TOPOLOGICAL AND GEOMETRIC TECHNIQUES IN GRAPH SEARCH-BASED ROBOT PLANNING

Subhrajit Bhattacharya

A DISSERTATION

in

Mechanical Engineering and Applied Mechanics

Presented to the Faculties of the University of Pennsylvania

in

Partial Fulfillment of the Requirements for the

Degree of Doctor of Philosophy

2012

TOPOLOGICAL AND GEOMETRIC TECHNIQUES IN
GRAPH SEARCH-BASED ROBOT PLANNING

Acknowledgments

I would like to express my sincere gratitude towards my advisors, Dr. Vijay Kumar and Dr. Maxim Likhachev, for their continuous support and guidance. Their mentoring and help all these years have been indispensable in my research that has culminated in this thesis.

I would like to thank the rest of my thesis committee, Dr. George J. Pappas, Dr. Daniel E. Koditschek and Dr. Robert Ghrist, for their valuable time, effort and suggestions.

My sincere gratitude goes to Dr. Robert Ghrist and Dr. David Lipsky for sharing their valuable insights into topics on algebraic topology, and for collaborating on the work related to homology of punctured Euclidean spaces. I would also like to thank Dr. Nathan Michael and Dr. Luciano Pimenta for their collaboration on the work on exploration and coverage.

My sincere thanks to the Department of Mechanical Engineering and Applied Mechanics, and the GRASP laboratory of University of Pennsylvania for providing such a wonderful opportunity and an amazing academic experience throughout my doctoral studies.

My special thanks goes to the Wikimedia foundation for providing easy and quick reference to a vast amount of knowledge through Wikipedia. Much of this thesis, especially the illustrations in it, wouldn't have been made possible without the amazing community-maintained software provided by the Blender foundation, Apache software foundation and other open-source software – my sincere appreciation and thanks to them. I would also like to thank Dr. N. J. Wildberger, UNSW, for his online video lectures on algebraic topology, and Mr. D. Simeonov and Mr. M. Fleder, MIT, for providing the example in Figure 3.3(a) of this thesis.

Finally, my heartiest thanks goes to my parents, friends and family for their support and encouragement all along.

ABSTRACT

TOPOLOGICAL AND GEOMETRIC TECHNIQUES IN GRAPH SEARCH-BASED ROBOT PLANNING

Subhrajit Bhattacharya

Vijay Kumar and Maxim Likhachev

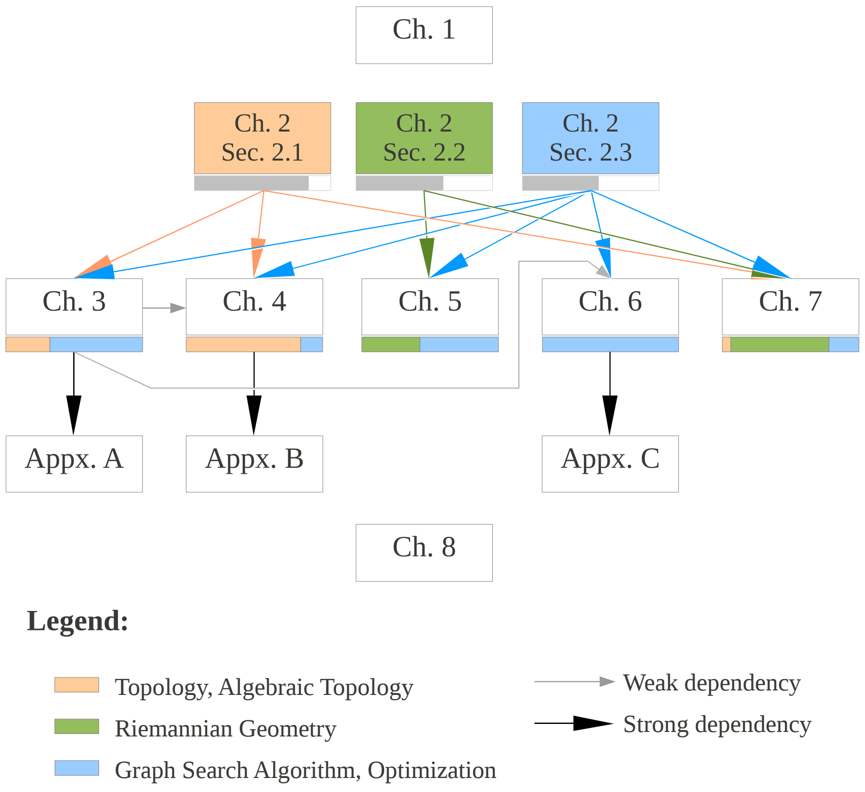
Search-based techniques have been widely used in robot path planning for finding optimal or close-to-optimal trajectories in configuration spaces. They have the advantages of being complete, optimal (up to the metric induced by the discretization) and efficient (in low dimensional problems), and broadly applicable, even to complex environments. Continuous techniques, on the other hand, that incorporate concepts from differential and algebraic topology and geometry, have the ability to exploit specific structures in the original configuration space and can be used to solve different problems that do not lend themselves to graph-search based techniques. We propose several novel ideas and develop new methodologies that will let us bring these two separate techniques under one umbrella. Using tools from algebraic topology we define differential forms with special properties whose integral reveal topological information about the solution path allowing us to impose topological constraints on the planning problems. Metric information can be used along with search-based techniques for creating Voronoi tessellations in coverage and exploration problems. In particular, we use entropy as a metric for multi-robot exploration and coverage of unknown or partially known non-convex environments. Finally, in multi-robot constrained planning problems we exploit certain special product structure in the high dimensional configuration space that combine the advantages of graph search methods and gradient descent algorithms allowing us to develop powerful tools to solve very high-dimensional planning problems.

Contents

List of Tables

List of Figures

Chapter Dependencies



The above diagram gives an overview of how the chapters and sections in this thesis depend on each other. The Chapter 2 discusses some of the preliminary mathematical and algorithmic tools used in this thesis. How much depth those discussions attain is approximately indicated by the length of the gray bar below each section of the chapter in the above diagram. The colored bars below each chapter approximately indicate how much the corresponding chapter uses the respective mathematical/algorithmic tools.

Chapter 1

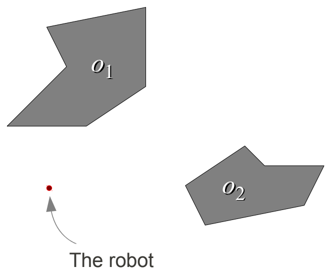
Introduction

1.1 Configuration Spaces

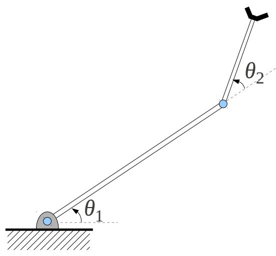
Many problems in robotics involve a configuration space. Configuration space or *C-space* of a robot or a system of robots is the abstract space of possible states or configurations that the system can attain. Thus each point in the C-space corresponds to a possible state of the robot(s) in the real environment. Robotics problems, especially planning problems, typically involve navigation of the system through the C-space space in order to achieve certain tasks or objectives. This translates to finding a 1-dimensional curve (a trajectory) in the C-space that the system needs to follow. Typically C-spaces are smooth manifolds and any curve on it is a possible trajectory for the system. However, presence of kinematic and dynamic constraints may require that the tangent at any point on the trajectory lies within a specific subset of the tangent space at the point of the C-space. Furthermore, in presence of a metric in the C-space or a more general measure for 1-dimensional curves in the space, one can talk about optimality of the trajectory.

Typically the C-space of a system can be parametrized by the different state variables corresponding to the different degrees of freedom. For example, the configuration space of a single point mobile robot navigating in a unbounded 2-dimensional flat plane with obstacles is simply $\mathbb{R}^2 - \mathcal{O}$, where $\mathcal{O}$ represents the set of points on the plane that make up the obstacles (Figure 1.1(a)). Similarly, the configuration space of a planar robot arm with two links and no joint angle limits (Figure 1.1(b)) is a *torus*, $\mathbb{T} = \mathbb{S}^1 \times \mathbb{S}^1$, each point on which correspond to an unique pair of joint angles θ_1, θ_2 (Figure 1.1(c)). One can generalize the notion of C-space for a system with multiple robots. For example, if $\mathcal{C} = \mathbb{R}^2 - \mathcal{O}$ is the configuration space of a point robot as described in Figure 1.1(a), the presence of n robots in the environment will result in a *joint* configuration space for the system of n robots described by $\overline{\mathcal{C}} = \mathcal{C} \times \mathcal{C} \times \cdots \times \mathcal{C} - \Delta = \mathcal{C}^n - \Delta$, where we take the product of n copies of the C-spaces corresponding to each robot, and remove from it the *diagonal* that represents collision of the robots (*i.e.*, $\Delta = \{[\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_n] \in \mathcal{C}^n \mid \mathbf{p}_i = \mathbf{p}_j \text{ for some } i \neq j\}$). Clearly, the joint configuration space is a $2n$ dimensional manifold.

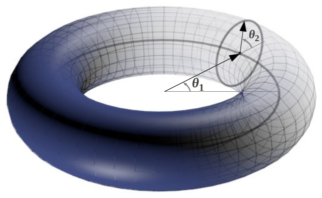
We now impose a very simple additional structure to the C-space of a point robot. Let us consider a *unicycle* model of the point robot [20], which means, in addition to the position (x, y), the robot has an orientation (θ). Thus the configuration space of the robot is now $(\mathbb{R}^2 - \mathcal{O}) \times \mathbb{S}^1$



(a) The configuration space of a point robot navigating on a plane with obstacles o_1 and o_2 is $\mathbb{R}^2 - (o_1 \cup o_2)$



(b) A 2-link robotic arm is described by the state variables θ_1 and θ_2.



(c) The configuration space of a 2-link robotic arm is the 2-torus.

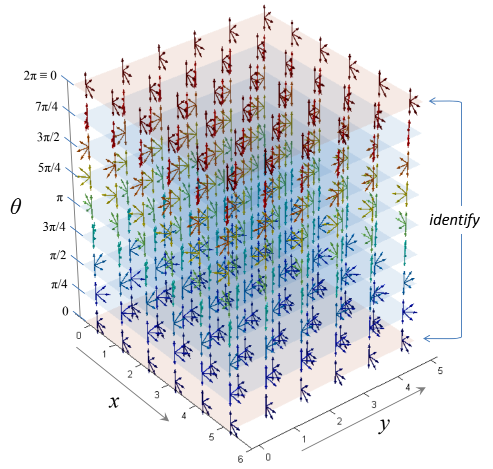
Figure 1.1: Examples of simple configuration spaces.

(a subset of $SE(2)$). Moreover, the unicycle model of the robot is non-holonomic since the robot can only move forward along the direction it is oriented, or rotate at a fixed place (*i.e.* x, y remains fixed, while θ changes). So at point $p = (x, y, \theta)$ in its configuration space, the robot can move along $(\dot{x}, \dot{y}, \dot{\theta}) = (v\cos(\theta), v\sin(\theta), \omega)$, for some $v \in \mathbb{R}_+$ and $\omega \in \mathbb{R}$ (representing forward and angular speeds). Thus the possible directions of motion is a 2-dimensional manifold generated by v and ω. This is the space of possible actions at (x, y, θ), which, in general, may not be a vector space, and in this particular example is a half space (a subset of the tangent space $T_p\mathcal{C}$). Attaching this space of possible actions to every point of the C-space (Figure 1.2(a)), we obtain a fiber bundle. The fiber bundle itself is a 5-dimensional manifold, sections of which give vector fields in the configuration space. A valid trajectory in the C-space needs to be such that the tangent at every point on it lies in the set of possible actions at that point (Figure 1.2(c)).

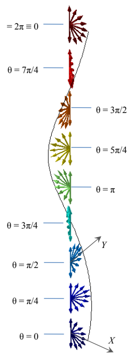
1.1.1 Continuous Approaches to Robot Motion Planning

The problem of navigating a robot (or a system of robots) from a start coordinate to a goal coordinate in the C-space is of much interest in robotics [55] and is typically referred to as "*goal directed navigation*". In continuous planning methods, one tries to derive either open loop trajectories [87] or closed loop feedback policies [69, 15] that avoid obstacles while satisfying constraints on the robot dynamics. However, it is difficult to establish completeness and convergence results except in special cases. One school of approach in solving the problem of *goal directed navigation* for a fixed goal coordinate in the C-space is to generate a vector field in the entire C-space that would drive the system towards the goal coordinate. However in most C-spaces, non-trivial topology and the presence of obstacles result in great challenges in generating such vector fields. In addition, kinematic and dynamic constraints pose additional difficulties.

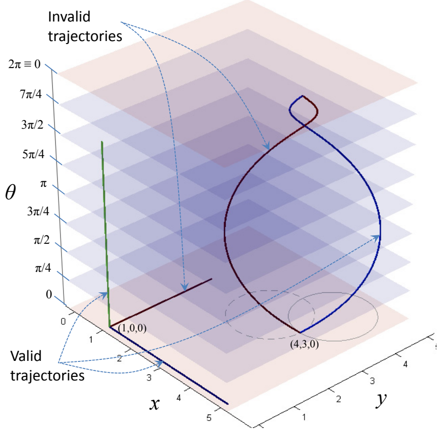
For a point robot navigating on a flat plane, one may hope to find a potential function, the gradient of which would give the desired vector field. However, initial attempts to construct potential functions like that suffered from lack of global convergence due to presence of local minima [51, 52, 15]. One of the most intriguing and elegant constructions to deal with the problem successfully was that of a *navigation function* introduced in [70]. The basic version of this machinery lets one



(a) The configuration space $\mathbb{R}^2 \times \mathbb{S}^1$, and the action space attached to every point in it (the arrows in the figure are some representative actions – the action spaces themselves are half-spaces).

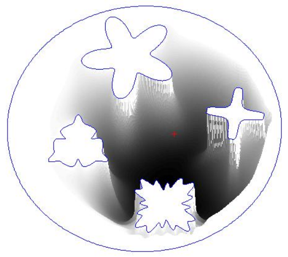


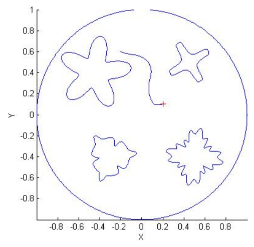
(b) Closer look at how the space of possible actions change with θ.



(c) Some valid and invalid trajectories in the C-space. The tangent at every point on the trajectory must lie inside the action fiber at the point.

Figure 1.2: Configuration space of a unicycle point robot, and actions due to kinematic constraints. For simplicity we have eliminated obstacles in the environment.


(a) Navigation function in a spherical environment with star-shaped obstacles.



(b) A robot following the gradient of the navigation function reaches goal.

Figure 1.3: Navigation function for robot path planning in $\mathbb{R}^2 - \mathcal{O}$.

construct a potential function in a *spherical* environment with *star-shaped* obstacles such that it has an unique stable minima at the the goal coordinate (Figure 1.3). While this ensures that in presence of a very small noise the robot will converge to its goal, the method suffers from issues of computational complexity, numerical difficulties and slow convergence. Besides, the approach works for a very limited variety of obstacle, and it is difficult to incorporate additional constraints in the problem. Generalizing such an approach for more complex nonlinear dynamical systems becomes increasingly challenging [60].

Vector fields have also been used for applications other than goal directed navigation. In problems like generation of patterns using mobile robots [44], surveillance [45] and transportation of objects via caging [29], vector fields have been successfully generated and employed for navigation of teams of robots in environments with simple obstacles or abstracted obstacles.

