) then
10             is_persistent = False
11             transients.add_all(component)
12             break
13     if is_persistent then
14         persistents.add_all(component)
15 for i in persistents do
16     for k in G.E_i do
17         R.E_1ki ← min(R.E_1ki, 1/len(TC[i]))
18 for i in transients do
19     for k in G.E_i do
20         R.E_1ki ← min(R.E_1ki, penalty)
21 return R
```

understanding the nature of the robot's behavior in which the robot is moving straight until a wall is hit. Examples in Figs **3-3** and **3-4** are presented using a smaller map for better appreciating different persistent sets. The robot starts at z_i facing east and moves straight east until it hits the wall. After the hit, the robot rotates following the action u until it can continue the straight motion. For the action $u \approx 15°$, four bounces occur when the first wall is hit, then a $\sim 90°$ rotation is performed and the robot can continue its straight movement. It is worth mentioning that this workspace is discretized into cells and that a set of configurations map to a single cell. Whenever the cell changes, this change is registered as a transition in the graph G. Starting from every cell in Z, the number of steps S and their transitions are recorded into G. The graph G is fully constructed at the end of this process. For a particular bouncing angle u, a specific graph is constructed.

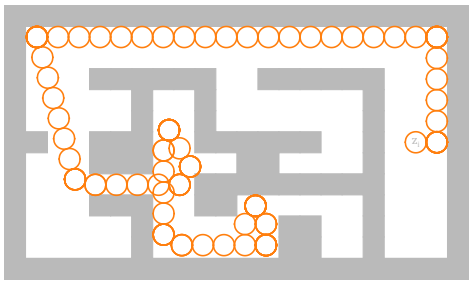


Figure 3-2: A generated path from the simulation of a bouncing robot (depicted by orange circles). The illustrated path is simulated for $S = 120$ steps starting from cell z_i facing east. The bouncing angle is $u \approx 15°$

.

In Fig. **3-3**(a) the graph G was constructed using $u \approx 25°$; for this action, a persistent set was found and the cells that comprise it are shown as red circles. Fig. **3-3**(b) shows cells (red and green circles) in two persistent sets found in the constructed graph G for another action $u \approx 26°$. Every cell is in either a persistent or a transient set. Nevertheless,

in our implementation, we disregarded persistent and transient sets consisting of a single cell. The edges inside the persistent set are kept for the calculation of the plan π. This complete process is repeated for different actions until the following three conditions are met:

1. Every cell in Z is in a persistent or in a transient set.

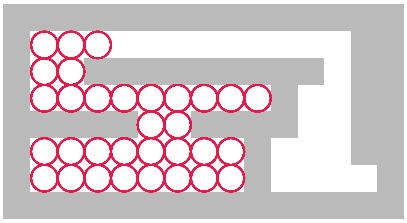
2. At least one cell in Q_G is in a persistent set.

3. A path exists between every cell in Z and at least one cell in Q_G.

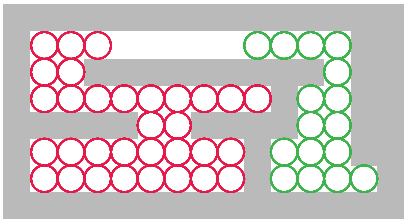
Fig. **3-4** illustrates the final feedback plan. The red circles represent the cells in Z. Configurations with the same (x,y) position and different orientations are superimposed in the Fig. **3-4**. The green lines going from the center of a cell z_i represent the actions $\pi(z_i)$ for the final plan π. The blue segments starting at the lower left corner cell z_0 show the followed path by a robot selecting the actions from the plan π. The robot finally arrives in the goal region marked as G following the path based on the plan π. Note that, in Fig. **3-4**, the goal region G is comprised of a set of cells at the same location but they have different orientations.

3.6. Conclusions and Future Work

This chapter presented a numerical method to compute a feedback-based plan in polynomial time that does not rely on a discretization of the action space. Therefore, the complexity does not depend on such discretization. Instead, the complexity of the method only depends on the number of cells defined in the free configuration space. The definition of this number of cells depends on the resolution desired, and could also be



(a)



(b)

Figure 3-3: *Persistent Sets*. (a) Cells (red circles) in one persistent set for a constructed graph using one action $u \approx 25°$. (b) Cells (red and green circles) in two persistent sets for a constructed graph using another action $u \approx 26°$.

applicable to a multi-resolution approach. The proposed method used the notion of the long-term behavior of the robot's motions using samples from the action space. We found the persistent sets and the transient sets as the long-term behavior of a robot's selected action. A feedback plan was successfully constructed by decomposing the long-term behavior of the bouncing robot's motions for different actions to reach a goal region from any location of an environment following a simple bouncing strategy.

If a discretization D of the action space U is fixed, and the action that can accomplish a particular task is not selected in D, then an algorithm A could not solve this task using D. Hence, our method outperforms A. Additionally, the low computational capabilities needed by our method makes it suitable for on-board applications. Furthermore, the modularity of our method also allows it to be used in a collaborative fashion by multiple

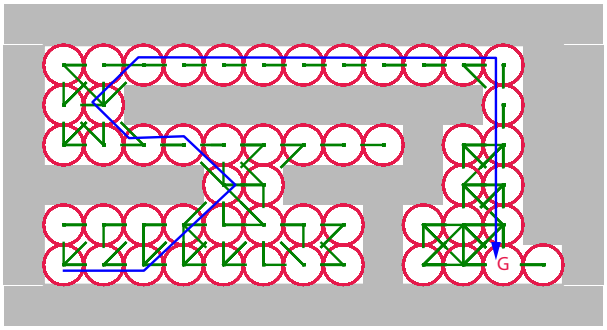


Figure 3-4: An illustration of the computed final plan π. Red circles represent a set of configurations with the same position (x,y) and different orientations. Green lines pointing out from the cell center z_i represent the direction of the next cell that will be reached after applying an action $\pi(z_i)$. Multiple green lines are drawn per cell to account for multiple orientations. The goal region is marked as G. The Blue lines show a path followed by a robot from z_0 (lower left corner) using the plan π.

robots. Our method is general and it only needs the equation of motion f and the action space U as input. This equation of motion can be nonlinear and might also include uncertainty. We tested our method with the simple bouncing strategy of a robot. However, our immediate goal is to test it in other commonly used autonomous vehicles [PČY+16b] and more concretely, we will be focusing on robots with higher degrees of freedom. Our method can be considered an alternative to popular methods such as value iteration and policy iteration [BBBB05] and feedback plan generation for systems with uncertainty [SB18]. Our immediate task is to compare experimentally our method to traditional methods in order to analyze its advantages and disadvantages. In our future research, we will look at different schemes to sample the action space. Our proposed method opens multiples avenues in this direction. An interesting approach to decide the sampling strategy based

on learning from previous samples would be a major contribution. Others algorithms for computing and replanning the shortest path, such as ARA*, D*, or RRT*[YF16b] would benefit to improve our method presented in this Chapter.

4 Toward Simultaneous Localization and Mapping in Aquatic Dynamic Environments

Part of this work was finalist in the student's poster competition at IEEE Oceanic Engenieering Society OCEANS Conference. [1]

4.1. Feedback Motion Planning on a self updating map.

Given the oceanic environment's stochastic nature and the significant spatial and temporal scales of most ocean processes, sampling is sparse at best. Predictive models are necessary to augment decision-making to ensure that robots are in the right place and time *for optimal or efficient sampling. Although interesting ocean features are aperiodic and stochastic in nature, they do exhibit coherent structure, which can generally be* seen *from aerial or remotely sensed data. In this paper, we provide a strategy to build a dynamic representation of an aquatic environment, based on different water parameters,*

[1] The author acknowledges Tauhidul Alam;Gregory M. Reis, Leonardo Bobadilla, Ryan N. Smith for their contribution in the developments of the ideas, the data and simulations

that allows a vehicle to find its position within the representation and keep the mentioned representation up to date simultaneously using a sampling strategy.

4.2. Introduction

The complexity of oceanic processes and the chronic under-sampling of the coastal oceans, and marine ecosystems in general, inevitably leads to the persistent call for sustained exploration and sampling efforts [RP03]. Recently, it has become more cost-effective and logistically sustainable to use Autonomous Underwater Vehicles (AUVs) and Autonomous Surface Vehicles (ASVs) as platforms of choice for marine sampling efforts, rather than traditional ship-based methods. The scientific literature has accepted the use of these assets, and operational oceanography is increasingly deploying more of these vehicles to observe dynamic processes [DMM+11]. These robots provide a non-intrusive and repeatable way to observe the oceans up close. They also allow augmenting the data provided by existing methods with endurances ranging from a few days to months. Moreover, they have speeds commensurate to observing spatiotemporal scales of the evolution of dynamic or episodic events, such as algal blooms, ocean fronts, and Lagrangian Coherent Structures. Given the stochastic environment and the large (> 50 km^2) spatial and temporal scales of significant ocean processes, sampling is sparse at best. Therefore, predictive models are necessary to augment decision making to ensure that robots are in the *right place and time* for sampling, and consequently registering the phenomena. However, no single model provides an informed view or representation of an ocean feature that enables intelligent or efficient sampling in a principled manner. Thus, forecasting where a robot should sample in the immediate future is challenging and has previously been addressed by pre-planned missions designed by oceanographic domain experts. To address this sampling design problem, i.e., deciding where to sample next, the task of localization must be addressed first. Ideally, an underwater robot deployed in an unknown aquatic environment will create a representation of its surroundings, localize in it, and use this map to

navigate. This problem is known as the *Simultaneous Localization and Mapping (SLAM)* problem [TBF05, T$^+$02, Thr00]. Here, we present the first steps towards developing a strategy that uses a predicted representation of an aquatic feature based on interpolated data to solve the SLAM problem. The rest of the chapter is organized as follows: Section 4.3 presents related work to the strategy proposed. In section 4.4 the model and problem are defined. Section 4.5 presents the methods to acquire and process the data. Finally, some conclusions and ideas for future work are presented in section 4.7.

4.3. Background and Related Work

There have been significant studies in recent years in the utility and implementation of autonomous underwater, and surface vehicles for persistent surveillance of the ocean [JEK$^+$15, SLS15]. Example studies enabled include the dynamics of physical phenomena, e.g., ocean fronts [ZGBR12] and algal assemblages [CSM$^+$08], temperature and salinity profiles [WZ11, SP11, LSYF08], and the onset of harmful algae blooms [SCL$^+$10, ZFP$^+$07, DPM$^+$12, CBKS08]. These works have primarily focused on developing sampling strategies for single, or a small number of, AUVs working in conjunction with stationary sensors, embedded networks, research vessels, and ASVs to maximize data collection for the various physical processes. However, ocean processes are intrinsically linked to the fluid dynamics and underlying bathymetry, yet most research does not explicitly account for these directly.

Although the ocean environment is naturally stochastic and aperiodic, it does exhibit coherent structures that can be exploited. Similar structures have been exploited in other scenarios, such as facial recognition [PEWF08, FLCS12, LSF07], city modeling [FZ03], novel view synthesis for 3D visualization [CMR10], and robotic localization tasks [SE12,

URO+08]. The reason for the limited adoption in the marine environment is that there are still significant engineering challenges for large-scale field deployments. Additionally, the structure exhibited by ocean processes is spatiotemporally dynamic, and we lack a sufficient understanding to know *a priori* how to best model/represent an ocean feature (what it should look like). Hence, it makes sense to leverage these processes' inherent structure or the bathymetry structure that forces these processes to occur in specific regions, given certain conditions, to enhance sampling and further our understanding.

Given this need for stochastic representations of the environment, we propose to develop a representative model to provide a robot an understanding of what it should expect in a given sampling scenario. The specifics and adaptations of how the plan is executed are left to on-board decisions, which are based on comparing the predicted model with samples. The model proposed takes advantage of the following observation: Given a set of water parameter readings, the current position can be narrowed down by determining the locations within a set of maps where the reading could have been taken. As the position gets refined, the maps can also be updated. Consistently, these updates should be incorporated using the readings. Also, the next best location to sample, i.e., where to take the readings from, should be determined to reduce the error variance. This process can be performed iteratively to keep the map up to date and would be particularly useful to keep dynamic environment maps updated and reliable.

From the realization of the SLAM problem's importance in the mid-80s, there have been significant advances in solving it, to a point where the problem is theoretically 'solved' [DWB06]. Despite this, the problem is far from being easily applicable, especially for realistic large and dynamic environments. The conventional solutions are formulated in a framework where a set of landmarks are defined and updated through time, as observations are taken. The vehicle pose is highly correlated to the sensing of the landmarks. This

observation is used to structure the problem in a Bayesian form. For dynamical environments, the landmark solution encounters the problems of displacements, cluttering, and removal of the landmarks. In the case of highly dynamical environments, being oceanic environments one of them, solutions to this problem such as the deletion of discarded landmarks [Bai02] are not suitable.