In environments with a metric, typically the problem that one is faced with is to find an optimal path. One can attempt to solve the Geodesic equation in a metric space [48]. However, even in metric spaces without discontinuities, the computation of the geodesic passing through two given points is highly non-trivial. One can employ a method like *shooting method* [82] for solving the Geodesic equation posed as a boundary value problem. However, in general, such methods are expensive. In presence of discontinuities due to obstacles, solving for the shortest path (which is informally called the *generalized geodesic*) between two points becomes practically infeasible.

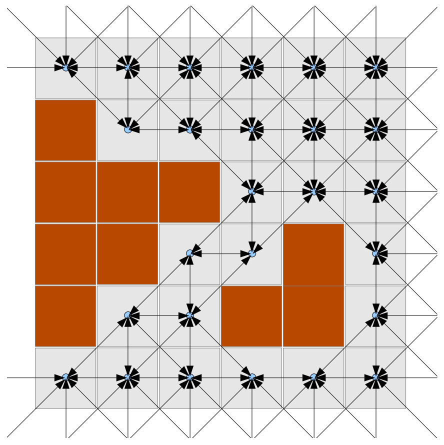
A more general optimal path planning problem arises when, instead of a metric, a general measure in terms of a cost or *action integral* is provided. In such problems, often the method of calculus of variation is adopted [31, 24, 54]. However, such approaches typically require an initial guess of a suboptimal trajectory, are computationally expensive, and compromise on the optimality when there are discontinuities in the environment (such as obstacles).

A related but different problem arises in robot coverage of an environment (*i.e.* how evenly the robots are distributed in an environment). Typically this is a multi-robot problem, in which an initial/start coordinates of the robots are provided (a single coordinate in the joint state-space). However there is no specific single goal coordinate as in *goal directed navigation.* Instead, a flow (a vector field) is given that governs the motion of the system. The vector field is typically a simple gradient of a potential function (which is designed such that it gives a measure of how bad the coverage is). This is the continuous-time version of the Lloyd's Algorithm [58] due to Cortez, *et al.* [18]. As with any potential function based approach, such a method suffers from problems of local minima. In fact Lloyd's algorithm only guarantees that the system will converge to a local minima of the potential. In a convex environment without obstacles this local minima is at least guaranteed to be one where the gradient is zero. However obstacles and non-convexity can result in creation of highly suboptimal local minima. Moreover in non-convex environments and environments with obstacles, the generalization of the potential function proposed in [18, 58] and computation of its gradient, becomes highly non-trivial and extremely computationally expensive.

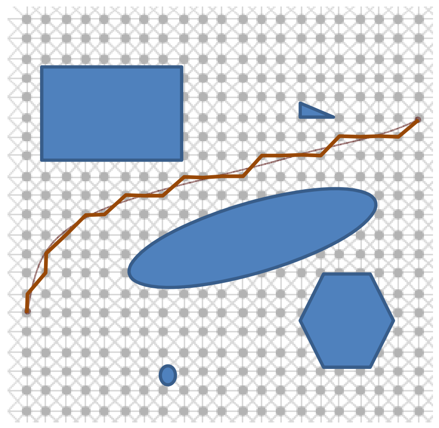
Planning in configuration spaces with arbitrary shaped obstacles, and in presence of nonholonomic constraints, kinematic constraints and dynamic constrains using any continuous approach is generally difficult. Vector field construction and calculus of variation can be employed, but it is difficult to provide guarantees and robustness.

1.1.2 The Discrete Approach

A robust alternative to the continuous approaches described above is the discrete approach of graph search-based planning. The basic idea behind the approach is to sample points from the configuration space, and construct a graph by considering those points as vertices of the graph and edges being sampled possible actions that can take the system from one vertex (a state) to another. A simple example is that of a point robot navigating on a plane with obstacles modeled as a 8-connected grid. The configuration space $\mathbb{R}^2 - \mathcal{O}$ can be partitioned into uniform square cells, and a vertex of the graph is placed at the center of each cell (Figure 1.4(a)). Edges are established between 8 neighboring vertices of each vertex, representing the possible transitions of the robot (the actions).



(a) A graph created by uniform square partitioning/discretization of an environment. The brown cells represent obstacles. Each vertex is connected to its 8 neighbors (except inaccessible vertices).



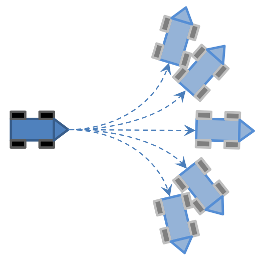
(b) A trajectory in the continuous configuration space can be approximated by a path in the graph.

Figure 1.4: Graph created by uniform discretization of an environment. This specific type of graph shown in the figures is referred to as the 8-connected grid.

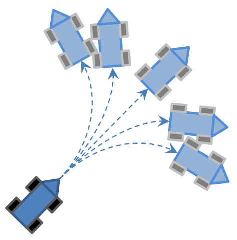
Following this construction, any trajectory in the original configuration space can be approximated by a path in the graph [7] (Figure 1.4(b)). This simple approximation eliminates much of the problems that continuous methods suffer from. Firstly, for goal directed navigation, the problem reduces to finding paths in the graph. One can then employ any search algorithm like Dijkstra's [26], A* [39], D* [81], ARA* [38] or R* [57] to search for the optimal path (or path with bounded sub-optimality depending on the chosen algorithm) in the graph from the start vertex to the goal vertex. Such algorithms are complete with guarantees on optimality or bounds on sub-optimality. Moreover in the graph, it is easy to incorporate metric information by assigning weights to edges. Thus, such discrete approaches work very well for planning trajectories in non-Euclidean metric spaces. Moreover, since actions in a graph are modeled by discrete set of edges, systems with non-holonomic constraints are effectively and easily modeled using this approach.

For example, in [28, 85], in order to effectively model the dynamics and non-holonomic constraints of a car, a *lattice graph* was used. The state variables consist of the position, orientation and forward speed of the vehicle, making the configuration space $SE(2) \times \mathbb{R}_+$. As discussed earlier, depending on the current state of the vehicle, (x, y, θ, v), the possible actions $(\dot{x}, \dot{y}, \dot{\theta}, \dot{v})$ are limited by the non-holonomic and dynamic constraints. This is difficult to model using a continuous approach, and even more difficult to find trajectories in such configuration space using any continuous approach. However a discrete approach of discretization of the configuration space and construction of a graph deals with this very effectively. The graph is constructed by partitioning the 4-dimensional configuration space into cells (may be an uniform partition with hypercubic cells). A vertex is associated with each cell, and edges emanating from the vertex are created so that they are samples from the space of possible actions at the vertex. Two configurations are considered to be the same if they fall in the same partition of the configuration space.

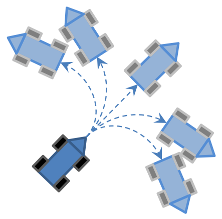
It is quite evident that a graph created in such a way may be too large for effective computation



(a) A discrete sample from the set of possible actions at a vertex at (x, y, θ, v) make up the edges emanating from the vertex.



(b) Actions from a vertex at (x, y, θ', v). Note that the orientation is different from the one shown in (a), but the speed is the same.



(c) Actions from a vertex at (x, y, θ', v'), with $v' < v$. Note that due to lower speed, it is possible for the car to take sharper turns.

Figure 1.5: Some vertices and edges emanating from them in a lattice graph created by discretizing the configuration space of a dynamic, non-holonomic car. A vertex is associated with each discretized cell. From each vertex, edges emanate such that the actions corresponding to the edges are sampled from the space of possible actions from the vertex. The figures show points and actions in configuration space projected to the $X - Y$ plane.

or even storage. However, as we will discuss later, graph search algorithms typically do not require that we explicitly create the graph and store it in memory before starting to solve the problem. In fact, in solving a goal directed navigation problem (*i.e.* searching for shortest path in the graph) it may not even be necessary to create the full graph. A well-informed *heuristic function* can guide heuristic searches to *expand* a small subset of the vertices in the graph. Moreover, efficient graph construction algorithms like Rapidly-exploring Random Tree (RRT) [56] and Probabilistic Roadmaps (PRM) [50] are often employed for efficient sampling of vertices and creating the graph in high dimensional configuration spaces [13].

In spite of having all such tools at hand, it is worth mentioning that given a fixed algorithm, the size of the graph (number of vertices and edges) as well as the complexity of the search algorithms increase exponentially with the dimensionality of the configuration space (number of state variables). While this does not pose a major problem for planning in low dimensional configuration spaces, in higher dimensional configuration spaces (*e.g.* in multi-robot problems, the joint configuration space of all the robots), this becomes a severe computational bottleneck.

Moreover, the process of discretization and sampling themselves tend to undermine much of the richness that was present in the original configuration space. For example, much of the topological information present in the original configuration space gets lost since the graph is rather indifferent to the global topology of the space. Moreover, metric of the original configuration space gets restricted to the graph. Thus, the shortest path in the graph need not necessarily be the shortest path in the original continuous metric space (Figure 1.4(b)). There has been some recent very interesting developments for addressing this later issue [21]. However such methods work reliably only in Euclidean metric spaces.

Most practical implementations in robotics, however, widely employ graph search-based approaches. The ease of implementation, robustness, completeness and guarantees on optimality within the graph most of the time outweigh the problems of sub-optimality due to discretization and loss of geometric and topological information due to graph construction. For example, almost every team in the DARPA Urban Challenge, 2007, used graph-search based planning approaches [85, 62, 42] for finding optimal paths in complex cluttered environments. Planning for robotic arms like that of the PR2 in cluttered environments has been achieved successfully using graph search-based planning [14].

1.2 Contributions of this Thesis

We thus observe that search-based techniques have been widely used in robot path planning for finding optimal trajectories in configuration spaces. They have the advantages of being complete, optimal (up to the metric induced by the discretization), robust and efficient (in low dimensional problems), even in complex environments. Moreover, there is a wide class of search-based techniques that allow one to trade-off optimality (with guaranteed bounds on sub-optimality) at substantial gain on search efficiency. Continuous techniques, on the other hand, lending concepts from differential and algebraic geometry and topology, have the ability of exploit specific structures in the original configuration space and help in solving a host of different problems that rarely come under the scope of graph-search based techniques. The main objective of this thesis is to propose certain ideas and methods that will let us bring these two separate techniques under one umbrella.

The first contribution of the thesis lies in the characterization of the topology of the configuration space and the solution using applications of algebraic topology. Search-based techniques, by virtue of discretization and graph construction, ignore the topological properties of the underlying configuration space and the solution trajectories. We try to account for that by defining differential forms whose integrals reveal topological information about the solution path. Appropriately defined 1-forms allow us to establish equivalence classes of trajectories (*e.g.*, homotopy or homology classes) and use it to guide the search. We show how to find trajectories that are constrained to lie in specified homotopy/homology classes or that avoid other specified classes.

The second contribution is to use search techniques to follow gradient of a potential function that is a generalization of the one used by Lloyd's Algorithm in coverage problems. We use graph search technique to partition the configuration space based on a metric, and hence create *generalized Voronoi tessellations*. The same search technique is used to compute gradient of the function that is to be minimized for attaining good coverage. We show application of this method to multi-robot coverage and exploration tasks in unknown or partially known non-convex environments.

In our third contribution we address the curse of dimensionality that is inherent in path planning for multi-robot systems. One of the main drawbacks of graph search algorithms is that with increase in the dimensionality of the configuration space, the number of nodes and edges in the graph increase exponentially. This poses a major challenge for finding optimal paths in high dimensional configuration spaces using graph search techniques. While gradient descent approaches scale much better with the dimensionality of the configuration space, these methods suffer from local minima, especially in non-convex environments. However, multirobot problems endow a special product structure to the configuration space allowing us to decouple robot directions and parallelize the search. Such decompositions can let us use a combination of graph search methods and gradient descent algorithms in complementary directions. We demonstrate how such decompositions are particularly suitable for multi-robot path planning problems with communication constraints.

Finally we address the problem of determining optimal trajectories for specified metrics. Trajectories found by discrete graph representations and searches suffer from sub-optimality induced by the discretization. However there are certain metric spaces (a trivial example being the Euclidean metric) in which we can conveniently use the notion of visibility to obtain the optimal trajectory with respect to the metric in the original space from the optimal trajectory on the graph. Our goal is to identify such special metric spaces and study the conditions under which we can transform a given non-Euclidean metric space into such special metric spaces.

Chapter 2

Preliminaries

2.1 Basic Algebraic Topology

2.1.1 Background: Point-set Topology

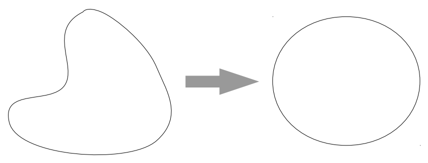
Set theory is the study of collections of some objects. In many cases that collection can be infinite and uncountable. For example, one may talk about the set of all the points on the surface of a sphere. However, set theory does little in establishing relationship between the objects in a set. For example, if we consider the set consisting of the points on a sphere, it's just a collection of points, each of which is distinct and there is no way of telling which point is "connected" to which other point in the set to give the sphere its familiar shape. That's where topology comes to the rescue. A *topology* consists of a set, along with the additional information on "grouping"/"collection" of the objects inside the set. Such "groupings" are called *open subsets* of the set.

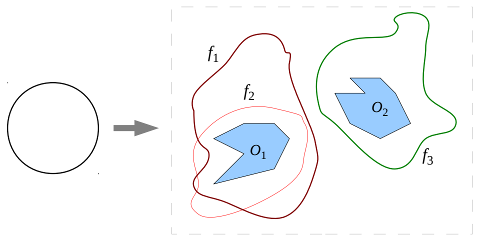
Definition 2.1.1 (Topology [63])**.** A topology on a set X is a collection, T, of subsets of X, containing both X and $\emptyset$, and closed under the operations of intersection and union. Together, the tuple (X, T) is called a *topological space*, and the elements of T are called *open sets* of the topological space.

Often, when there is a standard topology, by convention, for a space X, one can refer to the topological space simply as X. One of the most important consequences of defining topology is that we now have the notion of *continuity*.

Definition 2.1.2 (Continuous Functions Between Topological Spaces [63])**.** Given two topological spaces, (X_1, T_1) and (X_2, T_2), consider a map $f : X_1 \to X_2$. For any subset $U \subseteq X_2$, define $f^{-1}(U) = \{x \in X_1 \mid f(x) \in U\}$ (note that if $f^{-1} : Y \to X$ exists, this definition simply generalizes it to subsets of Y). Then f is said to be continuous if for each open set $V \in T_2$, the set $f^{-1}(V)$ is in T_1 (*i.e.* an open set).

Note that for continuity, f need not map open sets to open sets. That is, for $W \in T_1$, its image $f(W)$ need not be in T_2. When a continuous function is also injective, it is called an *embedding* (embedding of X in Y).


(a) Homeomorphic spaces (both equivalent to $\mathbb{S}^1$). The apparent difference between the two spaces is due to their embedding in $\mathbb{R}^2$. The spaces themselves are topologically equivalent.



(b) Continuous functions, $f_i : \mathbb{S}^1 \to (\mathbb{R}^2 - O_1 \cup O_2)$, which are also embeddings (injective). f_1 can be continuously deformed to f_2 (they are homotopic), but not to f_3 (f_1 and f_3 are not homotopic).

Figure 2.1: *Homeomorphic spaces* and *homotopic functions*.

Starting with these basic definitions, one can make assertions on certain properties of the topological space, construct one topological space from another, and establish relationships between them. This is the primary affair of the field of *point-set topology*. However, one can do little algebra or actual computation on a topological space using this. That's where the field of *algebraic topology* gets introduced.

Before we proceed to algebraic topology, we state a few definitions that follow from basic point-set topology.