Most solutions rely on the use of Bayesian filters, particularly the Extended Kalman Filters (EKF) and particle filters [DGA00]. The problems associated with the computational complexity of this kind of solution have been addressed by sparsification of information, partitioned updating, and sub-mapping [BDW06]. Previous work has been developed on map augmenting such as denseSLAM [NGN06], where the authors exploit the use of multi-layered maps; for denseSLAM, additional information such as occupancy, traversability, or elevation is kept to augment a navigation map. This idea is particularly useful for aquatic environments since many properties in the water can be used to determine a region. Also, the sensing of these properties is highly correlated to the vehicle pose and position. All this sheds light on using water properties as landmarks to solve the SLAM problem for aquatic environments. The SLAM problem solution's complexity is quadratic in the number of features [GN01]. The algorithm is also quadratic in space. This computational complexity is a big concern for large environments and augmented representations.

The spatial characteristic and the 'smoothness', i.e., the continuity of the values over the space, of the data collected present an opportunity to sample the data, alleviating the memory burden. Some other variable phenomena that exhibit these properties, for example, soil physicochemical properties [HHS04], has been accurately interpolated using Kriging [Ste12]. This method finds the best unbiased linear prediction of unknown values minimizing the error variance. Finally, In [RFA$^+$17b], the authors described a methodology to update the position p_0 on a terrain augmented map generated by a linear combination

of water parameters. These results are promising about the use of water parameters to augment information regarding location in the workspace.

This chapter presents the first steps in developing a model for predicting the spatiotemporal dynamics of interesting aquatic features. This work aims to exploit an updating schema of water parameters starting from 2D sensed parameters maps as a prior. Here, we present a data-driven approach to creating the foundation for model synthesis that considers water parameters, fluid dynamics, and the region of interest's underlying bathymetry. Following this scheme, an *augmented bathymetry map* is constructed using measured physical and biological variables, such as pH, salinity, turbidity, temperature, dissolved oxygen, and chlorophyll density that outperform traditional bathymetry strategies. Specifically, we apply the Kriging interpolation method to predict the non-sampled data points and determine the best next sample position, producing in the process the map-updating property.

The dynamical representation presented in this chapter has particular utility in planning efficient (time, energy, etc.) sampling missions that can focus on exploitation rather than exploration. This chapter's primary contribution is the development of an algorithmic tool-chain that generates the foundation for planning algorithms that promotes phenomena exploration by determining the best places to move in the navigation mission while decreasing the error variance in the prediction.

The rest of the chapter is organized as follows: In Section 4.4, the notation used though out this document is introduced, and the problem of interest is formulated, Section 4.5 develops the methods, Section 4.6 presents preliminary results. Finally, Some conclusions are drawn in Chapter 4.7.

Figure 4-1: YSI Ecomapper, the underwater vehicle used to gather physicochemical data in the environment.

4.4. Model and Problem Definition

4.4.1. Model Definition

The autonomous underwater vehicle (AUV) modeled in our chapter is the Ecomapper illustrated in Fig. **4-1**. The marine environment is modeled as a 2-D layer which is denoted as $\mathcal{W} \subset \mathbb{R}^2$. Let $\mathcal{O}$ represent the set of locations that are inaccessible for the robot (obstacles). The free space for the vehicle is represented by $E = \mathcal{W} \setminus \mathcal{O}$. The motion map M_m is modeled as a *discrete Markov chain* over a set of discrete locations in E and captures the long-term behavior of the water flow patterns. On the other hand, the augmented bathymetry map M_b is a collection of n functions $m_t^i : E \to \mathbb{R}$ for $i \in \{1, \ldots, n\}$ and time t, where each m_t^i is a spatial field representing a variable quantity that can be measured by the AUV at a specific time step. The map that the robot will use is $M = M_m \times M_b$. Let $\mathcal{M}$ be the collection of all possible such maps. The robot will receive sensing data from an *observation space* Y.

The AUV starts with an initial *information state* or *belief state* of the map $M_0 \in \mathcal{M}$ and an initial information state on his position in E denoted p_0. The initial map M_0 can be obtained from the previous knowledge of water current flows or from previously deployed missions in the case of physical and biological variables. As the vehicle moves, it needs

to update its belief state p_0 and M_0 by incorporating information from an observation $y \in Y$.

4.4.2. Problem Formulation

Our problem of interest is to update the map M_0 once a new observation y_t is obtained and to determine what is the best next position to sample.

Map Update and Next Sample Position: *Given the map M_k at the current time step k, incorporate a sensing observation y_k and determine the next position p_{k+1} to obtain an updated map estimate M_{k+1} that reduces the error variance on the map estimation.*

4.5. Method

In this section, we detail our method for solving the problem formulated in Section 4.4.

4.5.1. Algorithm Description

To make the problem tractable, we will create a parameterized version of each $m_k^i \in M_k$ in the augmented bathymetric map. The previous works showed that the more variables are taken into account, the more variation is found, and the more reliable is the localization subtask. For the SLAM problem, this is also the case.

Each m_k^i is constructed by interpolation using Kriging. This interpolation method is widely used in geological sciences to estimate distributions of properties based on spatially referenced samples. The method finds the best linear unbiased prediction of values in-between samples minimizing the error variance in the prediction. This kind of interpolation works naturally for the properties of aquatic environments. To estimate a water parameter

value at an unknown position $z(\boldsymbol{x_0}) \in m_k^i$ where $\boldsymbol{x}_0$ is a vector denoting a position in $\mathcal{W}$. $z(\boldsymbol{x_0})$ can be considered as a realization of a random function $Z(\boldsymbol{x})$ and can also be estimated using n samples at positions $\boldsymbol{x_1, x_2, ...x_n}$ such that

$$z^*(\boldsymbol{x_0}) = \sum_{i=1}^{n} \lambda_i z(\boldsymbol{x_i}) \tag{4-1}$$

is an estimation of the true unknown value of $z(\boldsymbol{x_0})$, where λ_i are the kriging coefficients. For this estimation to be unbiased, the coefficients λ_i need to compliance to:

$$\sum_{i=1}^{n} \lambda_i = 1 \tag{4-2}$$

Furthermore, since the aquatic environment is highly dynamic, the assumption on stationarity in equation 4-1 does not hold. Instead a set of L drift functions must be considered to account for the dynamics of the environment. The following formula is obtained:

$$z^*(x_0) = \sum_{i=1}^{n} \lambda_i z(\boldsymbol{x_i}) + \sum_{j=1}^{L} \mu_j f_j(\boldsymbol{x_0}) \tag{4-3}$$

where μ_l are Lagrange multipliers. This estimator is known as the Universal Kriging. The functions f_j must be established from the data. The advantage of using Kriging over other interpolation methods is on the ability to estimate also the variance in the prediction error:

$$\sigma^2 = \sum_{i=1}^{n} \lambda_i \gamma_{i0} + \sum_{j=1}^{L} \mu_j f_j(\boldsymbol{x_0}) \tag{4-4}$$

When new data is obtained and needs to be incorporated into the estimations, the naive approach is to recalculate all the coefficients and Lagrange multipliers. Fortunately, it has been shown [Eme09] that the values in λ_i are redistributed when a new data-point is added or removed from the sample set. Hence, the following relation is demonstrated:

$$\forall \alpha \in \{1, ..., n\}, \lambda_{\alpha|n}(\boldsymbol{x}) = \lambda_{\alpha|n+1}(\boldsymbol{x}) + \lambda_{n+1|n+1}(\boldsymbol{x})\lambda_{\alpha|n}(\boldsymbol{x} + \mathbf{1}) \tag{4-5}$$

where $\lambda_{\alpha|n}$ is the coefficient at position α given n sample points. Also the variance is updated by the following equation:

$$\sigma_n^2(\boldsymbol{x}) = \sigma_{n+1}^2(\boldsymbol{x}) + \lambda_{n+1|n+1}^2(\boldsymbol{x})\sigma_n^2(\boldsymbol{x_{n+1}}). \tag{4-6}$$

Where $\sigma_{n+1}^2(\boldsymbol{x})$ is the error variance when predicting $z(\boldsymbol{x})$ from the data at locations $\boldsymbol{x_1}, ..., \boldsymbol{x_n}, \boldsymbol{x_(n+1)}$. [Eme09] also shows how this relation can be used to determine, from a set of candidate sampling points, which one is better for sampling, i.e., which one reduces the variance on the error. Consequently, in this paper, we use this framework to conduct a surveillance mission, in which every following point is chosen to as reducing the variance in the error and therefore to gain more information about the phenomena surveyed. The complete scheme is summarized in algorithm 5.

Algorithm 5: SURVEILLANCE(M_0)

Input: M_0 – The set of maps at time 0 (prior)

Output: M_{k+1} – map updated.

```
k ← 0
y_k^i ← READ_DATA()
M_k ← M_0
while (true) do
    x_0 ← SLAM(M_k)
    x_candidates ← NEIGHBOURS(x_0)
    x_{n+1} ← BEST_CANDIDATE(x_candidates)
    MOVE(x_{n+1})
    M_(k + 1) ← UPDATE(M_k)
return M_{k+1}
```

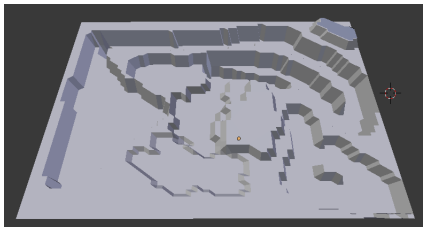


Figure 4-2: World used for simulation purpuses.

4.6. Experimental Validation

4.6.1. Data Acquisition

In this paper, we examine sensor data about water quality and bathymetry from multiple missions of a YSI EcoMapper AUV at the Lake Nighthorse, CO, USA ($37^o13'13{,}4''$ N, $107^o53'53{,}7''$ W) as illustrated in Fig. **4-3**, in April and May 2018 and July 2019. The duration of each mission was around 4 to 6 hours. The variables sensed used in this work include: Bathymetry, temperature, ph, salinity and turbidity. Fig. **4-4** shows the locations where the data was sampled measured from the upper-left corner of the inlet of lake.

Additionally, the algorithm was tested using a simulated environment. The Gazebo 9 physics engine was used along with a ROS (Robot Operating System) set of packages named Unmanned Underwater Vehicle Simulator [MSV+16] for aquatic environment simulations. The simulations are run in a 1km square region with designed bathymetry that includes different elevations.

4.6.2. Experimental Setup

To validate Algorithm 5, the data obtained should be meaningful to determine the current position. The data collected shows consistency, despite being gathered in different

Figure 4-3: Aerial image of our region of interest – the Lake Nighthorse, CO, USA. ($37^o13'13{,}4''$ N, $107^o53'53{,}7''$ W)

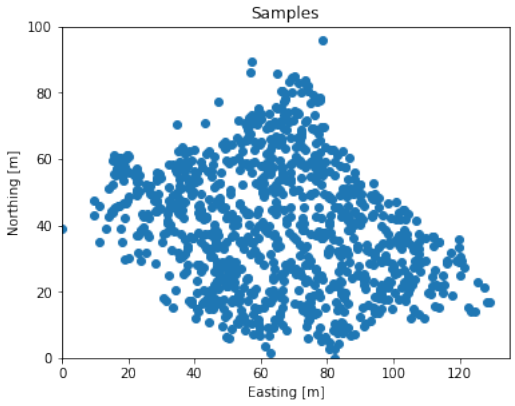


Figure 4-4: 1000 data points used to asses the region of interest. An area of $80m \times 80m$ was considered

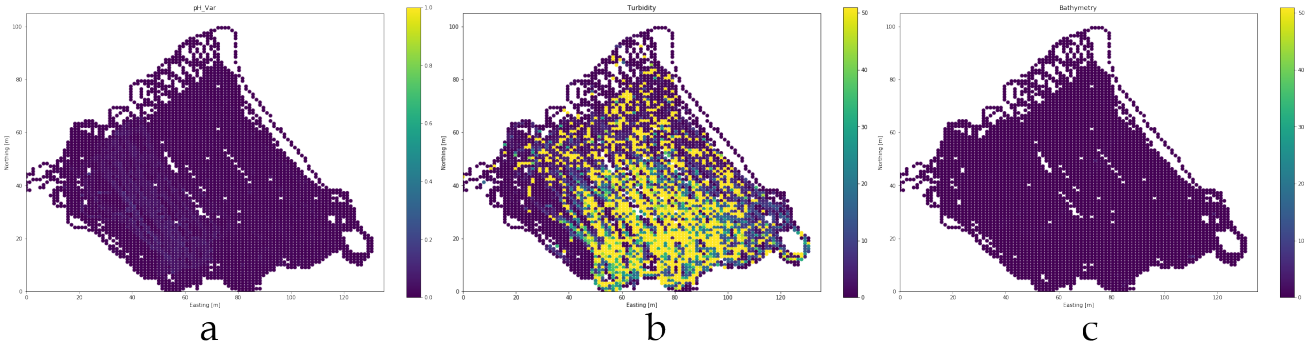


Figure 4-5: Variance in a. pH, b. Turbidity, c. Bathymetry for the data collected.

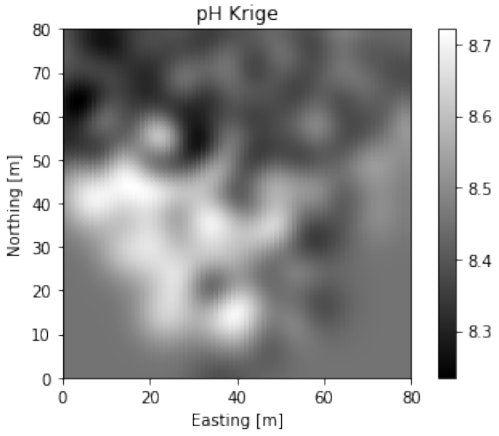


Figure 4-6: pH kriage

time stamps, even though some data was collected a year apart form the other. Figure **4-5** show the three more notable variables: pH, Turbidity and Bathymetry. The former helps as a comparison since Bathymetry is expected to suffer less changes. Figure **4-6** shows how pH exhibits the same behavior. Turbidity has some larger variations in the region where the river water flows throughout the inlet into the lake. Nevertheless, some portion of the space is still presenting less variation and this serves as additional information for localization purpuses.

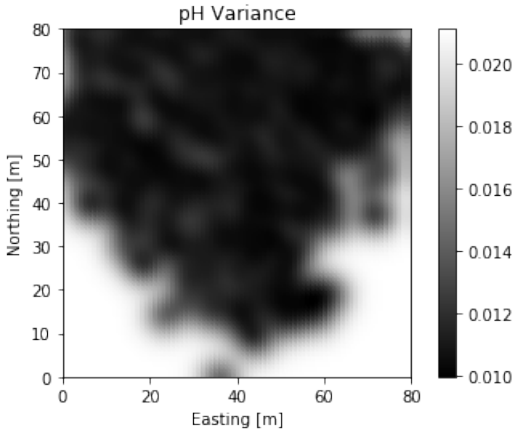


Figure 4-7: pH Kriage Error Variance

4.6.3. Experimental Result

4.7. Conclusions and Future Work

This chapter builds on previously developed representations of aquatic environments and constructed a preliminary solution to the SLAM problem in underwater settings. simple motion models are combined with physicochemical readings of the water to find initial candidate locations. The strategy presented here was tested preliminarily in a dataset acquired from deployments in the Lake Nighthorse, CO.

In the short-term, we would like to test our ideas with different datasets. We will proceed to validate our ideas in datasets in ocean deployments. We believe that due to the variability in ocean motions and science parameters, localization will be more precise once the map is built. On the other hand, richer models for constructing the maps should be explored to account for these factors.

Another avenue to be explored, based on Chapter 3, is a comparison of our ideas against established SLAM approaches [SNS11, CHL$^+$05, TBF05] and in particular, underwater

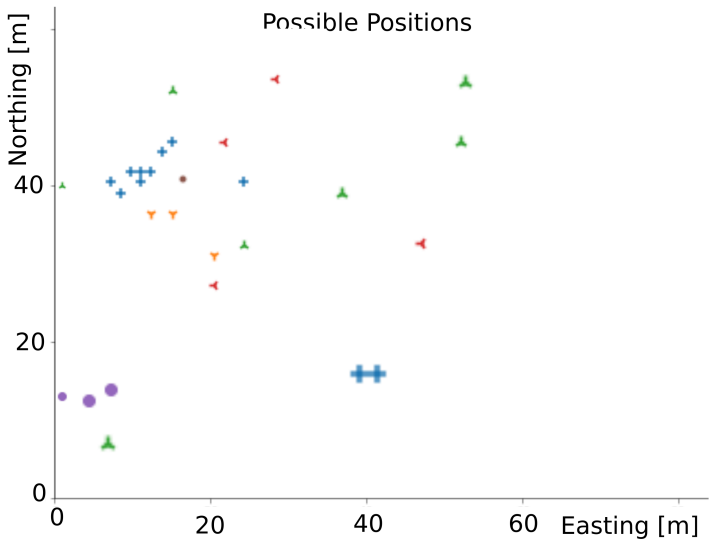


Figure 4-8: Possible positions based on the proposed SLAM

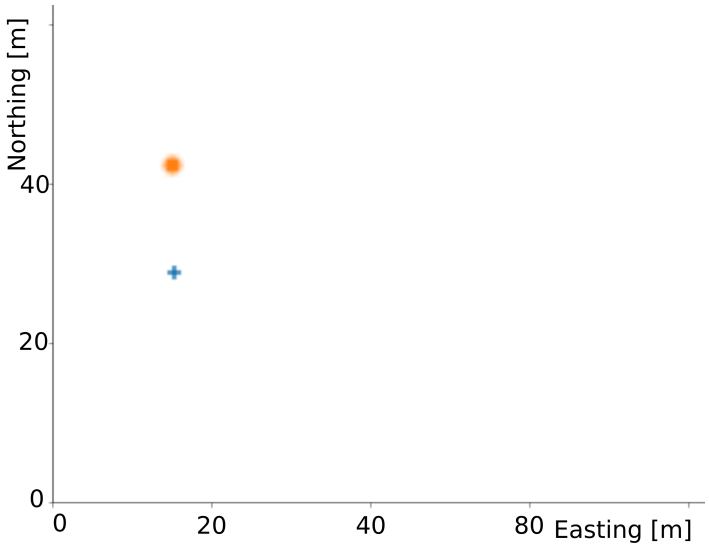


Figure 4-9: Inferred Position based on the proposed SLAM

SLAM approaches [RRTN07, RRNT06, RRTN08]. Although our approach shares the same high-level idea of updating a belief or *information state* [LaV06b, L$^+$12], we do not need a precise dynamical model of the vehicle's motion and our sensing modalities are different.

5 Coordinated multi-robot planning while preserving individual privacy

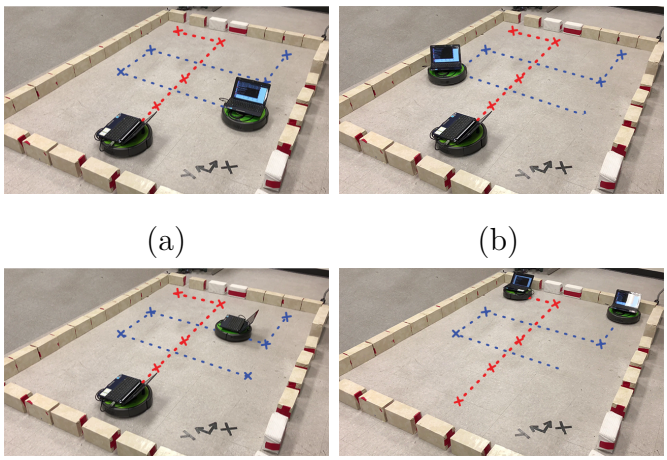
Part of this work was presented at IEEE International Conference on Robotics and Automation (ICRA) 2019[1]

5.1. Introduction

If many current predictions are to be believed, autonomous robots will be increasingly used in shared, contested, resource constrained, and adversarial scenarios. Certainly these traits underscore tasks such as automated delivery, battlefield awareness, and surveillance. In each of these cases, multiple robots operating concurrently often can achieve their ends more efficiently by cooperating to mediate their use of shared resources. But, as the information that the robots possess is sensitive or restricted, this poses the question of how to preserve individual privacy whilst coordinating. More broadly, privacy preservation is becoming a growing concern in robotics, and this paper examines several particular

[1]The author acknowledges the contribution of Li Li in the initial idea and the implementation of the circuits. Also, Tauhidul Alam, Dylan A. Shell, and Leonardo Bobadilla provided help on the development of the ideas and methodology.

scenarios where we envision it being relevant to the context of coordination among robots.



(a) (b)

(c) (d)

Figure 5-1: An experimental evaluation of the privacy-preserving monitoring task for two parties Alice and Bob using two iRobot Create 2.0 platforms: (a) At time = 0 second, Alice and Bob start executing their paths; (b) Since an intersection is found (without sharing information), Alice moves and Bob stays still; (c) At time = 20 seconds, Alice finishes her path; (d) At time = 40 seconds, both robots have reached their goals with no collisions and without either revealing the path details to the other party.

The following two examples, one within the commercial/civilian context and another with a military setting, provide motivating scenarios:

– An obstacle to the widespread commercial use of small Unmanned Aerial Vehicles (UAVs) is the potential for collisions, not only against each other but also against larger manned aircraft [faa]. As various airborne vehicles are owned and operated by different companies and stakeholders, it would be ideal if one could provide assurances of collision-free paths without mandating the disclosure of information that some

parties would be uneasy revealing.

– Consider a covert mission against some adversary where multiple robots must coordinate, for example, to rendezvous behind enemy lines. In the event of one of the robots being abducted or compromised, we would like guarantees that any information that might be extracted from the captured agent will not make the other robots vulnerable.

The above mobile robotics applications can be modeled in a Secure Multi-Party Computation (SMC) [Yao86, Yao82, GMW87] Framework. We think that SMC can model an important class of problems in mobile robotics for what we call *mutually distrusting cooperation*; in this type of problem one has multiple interacting robots that need to achieve tasks jointly within a shared environment but wish to limit the disclosure of their information. The contribution of our paper is showing the practical feasibility of Secure Multi-Party Computation in Robotics and Autonomous Systems. We hope to attract the attention of other researchers to this area. There have been few practical implementations of SMC, and to the best of our knowledge, this is one of the first implementations of SMC in Robotics. More concretely, the contributions of our paper are:

1. A secure path intersection protocol based on the polygon intersection ideas presented in [AD01] and simplified to make its implementation feasible using open source software packages.

2. A protocol that ensures that two robots will not collide while executing their task, its software implementation, and hardware proof of concept experiments.

3. A new, secure 3D path collision protocol that can be used in plans involving time or 3D workspaces.

The rest of the chapter is organized as follows. Section II describes related efforts in the literature. Section III introduces our model, the environment, and robot capabilities; and formulates the problem of interest. Section IV presents the proposed methods developed to address the problems of interest. Section V presents simulation and software development and physical deployment to show the feasibility of our approach. Finally, we show preliminary conclusions and directions for future work in Section VI.

5.2. Related Work

The motivation of our work comes from the area of Secure Multi-Party Computations [Yao86, Yao82] where a group of players needs to compute a joint function without disclosing their inputs. A well known particular case of this formulation is *Yao's Millionaire Problem* where two millionaires Alice and Bob want to know who has the most money without revealing their wealth to each other. The approach is general and can solve any function that can be encoded as a circuit. However, if the function is complicated, the use of *garbled circuits* will not be practical. This limitation has led to the development of specific protocols for SMC problems in particular domains [DA01] such as Linear Algebra, Statistics, Machine Learning, Networking, and Computational Geometry.

In particular, due to the close connection of planning algorithms to Computational Geometry, our ideas take inspiration from *Secure Computational Geometry* [FA04, AD01, LYHZ11, HAGS09, SYDP06] that proposes geometric constructions to computational problems involving intersections [AD01] (point in polygon, polygon-polygon intersection) or based on distance [FA04] (distance between parametric equations and line segments). These secure geometric constructions use the primitives such as Yao's millionaires, Garbled Circuits, 1-out-N oblivious transfer, and Homomorphic Encryption. In our work, we

build our protocols on top of some of these primitives, specifically from [AD01] and simplify them to implement using open source software and inexpensive robot platforms. We also create new secure primitives to compare paths in 3D as required by our applications. Our effort is also connected to practical system implementations of SMC [BCD+09].

Security issues in multi-robot networks have been the focus of recent research [RS17, GKM+17]. In [GKM+17], defense mechanisms for Sybil attacks have been proposed and implemented in commodity Wi-Fi radios. Their approach has also been tested in mobile robotic platforms. Privacy issues related to robots have also been investigated in several recent works [OS15, ZS18, PK16]. Among the investigated robotic privacy techniques, a differential privacy model is proposed in [PK16] for swarms of heterogeneous robots, also, combinatorial filters are designed in [OS15] satisfying privacy and utility constraints which have been explored for the privacy-preserving target tracking [ZS18].

5.3. Model and Problem Definition

5.3.1. Model Definition

The robots move in a 2-D workspace $\mathcal{W} = \mathbb{R}^2$. The free space of the environment is defined as $E = \mathcal{W} \setminus \mathcal{O}$, where $\mathcal{O} \subset \mathbb{R}^2$. We initially have two robots Alice and Bob that operate in a shared environment. Both robots have a representation of the environment E, can plan obstacle-free paths in the environment, and can communicate with each other. Let P_A be the path of Alice and P_B be the path of Bob.