Homeomorphism, Homotopy, Deformation Retract and Homotopy Equivalence

The first fundamental equivalence relation among topological spaces is that of *homeomorphism*. Topologically, two topological spaces are homeomorphic if they essentially are the same topological space (with, possibly, 'renaming'/'relabeling' of the items in one of the set and its topology to obtain the other). In presence of an embedding, informally, two spaces are homeomorphic if one can be *continuously deformed* into the other without causing cuts or tears in the space (*i.e.* open sets remain open). This is popularly exemplified using a donut and a coffee cup with a handle, and how, to a topologist, they are one and the same. Figure 2.1(a) shows a more modest example of homeomorphism.

Definition 2.1.3 (Homeomorphism [63], Fig. 2.1(a))**.** Two topological spaces X and Y are *homeomorphic* if there exists a bijective function $f : X \to Y$ (which implies the inverse, $f^{-1} : Y \to X$, exists and is bijective) such that both f and f^{-1} are continuous. f (which may not be unique) is called a homeomorphism between the spaces.

A fundamental equivalence relation among continuous functions defined between two fixed topological spaces is *homotopy*. Informally, two functions are homotopic if one can be continuously changed into another.

Definition 2.1.4 (Homotopy [40], Fig. 2.1(b))**.** Two continuous function between the same topological spaces, $f_1, f_2 : X \to Y$, are called *homotopic* if there exists a continuous function $F : X \times [0, 1] \to Y$ (where, $[0, 1]$ is assumed to have the standard *Euclidean topology*, and '$\times$'

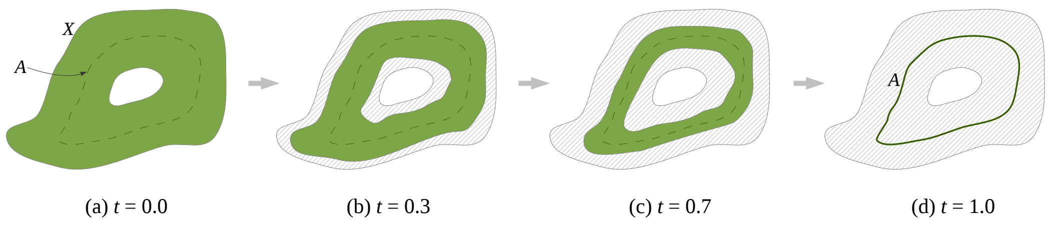


Figure 2.2: A deformation retraction of X to $A \subseteq X$. For each t, the green area is $F(X, t)$.

induces the *product topology* to the product space) such that $F(x, 0) = f_1(x)$ and $F(x, 1) = f_2(x)$, for all $x \in X$. Concisely we express this relationship as $f_1 \simeq f_2$. The function F (which may not be unique) is call a *homotopy* between f_1 and f_2. Informally, we say that f_1 can be *homotoped* to f_2 and vise-versa.

The idea of *deformation retract* is that given a topological space X, and a subspace A (a subset with *subspace topology* [63]), we ask the question whether or not the space X may be continuously 'shrunk' and 'deformed' to A without causing any 'cut' or 'tear'. If it can, we call A a deformation retract of X (Figure 2.2). Consider the identity map $\text{id}_X : X \to X$ (Figure 2.2(a)). Now start 'shrinking' X gradually to 'collapse' on to A. At every step of the shrinking process what we have is an embedding of X into itself such that the image of the embedding is the 'shrunk' version of X at that step (Figure 2.2(b,c)). Eventually we 'shrink' X to A (Figure 2.2(d)).

Definition 2.1.5 (Deformation Retract [40], Fig. 2.2)**.** A subspace A (with *subspace topology*) is called a deformation retract of a topological space X if there exists a continuous function $F : X \times [0, 1] \to X$ such that

- $F(x, 0) = x, \ \forall x \in X$ (*i.e.* $F(\cdot, 0) \equiv \text{id}_X$ is the identity map on X),
- $F(a, t) = a, \ \forall a \in A, t \in [0, 1]$, and,
- $F(x, 1) \in A, \ \forall x \in X$.

F (which may not be unique) is called a *deformation retraction* from X to A. Since A is a subspace of X, we can interpret F as a homotopy between the identity map id_X and the map $f_1 \equiv F(\cdot, 1)$ whose image is in A.

It is important to note that $f_1 \equiv F(\cdot, 1) : X \to X$ is homotopic to the identity map on X. Had A not been given beforehand, and instead, we were given a function $f_1 : X \to X$ that is homotopic to id_X, the image of f_1 would clearly be a deformation retract of X.

The fact that A needs to be a subspace of X in the definition of deformation retract essentially implies that there is an embedding $i : A \hookrightarrow X$, called the *inclusion*. However A, as an independent topological space, should not require an embedding in X to be described (*e.g* A in Figure 2.2 is topologically just a circle $\mathbb{S}^1$). We still should be able to describe a similar relationship between them. That's where a generalization of a deformation retract, called *homotopy equivalence*, comes into the picture.

The idea of homotopy equivalence is that instead of explicitly mentioning a subspace A of X, we look at the continuous functions from X to itself via a second space Y (the final image is of course a subspace of X). We then ask if this combination map is homotopic to the identity map

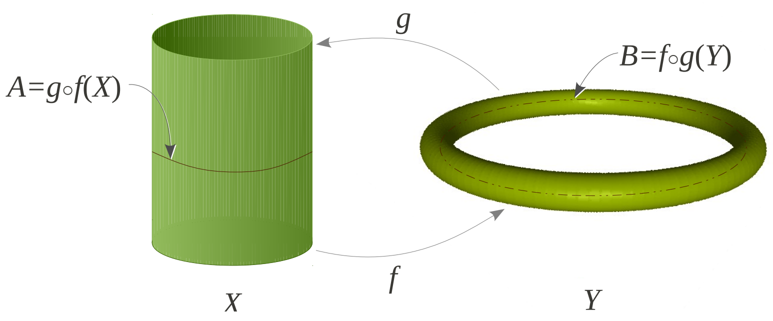


Figure 2.3: A cylinder (hollow, without lids) and a solid torus are homotopy equivalent. Each of them is homotopy equivalent to a circle.

on X. We do the same thing with the role of X and Y reversed. If the answer is 'yes' in both the cases, the spaces X and Y are said to be homotopy equivalents.

Definition 2.1.6 (Homotopy Equivalence [40], Fig. 2.3)**.** Two topological spaces X and Y are called *homotopy equivalent* if there exists continuous functions $f : X \to Y$ and $g : Y \to X$ such that $g \circ f$ is homotopic to the identity map id_X, and $f \circ g$ is homotopic to the identity map id_Y. The function f (and likewise g) is called a *homotopy equivalence*. X and Y are said to have same *homotopy type*, and informally we say one can be *homotoped* to the other.

If A is a deformation retract of X, then they are of course homotopy equivalents. However the converse is not always true. Out of many ways of determining if two spaces X and Y are homotopy equivalents, one approach is to check if each of them deformation retracts to a subspace that is topologically equivalent (homeomorphic). Then they are homotopy equivalent (Figure 2.3). The other, more formal, approach is to check if there exists a larger space with embeddings of X and Y into it, such that this larger space deformation retracts to both X and Y.

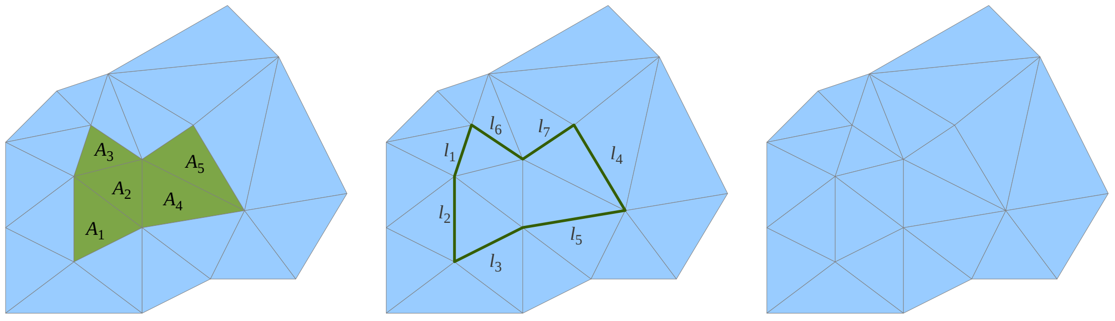
Contractible Space

Definition 2.1.7 (Contractible Space [40])**.** A topological space X is called *contractible* if the identity map on it, id_X, is homotopic to a constant map (a function taking every points in X to a fixed point $x_0 \in X$).

The intuition behind contractibility is that the space can be pulled (contracted) continuously towards a point inside it. It is important to note that a contractible space need not be finite. For example, $\mathbb{R}^D$ is contractible for any finite D. This is because, for any point $p \in \mathbb{R}^D$ one can construct a map $f_p : [0,1] \to \mathbb{R}^D$ so that $f_p(0) = p$, $f_p(0) = 0$ (the origin), and $F(p,t) = f_p(t)$ is continuous.

2.1.2 Motivation of Algebraic Topology

Algebraic topology imparts certain algebraic (primarily *group*) structures to a topological space, and allows interpretation of the structure of the topological space by analysis of the algebraic structures.



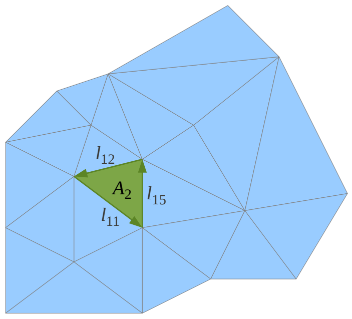
(a) An area (a 2-chain). (b) The boundary of the area (a 1-cycle). (c) The boundary of the boundary is empty.

Figure 2.4: Boundary operator acting on a chain twice gives an empty chain.

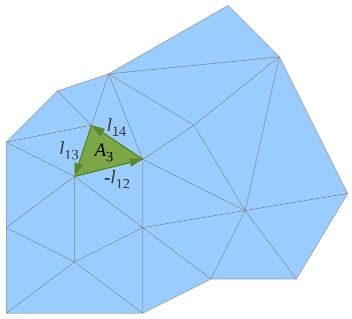
In this section we will motivate some basic ideas behind algebraic topology without going into too much technical details. Instead, we will use some simple illustrations to explain them.

The first step in imparting an algebraic structure to a topological space is to describe the space in terms of a sequence of groups (in simple cases, vector spaces, which are themselves groups with additional structures), and maps between them. This algebraic object will be called a *chain complex*. While it is not necessary to discretize/triangulate a topological space to describe a chain complex on it (as we will shortly do), for ease of understanding we make this simplifying discretization. Each discrete element in this discritization is called a simplex (Figure 2.4) – the vertices will be called 0-simplices, the edges 1-simplices, the triangles 2-simplices, tetrahedrons 3-simplices, etc. Formally, a n-simplex on a topological space, X, is a map from a *standard n-simplex* [40] to X. However, most often, we will informally refer to the image of a n-simplex as the n-simplex itself.

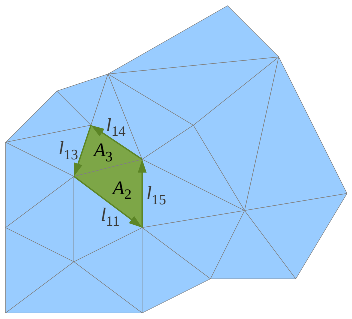
Boundary Operator: Consider a patch of area on the plane that is discretized into simplices as in Figure 2.4. Out of all the 2-simplices (triangles), we pick a few – ones marked by green color as in Figure 2.4(a). We simply call those triangles $A_1, A_2, \cdots, A_5$ (note that by A_i we do not mean the 'area', but the whole triangle as an abstract object/set). Thus the region they cover is denoted by $A_1 + A_2 + \cdots + A_5$ (where, for now, we can interpret '+' as an union). Figure 2.4(b) shows the boundary of the chosen area and is likewise represented by $\sum_{i=1}^{7} l_i$ (each 1-simplex or edge is arbitrarily labeled l_i). However, if we now look at the boundary of $\sum_{i=1}^{7} l_i$, it is clearly empty (in general, such boundary could have been made up of 0-simplices or vertices). This last observation is a key motivation behind constructing a *chain complex.* This observation extends to higher dimensions and any topological space as well. For example, in a 3 dimensional space discretized into tetrahedrons, if we pick a few of those tetrahedrons (3-simplices) to define a volume, and take the boundary of that volume (which will be a closed surface), this boundary will itself have an empty boundary. Thus, boundary of a boundary is always empty. In a naive notation, if $\partial_2(A)$ represents the boundary of an area A, and $\partial_1(l)$ represents the boundary of a curve l, what we just stated can be summarized as $\partial_2(\sum_{j=1}^{5} A_j) = \sum_{i=1}^{7} l_i$, and $\partial_1(\sum_{i=1}^{7} l_i) \ = \ 0 \ \ = \partial_1 \circ \partial_2(\sum_{j=1}^{5} A_j)$. In general, $\partial_n \circ \partial_{n+1} = 0$.



(a) In the boundary of A_2 we have already labeled the edge l_{12}.



(b) The boundary of A_3 needs to have $-l_{12}$ as an edge. We can relabel it to, say, l_{21}, but then we will need to equate it to $-l_{12}$ to ensure distributivity of ∂_2.

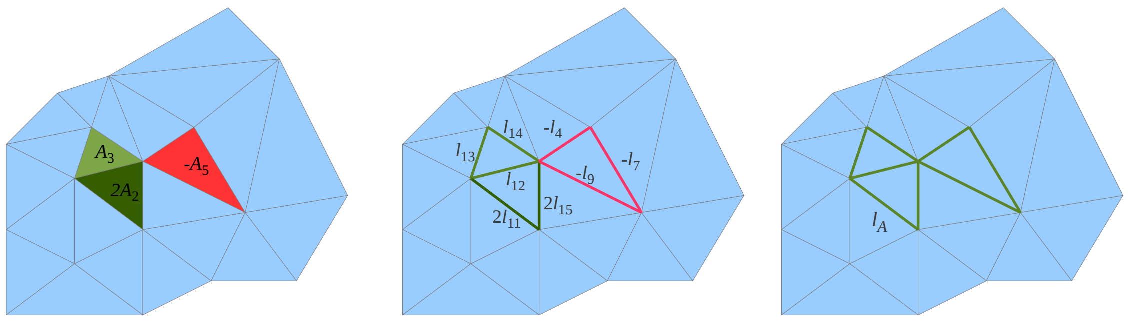


(c) Upon adding the boundaries of A_2 and A_3, the edges l_{12} and $-l_{12}$ cancel out, and we obtain the boundary of $A_2 + A_3$.

Figure 2.5: Distributivity of Boundary Operator.

Distributivity of Boundary Operator and Orientation: One of the properties that we would like the boundary operator, ∂_n, to have is distributivity. That is, for example, we would like to be able to write $\partial_2(A_i + A_j) = \partial_2(A_i) + \partial_2(A_j)$. This will let us assert that boundary of boundary is empty, purely from algebraic conditions, without looking at a picture: $\partial_1 \circ \partial_2(\sum_i A_i) = \sum_i \partial_1 \circ \partial_2(A_i) = \sum_i 0 = 0$ (since boundary of boundary of an individual triangle is always empty). This would enable us develop a linear algebra. This requires that we assign some sign (directionality) to each of l_i. Consider a single 2-simplex, A_2, as shown in Figure 2.5(a). Its boundary is $l_{11} + l_{15} + l_{12}$. Now consider the 2-simplex A_3 (Figure 2.5(b)). Since A_2 and A_3 share the common edge, l_{12}, which will lie inside $A_2 + A_3$, we need to make sure that somehow this edge gets canceled out when we add $\partial_2(A_2)$ to $\partial_2(A_3)$ to obtain $\partial_2(A_2 + A_3)$ (Figure 2.5(c)). This is attained by giving a directionality of every segment l_i, represented as $\pm l_i$, and noting that $l_i + (-l_i) = 0$. There is nothing special about the dimension 1 of the 1-simplices, and we can in fact assign directionality to simplices of every dimension (vertices, edges, triangles, tertahedrons, etc.). The definition of direction has to be such that it admits distributivity of the boundary operators ∂_n. For example, if one considers $-A_2$ in Figure 2.5(a), in order to be consistent with the fact that $\partial_2(A_2 + (-A_2)) = \partial_2(A_2) + \partial_2(-A_2) = 0$, we need to have the boundary of $-A_2$ (*i.e.* $\partial_2(-A_2)$) as $(-l_{11}) + (-l_{15}) + (-l_{12})$, that is the original line segments with reverse orientation.