5.3.2. Problem Formulation

In our first primitive, we require a privacy-preserving mechanism to allow for Alice and Bob to determine whether their paths (P_A and P_B) collide without revealing the path information to the other party. We propose to do this in a decentralized fashion and without relying on the existence of any trusted third-party. These constraints motivate our first problem of interest.

Problem 0. Privacy-Preserving Path Intersection: *Given two robots, Alice and Bob with paths P_A and P_B, inform the robots whether the paths have at least one point of intersection without sharing the path information.*

We will use the above problem as a building block for solving more pragmatic tasks in a privacy-preserving fashion. We are interested in a continuous monitoring task where both robots are executing the task but need to guarantee collision avoidance. This motivates the following problem of interest. **Problem 1. Privacy-Preserving Persistent Monitoring:** *Given two robots Alice and Bob that are executing a mission in E, ensure, without sharing any information about their paths or position, that they will not collide.*

We are also interested in problems involving time-parametrized trajectories. For this purpose, we need to construct secure primitives to calculate if two 3D line segments intersect. This primitive will allow us to solve the following problem.

Problem 2. Ascertaining Rendezvous Securely: *There are two robots, Alice and Bob, and each of them has a time-parametrized trajectory. They want to know if their paths intersect without sharing either respective paths or the intersection point and time.*

5.4. Methods

In this section, we detail our method for solving the problems formulated in Section 5.3.2.

Initially, we need to implement a primitive that can test if two paths intersect. To create a practical implementation, we build our protocol based on the polygon intersection protocol presented in [AD01]. This protocol uses a secure dot product implementation along with Yao's Millionaires, Garbled circuits, 1-out-of-N oblivious transfer, and homomorphic encryption. We simplify this protocol, adapt it for the 2-D path intersection, and make it suitable for implementation on mobile robots.

5.4.1. Privacy-Preserving Path Intersection

A path P of a robot is represented by a sequence of contiguous segments $P = (S_1, S_2, \ldots, S_n)$. Each segment S_i is composed by its two points $\mathbf{u_i} = (x_i, y_i)$ and $\mathbf{v_i} = (x'_i, y'_i)$. Meanwhile, we use a line equation $f(x, y) = 0$ to represent the line that contains the segment, where $f(\mathbf{u}) = f(x, y) = ax + by + c$. We can easily calculate a, b, and c by providing $\mathbf{u_i}$ and $\mathbf{v_i}$.

Suppose Alice has a segment $S_A = (\mathbf{u_A}, \mathbf{v_A})$ and the line equation $f_A(x, y) = 0$ that contains the vertices $\mathbf{u_A}, \mathbf{v_A}$, and Bob has a segment $S_B = (\mathbf{u_B}, \mathbf{v_B})$ and the line equation $f_B(x, y) = 0$ that contains the vertices $\mathbf{u_B}, \mathbf{v_B}$. Then, S_A intersects with S_B if and only if one of the following expression is true:

$f_A(\mathbf{u_B}) \leq 0 \wedge f_A(\mathbf{v_B}) \geq 0 \wedge f_B(\mathbf{u_A}) \leq 0 \wedge f_B(\mathbf{v_A}) \geq 0,$

$f_A(\mathbf{u_B}) \leq 0 \wedge f_A(\mathbf{v_B}) \geq 0 \wedge f_B(\mathbf{u_A}) \geq 0 \wedge f_B(\mathbf{v_A}) \leq 0,$

$f_A(\mathbf{u_B}) \geq 0 \wedge f_A(\mathbf{v_B}) \leq 0 \wedge f_B(\mathbf{u_A}) \leq 0 \wedge f_B(\mathbf{v_A}) \geq 0,$

$f_A(\mathbf{u_B}) \geq 0 \wedge f_A(\mathbf{v_B}) \leq 0 \wedge f_B(\mathbf{u_A}) \geq 0 \wedge f_B(\mathbf{v_A}) \leq 0.$

To make the computation of the substitution easier, we define an operator $\sqcup$ such that

$\mathbf{u} \sqcup 1 = (x, y, 1)$, where $\mathbf{u} = (x, y)$ is a vector.

Protocol 1 Secure Path Intersection Protocol

Input: Given Alice's path $P_A = (S_{A_1}, S_{A_2}, \ldots, S_{A_n})$ and Bob's path $P_B = (S_{B_1}, S_{B_2}, \ldots, S_{B_n})$.
Output: Whether there is a collision between P_A and P_B.

1. Alice generates a public/private key pair (k_A^{pub}, k_A^{pri}) using the Paillier homomorphic encryption system.

2. For each pair of segments (S_A, S_B) where $S_A = (\mathbf{u_A}, \mathbf{v_A})$ and $S_B = (\mathbf{u_B}, \mathbf{v_B})$:

   a) Alice calculates (a_1, b_1, c_1). Bob calculates (a_2, b_2, c_2).

   b) Alice generates a vector $\mathbf{m} = (E_{k_A^{pub}}(a_1), E_{k_A^{pub}}(b_1), E_{k_A^{pub}}(c_1))$.

   Bob assigns a vector $\mathbf{n} = (a_2, b_2, c_2)$.

   c) Alice generates two vectors $\mathbf{p_A} = (E_{k_A^{pub}}(\mathbf{u_A}.x), E_{k_A^{pub}}(\mathbf{u_A}.y), E_{k_A^{pub}}(1))$ and $\mathbf{q_A} = (E_{k_A^{pub}}(\mathbf{v_A}.x), E_{k_A^{pub}}(\mathbf{v_A}.y), E_{k_A^{pub}}(1))$. Bob generates two vectors $\mathbf{p_B} = \mathbf{u_B} \sqcup 1$ and $\mathbf{q_B} = \mathbf{v_B} \sqcup 1$.

   d) Alice sends $\mathbf{m}, \mathbf{p_A}$ and $\mathbf{q_A}$ to Bob.

   e) Bob calculates w_1, w_2, w_3, w_4, where $w_1 = \mathbf{m} \cdot \mathbf{p_B}, w_2 = \mathbf{m} \cdot \mathbf{q_B}, w_3 = \mathbf{n} \cdot \mathbf{p_A}, w_4 = \mathbf{n} \cdot \mathbf{q_A}$.

   f) Bob generates 4 random numbers r_1, r_2, r_3, r_4 and calculates h_1, h_2, h_3, h_4, where $h_i = w_i + r_i, i = 1, 2, 3, 4$.

   g) Bob sends h_1, h_2, h_3, h_4 back to Alice.

   h) Alice computes $t_i = D_{k_A^{pri}}(h_i)$ where $i = 1, 2, 3, 4$.

   i) Alice and Bob use a garbled circuit to check if the following expression is true: $(t_1 \leq r_1 \wedge t_2 \geq r_2 \wedge t_3 \leq r_3 \wedge t_4 \geq r_4) \vee (t_1 \leq r_1 \wedge t_2 \geq r_2 \wedge t_3 \geq r_3 \wedge t_4 \leq r_4) \vee (t_1 \geq r_1 \wedge t_2 \leq r_2 \wedge t_3 \leq r_3 \wedge t_4 \geq r_4) \vee (t_1 \geq r_1 \wedge t_2 \leq r_2 \wedge t_3 \geq r_3 \wedge t_4 \leq r_4)$.

   j) Return **True** if the garbled circuit returns *true*.

3. Return **False**.

Let Alice compose three vectors $\mathbf{m} = (a_1, b_1, c_1), \mathbf{p_A} = \mathbf{u_A} \sqcup 1, \mathbf{q_A} = \mathbf{v_A} \sqcup 1$, and let Bob compose three vectors $\mathbf{n} = (a_2, b_2, c_2), \mathbf{p_B} = \mathbf{u_B} \sqcup 1, \mathbf{q_B} = \mathbf{v_B} \sqcup 1$, then

$i = \mathbf{m} \cdot \mathbf{p_B}, i' = \mathbf{m} \cdot \mathbf{q_B}, j = \mathbf{n} \cdot \mathbf{p_A}, j' = \mathbf{n} \cdot \mathbf{q_A}$.

So far, if Alice sends $\mathbf{m}, \mathbf{p_A}, \mathbf{q_A}$ to Bob, Bob can easily compute i, i', j, j' and do the

intersection determination logic and finally send the result back to Alice, then each of them would know if there is a collision, but this reveals detailed information of Alice's path, and Bob may send a spurious result to Alice. However, this problem can be addressed by taking advantage of the Paillier homomorphic encryption system (PES). PES is an additive homomorphic encryption system [Pai99], meaning that the sum of two encrypted numbers is the encrypted sum of the plain numbers. Paillier also supports scalar multiplication, which means that a scalar multiplied by an encrypted number yields the encryption of the scalar multiplied by the plain number. We emphasize that, though not a fully homomorphic encryption system, it suffices for the protocol we outline. To prevent the path information from being revealed, Alice can send encrypted information using PES, and Bob may perform the computation using these encrypted numbers only. However, in doing so, Bob might expose his information as stated in [AD01]. To prevent this, Bob adds some random numbers to the intermediate results (h_1 to h_4), then sends the outcomes to Alice. Then, Alice decrypts them and uses them as the inputs to the garbled circuit. Thus, Protocol 1 securely decides if two paths collide, avoiding leaking any path information.

5.4.2. Privacy-Preserving Persistent Monitoring

Algorithms 6 and 7 show the implementation of the privacy-preserving persistent monitoring task for two parties Alice and Bob. A circuit that permits comparison of k segments at the time is used to compute whether a collision among any combination of the $k \times k$ segments collides or not; for larger paths, k-length subsets are compared at a time, each referred to as a round. In this scheme, one robot moves k segments at each round. Both algorithms receive as input the list of segments, the number of segments per round, and the IP address to communicate with each other via network sockets. The algorithms can

Algorithm 6: AliceTrajectory(P, k, IP_addr_B)

Input: $P = (S_1, S_2, \ldots, S_n)$; $k < n$ the number of segments per round; IP_addr_B Bob's IP address

Output: $C = (c_1, c_2, \ldots, c_m)$, $m = \lceil n/k \rceil$, $c_j \in \{False, True\}$ for $j = 1, 2, \ldots, m$, and $c_0 = True$ if both move for the first round.

```
1  CONNECT(IP_addr_B)
2  round ← 0      low ← 0      high ← low + k
3  k_A^pub, k_A^priv ← paillier()
4  while high ≤ n do
5      R ← P[low : high]
6      for i ← low to high − 1 do
7          m ← E(k_A^pub, Equation(R_u, R_v))
8          SEND(m)                              // encoded parameters
9          RECEIVE(ACK)                               // ACK awaiting
10         p_A ← E(k_A^pub, R_{i,u} ⊔ 1)
11         q_A ← E(k_A^pub, R_{i,v} ⊔ 1)
12         SEND(p_A; q_A)
13         RECEIVE(ACK)
14         d_r ← RECEIVE()
15         SEND(ACK)
16         u ← D(k_A^pri, d_r)
17         result_round ← AliceCircuit(u)
18         round ← round + 1
19         low, high ← high, high + k
20         if low <n and high >n then
21             high, low ← n, n − k
22 return result
```

be run to plot the resulting behavior or to send commands to the robot platform.

Algorithm 7: BOBTRAJECTORY(P, k, IP_addr_A)

Input: $P = (S_1, S_2, ..., S_n)$; $k < n$ the number of segments per round; IP_addr_A Alice's IP address

Output: $C = (c_1, c_2, ..., c_m)$, $m = \lceil n/k \rceil$, $c_j \in \{False, True\}$ for $j = 1, 2, \ldots, m$, and $c_0 = True$ if both move for the first round

```
1  CONNECT(IP_addr_A)
2  round ← 0      low ← 0      high ← low + k
3  while high ≤ n do
4      R ← P[low : high]
5      for i ← low to high − 1 do
6          m ← RECEIVE()                                  // encoded
7          SEND(ACK)                                      // ACK receipt
8          p_A; q_A ← receive()
9          SEND(ACK)
10         r = (rand(),rand(),rand(),rand())*(k*k)
11         n ← Equation(R_u, R_v)
12         p_B ← R_u ⊔ 1      q_B ← R_j ⊔ 1
           /* d_r = h, h_i = w_i + r_i, w_1 = (m · p_B)                                  */
13         d_r ← [(m · p_B), (m · q_B), (n · p_A), (n · q_A)]
14         d_r ← d_r + r
15         SEND(d_r)
16         RECEIVE(ACK)
17         result_round ← BOBCIRCUIT(r)
18         round ← round + 1
19         low, high ← high, high + k
20         if low <n and high >n then
21             high, low ← n, n − k
22 return result
```

Algorithms 6 and 7 detail how the agents share the data they need. Algorithm 6 sends a variable to Bob (line 8). Bob receives this variable in Algorithm 7 (line 6). An " ACK" command is used to coordinate this data exchange since the function RECEIVE(ACK) will block the execution until " ACK" arrives. Once both algorithms have calculated and

received their data, an instance of the circuit is launched. This circuit instance receives both Alice and Bob's input and sends the response to both of them. The circuit is launched via a system call, and the communication is achieved via sockets. If no collision is detected, Alice and Bob may safely move simultaneously. Otherwise, they will have to take turns.

5.4.3. Rendezvous Using Secure 3D Intersection

For the parties to find each other in the rendezvous problem, it is not enough to detect the intersection of the paths, but the meeting must occur at the same time. This is why the problem of secure rendezvous reduces to the calculation of the intersection of two time-parametrized paths with coordinates (x, y, t). In the 3D case, we need to use a fully homomorphic encryption system [GB09] since the homomorphic multiplication property is required between two encrypted numbers. The intersection of segments in the 3D case can be resolved using the following steps:

1. Determine whether two segments are coplanar.

2. If they are coplanar, solve the problem in a subspace.

Suppose we have four points P_1, P_2, P_3, P_4 in a 3D Cartesian space, then segment $\overline{P_1P_2}$ and $\overline{P_3P_4}$ are coplanar if and only if $\overrightarrow{P_1P_2}$ is perpendicular to $\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_4}$, that is, $\overrightarrow{P_1P_2} \cdot (\overrightarrow{P_1P_3} \times \overrightarrow{P_1P_4}) = 0$. Accordingly, we introduce a secure coplanar protocol in Protocol 2. A robust algorithm to determine whether two 2D segments intersect or not was given by [CLRS01]. The procedure works with 2D vectors. Since the plane (or a line if two segments are co-linear) spanned by two co-planar line segments is a subspace of the 3D Cartesian space, the algorithm also works in this subspace. Along these lines, we extended the algorithm (in Protocol 3) to address secure computation in 3D. This protocol can be used

to determine whether there is a collision between two 3D paths. Concretely, we have used it to detect rendezvous securely in two time-parameterized 2D trajectories. The function $DIR(\mathbf{u_1}, \mathbf{u_2}, \mathbf{u_3},)$ used in Protocol 3 computes the normal vector to the plane formed by $\overrightarrow{u_3u_1}$ and $\overrightarrow{u_2u_1}$. It also must be pointed out that Protocol 3 does not consider the case when one vertex from a segment lies directly on the other segment. Additional comparisons are involved in order to handle this particular case.

Protocol 2 Secure Coplanar Protocol

Input: Given Alice's segment S_A and Bob's segment S_B.
Output: Whether S_A and S_B are coplanar.

1. Alice generates key pair (k_A^{pub}, k_A^{pri}) using fully homomorphic encryption system.
2. Alice computes $C_{\mathbf{u_A}}$ and $C_{\mathbf{v_A}}$, i.e., S_A encrypted components using k_A^{pub}.
3. Alice sends $C_{\mathbf{u_A}}$ and $C_{\mathbf{v_A}}$ to Bob.
4. Bob $\mathbf{p} = C_{\mathbf{v_A}} - C_{\mathbf{u_A}}, \mathbf{m} = \mathbf{u_B} - C_{\mathbf{u_A}}, \mathbf{n} = \mathbf{v_B} - C_{\mathbf{u_A}}$.
5. Bob computes $w = \mathbf{p} \cdot (\mathbf{m} \times \mathbf{n})$.
6. Bob computes $h = w + r_B$, r_B is a random number.
7. Bob sends h to Alice.
8. Alice computes t, decryption of h using k_A^{pri}.
9. Alice and Bob use a garbled circuit to check whether t is equal to r_B.
10. Return **True** if the circuit returns *true*, otherwise, return **False**.

Protocol 3 Secure 3D-Intersection Protocol

Input: Given Alice's path $P_A = (S_{A_1}, S_{A_2}, \ldots, S_{A_n})$, and Bob's path $P_B = (S_{B_1}, S_{B_2}, \ldots, S_{B_n})$.
Output: Whether P_A and P_B intersects.

1. Alice and Bob generates public / private key pairs (k_A^{pub}, k_A^{pri}) and (k_B^{pub}, k_B^{pri}) respectively using fully homomorphic encryption system.

2. For each pair of segments (S_A, S_B) where $S_A = (\mathbf{u_A}, \mathbf{v_A})$ from Alice and $S_B = (\mathbf{u_B}, \mathbf{v_B})$ from Bob.

   a) Both Alice and Bob execute Protocol 2 steps 2 to 7, Bob gets r_B and h.

   b) Bob computes $C_{\mathbf{u_B}} = (E(k_B^{pub}, \mathbf{u_B}.x), E(k_B^{pub}, \mathbf{u_B}.y), E(k_B^{pub}, \mathbf{u_B}.z))$, $C_{\mathbf{v_B}}$ $= (E(k_B^{pub}, \mathbf{v_B}.x), E(k_B^{pub}, \mathbf{v_B}.y), E(k_B^{pub}, \mathbf{v_B}.z))$.

   c) Bob sends $C_{\mathbf{u_B}}$, $C_{\mathbf{v_B}}$ to Alice.

   d) Alice computes: $\mathbf{d_1} = DIR(C_{\mathbf{u_B}}, C_{\mathbf{v_B}}, \mathbf{u_A})$,

   $$\mathbf{d_2} = DIR(C_{\mathbf{u_B}}, C_{\mathbf{v_B}}, \mathbf{v_A}),$$

   $$\mathbf{d_3} = DIR(\mathbf{u_A}, \mathbf{v_A}, C_{\mathbf{u_B}}),$$

   $$\mathbf{d_4} = DIR(\mathbf{u_A}, \mathbf{v_A}, C_{\mathbf{v_B}}).$$

   e) Alice generates 2 random numbers r_{A_1}, r_{A_2} and computes $l_1 = (\mathbf{d_1} \cdot \mathbf{d_2}) + r_{A_1}, l_2 = (\mathbf{d_3} \cdot \mathbf{d_4}) + r_{A_2}$.

   f) Alice sends l_1, l_2 to Bob; Bob sends the output h from Protocol 2 to Alice.

   g) Alice computes $t = D(k_A^{pri}, h)$ and Bob computes $t_1 = D(k_B^{pri}, l_1), t_2 = D(k_B^{pri}, l_2)$

   h) Alice and Bob use a garbled circuit to check if the following expression is true: $((t == r_B) \wedge (r_{A_1} > l_1) \wedge (r_{A_2} > l_2))$ where t, r_{A_1}, r_{A_2} are inputs fed by Alice and r_B, l_1, l_2 are inputs fed by Bob.

   i) Return **True** if the garbled circuit returns *true*.

3. Return **False**.

5.5. Experimental Results

5.5.1. Software Implementation of Persistent Monitoring

The online version described in the problem formulation section was implemented in Python 3. The Fairplay software [BDNP08, MNP+04] for Secure Multi-party Computation using garbled circuits was integrated into our implementation. Communication between the parties was achieved via sockets. We also used the Python library `python-paillier` [pyt] as it implements the scalar multiplicative and additive homomorphic cryptosystem Paillier [Pai99]. Protocol 1 was implemented in a multi-round fashion, each round consisting of two k-length paths P_A and P_B. After each round, the robots decide whether to move simultaneously, if no collision is detected, or to determine who moves first otherwise. In the latter scenario, a collision was detected and the robots must move sequentially. A prior random agreement is used to settle which party has right of way. Fig. **5-2** shows two desired paths, the protocol rounds are highlighted accordingly for each party. It shows when a collision would only occur (though only if the paths collide in the same round). The green circles representing potential collisions never happen owing to the use of our strategy. The results are plotted in a simulation using Python.

Fig. **5-3** shows the execution of the paths shown in Fig. **5-2**; the paths consist of 12 segments each. The implementation uses a subpath of length 4, i.e., $k = 4$, implying that 3 rounds must occur. In the first round, see Fig. **5-3**(a), the paths in the round are collision-free and both robots can safely move simultaneously. Fig. **5-3**(b) and Fig. **5-3**(c) show round 2, wherein a collision is detected, and the robots decide to move one after the other. (In this simulation, Alice was randomly selected to go first.) Fig. **5-3**(d) shows the last round. No collision is detected within this round so both Alice and Bob head to their

destinations simultaneously.

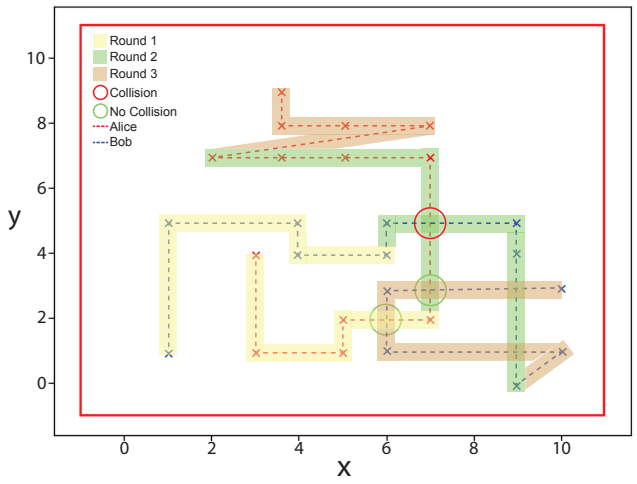


Figure 5-2: Two desired paths of Alice and Bob in a multi-round simulation run. Three collisions (red and green circles) are found if two paths are compared as a whole. Only one collision (red circle) is found in our strategy as we divide the paths into different segments and compare these segments in several rounds.

5.5.2. Implementation of 3D Intersection

Since the secure intersection decision in 3D environments involves multiplication between encrypted numbers, a partial homomorphic cryptosystem like Paillier cannot implement Protocol 3. Hence, a fully homomorphic cryptosystem is needed instead. The Simple Encrypted Arithmetic Library (SEAL) [CLP17, SEA] meets the necessary requirements. This library was developed by the Cryptography Research Group at Microsoft Research and has a Python wrapper PySEAL [TKS+18, PYS] making it possible to use within Python.

Unlike the 2D case, where only Alice generates a public/private key pair, in the 3D

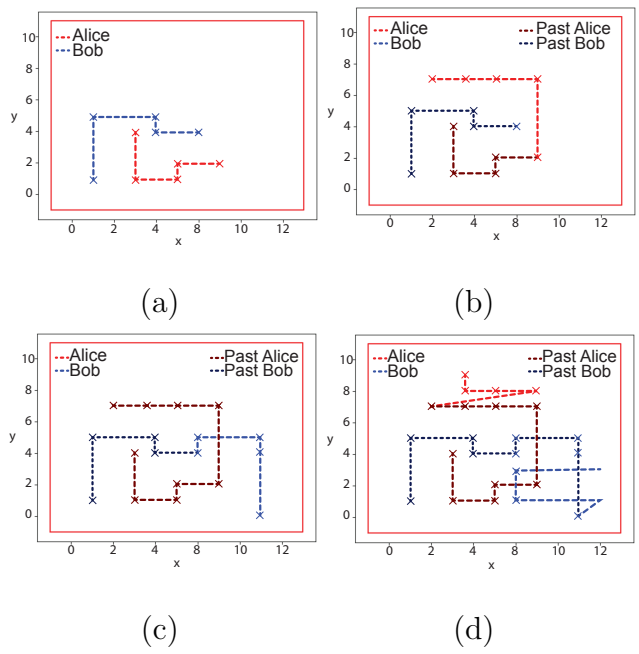


Figure 5-3: Snapshots of the execution of paths shown in Fig. **5-2**: (a) Alice and Bob move together since no collision exists in the first four segments; (b) A collision is detected in the next four segments, thus, Alice moves first; (c) Bob moves next; (d) Both Alice and Bob move simultaneously as no collision exists in the final four segments.

case, both Alice and Bob generate key pairs. Thereafter, they exchange their encrypted points. We let Bob handle the vector computations related to the co-planar determination and have Alice handle the vector computations related to intersection calculation. This split is not essential for solving the problem: either Alice or Bob could perform all the calculations but doing so allows us to distribute the computational load.

Fig. **5-4** presents the simulation results for the algorithm that computes time-parameterized path collisions for objects moving in 3D space. This exemplifies our motivating example

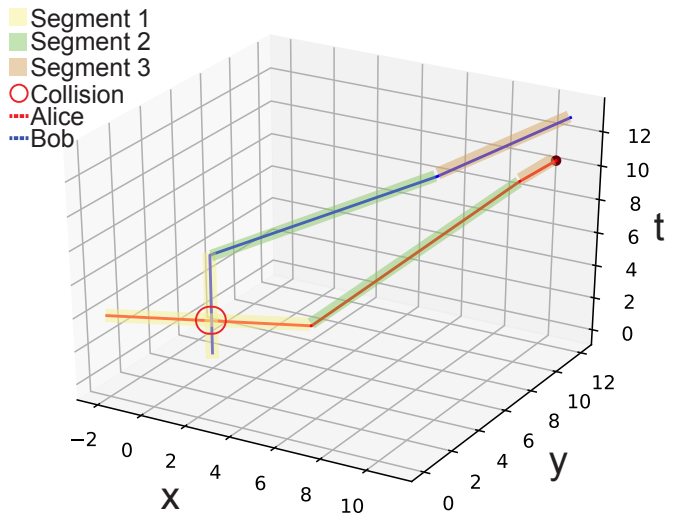


Figure 5-4: Collision detection (red circle) in three segments of Alice's time-parameterized path (red) and Bob's time-parameterized path (blue).