Group Construction: By now it is easy to see a group structure emerging. For example, for every line segment l_i we have defined an inverse element, $-l_i$ so that they add up to 0, the identity element. Also, we have developed the intuition of how the binary operator '+' works between l_i and l_j for $i \neq j$ or $l_j = -l_i$. All that we now need to do to make the set of possible combinations of the 1-simplices (*e.g.* $l_{i_1} + l_{i_2} + \cdots$ is an arbitrary combination – called a 1-chain) an algebraic group is to *close* it under the operation of addition. Earlier we have related $l_i + l_j$ with taking union of the line segments l_i and l_j. However, if we write $l_i + l_i$, an interpretation in terms of union, will simply mean l_i. This will not be consistent with our attempt to define a group. Thus,



(a) A graphical representation of $A_3 + 2A_2 + (-A_5) \in C_2(X)$. Colors represent the integer coefficients – green positive, red negative, darker is higher. Note how the coefficient for the other 2-simplices is 0 represented by light blue. This type of arbitrary combination is called a *chain* (a 2-chain in this case). Thus, one can interpret this as a order set of coefficients, $[0, 2, 1, 0, -1, 0, 0, 0, \cdots]$, for the corresponding 2-simplices.

(b) The boundary of the 2-chain shown in (a). The boundary is $l_{13} + l_{14} + l_{12} + 2l_{11} + 2l_{15} - l_4 - l_9 - l_7$. Note that we no more use an 'arrow' to represent the direction of the 1-simplices. Since we can now have arbitrary integer coefficients, color representation of the coefficients is the preferred way of visualization. Green indicates positive, red indicates negative. Note how a $-l_{12}$ from $\partial_2(A_3)$ adds to $2l_{12}$ from $\partial(A_2)$ to give the l_{12} in the figure.

(c) However, one can set $l_A = l_{13} + l_{14} + l_{12} + 2l_{11} + 2l_{15} - l_4 - l_9 - l_7$, and considered to be a generating element of $C_1(X)$ (by change of basis). Then we will be using light green to represent l_A with coefficient 1. Then, for example, $l_A + l_{12}$ will be represented by this same figure as above, except with the edge corresponding to l_{12} being marked with darker green.

Figure 2.6: A 2-chain and its boundary, with coefficients (which includes direction information) represented by colors.

we define a new element $2l_i := l_i + l_i$. This can be interpreted as taking the line segment two times on top of itself (similar to *disjoint union*). However it is to be kept in mind that this is absolutely an algebraic construction. Following along similar lines, we can define $2l_i, 3l_i, 4l_i, \cdots$, and the corresponding inverses $-2l_i, -3l_i, -4l_i, \cdots$. In general $nl_i := l_i + l_i + \cdots (n \text{ times})$, and $-ml_j := (-l_j) + (-l_j) + \cdots (m \text{ times})$. Thus we have constructed an abelian group that is *freely generated* by $l_1, l_2, l_3, \cdots$. We represent this group by $C_1(X)$ (where, X is the topological space which we discretized to create the simplices), where the subscript 1 refers to the dimension of the simplices. Of course we can do similar treatment for simplices of all dimensions (*e.g.* Figure 2.6). The group for the n-dimensional simplices is written as $C_n(X)$.

Coefficients: One further generalization of the said group construction is that with arbitrary coefficients. The idea can be described as follows: So far we have constructed elements like $2l_i, 3l_i, \cdots$ (*i.e.* l_i with integer coefficients). However, very often, one may come across problems where non-integer coefficients arise very naturally. One typical inspiration comes from an electrical network that can be modeled as a simplicial 1-complex (the coefficients on the 1-simplices represent the currents passing through them, and the coefficients on the 0-simplices represent the voltage at the nodes). Then, because of Kirchhoff's law, the sum of the incoming currents at any node is equal to the sum of the outgoing currents at the node. Consequently, any closed loop of current represents a 1-cycle in the complex [34]. However, note that now we need to define coefficients over $\mathbb{R}$ since the currents can assume any real number (*e.g.* $2.56\ l_i$). Moreover, due to the *superposition theorem* *for electrical circuits* [11], one can add currents, and hence add the 1-cycles. For such additions we naturally borrow the definition of additions from the real numbers, $\mathbb{R}$ (which is a group under '+' operator with 0 as identity element), and the additions happen simply on the coefficients of l_i. In fact we can generalize the coefficients to arbitrary algebraic structures (like *groups*, *rings*, *fields*, etc.). Thus, if G is an algebraic structure and an abelian group under an addition operation, '+', then we define $C_1(X;G)$ (as a generalization of $C_1(X)$) to be abelian group in which every element is represented by an ordered set of coefficients $[g_1, g_2, g_3, \cdots],\ g_i \in G$ (these are the coefficient of $l_1, l_2, l_3, \cdots$), that inherits the operator '+' from G by the element-wise operation $[g_1, g_2, g_3, \cdots] + [g'_1, g'_2, g'_3, \cdots] = [g_1 + g'_1, g_2 + g'_2, g_3 + g'_3, \cdots]$. It is to be noted that the addition operator is a chosen preferred operator of G. The inheritance of other operators from G to $C_1(X;G)$ are subject to independent definitions. Of course, once again, this is general for arbitrary dimensions, and for n-dimensional simplices we have the algebraic structures $C_n(X;G)$.

The boundary operator, which was distributive, is extended to being *linear* (informally. More precisely it is a group homomorphism) when we have the coefficients (this is mostly by definition than anything else – by extending the definition of boundary operator to chains with coefficients). Thus, from our previous example, if $A = g_a A_2 + g_b A_3 \in C_2(X;G)$ for $g_a, g_b \in G$, we have $\partial_2(A) = g_a\partial_2(A_2) + g_b\partial_2(A_3) = g_a(l_{11} + l_{15} + l_{12}) + g_b(l_{13} - l_{12} + l_{14}) = g_a(l_{11} + l_{15}) + g_b(l_{13} + l_{14}) + (g_a - g_b)l_{12}$. Substituting $g_a = g_b = 1$ for $G = \mathbb{Z}$, we obtain our previous result. Also, by linearity, $\partial_1 \circ \partial_2(A) = g_a\partial_1 \circ \partial_2(A_2) + g_b\partial_1 \circ \partial_2(A_3) = g_a 0 + g_b 0 = 0$. In general $\partial_n \circ \partial_{n+1} = 0$ due to linearity and the fact that for a $(n+1)$-simplex σ^{n+1}, $\ \partial_n \circ \partial_{n+1}\sigma^{n+1} = 0$. More precisely, the boundary operator is a *group homomorphism.*

Vector Space: One can very well compare $C_n(X)$ to a vector space (especially when the coefficients are in field $\mathbb{R}$). For example, in Figure 2.6, each l_i may be thought to be an basis vector forming a basis set in a N-dimensional vector space (where N is the total number of 'edges' or 1-simplices in the discretization of X). Thus, any linear combination of the basis vectors, $\sigma = a_1 l_1 + a_2 l_2 + \cdots + a_N l_N$, will represent some 1-chain (Figure 2.6(b)). Then, $C_n(X)$ is very much like a vector space spanned by those basis vectors. One can even talk about change of basis (Figure 2.6(c)). In particular, if the coefficients are in a field, then $C_n(X)$ is indeed a vector space. If elements from this vector space can be represented by coefficient vectors $[a_1, a_2, a_3, \cdots, a_N]^T$ (as described earlier, and assuming there are N numbers of n-simplices in the representation of X), and the elements of the vector space $C_{n-1}(X)$ is represented by coefficient vectors $[b_1, b_2, b_3, \cdots, b_M]^T$ (where we assume that there are M numbers of $(n-1)$-simplices in the finitely discretized representation of X), then the boundary operator ∂_n may be represented by a $M \times N$ matrix. Thus, when a N-dimensional coefficient vector representing a vector from from $C_n(X)$ is left-multiplied by this matrix, we obtain a M-dimensional coefficient vector representing a vector in $C_{n-1}(X)$. Then the kernel and image of ∂_n have simple interpretations borrowing the corresponding concepts from linear algebra.

2.1.3 Formal Description of Homology

Once we have established the motivation behind defining chain complexes, we can formally define it in the most general way as follows.

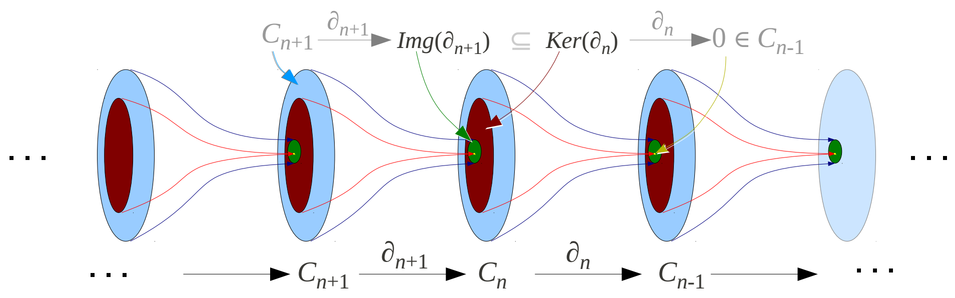


Figure 2.7: A schematic representation of a chain complex (see [34]). It consists of the sequence of groups, $C_\bullet$, along with homomorphisms $\partial_\bullet$, with the property that $\partial_n \circ \partial_{n+1}(\sigma) = 0$ for any $\sigma \in C_{n+1}$.

Definition 2.1.8 (Chain Complex [40], Fig. 2.7)**.** A *chain complex* is a sequence of abelian groups, $\cdots, C_3, C_2, C_1, C_0, C_{-1}, \cdots$, along with homomorphisms $\partial_n : C_n \to C_{n-1}$ such that $\partial_{n-1} \circ \partial_n = 0$ for all $n = \cdots, 3, 2, 1, 0, -1, \cdots$. It is commonly represented using the following diagram:

$$\begin{tikzcd} \cdots \arrow[r] & C_{n+3} \arrow[r, "\partial_{n+3}"] & C_{n+2} \arrow[r, "\partial_{n+2}"] & C_{n+1} \arrow[r, "\partial_{n+1}"] & C_n \arrow[r, "\partial_n"] & C_{n-1} \arrow[r, "\partial_{n-1}"] & \cdots \end{tikzcd}$$

with, $\partial_{n-1} \circ \partial_n = 0,\ \forall n$.

Note that in general, chain complexes need not be related to any topological space, as in the independent definition stated above. It is simply a sequence of abelian groups $C_\bullet$ with the operators $\partial_\bullet$ (by subscript '$\bullet$' we informally mean the collection for all n). Such independent studies of chain complex without any reference to topology is known as *homological algebra* and is a field of study by its own right. Algebraic topology borrows significant amount of tools from that field.

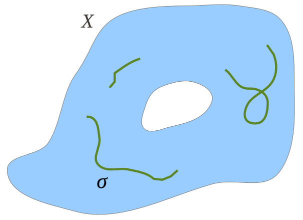
In algebraic topology, these groups C_n obviously corresponds to the groups freely generated by the n-simplices on the topological space X, and are written as $C_n(X)$. Chain complexes generated by a finite number of simplices as described earlier are known as *simplicial complex*. However there are other, and more general forms of chain complex that one can define on a topological space – Δ-complex, singular complex, cellular complex, cubical complex, etc.

Since $\partial_n \circ \partial_{n+1} = 0$, we have (see Figure 2.7)

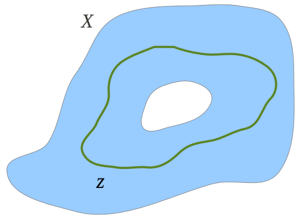
$$Img(\partial_{n+1}) \subseteq Ker(\partial_n) \subseteq C_n(X)$$

Definition 2.1.9 (Subgroup of Boundaries [40])**.** $B_n(X) := Img(\partial_{n+1}) \subseteq C_n(X)$ is called the group of n-boundaries (a subgroup of C_n), and is the image of the whole of $C_{n+1}(X)$ under the action of ∂_{n+1}. Elements in $B_n(X)$ (called n-boundaries) are n-chains, each of which are boundaries of some $(n+1)$-chains in X (Figure 2.8(c)).

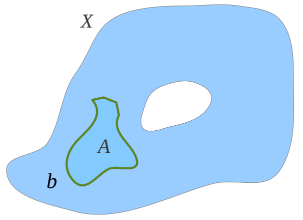
Definition 2.1.10 (Subgroup of Cycles [40])**.** $Z_n(X) := Ker(\partial_n) \subseteq C_n(X)$ is called the group of n-cycles (a subgroup of C_n), and is the *kernel* of ∂_n (*i.e.* all the elements in C_n that maps to the identity element in $C_{n-1}(X)$ under the action of ∂_n). Elements in $Z_n(X)$ (called n-cycles) are n-chains that have empty boundary under the action of ∂_n (Figure 2.8(b)). Of course all n-boundaries are n-cycles as well, but the converse is not true.



(a) A 1-chain. The boundary of this chain is not empty since the curves have end-points (0-chains). This is an arbitrary element of $C_1(X)$.



(b) A 1-chain without a boundary is a 1-cycle. This is an element of $Z_1(X)$.



(c) A chain which is boundary of a 2-chain (the one marked by A). This is an element of $B_1(X)$.

Figure 2.8: Illustration of *chain*, *cycle* and *boundary*. The topological space X is shown in blue. The figures do not show a discretization explicitly, however one can think it to be discretized into very small simplices (much like what we had in Figure 2.5 or 2.6) in order to accommodate almost arbitrary-shaped chains. Every 1-chain shown in this figure are made of 1-simplices, being labeled for the first time, with *coefficients as* 1 (indicated by the color light green – refer to Figure 2.6(b)).

It is easy to see that both $Z_n(X)$ and $B_n(X)$ are closed under addition and also the inverse of elements in each of them belong to the sets themselves. Thus they are subgroups.

As discussed earlier, an element $\sigma \in C_n(X)$ is an arbitrary linear combination of n-simplices on X (Figure 2.8(a)). However, a subset of those will be such that the boundary operator, ∂_n, acts on them to give zero (*i.e.* they are in $Ker(\partial_n)$). These are elements $z \in Z_n(X) \subseteq C_n(X)$ (Figure 2.8(b)). Furthermore, some out of those are such that they themselves are boundaries of some one higher dimensional chain (*i.e.* they are in $Img(\partial_{n+1})$). Those are elements $b \in B_n(X) \subseteq Z_n(X)$ (Figure 2.8(c)).