of shared areas where manned aircraft and small UAVs need to navigate without revealing their path information.

Fig. **5-5** shows a secure rendezvous experiment for Alice and Bob using 3D intersection. If only two dimensions (Fig. **5-5** right) are considered, then two rendezvous points might be found. However, when the time is considered, by the moment Bob arrives at the meeting point, Alice may have been there in the past, or she may only arrive in the future and so no real rendezvous will occur. A rendezvous (Fig. **5-5** left) only occurs when there is an intersection in $\mathbb{R}^2 \times \mathbb{T}$.

5.5.3. Hardware Experiment

A physical implementation was also conducted to test the persistent monitoring algorithms with two iRobot Create 2 platforms (see Fig. **5-1**). The mobile robot platforms are

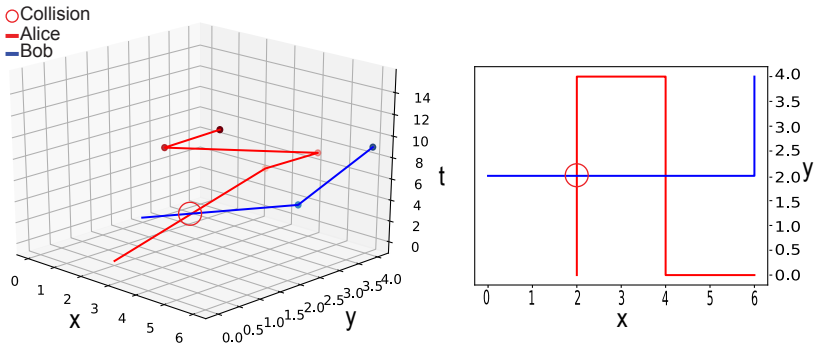


Figure 5-5: Rendezvous determination using secure 3D intersection: (left) Only one point of intersection is found when the time is included as an extra dimension; (right) An additional false point of intersection is found in 2D that is resolved in time.

connected to laptops running Ubuntu 12.0 SO, Intel Atom at 2.0 GHz, and 2 GB RAM. Two robots communicate with each other using the `pycreate2` Python library [Pyc]. Fig. **5-1**(a) shows the initial configurations and the intended path. In this experiment, all the segments are compared to each other. Since one collision is detected, Alice (blue line) moves first (Fig. **5-1**(b) and **5-1**(c)). Fig. **5-1**(d) shows their final position. The interface to send commands to the robots takes care of the orientation of the robot, and the distance traveled at each segment. Finally, the robots face "EAST". More experiments and simulation videos can be found at: `http://users.cis.fiu.edu/\~jabobadi/securemp/`.

5.6. Conclusion and Future Work

This chapter demonstrated the feasibility of privacy-preserving multi-robot coordination. We believe that we have just scratched the surface and there are several practi-

cal avenues for future research. In this work, privacy-preserving computational geometry primitives are used which *verify* properties such as the distance between points, line intersections [FA04], and point in polygons [AD01]. There also exists prior work in privacy computational geometry that *constructs* geometric objects. One such example is the privacy-preserving calculation of convex hulls [HAGS09, EGT10] which has an initial phase based on a data oblivious transfer algorithm that is then followed by secure protocols. We will explore this route in the future to extend the range of privacy-preserving robotic tasks that we can solve.

Hereby, a model where both agents need to cooperate in a shared environment but need to limit the disclosure of information in their coordination is considered. Fully adversarial motion planning where both Bob and Alice actively try to learn each other's information would be a further improvement of this work. It would also be interesting to explore semi-honest models [BS05] where robots will follow the protocol but one robot is curious to learn the other robot's information.

A scheme to allow more than two parties is also a worthy aim, enabling multiple robots to decide how to move to prevent any collision. It presents several system challenges as to how many active connections they would manage and how to dynamically handle robots entering and leaving groups of interacting robots.

Finally, other mobile platforms, such as micro aerial vehicles, could be used for validation in future implementations. From a motion planning perspective, these ideas can be used as a final process in motion planning pipelines of autonomous vehicles [LaV11a, LaV11b]. Although we did not consider kinematic constraints for this version, it would be interesting to consider them in future works to allow for more complex robots.

6 Conclusions and Perspectivess

Conclusions and perspectives were mentioned in the final section of each of the previous chapters; in this chapter, we present a brief summary of the dissertation, some broader conclusions, and outline some potential directions for future research.

6.1. Summary

In this research, motion planning for autonomous vehicles is studied, based on the literature three scenarios that present open challenges are considered: Unknown or hard to calculate vehicle's model, unknown vehicle configuration, unknown map and unknown hostility. For each one of these scenarios, a concrete setting is defined and a strategy is proposed, tested and evaluated. Most of the effort is devoted to underwater vehicles since they are presented as a generalization of others types of vehicles, such as aerial and terrestrial; always with the expectation of being able to accommodate the strategies to other types. Also, it is considered throughout this work, that simplifying sensors, robots or strategies is a key element to build up for more complex task and settings. That is why communications among multiple simple robots is inevitable, and also privacy preserving must be accounted. Consistently, the second scenario is considered for a simple bouncing robot, but can perfectly be extended to any kind of vehicle using the strategy proposed

in Chapter 2. In the same line of thought, all the contribution of this work can be put together to solve larger scale scenarios such that, the model is hard to calculate, the position is hard to precise, the map is partially unknown and the required task must be performed avoiding collision with other entities from which the plan is meant to be kept private.

6.2. Conclusions

The contributions in this research open multiple avenues, unlike the *Theory of Computation* where a set of equivalence classes are established along with the relationship between each other, also whether the problems from each class can be solved by a particular computer and the complexity for the solution, Robotic Systems do not have such a perk. Probably because robotic systems has being though as practical implementation from the beginning, or perhaps because of the complexity of being deployed into the real world instead of just being abstracted. Simple robots, such as the bouncing robot presented in Chapter 3 are means to define such a hierarchy and relationship between different robotics systems. Some work in this direction have been proposed in [OL08], where a dominance order is established based on the ability of robots to complete tasks. This equivalence classes would also be interesting for the environment in which the robot is deployed, it would be interesting to define the equivalence of robotic system based on the environment classification. Some clues can be found in the way the properties of the aquatic environment are used to localize, since similar properties can also be exploited in other environment either natural or artificial, such as luminosity, WIFI signals, Radio Signals, intensity of visual guides, etc. In that sense, the strategy presented in this research can be extended to others environments and furthermore, equivalence classes can be defined.

6.3. Perspectives

As stated before, a myriad of avenues is unleashed from every contribution. In each chapter, some perspectives have already been pointed out; in this section, some general themes will be touched on regarding how to follow the development of this research. The aforementioned integration of every situation would be the first avenue worth to mention, for its interesting implications. Extending the planning algorithm for multiple vehicles at the same time would be a significant contribution to the idea of dividing a task into several simple robots, not just for achieving coverage but also to another type of task in which non-centralized communication is the premise for collaboration. Finally, we envisioned great potential in the further development of secure communication since it is crucial for the reliability of multiple simple robots developing a particular task.

Bibliografía

[ABS17a] Tauhidul Alam, Leonardo Bobadilla, and Dylan A. Shell. Minimalist Robot Navigation and Coverage Using a Dynamical System Approach. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 249–256. IEEE, 4 2017.

[ABS17b] Tauhidul Alam, Leonardo Bobadilla, and Dylan A Shell. Minimalist robot navigation and coverage using a dynamical system approach. In *Proceedings of IEEE International Conference on Robotic Computing*, pages 249–256, 2017.

[ABS18] Tauhidul Alam, Leonardo Bobadilla, and Dylan A. Shell. Space-Efficient Filters for Mobile Robot Localization from Discrete Limit Cycles. *IEEE Robotics and Automation Letters*, 3(1):257–264, 1 2018.

[AD01] Mikhail J Atallah and Wenliang Du. Secure multi-party computational geometry. In *Proceedings of the Workshop on Algorithms and Data Structures*, pages 165–179. Springer, 2001.

[AmAK+18] Ali-akbar Agha-mohammadi, Saurav Agarwal, Sung-Kyun Kim, Suman Chakravorty, and Nancy M. Amato. SLAP: Simultaneous Localization and

Planning Under Uncertainty via Dynamic Replanning in Belief Space. *IEEE Transactions on Robotics*, 34(5):1195–1214, 10 2018.

[AmCA14a] Ali-akbar Agha-mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2 2014.

[AMCA14b] Ali-Akbar Agha-Mohammadi, Suman Chakravorty, and Nancy M Amato. FIRM: Sampling-based feedback motion-planning under motion uncertainty and imperfect measurements. *The International Journal of Robotics Research*, 33(2):268–304, 2014.

[ARBS18a] Tauhidul Alam, Gregory Murad Reis, Leonardo Bobadilla, and Ryan N. Smith. A Data-Driven Deployment Approach for Persistent Monitoring in Aquatic Environments. In *2018 Second IEEE International Conference on Robotic Computing (IRC)*, pages 147–154. IEEE, 1 2018.

[ARBS18b] Tauhidul Alam, Gregory Murad Reis, Leonardo Bobadilla, and Ryan N Smith. A data-driven deployment approach for persistent monitoring in aquatic environments. In *Proceedings of IEEE International Conference on Robotic Computing*, pages 147–154, 2018.

[ARBS18c] Tauhidul Alam, Gregory Murad Reis, Leonardo Bobadilla, and Ryan N Smith. An underactuated vehicle localization method in marine environments. Technical report, 2018.

[ARBS18d] Tauhidul Alam, Gregory Murad Reis, Leonardo Bobadilla, and Ryan N

Smith. An underactuated vehicle localization method in marine environments. In *Proceedings of MTS/IEEE OCEANS Charleston*, pages 1–8, 2018.

[AS18] Jacob Anderson and Ryan N. Smith. Predicting Water Properties with Markov Random Fields for Augmented Terrain-Based Navigation in Autonomous Underwater Vehicles. In *2018 OCEANS - MTS/IEEE Kobe Techno-Oceans (OTO)*, pages 1–5. IEEE, 5 2018.

[Bai02] Tim Bailey. Mobile robot localisation and mapping in extensive outdoor environments. pages 121–125. Ph.D. dissertation, 2002.

[BBBB05] Dimitri P Bertsekas, Dimitri P Bertsekas, Dimitri P Bertsekas, and Dimitri P Bertsekas. *Dynamic programming and optimal control*, volume 1. Athena scientific Belmont, MA, 2005.

[BCD+09] Peter Bogetoft, Dan Lund Christensen, Ivan Damgård, Martin Geisler, Thomas Jakobsen, Mikkel Krøigaard, Janus Dam Nielsen, Jesper Buus Nielsen, Kurt Nielsen, Jakob Pagter, et al. Secure multiparty computation goes live. In *Proceedings of the International Conference on Financial Cryptography and Data Security*, pages 325–343, 2009.

[BDNP08] Assaf Ben-David, Noam Nisan, and Benny Pinkas. Fairplaymp: a system for secure multi-party computation. In *Proceedings of the 15th ACM Conference on Computer and Communications Security*, pages 257–266. ACM, 2008.

[BDW06] Tim Bailey and Hugh Durrant-Whyte. Simultaneous localization and mapping (slam): Part ii. *IEEE Robotics & Automation Magazine*, 13(3):108–117, 2006.

[BR11] Adam Bry and Nicholas Roy. Rapidly-exploring Random Belief Trees for motion planning under uncertainty. In *2011 IEEE International Conference on Robotics and Automation*, pages 723–730. IEEE, 5 2011.

[BRK99] Robert R Burridge, Alfred A Rizzi, and Daniel E Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The International Journal of Robotics Research*, 18(6):534–555, 1999.

[BS05] Justin Brickell and Vitaly Shmatikov. Privacy-preserving graph algorithms in the semi-honest model. In *Proceedings of the International Conference on the Theory and Application of Cryptology and Information Security*, pages 236–252. Springer, 2005.

[CBKS08] V. Chen, M. Batalin, W. Kaiser, and Gaurav S. Sukhatme. Towards spatial and semantic mapping in aquatic environments. In *IEEE International Conference on Robotics and Automation*, pages 629 – 636, Pasadena, CA, May 2008.

[CHL+05] Howie M Choset, Seth Hutchinson, Kevin M Lynch, George Kantor, Wolfram Burgard, Lydia E Kavraki, and Sebastian Thrun. *Principles of Robot Motion: Theory, Algorithms, and Implementation*. MIT press, 2005.