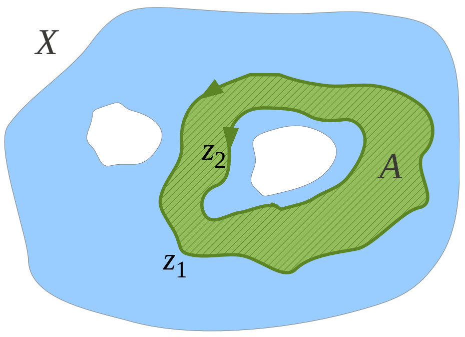
Since, by definition, for any $b \in B_n(X)$ we can find a $\omega \in C_{n+1}(X)$ such that $\partial_n(b) = \partial_n \circ \partial_{n+1}(\omega) = 0$ (by definition of chain complex), we have $B_n(X)$ a subgroup of $Z_n(X)$. Thus, one can now construct the following *quotient group*,

Definition 2.1.11 (Homology Groups [40])**.** We define the n^{th} homology group of X as ,

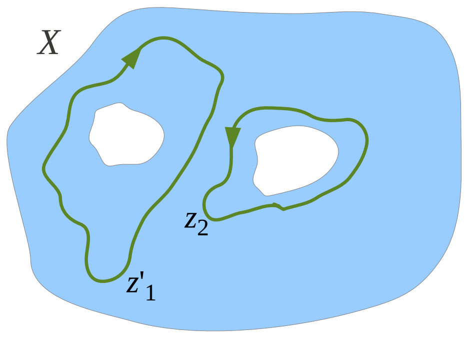
$$H_n(X) \;=\; Z_n(X) \;/\; B_n(X)$$

The intuitive description of $H_n(X)$ is as follows: We look at two n-cycles $z_1, z_2 \in Z_n(X)$ (Figure 2.9). If their difference is boundary of some one-dimensional higher chain (*i.e.* $z_1 - z_2 \in B_n(X)$), we say that they belong to the same *homology class* or are *homologous* (Figure 2.9(a) and 2.9(d)), otherwise we say that they are in different *homology classes* (Figure 2.9(b) and 2.9(c)). Thus, we have several different homology classes of n-cycles. $H_n(X)$ essentially is the set of all those homology classes. This can be seen clearly by the definition of *group quotient*. Each element of $H_n(X)$ is a partition of $Z_n(X)$ such within each partitions (*i.e.* a homology class) two elements z_1 and $\overline{z}_1$ can be related as $\overline{z}_1 = z_1 + b$ for some $b \in B_n(X)$.

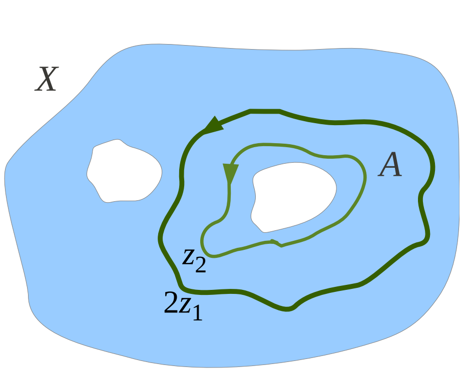
Group Structure of $H_n(X)$: For a given $z \in Z_n(X)$, we write $[z] \in H_n(X)$ for the homology class of Z. The addition operator of $H_n(X)$ is inherited from $Z_n(X)$ in a rather natural way – we define



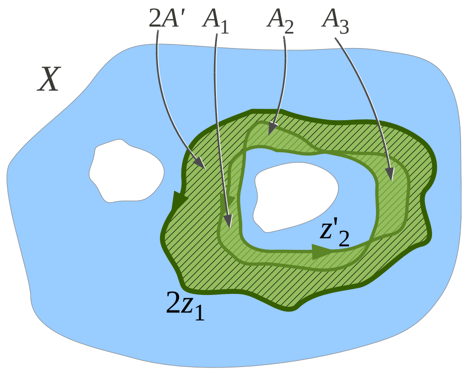
(a) z_1 and z_2 are 1-cycles that are in the **same** *homology class*. This is because z_1 and $-z_2$ (z_2 with reversed orientation) forms the boundary of a 2-chain, A, (one consisting of all the 2-simplices in the annular region with unit coefficients). That is, $z_1 - z_2 = \partial_2(A) \in B_1(X)$



(b) z_1' and z_2 are 1-cycles that are in **different** homology classes. This is because one cannot find a 2-cycle A such that $z_1 - z_2 = \partial_2(A) \in B_n(X)$



(c) $2z_1$ (where z_1 is the same as in (a)) and z_2 are 1-cycles that are in **different** homology classes. The coefficient 2 for z_1 (which may be thought of as 2 copies of z_1 placed on top of one another) is indicated by the darker color (refer to Figure 2.6(b)).

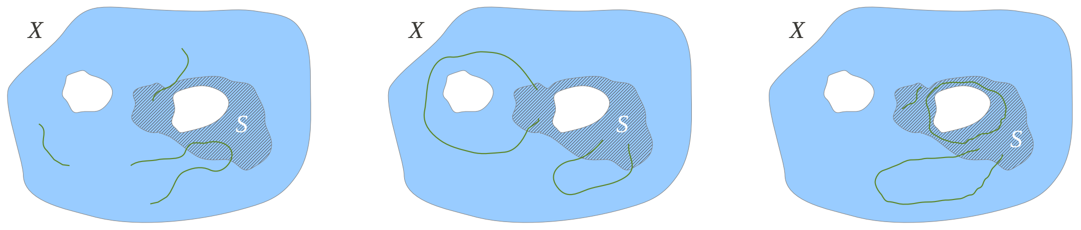


(d) $2z_1$ and z_2' are 1-cycles that are in **same** homology classes. This is because we can write $2z_1 - z_2 = \partial_2(2A' - A_1 - A_2 - A_3) \in B_1(X)$.

Figure 2.9: Cycles in same and different homology classes. Discretization and color coding of coefficients similar to before (Figure 2.8).

$[z_1] + [z_2] = [z_1 + z_2]$, where the first addition is the one in $H_n(X)$ (one we are defining), while the addition on the right is well-known for $Z_n(X)$. Also, the identity element (or 'zero') of $H_n(X)$ is the homology class of the boundaries (elements of $B_N(X)$). This can be observed as follows: If z_1 and $\overline{z}_1$ belong to the same homology class (*i.e.* $\overline{z}_1 = z_1 + b$ for some $b \in B_n(X)$), then by definition, $[\overline{z}_1] = [z_1] \implies [z_1 + b] = [z_1] \implies [z_1] + [b] = [z_1] \implies [b] = 0$.

Resorting briefly to our earlier comparison of $C_n(X)$ with a vector space, we can see that $Z_n(X)$ is like a vector subspace (a vector space by its own right). Thus, $B_n(X)$ is a subspace of $Z_n(X)$.



(a) Relative chains. The 1-chains in X have some boundary (end point, 0-chain) lying outside S.

(b) Relative cycles. The 1-chains in X have boundaries lying inside S. Thus in the relative form (*i.e.* their image under quotient map j), they have no boundary. Hence these are relative cycles.

(c) Relative boundaries. For these 1-chains in X, either the whole of it lies inside S, thus making it trivial, or they are boundaries of some relative 2-chain.

Figure 2.10: Relative chains on $C_1(X, S)$, and chains that are relative cycles and relative boundaries.

Then $H_n(X)$ may be thought of as the vector subspace of $Z_n(X)$ which is orthogonal to $B_n(X)$ such that $H_n(X)$ and $B_n(X)$ spans the whole of $Z_n(X)$.

Relative Homology: Given $C_\bullet(X)$, a chain complex on X, and a subspace S of X (S and X are together written as (X, S) and is called a *pair of spaces*), one can construct the *subcomplex* $C_\bullet(S)$, where each $C_n(S)$ is a subgroup of $C_n(X)$ freely generated by the n-simplices that fall inside S. Then one can talk about quotient groups $C_n(X)/C_n(S)$. These quotient groups are written as $C_n(X, S)$ for brevity. Thus, there is a quotient map $j : C_n(X) \to C_n(X, S)$ such that given any n-chain $\sigma \in C_n(X)$, if we *trivialize* the part of the chain that lies inside S, we obtain $j(\sigma) \in C_n(X, S)$. It is analogous to taking projection of σ on the subspace orthogonal to the subspace $C_n(S)$. One can then extend the boundary operator ∂_n quite naturally to define $\partial_n : C_n(X, S) \to C_{n-1}(X, S)$. Then it is not difficult to see that $C_\bullet(X, S)$ form a chain complex,

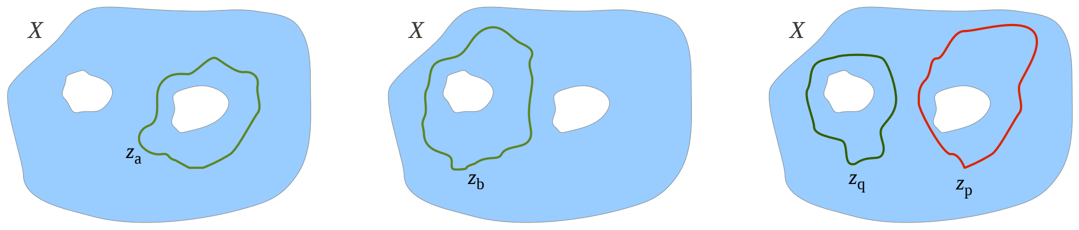
$$\cdots \longrightarrow C_{n+1}(X,S) \xrightarrow{\partial_{n+1}} C_n(X,S) \xrightarrow{\partial_n} C_{n-1}(X,S) \xrightarrow{\partial_{n-1}} \cdots$$

with $\partial_{n-1} \circ \partial_n = 0, \ \forall n$. The image of a n-chain $\sigma \in C_n(X)$ under the action of j is typically called a *relative chain* and is an element of $C_n(X, S)$. We can compute homology groups for the above chain complex. Those are called *relative homology* groups, and are represented by $H_n(X, S)$. It is to be noted that these, in general, are not same as $H_n(X)/H_n(S)$. Any $\sigma \in C_n(X)$ lying inside completely S will be a trivial relative chain $j(\sigma) = 0 \in C_n(X, S)$ and is a *relative boundary* (Figure 2.10(c)). And any n-chain $\sigma \in C_n(X)$ with boundary lying completely inside S will be a *relative n-cycle* $j(\sigma) \in Z_n(X, S)$ (Figure 2.10(b)).

2.1.4 Properties of Homology

In this section we will mostly state some results and explain their implications, but without detailed proofs. The reader may refer to [40] for the proofs and detailed discussion.

Interpretation of Homology Groups: Each element of the n^{th} homology group, $H_n(X)$, as we



(a) $[z_a]$ is a generator of $H_1(X)$.

(b) $[z_b]$ is another generator of $H_1(X)$.

(c) $z = az_p + bz_q \in Z_1(X)$ is an arbitrary 1-cycle. Its homology class however is $[z] = a[z_p] + b[z_q] = a[z_a] + b[z_b]$, *i.e.* can be expressed as a linear combination of the generating homology classes.

Figure 2.11: The rank of homology group gives the Betti number. The homology class of an arbitrary cycle can be expressed as the linear combination of the generators of the homology group.

just saw, represents a class of n-cycles that differ by n-boundaries. There is however an even more intuitive and useful interpretation of the homology groups – the rank (or informally, the dimensionality) of the group tells us about the n^{th} Betti number (informally, the number of $(n+1)$-dimensional 'holes' when $n > 0$, and number of connected components when $n = 0$) of the topological space X. This is not difficult to see from the example in Figure 2.9. One can see that corresponding to the two holes in the space X, there are two distinct types of cycles that are not boundary (called non-trivial cycles) – one that goes around the right hole (Figure 2.11(a)), and other that goes around the left hole (Figure 2.11(b)). These are the generators of $H_1(X)$. In fact a direct computation of $H_1(X)$ reveals that it is isomorphic to the group $\mathbb{Z}^2$ (*direct sum* of two copies of the integers' group, which is a group under addition) – thus, the first Betti number of the space, b_1, is 2. The homology class of any other cycle in the space can be expressed as a linear combination of these two homology classes (Figure 2.11(c)).

Indifference to Method of Construction of Chain Complex: The homology groups of a topological space are indifferent to the method of construction of the chain complex used to compute the homology groups. The intuitive idea is that given a topological space X, one can create a chain complex in many different ways on it. Even a finite simplicial discretization (*e.g.* Figure 2.6) can be created in infinite variety. Besides, there are other types of chain complexes that can be constructed on a topological space (like Δ-complex, singular complex, cellular complex, cubical complex – see [40] for details). Although these create vastly different chain complexes, $\{C_\bullet(X), \partial_\bullet\}$, the homology groups $H_n(X)$ computed using any of them will however be the same as long as we stick to the same coefficient group. This may be intuitive from the discussion in the previous paragraph, where we saw that the homology groups provide information about the Betti numbers of X – a topological invariant of X. However a rigorous proof of this fact is quite elaborate and involved [40].

Functoriality: Homology is a functor from the *category* of topological spaces to the category of groups [40]. The simple meaning of this statement is that if there are two topological spaces X and

Y, and if there is a continuous function $f : X \to Y$, then there exists a group homomorphism $f_{*:n} : H_n(X) \to H_n(Y)$, $\forall n$ such that for a cycle $z \in Z_n(X)$ the following holds: $f_{*:n}([z]_X) = [f(z)]_Y$. Here, by $f(z)$ we mean the image of z in Y under the action of f (which will still be a cycle), and by the subscripts X and Y of the square brackets we mean the homology class of the cycle in the respective topological space (*i.e.* elements of $H_n(X)$ or $H_n(Y)$ respectively). We say that the map f *induces* the homomorphisms f_* between the homology groups (where, by the subscript '$*$' we mean the collection of all the induced homomorphisms $\cdots, f_{*:2}, f_{*:1}, f_{*:0}$). Also, due to functionality, if there are maps $f : X \to Y$ and $g : Y \to Z$, then $(g \circ f)_* = g_* \circ f_*$.

Homotopy Invariance: If two spaces X and Y are *homotopy equivalent* (Definition 2.1.6), then their homology groups are isomorphic (*i.e.* they essentially are the same groups). In notation, $H_n(X) \approxeq H_n(Y)$, $\forall n$. This is an important result in algebraic topology.

However, if we know that the n^{th} homology groups of two spaces are isomorphic (*i.e.* $H_n(X) \approxeq H_n(Y)$), it is often nontrivial to find a map $f : X \to Y$ that *induces* the isomorphism. For example we can have $H_n(X) \approxeq H_n(Y) \approxeq \mathbb{Z}^p$, but if $f : X \to Y$ is such that every point on X is mapped to a single point $y_0 \in Y$ (*i.e.* a constant map), then the map $f_{*:n}$ is a zero map (which is still a homomorphism, but not an isomorphism). On the other hand, if f is a *homotopy equivalence* between X and Y, then f_* are isomorphisms.

Long Exact Sequence (LES): A long exact sequence is a special type of chain complex consisting of sequence of abelian groups, $A_\bullet$, and chain maps between the groups, $p_\bullet$, with the property that $Img(p_{n+1}) = Ker(p_n)$, $\forall n$ (instead of just being subset as it was the case for chain complex). Thus long exact sequences are obviously chain complexes as well. The sequence can be finite or infinite as in the case of chain complex.

An important result in algebraic topology is that given a pair of spaces, (X, S), the following sequence is a LES:

$$\cdots \longrightarrow H_n(S) \xrightarrow{i_*} H_n(X) \xrightarrow{j_*} H_n(X,S) \xrightarrow{\partial_*} H_{n-1}(S) \xrightarrow{i_*} H_{n-1}(X) \xrightarrow{j_*} \cdots$$

where, by the subscripts '$*$' we mean the corresponding induced homomorphism to be used for appropriate value of n. i_* is the homomorphism induced (due to *functoriality* of homology) by the inclusion map $i : S \hookrightarrow X$. j_* is induced by the quotient map at the chain level (as opposed to the topological space) $j : C_n(X) \to C_n(X)/C_n(S)$. And ∂_* is a homomorphism that maps the homology class of relative cycles in (X, S) to the homology class of its boundary in S. A more detailed discussion on properties of LES can be found in pp. 114 of [40].