[CK09] Suman Chakravorty and S. Kumar. Generalized sampling based motion planners with application to nonholonomic systems. In *2009 IEEE International Conference on Systems, Man and Cybernetics*, pages 4077–4082. IEEE, 10 2009.

[CLP17] Hao Chen, Kim Laine, and Rachel Player. Simple encrypted arithmetic

library-seal v2. 1. In *Proceedings of the International Conference on Financial Cryptography and Data Security*, pages 3–18. Springer, 2017.

[CLRS01] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press, Cambridge, MA, 2001.

[CLY+20] Qinyue Chen, Sheue-Er Low, Jeremiah WE Yap, Adjovi KX Sim, Yu-Yang Tan, Benjamin WJ Kwok, Jeannie SA Lee, Chek-Tien Tan, Wan-Ping Loh, Bernard LW Loo, et al. Immersive virtual reality training of bioreactor operations. In *2020 IEEE International Conference on Teaching, Assessment, and Learning for Engineering (TALE)*, pages 873–878. IEEE, 2020.

[CMR10] A. Comport, E. Malis, and P. Rives. Real-time quadrifocal visual odometry. *International Journal of Robotics Research, Special issue on Robot Vision*, 29(2 - 3):245 – 266, 2010.

[CS02] L. G. Crespo and J. Q. Sun. Stochastic Optimal Control of Nonlinear Systems via Short-Time Gaussian Approximation and Cell Mapping. *Nonlinear Dynamics*, 28(3/4):323–342, 2002.

[CSM+08] David A Caron, Beth Stauffer, Steffi Moorthi, Amarjeet Singh, Maxim Batalin, Eric Graham, Mark Hansen, William Kaiser, Jnaneshwar Das, Arvind A Pereira, Amit Dhariwal, Bin Zhang, Carl Oberg, and Gaurav S Sukhatme. Macro- to fine-scale spatial and temporal distributions and dynamics of phytoplankton and their environmental driving forces in a small subalpine lake in southern {c}alifornia, {usa}. *Journal of Limnology and Oceanography*, 53(5):2333–2349, 2008.

[DA01] Wenliang Du and Mikhail J Atallah. Secure multi-party computation problems and their applications: a review and open problems. In *Proceedings of the Workshop on New Security Paradigms*, pages 13–22, 2001.

[DGA00] Arnaud Doucet, Simon Godsill, and Christophe Andrieu. On sequential Monte Carlo sampling methods for Bayesian filtering. *Statistics and Computing*, 10(3):197–208, 2000.

[DMM+11] J Das, T Maughan, M McCann, M Godin, T O'Reilly, M Messie, F Bahr, K Gomes, F Py, J Bellingham, G Sukhatme, and K Rajan. Towards mixed-initiative, multi-robot field experiments: Design, deployment, and lessons learned. In *Proceedings of the Intelligent Robots and Systems (IROS) Conference*, 2011.

[DPM+12] J Das, F Py, T Maughan, M Messie, T O'Reilly, J Ryan, G S Sukhatme, and K Rajan. Coordinated Sampling of Dynamic Oceanographic Features with AUVs and Drifters. *Intnl. J. of Robotics Research*, 31(5):626–646, 2012.

[dSTMdS07] Jorge Estrela da Silva, Bruno Terra, Ricardo Martins, and Joao Borges de Sousa. Modeling and simulation of the lauv autonomous underwater vehicle. In *13th IEEE IFAC international conference on methods and models in automation and robotics*, volume 1. Szczecin, Poland Szczecin, Poland, 2007.

[DT15] Robin Deits and Russ Tedrake. Efficient mixed-integer planning for UAVs in cluttered environments. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 42–49. IEEE, 5 2015.

[DWB06] Hugh Durrant-Whyte and Tim Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics & Automation Magazine*, 13(2):99–110, 2006.

[Dyna] Boston Dynamics. `https://www.bostondynamics.com/atlas`.

[Dynb] Boston Dynamics. `https://www.bostondynamics.com/spot`.

[EGT10] David Eppstein, Michael T Goodrich, and Roberto Tamassia. Privacy-preserving data-oblivious geometric algorithms for geographic data. In *Proceedings of the 18th SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 13–22. ACM, 2010.

[Eme09] Xavier Emery. The kriging update equations and their application to the selection of neighboring data. *Computational Geosciences*, 13(3):269–280, Sep 2009.

[FA04] Keith B Frikken and Mikhail J Atallah. Privacy preserving route planning. In *Proceedings of the ACM Workshop on Privacy in the Electronic Society*, pages 8–15, 2004.

[faa] Drone airborne collisions report. `https://www.faa.gov/newsroom/researchers-release-report-drone-airborne-collisions?newsId=89246`. Published: 28-11-2017, Accessed: 01-10-2021.

[FLCS12] C. Fookes, F. Lin, V. Chandran, and S. Sridharan. Evaluation of image resolution and super-resolution on face recognition performance. *Journal of Visual Communication and Image Representation*, 23(1):75 – 93, 2012.

[FZ03] C. Fruh and A. Zakhor. Constructing 3d city models by merging aerial and

ground views. *IEEE Computer Graphics and Applications*, 23(6):52 – 61, 2003.

[GB09] Craig Gentry and Dan Boneh. *A fully homomorphic encryption scheme*, volume 20. Stanford University Stanford, 2009.

[GKM+17] Stephanie Gil, Swarun Kumar, Mark Mazumder, Dina Katabi, and Daniela Rus. Guaranteeing spoof-resilient multi-robot networks. *Autonomous Robots*, 41(6):1383–1400, 2017.

[GMF19] Alessio Gambi, Marc Mueller, and Gordon Fraser. Automatically testing self-driving cars with search-based procedural content generation. In *Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis*, pages 318–328, 2019.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, pages 218–229, 1987.

[GN01] Jose E Guivant and Eduardo Mario Nebot. Optimization of the simultaneous localization and map-building algorithm for real-time implementation. *IEEE Transactions on Robotics and Automation*, 17(3):242–257, 2001.

[HAGS09] Sandeep Hans, Sarat C Addepalli, Anuj Gupta, and Kannan Srinathan. On privacy preserving convex hull. In *Proceedings of the International Conference on Availability, Reliability and Security*, pages 187–192, 2009.

[HHS04] Tomislav Hengl, Gerard BM Heuvelink, and Alfred Stein. A generic fra-

mework for spatial prediction of soil variables based on regression-kriging. *Geoderma*, 120(1-2):75–93, 2004.

[hJKF15] Jeong hwan Jeon, Sertac Karaman, and Emilio Frazzoli. Optimal sampling-based feedback motion trees among obstacles for controllable linear systems with linear constraints. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 4195–4201, 2015.

[HSF+13] Daniel Hernandez, Ryan N Smith, Enrique Fernandez, Josep Isern, Jorge Cabrera, Antonio Dominguez, and Victor Prieto. Glider path-planning for optimal sampling of mesoscale eddies. In *Fourteenth International Conference on Computer Aided Systems Theory, Workshop on Marine Robotics and Applications*, 2 2013.

[Hsu13] Chieh Su Hsu. *Cell-to-cell mapping: A method of global analysis for nonlinear systems*, volume 64. Springer Science & Business Media, 2013.

[JEK+15] Raja Jurdak, Alberto Elfes, Branislav Kusy, Ashley Tews, Wen Hu, Emili Hernandez, Navinda Kottege, and Pavan Sikka. Autonomous surveillance for biosecurity. *Trends in biotechnology*, 33(4):201–207, 2015.

[JKR+19] Rae Jeong, Jackie Kay, Francesco Romano, Thomas Lampe, Tom Rothorl, Abbas Abdolmaleki, Tom Erez, Yuval Tassa, and Francesco Nori. Modelling generalized forces with reinforcement learning for sim-to-real transfer. *arXiv preprint arXiv:1910.09471*, 2019.

[JP12] Léonard Jaillet and Josep M Porta. Path planning under kinematic cons-

traints by rapidly exploring manifolds. *IEEE Transactions on Robotics*, 29(1):105–117, 2012.

[KF11] Sertac Karaman and Emilio Frazzoli. Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, 30(7):846–894, 2011.

[KH04] Nathan Koenig and Andrew Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)(IEEE Cat. No. 04CH37566)*, volume 3, pages 2149–2154. IEEE, 2004.

[kiw] Kiwi campus. `https://commons.wikimedia.org/wiki/File:Kiwibot_Berkeley_Front.jpg`.

[KLM96] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. Reinforcement learning: A survey. *Journal of Artificial Intelligence Research*, 4:237–285, 1996.

[L+12] Steven M LaValle et al. Sensing and filtering: A fresh perspective based on preimages and information spaces. *Foundations and Trends® in Robotics*, 1(4):253–372, 2012.

[LaV98] Steven M LaValle. Rapidly-exploring random trees: A new tool for path planning. 1998.

[LaV06a] Steven M. LaValle. Planning Algorithms. *Cambridge University Press*, 2006.

[LaV06b] Steven M LaValle. *Planning Algorithms*. Cambridge University Press, Cambridge, U.K., 2006. Available at http://planning.cs.uiuc.edu/.

[LaV11a] S. M. LaValle. Motion planning. *IEEE Robotics & Automation Magazine*, 18(2):108–118, 6 2011.

[LaV11b] S. M. LaValle. Motion planning. *IEEE Robotics & Automation Magazine*, 18(1):79–89, March 2011.

[LaV11c] Steven M. LaValle. Motion planning: Wild frontiers. *IEEE Robotics Automation Magazine*, 18(2):108–118, 2011.

[LDFT16] Benoit Landry, Robin Deits, Peter R Florence, and Russ Tedrake. Aggressive quadrotor flight through cluttered environments using mixed integer programming. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1469–1475. IEEE, 5 2016.

[Led99] Jim A Ledin. Hardware-in-the-loop simulation. *Embedded Systems Programming*, 12:42–62, 1999.

[LL06] Stephen R Lindemann and Steven M LaValle. Multiresolution approach for motion planning under differential constraints. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 139–144. IEEE, 2006.

[LSF07] C. Liu, H.-Y. Shum, and W. Freeman. Face hallucination: Theory and practice. *Internaltional Journal of Computer Vision*, 75(1):115 – 134, 10 2007.

[LSYF08] Kevin M. Lynch, Ira B. Schwartz, Peng Yang, and Randy A. Freeman. Decentralized Environmental Modeling by Mobile Sensor Networks. *IEEE Transactions on Robotics*, 24(3):710–724, June 2008.

[LV11] Steven La Valle. Motion Planning. *IEEE Robotics & Automation Magazine*, 18(2):108–118, 6 2011.

[LYHZ11] Kaitai Liang, Bo Yang, Dake He, and Min Zhou. Privacy-preserving computational geometry problems on conic sections. *Journal of Computational Information Systems*, 7(6):1910–1923, 2011.

[M+65] Gordon E Moore et al. Cramming more components onto integrated circuits, 1965.

[man] `https://commons.wikimedia.org/wiki/File:Manufacturing_equipment_116.jpg`.

[Mas85a] M. Mason. The mechanics of manipulation. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 544–548. Institute of Electrical and Electronics Engineers, 1985.

[Mas85b] Matthew Mason. The mechanics of manipulation. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 2, pages 544–548, 1985.

[McC21] Bethany H. McCarthy. Ibm unveils world's first 2 nanometer chip technology, opening a new frontier for semiconductors. `https://newsroom.ibm.com/2021-05-06-IBM-Unveils-Worlds-First-2-Nanometer-Chip-Technology,-Opening-a-New-Frontier-for-Semiconductors#assets_all`, May 2021. Accessed: 01-10-2021.

[MNP+04] Dahlia Malkhi, Noam Nisan, Benny Pinkas, Yaron Sella, et al. Fairplay-

secure two-party computation system. In *Proceedings of the USENIX Security Symposium*, volume 4, page 9. San Diego, CA, USA, 2004.

[MRCS17] Yuriy Mileyko, Gregory Reis, Monique Chyba, and Ryan N Smith. Energy-efficient control strategies for updating an augmented terrain-based navigation map for autonomous underwater navigation. In *2017 IEEE Conference on Control Technology and Applications (CCTA)*, pages 223–228. IEEE, 8 2017.

[MSV+16] Musa Morena Marcusso Manhães, Sebastian A. Scherer, Martin Voss, Luiz Ricardo Douat, and Thomas Rauschenbach. UUV simulator: A gazebo-based package for underwater intervention and multi-robot simulation. In *OCEANS 2016 MTS/IEEE Monterey*. IEEE, sep 2016.

[MT16] Anirudha Majumdar and Russ Tedrake. Funnel Libraries for Real-Time Robust Feedback Motion Planning. 1 2016.

[MT17] Anirudha Majumdar and Russ Tedrake. Funnel libraries for real-time robust feedback motion planning. *The International Journal of Robotics Research*, 36(8):947–982, 2017.