2.1.5 Cohomology

Before we conclude, we will briefly describe the concept of cohomology [40] and in particular, the *de Rham cohomology* [8]. The concept of cohomology is very closely related to homology, and most of the concepts in cohomology (*e.g. cochain, cocycle* and *coboundary*) are derived from the related concepts in homology.

Given a chain complex $\{C_\bullet, \partial_\bullet\}$ (with coefficients in group G), consider a homomorphism (informally, a linear function) $\alpha : C_n \to G'$ for some group G'. That is, given any chain $\sigma \in C_n$ we can

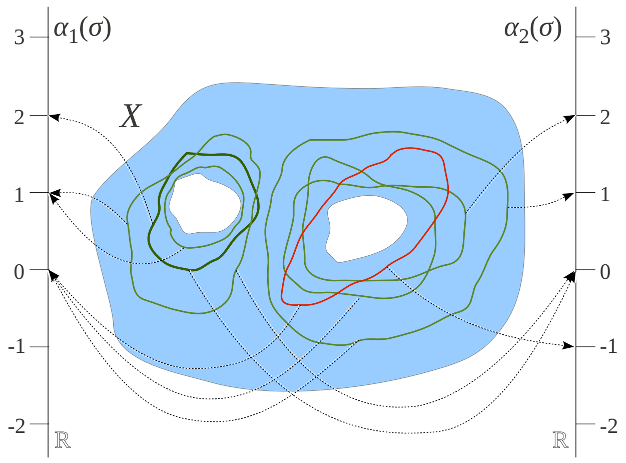


Figure 2.12: Two cocycles, $\alpha_1, \alpha_2 : C_1(X) \to \mathbb{R}$, acting on various cycles, σ. The cohomology class of these two cocycles are generators of $H^1(X)$. In this example, $H_1(X) \approxeq \mathbb{R}^2$ as well as $H^1(X) \approxeq \mathbb{R}^2$ (with all coefficients in $\mathbb{R}$).

compute $\alpha(\sigma)$, which will give us a value in G'. Now consider the set of all such linear functions $C_n \to G'$ that are possible, and call this set C^n. Thus $\alpha \in C^n$. One can define an addition, $\uplus$, on elements of C^n: For $\alpha_1, \alpha_2 \in C^n$, we define $(\alpha_1 \uplus \alpha_2)(\sigma) = \alpha_1(\sigma) \;+'\; \alpha_2(\sigma)$, where '$+'$' is the addition operator in G'. It is easy to see that C^n, along with the addition $\uplus$, forms a group. Concisely one writes $C^n = \text{Hom}(C_n, G')$ and call it the *dual group* of C_n. Elements of C^n are called n-cochains, which essentially are functions from C_n to G'. Thus, for every cochain $\alpha \in C^n$ and every chain $\sigma \in C_n$, the *evaluation* of α on σ, *i.e.*, $\alpha(\sigma)$, gives a value in G'.

Also, corresponding to every ∂_n in the chain complex, one can define a *dual map* $\delta^{n-1} : C^{n-1} \to C^n$ as follows: Consider a $\beta \in C^{n-1}$ (which is a function from C_{n-1} to G'). Then $\delta^{n-1}(\beta)$ is an element of C^n such that for every $\sigma \in C_n$ the following holds: $(\delta^{n-1}(\beta))(\sigma) = \beta(\partial_n(\sigma))$. It is easily seen that $\{C^\bullet, \delta^\bullet\}$ forms a sequence

$$\cdots \longleftarrow C^{n+3} \xleftarrow{\delta^{n+2}} C^{n+2} \xleftarrow{\delta^{n+1}} C^{n+1} \xleftarrow{\delta^{n}} C^{n} \xleftarrow{\delta^{n-1}} C^{n-1} \xleftarrow{\delta^{n-1}} \cdots$$

with $\delta^{n+1} \circ \delta^n = 0, \ \forall n$. This, in fact, is a chain complex with a increasing sequence of indices (which is no different from a standard chain complex since it can always be converted to a decreasing sequence by a variable substitution). This chain complex is however called a *cochain complex*, the *dual* of a chain complex. Just as in a chain complex, we can compute the homologies. Such groups are called the *cohomology groups*: $H^n(X) = Ker(\delta^n)/Img(\delta^{n-1})$.

n-cocycles are elements of the group $Z^n := Ker(\delta^n)$. They have the property that they evaluate to zero on every n-boundary. *n-coboundaries* are elements of the group $B^n := Img(\delta^{n-1})$. They have the property that they evaluate to zero on every n-cycle and n-boundary. Cohomology classes of cocycles have similar interpretation as homology classes: Each cohomology class contains the

cocycles that differ by a coboundary. Thus, two cocycles from the same cohomology class will evaluate to give the same value on two cycles from the same homology class. Figure 2.12 illustrates how two cocycles evaluate on different cycles in the space X.

De Rham Cohomology Groups

A related concept is that of the *De Rham Cohomology.* Here, instead of looking at functions from $C_n(X)$ to a group G' (functions that can be evaluated on n-cycles), we look at *differential n-forms* [8] that can be integrated on n-cycles. In fact, due to the linearity of the cochains (*i.e.* the functions $C_n(X) \to G'$), one would expect that the cochains can be written as some form of an integral – for example, $\int_{\sigma_1+\sigma_2} \omega = \int_{\sigma_1} \omega + \int_{\sigma_2} \omega$ lets us define a cochain $\alpha(\sigma) = \int_{\sigma} \omega$. It can in fact be shown that there is a 1-to-1 correspondence between the cohomology classes of n-cocycles and the de Rham cohomology classes of *closed differential n-forms*. This is formalized by the *de Rham theorem* that gives an isomorphism between the cohomology groups with coefficients in $\mathbb{R}$ (*i.e.* $H^*(X;\mathbb{R})$) and the *de Rham cohomology groups*, $H^*_{dR}(X)$, of a space X.

2.2 Elementary Riemannian Geometry

Topology, due to its invariance to homeomorphism, does not tell us anything about the distance between points on a topological space. One needs to define a metric or distance function on a topological space for that.

Definition 2.2.1 (Distance Function [47]). A *distance function* on a topological space X is a function $d : X \times X \to \mathbb{R}$ such that, for any $p, q \in X$, the following conditions are satisfied,

i. $d(p,q) \geq 0$,

ii. $d(p,q) = 0$ if and only if $p = q$,

iii. $d(p,q) = d(q,p)$, and,

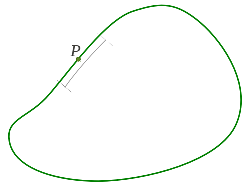
iv. $d(p,q) \leq d(p,r) + d(r,q)$ for all $p, q, r, \in X$ (this is called the *triangle inequality*).

A distance function is also called a *metric*. However, we will reserve that term for informally referring to Riemannian metric, and will avoid using the term to indicate a distance function in order to avoid confusion.

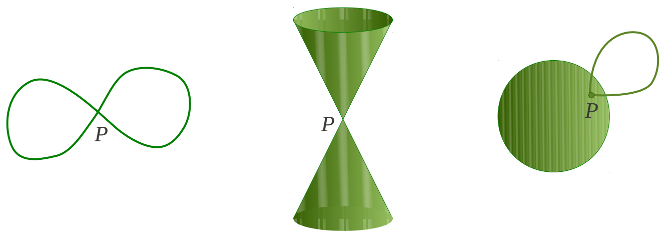
Simple example of distance functions include the p-norm in the familiar vector space of $\mathbb{R}^D$ (which is a topological space with the standard Euclidean topology, with additional structure of a vector space). Thus, if $x = [x^1, x^2, \cdots, x^D]$ and $y = [y^1, y^2, \cdots, y^D]$ are points in $\mathbb{R}^D$, then $d(x,y) = \left(|x^1 - y^1|^p + |x^2 - y^2|^p + \cdots + |x^D - y^D|^p\right)^{1/p}, \quad p > 0$, defines a distance function on $\mathbb{R}^D$.

2.2.1 Manifolds, Coordinate Charts, Atlases and Tangent Space

In Riemannian Geometry we will mostly be concerned with a specific class of topological spaces, namely *manifolds*.



(a) A circle is a 1-manifold. A open neighborhood of every point on it resembles an open segment of $\mathbb{R}$.



(b) Topological spaces that are not manifolds. The spaces look locally Euclidean everywhere, except for the points marked as 'P'.

Figure 2.13: Topological spaces that are manifolds (a) and that are not manifolds (b).

Definition 2.2.2 (Manifold [47])**.** A manifold is a topological space that locally looks like an Euclidean space everywhere. That is, if M is a topological space, and $p \in M$ is a point in it, then there exists a open neighborhood U of p (*i.e.* an open set U with $p \in U$), such that one can construct homeomorphisms $\psi : U \to \mathbb{R}^D$ for some nonnegative integer D. The minimum value of D for which it is possible to construct such homeomorphisms is called dimension of the manifold.

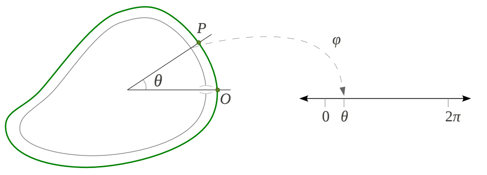
Of course all manifolds are topological spaces, but the converse is not true. A circle is a 1-dimensional manifold since the neighborhood of every point on it resembles a line segment, a subset of $\mathbb{R}^1$ (Figure 2.13(a)). Similarly a sphere or a torus are 2-dimensional manifolds. A solid ball is a 3-dimensional manifold with boundary (*i.e.* points on the boundary locally looks like half-space instead of $\mathbb{R}^3$). Figure 2.13(b) shows some simple topological spaces that are not manifolds. This is because at least at some point in the spaces there does not exist an open neighborhood that resembles an Euclidean space.

Next we proceed towards defining a coordinate chart on a manifold. There are many natural algebraic tools associated with the standard Euclidean space, $\mathbb{R}^D$ (*e.g.* its vector-space structure, natural definition of differentiation along the axes, etc.). The main purpose of defining a coordinate chart is to borrow those concepts to arbitrary manifolds.

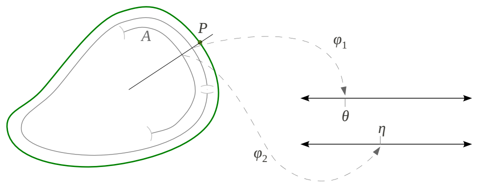
Definition 2.2.3 (Coordinate Chart)**.** Given an open subset U of a D-dimensional manifold M, and a continuous injective function $\phi : U \to \mathbb{R}^D$, we say $C = (U, \phi)$ is a coordinate chart on U. ϕ in fact needs to be a homeomorphism as well (*i.e.* have a continuous inverse over its image).

Thus, the polar coordinate $\theta \in (0, 2\pi)$ used to describe points on a circle (Figure 2.14(a)) constitutes a coordinate chart. The map $\phi : (\mathbb{S}^1 - O) \to (0, 2\pi)$ maps every points on the circle, except one (exclusion of which makes the pre-image of ϕ an open subset of $\mathbb{S}^1$, and the image an open subset of $\mathbb{R}^1$), to the open interval $(0, 2\pi)$ on $\mathbb{R}$. Similarly, the familiar polar coordinate on a 2-sphere constitutes a coordinate chart. The open subset under consideration here is the entire surface of the sphere except the polar points and one longitudinal line, and the function ϕ maps every point on it to a point in $\mathbb{R}^2$, namely, (θ, γ) – the latitudinal and longitudinal angles.

The variables (θ in case of the circle, (θ, γ) in case of a sphere), natural to $\mathbb{R}^D$, can now be used



(a) The map $\phi : (\mathbb{S}^1 - O) \to (0, 2\pi)$ constitutes a coordinate chart. Note that $(\mathbb{S}^1 - O)$ is an open subset of $\mathbb{S}^1$.



(b) Two maps, $\phi_1 : (\mathbb{S}^1 - O) \to \mathbb{R}$ and $\phi_2 : A \to \mathbb{R}$, with A being an open subset of $\mathbb{S}^1$, constitutes an atlas on $\mathbb{S}^1$. This is because $(\mathbb{S}^1 - O) \cup A = \mathbb{S}^1$.

Figure 2.14: A chart and an atlas on the topological circle.

to describe points on the pre-image of ϕ, *i.e.* the open subset U (since ϕ has a continuous inverse as well). These are called the coordinate variables of the chart C. Very often, if $\mathbf{x} = (x^1, x^2, \cdots, x^D)$ are the coordinate variables corresponding to a given chart, one simply writes $\mathbf{x}$ to refer to points on U instead of writing $\phi(\mathbf{x})$.

Definition 2.2.4 (Atlas [47], Fig. 2.14(b))**.** An atlas is a collection of coordinate charts $\{U_\alpha, \phi_\alpha\}$ on a manifold M, such that the union of the open subsets covers the entire of M (we say U_α is an *open covering* of M). That is $\cup_\alpha U_\alpha \supseteq M$.

Going back to the example of the chart on a open subset of circle, we were unable to incorporate one single point on the circle for being described by the chart in Figure 2.14(a). However, with an atlas (Figure 2.14(b)), we can have multiple separate charts, using which we can cover the entire circle.

Definition 2.2.5 (Chart Transition [47], Fig. 2.15)**.** Consider two charts (U_m, ϕ_m) and (U_n, ϕ_n), for some open subsets U_m and U_n of a D-dimensional manifold M such that $U_m \cap U_n \neq \emptyset$. Then one can define the map $\phi_n \circ \phi_m^{-1} : \phi_m(U_m \cap U_n) \to \phi_n(U_m \cap U_n)$. Note that both the pre-image and image of this map are subsets of $\mathbb{R}^D$. Such a map is called a *chart transition* or a *coordinate transformation* or simply a *coordinate change* (the later two terms are used more frequently when $U_m = U_n = U_m \cap U_n$).

Thus, if $\mathbf{x} \in \mathbb{R}^D$ are the coordinate variables for (U_m, ϕ_m), and $\overline{\mathbf{x}} \in \mathbb{R}^D$ are the ones for (U_n, ϕ_n), then the chart transition is given by $\overline{\mathbf{x}} = \phi_n \circ \phi_m^{-1}(\mathbf{x})$. If the charts are assumed to be implicitly defined with the coordinate variables, one simply writes $\overline{\mathbf{x}} = f(\mathbf{x})$, or informally, $\overline{\mathbf{x}} = \overline{\mathbf{x}}(\mathbf{x})$ (*i.e.* the variables $\overline{x}_i$ are functions of the variables x_j).

A manifold is said to be differentiable if for every pair of charts, (U_m, ϕ_m) and (U_n, ϕ_n), the transitions $\phi_n \circ \phi_m^{-1}$ are differentiable functions of class C^∞ (where the notion of differentiability on $\mathbb{R}^D$ is natural due to the vector-space structure on it). Most of the manifolds that we will be concerned with in this thesis are differentiable.

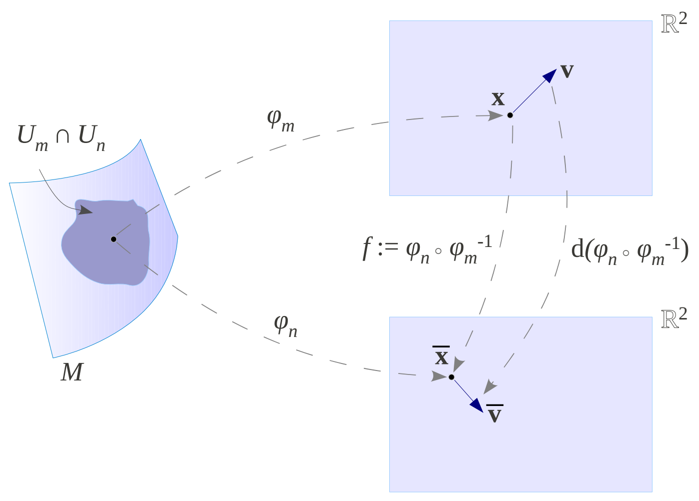


Figure 2.15: Chart Transition.

Tangent Space

We are familiar with the notion of a tangent to a curve or the tangent plane to a surface embedded in $\mathbb{R}^3$. The notion of such tangents is generalized to arbitrary manifolds by *tangent space*. Consider a chart (U_m, ϕ_m). A point, p, in U_m is represented by its image, $\mathbf{x} = [x^1, x^2, \cdots, x^D] \in \mathbb{R}^D$, under the action of ϕ. By the virtue of the vector-space structure of $\mathbb{R}^D$, one can imagine a vector centered at $\mathbf{x}$ and having components $v^1, v^2, \cdots, v^D$ along the directions of increasing values of $x^1, x^2, \cdots, x^D$ (Figure 2.15).

Fundamental to the definition of such a vector is the notion of transformation of the coefficients, $[v^1, v^2, \cdots, v^D]$, under a chart transition. Consider the chart transition $\overline{\mathbf{x}} = f(\mathbf{x})$ as discussed earlier. The vector (sitting at $\mathbf{x}$) with components $[v^1, v^2, \cdots, v^D]$ in the un-barred coordinate variables will have certain components $[\overline{v}^1, \overline{v}^2, \cdots, \overline{v}^D]$ in the barred coordinates (sitting at $\overline{\mathbf{x}}$) (Figure 2.15), which is determined by some transformation rule that we are yet to define. However, whatever that definition be, the property that such a transformation rule on coefficient vectors attached to points, $\mathbf{x}$ or $\overline{\mathbf{x}}$, *must* satisfy is invariance under a composition of forward and inverse transformation. That is, if corresponding to the chart transition $\overline{\mathbf{x}} = f(\mathbf{x})$ the transformation of the vector components (at the specific point p) is given by some function $\mathrm{d}_{\square} f_{\mathbf{x}} : \mathbb{R}^D \rightarrow \mathbb{R}^D$ such that $[\overline{v}^1, \overline{v}^2, \cdots, \overline{v}^D] = \mathrm{d}_{\square} f_{\mathbf{x}}([v^1, v^2, \cdots, v^D])$, and if corresponding to the inverse chart transition $\mathbf{x} = g(\overline{\mathbf{x}})$ the transformation of the previously transformed vector components is given by some function $\mathrm{d}_{\square} g_{\overline{\mathbf{x}}}$ such that $[\overline{\overline{v}}^1, \overline{\overline{v}}^2, \cdots, \overline{\overline{v}}^D] = \mathrm{d}_{\square} g_{\overline{\mathbf{x}}}([\overline{v}^1, \overline{v}^2, \cdots, \overline{v}^D])$, then we should have $[\overline{\overline{v}}^1, \overline{\overline{v}}^2, \cdots, \overline{\overline{v}}^D] = [v^1, v^2, \cdots, v^D]$. That is, $\mathrm{d}_{\square} g_{\overline{\mathbf{x}}} \circ \mathrm{d}_{\square} f_{\mathbf{x}}$ acts as an identity transformation on the vector components whenever $g = f^{-1}$. Moreover it should also satisfy the composition property $\mathrm{d}_{\square}(h \circ f)_{\mathbf{x}} = \mathrm{d}_{\square} h_{f(\mathbf{x})} \circ \mathrm{d}_{\square} f_{\mathbf{x}}$, where f and h are two chart transitions. These properties should hold true at every $\mathbf{x}$ and every component vector of vector attached to $\mathbf{x}$. The conditions are known as *cocycle conditions* [53, 8].

Contravariant Vectors: One such transformation rule for vectors that satisfies the above conditions is $\overline{v}^j = \sum_{i=1}^{D} \frac{\partial f^j}{\partial x^i}\Big|_{\mathbf{x}} v^i$ (where by f^j we simply mean the j^{th} component of the vector function f). This can be easily seen by noting that $[\overline{v}] = J_f[v]$ (where J_f is the Jacobian matrix of f at $\mathbf{x}$, and $[v]$ represents the coefficient vector as a column vector). Then a subsequent transformation under $\mathrm{d}_{[]}g$ will give $J_g[\overline{v}] = J_g J_f[v] = [v]$ (since g is the inverse transformation of f, the product of the Jacobian matrices, $J_g J_f$, is the identity matrix). Moreover, it is a standard exercise to check that $J_h J_f = J_{h\circ f}$. Such vectors, the coefficients of which follow such transformation rules, are known as *contravariant vectors* (or simply, *vectors*). There are other types of vectors that transform differently, but satisfy the said properties under composition of transformations. One such example is that of *covariant vectors*, which we will not discuss in details in this section.

A systematic way of writing contravariant vectors is by choosing $\frac{\partial}{\partial x^i}$ as basis for such vectors (*i.e.* quantities to which we 'multiply' the said coefficients/components, v^i, and take sum to write the full vector). This is purely done to avoid writing the said transformation rule for vector components explicitly, and instead take advantage of the standard rule for transformation of partial derivatives from one set of variables to another. Thus, under this representation, one would write for a vector, $\mathbf{v} = \sum_{i=1}^{D} v^i \frac{\partial}{\partial x^i}$ (where as usual the v^i are the components in the specific chart), and asserts that this $\mathbf{v}$ is independent of the choice of coordinate chart. That is,

$$\mathbf{v} \;=\; \sum_{i=1}^{D} v^i \frac{\partial}{\partial x^i} \;=\; \sum_{j=1}^{D} \overline{v}^j \frac{\partial}{\partial \overline{x}^j} \tag{2.2.1}$$

Now, from the property of partial derivatives,

$$\frac{\partial}{\partial x^i} \;=\; \sum_{j=1}^{D} \frac{\partial \overline{x}^j}{\partial x^i} \frac{\partial}{\partial \overline{x}^j}$$

Plugging this in the middle terms of (2.2.1) naturally reveals the transformation rule

$$\overline{v}^j \;=\; \sum_{i=1}^{D} \frac{\partial \overline{x}^j}{\partial x^i} v^i \tag{2.2.2}$$

where, we have used the informal notation for transformation, $\overline{\mathbf{x}} = \overline{\mathbf{x}}(\mathbf{x})$.

Definition 2.2.6 (Tangent Space [47]). Given a coordinate chart (U, ϕ) on a smooth manifold M, the tangent space at a point p on it represented by the coordinate variable $\mathbf{x} \in \Omega = Img(\phi) \subset \mathbb{R}^D$ is a D-dimensional vector space, $T_pM = T_{\mathbf{x}}\Omega$, spanned by the basis $\frac{\partial}{\partial x^1}, \frac{\partial}{\partial x^2}, \cdots, \frac{\partial}{\partial x^D}$.

The whole point of choosing such a basis is to make the vectors in the tangent space independent of the choice of the coordinate chart, as described earlier, and to make the components of the vectors in each coordinate chart follow certain transformation rule.

Einstein's Summation Convention for Repeated Indices: Purely for the convenience of writing, we would often drop the summation sign (capital sigma) inside expressions like $\sum_{j=1}^{D} \frac{\partial \overline{x}^j}{\partial x^i} \frac{\partial}{\partial \overline{x}^j}$ and $\sum_{i=1}^{D} v^i \frac{\partial}{\partial x^i}$, and whenever there is a repeated index (j and i in these cases respectively), we will assume the summation to be implied. Thus, for these expressions we will simply write $\frac{\partial \overline{x}^j}{\partial x^i} \frac{\partial}{\partial \overline{x}^j}$ and

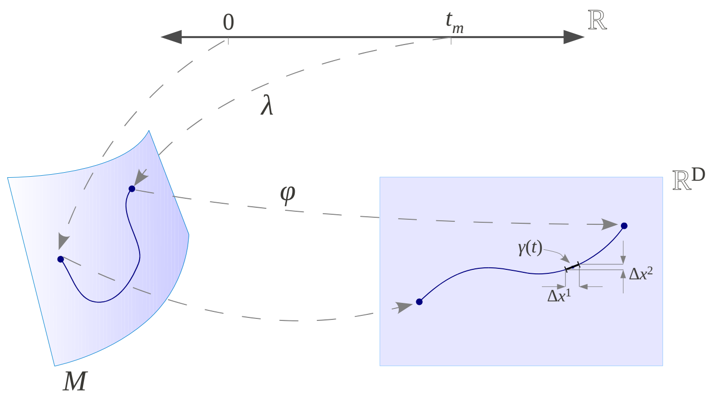


Figure 2.16: An infinitesimal element on a curve.

$v^i \frac{\partial}{\partial x^i}$ respectively.

2.2.2 Riemannian Metric, Geodesics and Curvature

We have previously described the notion of a distance function on a general topological space. There is a more restricted (and probably more intuitive) notion of distance – one that is induced by the 'length' of the shortest curve connecting two points. Before we can describe the shortest curve, we need to define length of a curve. Throughout, we will mostly be working with differentiable manifolds as our topological spaces.

Informal Description

The definition of length of a curve on a D-dimensional manifold, M, represented by $\lambda : [0, t_m] \to M$, is closely related with the definition of the length of an infinitesimal element on the curve. One can then integrate the infinitesimal lengths to obtain the total length of the curve. While the length of an infinitesimal element on a curve can have a variety of possible definitions (including ones based on arbitrary norms and higher order derivatives of the curve), the type of definition that we will be interested in is based on quadratic forms on the tangent spaces of the manifold. Such a definition of length arises naturally in many physical and practical applications, and forms the motivation for Riemannian metric.

For a given coordinate chart $C = (U, \phi)$ with coordinate variables x^i, such that the curve λ lies entirely in U, we define the curve in the given coordinates as $\gamma = \phi \circ \lambda : [0, t_m] \to \mathbb{R}^D$ (Figure 2.16). The domain of γ is called the parameter space of the curve. A small infinitesimal element on the curve between t and $t + \Delta t$ in the parameter space is then represented by $\dot{\gamma}(t)\Delta t =: [\Delta x^1, \Delta x^2, \cdots, \Delta x^D]$ (where Δx^i represent the change in the i^{th} coordinate variable across the infinitesimal element in the chart C). It is not difficult to note that $[\Delta x] := [\Delta x^1, \Delta x^2, \cdots, \Delta x^D]$ behaves like coefficients of a contravariant vector (since in a different coordinate chart one would get $\Delta \overline{x}^j = \frac{\partial \overline{x}^j}{\partial x^i} \Delta x^i$ – a consequence of elementary calculus).

Then we define the 'length' of the element as

$$\Delta s = \sqrt{g_{ij} \Delta x^i \Delta x^j} \tag{2.2.3}$$

where, as agreed, we have assumed summation over the repeated indices. g_{ij} hence represents elements of a matrix, g (which is specific to the given coordinate chart), such that the length is given by $[\Delta x]^T g [\Delta x]$ (where we assumed $[\Delta x]$ to be a column vector of the coefficients).

The condition that the definition of length must satisfy is that the length of an infinitesimal element should not change upon change of coordinates. Thus, if we are given another coordinate chart with coordinate variables $\overline{x}^i$, the following condition must hold,

$$(\Delta s)^2 \quad = \quad g_{ij} \Delta x^i \Delta x^j \quad = \quad \overline{g}_{pq} \Delta \overline{x}^p \Delta \overline{x}^q \tag{2.2.4}$$

Using the transformation rule for coefficients of contravariant vector, $\Delta x^i = \frac{\partial x^i}{\partial \overline{x}^p} \Delta \overline{x}^p$, and substituting it in the middle term of the above equation, one gets the relationship between g_{ij} and $\overline{g}_{ij}$,

$$\overline{g}_{pq} = \frac{\partial x^i}{\partial \overline{x}^p} \frac{\partial x^j}{\partial \overline{x}^q} g_{ij} \tag{2.2.5}$$

This gives a transformation rule for the elements of the matrix used to define length as in (2.2.3).

The discussion so far indicates the definition of a bilinear scalar product on contravariant vectors (*e.g.* acting on two copies of $\mathbf{\Delta x} = \Delta x^i \frac{\partial}{\partial x^i}$ to give the square of length of the segment as prescribed by (2.2.4)). It is important to note that the value of this product for two contravariant vectors does not depend on the choice of the coordinate chart (this is achieved by the way we constructed the transformation rule (2.2.5)). Thus it is a product defined on the tangent spaces of the manifold itself.

Definition 2.2.7 (Riemannian Metric [47])**.** A Riemannian metric on a differentiable manifold, M, is a symmetric bilinear scalar product on each tangent space, T_pM, for every $p \in M$, such that it varies smoothly with p. That is, it is a bilinear function $\mathbf{g}(p) : T_pM \times T_pM \to \mathbb{R}$, that is symmetric in its two parameters, and it itself is smooth in p.

$\mathbf{g}$ is called the metric tensor, and the g_{ij}, from our previous discussion, the matrix representation of the metric in a particular coordinate chart. The symmetry condition implies that the matrix representation is a symmetric matrix in any coordinate chart.

It is important to note that matrix representation of the metric in a particular coordinate chart simply constitutes a collections of $D(D+1)/2$ functions in the coordinate variables. That is,

$$g \quad = \quad \begin{bmatrix} g_{11}(\mathbf{x}) & g_{12}(\mathbf{x}) & \cdots & g_{1D}(\mathbf{x}) \\ g_{21}(\mathbf{x}) & g_{22}(\mathbf{x}) & \cdots & g_{2D}(\mathbf{x}) \\ \vdots & \vdots & \ddots & \vdots \\ g_{D1}(\mathbf{x}) & g_{D2}(\mathbf{x}) & \cdots & g_{DD}(\mathbf{x}) \end{bmatrix}$$

with $g_{ij}(\mathbf{x}) = g_{ji}(\mathbf{x})$.

We use the following notation to write derivatives of the components of the matrix representation of the metric in a particular coordinate chart,

$$g_{ij,k} \quad := \quad \frac{\partial g_{ij}}{\partial x^k}$$

Thus, for a different coordinate system, $\overline{g}_{ij,k} := \frac{\partial \overline{g}_{ij}}{\partial \overline{x}^k}$.

One can write the inverse of the matrix g. The components of the inverse matrix are once again functions of the coordinate variables. The components of the matrix of g^{-1} are written as g^{ij} (note that the indices are written as superscript). It is easy to verify that $g_{ik}g^{kj}$ gives components of the product of the two matrices. Thus,

$$g_{ik}g^{kj} = \delta_i^j$$

where, δ_i^j represents the Kronecker delta (*i.e.* the components of the identity matrix).

We will next describe a few quantities (specific to a particular coordinate chart) derived directly from the components of the matrix representation of the metric tensor on a particular coordinate system without detailing their immediate significance. We will hence use those quantities in order to state some results. The reader can refer to [47] for more detailed discussion on these quantities.

Definition 2.2.8 (Christoffel Symbols)**.**

$$\Gamma^i_{kl} \quad := \quad \frac{1}{2}\, g^{im} \left(g_{mk,l} + g_{ml,k} - g_{kl,m} \right) \tag{2.2.6}$$

Similar to the components of the metric, for notational convenience we define

$$\Gamma^i_{kl,p} := \frac{\partial \Gamma^i_{kl}}{\partial x^p}$$

Definition 2.2.9 (Ricci Scalar Curvature)**.**
We define the *Riemannian curvature tensor* as

$$R^\rho{}_{\sigma\mu\nu} \quad = \quad \Gamma^\rho_{\nu\sigma,\mu} - \Gamma^\rho_{\mu\sigma,\nu} + \Gamma^\rho_{\mu\lambda}\Gamma^\lambda_{\nu\sigma} - \Gamma^\rho_{\nu\lambda}\Gamma^\lambda_{\mu\sigma}$$

We hence define the *Ricci curvature* tensor,

$$R_{ij} \quad = \quad R^k{}_{ikj}$$

Following that, we define the *Ricci scalar curvature*,

$$\begin{aligned} R \quad &= \quad g^{ij} R_{ij} \\ &= \quad g^{ij} (\Gamma^k_{ij,k} - \Gamma^k_{ik,j} + \Gamma^l_{ij}\Gamma^k_{kl} - \Gamma^l_{ik}\Gamma^k_{jl}) \end{aligned} \tag{2.2.7}$$

It is to be noted that Ricci scalar curvature is a scalar function on a manifold, and is independent of the choice of the coordinate system.