[Mus] Elon Musk. Tesla ai day. `https://www.youtube.com/watch?v=j0z4FweCy4M`. Accessed: 01-10-2021.

[N+19] Sen Nag et al. How much of the ocean have we explored? *WorldAtlas*, 2019.

[NGN06] Juan Nieto, Jose Guivant, and Eduardo Nebot. Denseslam: Simultaneous localization and dense mapping. *The International Journal of Robotics Research*, 25(8):711–744, 2006.

[OL08] Jason M O'Kane and Steven M LaValle. Comparing the power of robots. *The International Journal of Robotics Research*, 27(1):5–23, 2008.

[OS15] Jason M O'Kane and Dylan A Shell. Automatic design of discreet discrete filters. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 353–360, 2015.

[Pai99] Pascal Paillier. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of the International Conference on the Theory and Applications of Cryptographic Techniques*, pages 223–238. Springer, 1999.

[PCY+16a] Brian Paden, Michal Cap, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A Survey of Motion Planning and Control Techniques for Self-Driving Urban Vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 3 2016.

[PČY+16b] Brian Paden, Michal Čáp, Sze Zheng Yong, Dmitry Yershov, and Emilio Frazzoli. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on Intelligent Vehicles*, 1(1):33–55, 2016.

[PEWF08] S. Prince, J. Elder, J. Warrell, and F. Felisberti. Tied factor analysis for face recognition across large pose differences. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(6):970 – 984, 2008.

[Pix] Pixabay. `https://pixabay.com/de/photos/drohne-flugdrohne-himmel-sonne-1237439/`.

[PK16] Amanda Prorok and Vijay Kumar. A macroscopic privacy model for heterogeneous robot swarms. In *International Conference on Swarm Intelligence*, pages 15–27. Springer, 2016.

[Pyc] Pycreate. `https://pypi.org/project/pycreate2/0.5.2/`.

[PYS] Pyseal. `https://github.com/Lab41/PySEAL`https://github.com/Lab41/PySEAL.

[pyt] Python paillier implementation. `https://python-paillier.readthedocs.io/en/develop/`.

[QGC+09] M Quigley, B Gerkey, K Conley, J Faust, T Foote, J Leibs, E Berger, R Wheeler, and A Y Ng. Ros: an open-source robot operating system. In *Proceedings of the Open-Source Software workshop of the International Conference on Robotics and Automation*, 2009.

[RAK21] Stephan R Richter, Hassan Abu AlHaija, and Vladlen Koltun. Enhancing photorealism enhancement. *arXiv preprint arXiv:2105.04619*, 2021.

[RFA+17a] Gregory Murad Reis, Michael Fitzpatrick, Jacob Anderson, Leonardo Bobadilla, and Ryan N. Smith. Augmented Terrain-Based Navigation to Enable Persistent Autonomy for Underwater Vehicles. In *2017 First IEEE International Conference on Robotic Computing (IRC)*, pages 292–298. IEEE, 4 2017.

[RFA+17b] Gregory Murad Reis, Michael Fitzpatrick, Jacob Anderson, Leonardo Bobadilla, and Ryan N. Smith. Increasing persistent navigation capabilities for underwater vehicles with augmented terrain-based navigation. In *MTS/IEEE Oceans*, Aberdeen, Scotland, April 2017. Best Student Paper Finalist.

[RFA+17c] Gregory Murad Reis, Michael Fitzpatrick, Jacob Anderson, Jonathan Kelly, Leonardo Bobadilla, and Ryan N. Smith. Increasing persistent navigation capabilities for underwater vehicles with augmented terrain-based navigation. In *OCEANS 2017 - Aberdeen*, pages 1–8. IEEE, 6 2017.

[RK92] E Rimon and DE Koditschek. Exact robot navigation using artificial potential functions. *IEEE Transactions on Robotics and Automation*, 8(5):501–518, 1992.

[rov] Mars exploration rovers. `https://mars.nasa.gov/mer/`.

[RP03] D L Rudnick and M J Perry. ALPS: Autonomous and Lagrangian Platforms and Sensors, Workshop Report. Technical report, http://www.geoprose.com/ALPS, 2003.

[RRNT06] David Ribas, Pere Ridao, Jose Neira, and Juan D Tardos. SLAM using an imaging sonar for partially structured underwater environments. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5040–5045, 2006.

[RRTN07] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. Underwater SLAM in a marina environment. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 1455–1460, 2007.

[RRTN08] David Ribas, Pere Ridao, Juan Domingo Tardós, and José Neira. Underwater SLAM in man-made structured environments. *Journal of Field Robotics*, 25(11-12):898–921, 2008.

[RS17] Venkatraman Renganathan and Tyler Summers. Spoof resilient coordination for distributed multi-robot systems. In *Proceedings of the International Symposium on Multi-Robot and Multi-Agent Systems*, pages 135–141, 2017.

[RVR+17] Andrei A Rusu, Matej Večerík, Thomas Rothörl, Nicolas Heess, Razvan Pascanu, and Raia Hadsell. Sim-to-real robot learning from pixels with progressive nets. In *Conference on Robot Learning*, pages 262–270. PMLR, 2017.

[SB18] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[SCL+10] Ryan N Smith, Yi Chao, Peggy P Li, David A Caron, Burton H Jones, and Gaurav S Sukhatme. Planning and Implementing Trajectories for Autonomous Underwater Vehicles to Track Evolving Ocean Processes based on Predictions from a Regional Ocean Model. *International Journal of Robotics Research*, 29(12):1475–1497, October 2010.

[SE12] T. Senlet and A. Elgammal. Satellite image based precise robot localization on sidewalks. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 2647 – 2653, 5 2012.

[SEA] Simple encrypted arithmetic library (seal). `https://python-paillier.readthedocs.io/en/develop/`.

[SH90] J Q Sun and CS T Hsu. The generalized cell mapping method in nonlinear random vibration based upon short-time gaussian approximation. *Journal of Applied Mechanics*, 57(4):1018–1025, 1990.

[SJP15] Edward Schmerling, Lucas Janson, and Marco Pavone. Optimal sampling-based motion planning under differential constraints: the driftless case. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2368–2375. IEEE, 2015.

[SLOK96] P Svestka, JC Latombe, and LE Overmars Kavraki. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[SLS15] Andrew Stuntz, David Liebel, and Ryan N Smith. Enabling persistent autonomy for underwater gliders through terrain based navigation. In *OCEANS 2015-Genova*, pages 1–10. IEEE, 2015.

[SM05] 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(4):347–404, 1 2005.

[SMC+12] Alexandre Sousa, Luis Madureira, Jorge Coelho, José Pinto, João Pereira, João Borges Sousa, and Paulo Dias. Lauv: The man-portable autonomous underwater vehicle. *IFAC Proceedings Volumes*, 45(5):268–274, 2012.

[SNL20] Markku Suomalainen, Alexandra Q Nilles, and Steven M LaValle. Virtual reality for robots. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11458–11465. IEEE, 2020.

[SNS11] Roland Siegwart, Illah Reza Nourbakhsh, and Davide Scaramuzza. *Introduction to Autonomous Mobile Robots*. MIT press, 2011.

[SP11] Nitin Sydney and Derek A. Paley. Multi-vehicle control and optimization for spatiotemporal sampling. In *IEEE Conference on Decision and Control and European Control Conference*, pages 5607–5612. IEEE, December 2011.

[Ste12] Michael L Stein. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[SYDP06] Li Shundong, Dai Yigi, Wang Daoshun, and Luo Ping. A secure multi-party computation solution to intersection problems of sets and rectangles. *Progress in Natural Science*, 16(5):538–545, 2006.

[T+02] Sebastian Thrun et al. Robotic mapping: A survey. *Exploring artificial intelligence in the new millennium*, 1(1-35):1, 2002.

[Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1(2):146–160, 1972.

[TBF05] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics*. MIT press, 2005.

[Tes] Artificial intelligence & autopilot. `https://www.tesla.com/AI`. Accessed: 01-10-2021.

[Thr00] Sebastian Thrun. Probabilistic algorithms in robotics. *AI Magazine*, 21(4):93–93, 2000.

[TKS+18] A. J. Titus, S. Kishore, T. Stavish, S. M. Rogers, and K. Ni. PySEAL: A Python wrapper implementation of the SEAL homomorphic encryption library. *ArXiv e-prints*, March 2018.

[TMTR10] Russ Tedrake, Ian R Manchester, Mark Tobenkin, and John W Roberts. LQR-trees: Feedback motion planning via sums-of-squares verification. *The International Journal of Robotics Research*, 29(8):1038–1052, 2010.

[URO+08] Ben Upcroft, M.F. Ridley, L. Ong, B. Douillard, T. Kaupp, S. Kumar, T.A. Bailey, Fabio Ramos, A. Makarenko, A. Brooks, S. Sukkarieh, and H F Durrant-Whyte. Multi-level state estimation in an outdoor decentralised sensor network. *Springer Tracts in Advanced Robotics - Experimental Robotics*, 39:355 – 365, 2008.

[usv] Usv, los drones marítimos. `https://www.feindef.com/blog/usv-los-drones-maritimos/`. [Online; accessed 01-10-2021].

[uuv] Marum center for marine environmental sciences, university of bremen. `https://www.marum.de/en/Discover/Technology.html`.

[VDP+18] António AC Vieira, Luís MS Dias, Guilherme AB Pereira, José A Oliveira, Maria Do Sameiro Carvalho, and Paulo Martins. Simulation model generation for warehouse management: Case study to test different storage strategies. *International Journal of Simulation and Process Modelling*, 13(4):324–336, 2018.

[VPSP17] Ulrich Viereck, Andreas Pas, Kate Saenko, and Robert Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In Sergey Levine, Vincent Vanhoucke, and Ken Goldberg, editors, *Proceedings of the 1st Annual Conference on Robot Learning*, volume 78 of *Proceedings of Machine Learning Research*, pages 291–300. PMLR, 13–15 Nov 2017.

[way] Waymo pacifica driverless minivan. `https://commons.wikimedia.org/wiki/File:Waymo-Pacifica.jpg`.

[WSF+18] Florian Wirnshofer, Philipp S. Schmitt, Wendelin Feiten, Georg v. Wichert, and Wolfram Burgard. Robust, Compliant Assembly via Optimal Belief Space Planning. 11 2018.

[WVDB13] Dustin J Webb and Jur Van Den Berg. Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 5054–5061, 2013.

[WZ11] Wencen Wu and Fumin Zhang. Cooperative exploration of level surfaces of three dimensional scalar fields. *Automatica*, 47(9):2044–2051, September 2011.

[XQX+14] Fu-Rui Xiong, Zhi-Chang Qin, Yang Xue, Oliver Schütze, Qian Ding, and Jian-Qiao Sun. Multi-objective optimal design of feedback controls for dynamical systems with hybrid simple cell mapping algorithm. *Communications in Nonlinear Science and Numerical Simulation*, 19(5):1465–1473, 5 2014.

[Yao82] Andrew C Yao. Protocols for secure computations. In *Proceedings of the 23rd Annual Symposium on Foundations of Computer Science*, pages 160–164, 1982.

[Yao86] Andrew Chi-Chih Yao. How to generate and exchange secrets. In *Proceedings of the 27th Annual Symposium on Foundations of Computer Science*, pages 162–167, 1986.

[YF16a] Dmitry S. Yershov and Emilio Frazzoli. Asymptotically optimal feedback planning using a numerical Hamilton-Jacobi-Bellman solver and an adaptive mesh refinement. *The International Journal of Robotics Research*, 35(5):565–584, 4 2016.

[YF16b] Dmitry S Yershov and Emilio Frazzoli. Asymptotically optimal feedback planning using a numerical Hamilton-Jacobi-Bellman solver and an adaptive mesh refinement. *The International Journal of Robotics Research*, 35(5):565–584, 2016.

[ZFP+07] Fumin Zhang, D. M. Fratantoni, Derek Paley, J. Lund, and Naomi E. Leonard. Control of coordinated patterns for ocean sampling. *International Journal of Control*, 80(7):1186 – 1199, 2007.

[ZGBR12] Y Zhang, M A Godin, J G Bellingham, and J P Ryan. Using an Autonomous Underwater Vehicle to Track a Coastal Upwelling Front. *IEEE Journal of Oceanic Engineering*, 37(3):338–347, July 2012.

[ZLM09] Liangjun Zhang, Steven M LaValle, and Dinesh Manocha. Global vector field computation for feedback motion planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, pages 477–482, 2009.

[ZQW20] Wenshuai Zhao, Jorge Peña Queralta, and Tomi Westerlund. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 737–744. IEEE, 2020.

[ZS18] Yulin Zhang and Dylan A Shell. Complete characterization of a class of

privacy-preserving tracking problems. *International Journal of Robotics Research*, 2018.