In spite of all the intricacies in the definitions and their interpretations, both Γ^i_{kl} and R are essentially functions of the components of the matrix representation of the metric, g_{ij}, and their derivatives. Hence both of these quantities can be expressed as functions of the coordinate variables $x^1, x^2, \cdots, x^D$.

Geodesic

Consider the length of a curve $\gamma : [0, t_m] \to \mathbb{R}^D$ on a Riemannian manifold (expressed in a particular coordinate chart – *i.e.* if $\lambda : [0, t_m] \to U \subseteq M$ was the original curve, and (U, ϕ) is a coordinate chart, then we define $\gamma = \phi \circ \lambda$),

$$L(\gamma) = \int_0^{t_m} \sqrt{g_{ij}\ \dot{\gamma}^i\ \dot{\gamma}^j}\ \mathrm{d}t \tag{2.2.8}$$

where $\dot{\gamma}^i$ represent the i^{th} component of the coefficients of the tangent vector in the particular coordinate. Due to the way we defined g and its transformation, the length of a fixed curve on M is independent of the choice of the coordinate system.

If the start and end points of γ are constrained to two specific points (*i.e.* $\gamma(0) = \mathbf{s} \in \mathbb{R}^D$ and $\gamma(t_m) = \mathbf{e} \in \mathbb{R}^D$ are the constraints), then one can consider the problem of minimizing $L(\gamma)$ over the different curves, γ, that connect the two points. It can be shown that the γ that minimizes L, also minimizes the integral

$$E(\gamma) = \int_0^{t_m} g_{ij}\ \dot{\gamma}^i\ \dot{\gamma}^j\ \mathrm{d}t$$

Typically a *calculus of variation* approach is taken to solve this problem. The solution thus obtained is in form of differential equation, known as the *Geodesic Equation*,

$$\frac{\mathrm{d}^2\gamma^i}{\mathrm{d}t^2} + \Gamma^i_{jk}\frac{\mathrm{d}\gamma^j}{\mathrm{d}t}\frac{\mathrm{d}\gamma^k}{\mathrm{d}t} = 0 \tag{2.2.9}$$

That is, any curve that minimizes the integral of (2.2.8) between any two points (even arbitrarily close) satisfies the geodesic equation (2.2.9), and is called a *geodesic*.

Geodesics are of course independent of the choice of the coordinate chart – a geodesic computed in one coordinate chart will map to the corresponding geodesic computed in a different coordinate chart under the coordinate transformation.

The solution by the calculus of variation approach to obtain (2.2.9) is a locally optimal one, not a globally optimal one. Thus, it is possible that more than one curve connecting two points satisfy the geodesic equation. In fact it can be shown [47] that there is an unique geodesic in every homotopy class of curves connecting two points in a Riemannian metric space.

The length of the *shortest geodesic* (*i.e.* minimum over the lengths of all the possible geodesics between those points) between two points describe a distance function.

Curvature

There are essentially two notions of curvature of any metric space – extrinsic and intrinsic. The Ricci Scalar (Definition 2.2.9) is a measure of intrinsic curvature. Intrinsic curvatures are more fundamental to a metric space, whereas extrinsic curvature arises mostly due to embedding.

Intuitively, a space (or a subset of a space) has zero intrinsic curvature everywhere if it can

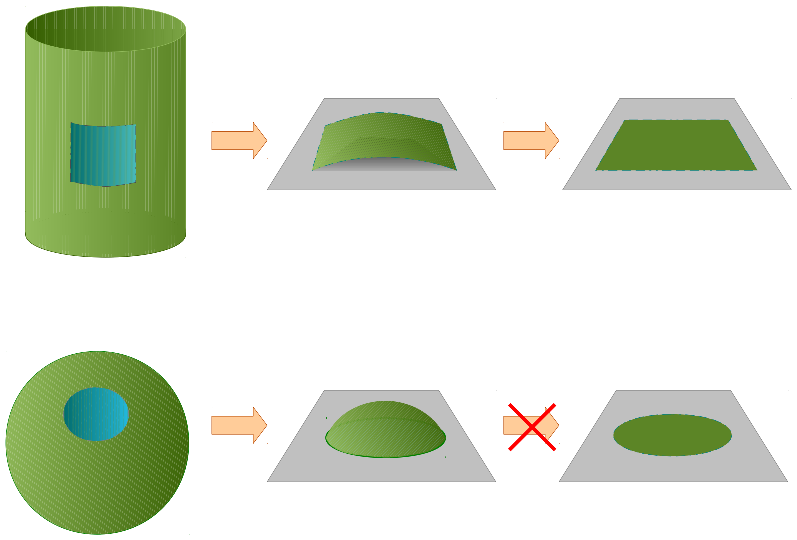


Figure 2.17: Curvature: Extrinsic vs. Intrinsic. A small patch on the cylinder can be flattened without *stretching* or *compressing* any part of it. However that is not possible for a patch from a sphere. Thus the curvature of a cylinder is extrinsic, while that of a sphere is intrinsic. The Ricci scalar curvature computed using any coordinate chart on an open set on the surface of a symmetric cylinder will give a constant value of zero. While the same for a symmetric sphere will be a constant positive value.

be deformed isometrically (*i.e.* without stretching/squeezing any part) so as to embed it in a flat Euclidean space of same dimension (Figure 2.17).

Topology and Metric

Since metric itself is described by its matrix representation in a specific chart (and possibly multiple representations in an atlas), in order to describe metric on a topological space independent of a coordinate chart, one often refers to certain description of the Ricci scalar curvature of the space or properties of geodesics on the space (since Ricci scalar curvature and properties, like intersection, of geodesics are independent of the choice of the coordinate chart). For example, one can refer to a metric space with zero Ricci scalar curvature everywhere on it. An example of such a space is a flat Euclidean space with Euclidean metric. The surface of a cylinder also has zero scalar curvature (Figure 2.17). Again, one can refer to a space with constant positive scalar curvature everywhere. A sphere is such a space. Similarly, a hyperbolic space is one with constant negative Ricci scalar curvature everywhere.

However, such a description of a metric using intrinsic curvature, although describes a space locally, is not enough to define the global topology of a space. The zero curvature of $\mathbb{R}^2$ and the cylinder was one such example. Again, the same topological space can admit two totally different descriptions of metric. An interesting example of such a case is that of a flat torus. The *flat torus* is a specific embedding of the standard topological 2-torus in $\mathbb{R}^4$. The metric inherited from the embedding space by the torus due to the specific embedding turns out to be flat (*i.e.* the Ricci scalar curvature of the flat torus is zero everywhere). However, in the standard embedding of the torus in $\mathbb{R}^3$, it is not possible to achieve zero scalar curvature at every point on it.

A particular topological space can admit a variety of metrics. However it may not admit an arbitrary description of a metric. For example, one cannot construct a metric on a topological 2-sphere so that the Ricci curvature is zero everywhere on it. This is closely related to the notion of *total curvature* of a space, which is indicative of the global topology of the space.

2.3 Graph Search Algorithms

As discussed in Section 1.1.2, in the discrete approach towards planning path, a graph is constructed by distritizing the configuration space and placing a vertex/node at each discretized cell. Edges are established based on available actions between neighboring vertices.

Thus a graph consists of 3 components: A vertex set $\mathcal{V}(G)$, an edge set $\mathcal{E}(G) \subseteq \mathcal{V}(G) \times \mathcal{V}(G)$, and a cost function $\mathcal{C}_G : \mathcal{E}(G) \to \mathbb{R}^+$. An element in $\mathcal{V}(G)$ is called a vertex or a node. An element in $\mathcal{E}(G)$ is represented by the ordered pair $[a, b] \in \mathcal{E}(G)$, which implies that there exists an edge in G connecting $a \in \mathcal{V}(G)$ to $b \in \mathcal{V}(G)$.

2.3.1 Dijkstra's Algorithm

Dijkstra's Algorithm [26] is the most fundamental in finding minimum cost (or shortest distance) paths between a vertex, $p \in \mathcal{V}(G)$, in the graph to every other vertex in the graph that is reachable from p. It is complete and guaranteed to be optimal. The intuition behind the Dijkstra's Algorithm is that starting from the start node p, a 'wavefront' of *almost* equal geodesic distances (*i.e.* cost of shortest paths) is propagated through the graph. Figure 2.18 illustrates the progress of Dijkstra's algorithm. Notice the wavefront marked by the empty blue circles. This is the set of nodes $\{u \in Q \mid g(u) \text{ is finite}\}$.

Algorithm 2.3.1

```
$g$ = **Dijkstras** $(G, p)$
  Inputs:    a. Graph $G$
             b. Start node $p \in \mathcal{V}(G)$
  Outputs:   a. The shortest distance map $g : \mathcal{V}(G) \to \mathbb{R}^+$
1  Initiate $g$: Set $g(v) := \infty$, for all $v \in \mathcal{V}(G)$   // Minimum distance
2  Set $g(p) = 0$
3  Set $Q := \mathcal{V}(G)$   // Set of un-expanded nodes
4  while $(Q \neq \emptyset)$
5      $q := \text{argmin}_{q' \in Q} \; g(q')$   // Vertex to expand. $Q$ is maintained by a heap data-structure.
6      if $(g(q) == \infty)$
7          break
8      $Q = Q - q$   // Remove $q$ from $Q$
9      for each $(\{w \in \mathcal{N}_G(q)\})$   // For each neighbor of $q$
10         Set $g' := g(q) + \mathcal{C}_G([q, w])$
11         if $(g' < g(w))$
12             Set $g(w) = g'$
13 return $g$
```

where, $\mathcal{N}_G(u) = \{w' \in \mathcal{V}(G) \mid [u, w'] \in \mathcal{E}(G)\}$, the set of neighbors of u.

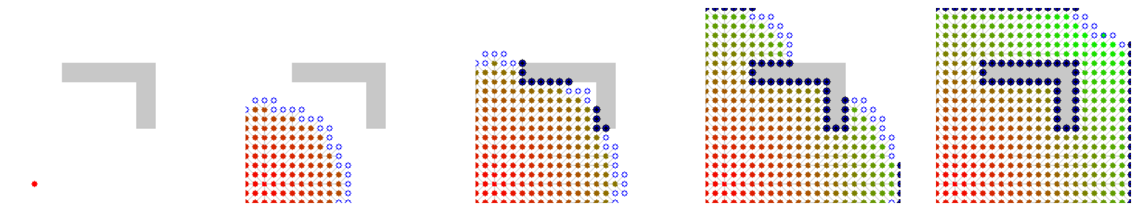

(a) *iter* = 0. Start node in red.
(b) *iter* = 96. The wave front (empty blue circles) is visible.
(c) *iter* = 190.
(d) *iter* = 286.
(e) *iter* = 376.

Figure 2.18: Illustration of progress of Dijkstra's algorithm.

Once one has the map $g : \mathcal{V}(G) \to \mathbb{R}^+$, one can construct a shortest path from any given node, r, to the start node p using Algorithm 2.3.2.

Algorithm 2.3.2

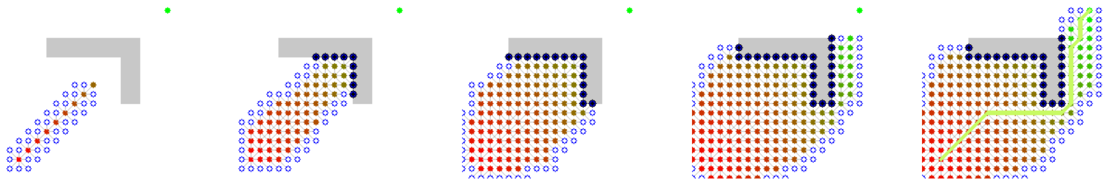
```
P = Reconstruct_Path (G, g, r)
  Inputs:    a. Graph G
             b. The shortest distance map $g : \mathcal{V}(G) \to \mathbb{R}^+$
             c. Vertex to which to find the shortest path, $r \in \mathcal{V}(G)$
  Outputs:   a. A path (ordered set of vertices) in the graph, $P = [\rho_1, \rho_2, \rho_3, \cdots, \rho_n = r]$
1  Initiate $P = [\,]$
2  if $(g(r) == \infty)$    // $r$ unreachable from the start node
3      return $P$
4  Set $v := r$
5  while $(g(v) \neq 0)$
6      $P = v \oplus P$    // Insert $v$ at the beginning of $P$.
7      $v = \text{argmin}_{w \in \mathcal{N}_G^{-1}(v)} g(w)$    // back-trace predecessor that led to $v$.
8  $P = v \oplus P$    // Insert the final vertex (the start node) at the beginning of $P$.
9  return $P$
```

where, $\mathcal{N}_G^{-1}(u) = \{w' \in \mathcal{V}(G) \mid [w', u] \in \mathcal{E}(G)\}$, the set of vertices in G that has u as a neighbor. Note that for undirected graphs $\mathcal{N}_G^{-1}(u) = \mathcal{N}_G(u)$.

Note that the shortest path may not be unique, since in many cases there can be multiple paths with the same least cost.

2.3.2 A* Algorithm

The A* Algorithm [39] is, in essence, much similar to the Dijkstra's algorithm. While in Dijkstra's algorithm one is typically interested in finding least cost paths to every vertex in the graph from the start vertex, in A* search typically a fixed goal is provided. This allows us to direct the search in a more informed manner. Instead of expanding the 'wavefront' (mentioned earlier) uniformly in all direction, we expand it with a bias towards the direction of the goal vertex (Figure 2.19). This bias is governed by what is known as a *heuristic function*. A heuristic function for a graph G is a function $h : \mathcal{V}(G) \times \mathcal{V}(G) \to \mathbb{R}^+$, such that $h(v_a, v_b)$ give some estimate of the minimum distance (*i.e.* total cost of shortest path) between the vertices v_a and v_b. An *admissible heuristic function* is such a heuristic function that always underestimates the actual minimum cost. It can be shown that the A* algorithm (Algorithm 2.3.3) with an admissible heuristic returns the optimal (least



(a) $iter = 8$. Goal vertex in green. (b) $iter = 57$. (c) $iter = 114$. (d) $iter = 171$. (e) $iter = 376$.

Figure 2.19: Illustration of progress of A* algorithm. The open set is marked by empty blue circles.

cost) path to the goal. Of course, the constant function $h(v_a, v_b) = 0$ is an admissible heuristic, and in that case it can be shown that the A* algorithm becomes equivalent to the Dijkstra's algorithm. In fact, the only major algorithmic difference of A* from Dijkstra's (besides a few other structural differences) is that in place of line 7 of Algorithm 2.3.1, one selects the vertex q to be expanded as the one with lowest value of f instead of g (line 8 of Algorithm 2.3.3), where the f-value is g-value plus the heuristic from that vertex to the goal.

Algorithm 2.3.3

```
$g$ = **A_star** $(G, p, r, h)$
  Inputs:    a. Graph $G$
             b. Start node $p \in \mathcal{V}(G)$
             c. Goal node $r \in \mathcal{V}(G)$
             d. An admissible heuristic function $h : \mathcal{V}(G) \times \mathcal{V}(G) \rightarrow \mathbb{R}^+$
  Outputs:   a. A path connecting start vertex to goal vertex, $P = [p=\rho_1,\ \rho_2,\ \rho_3,\ \cdots,\ \rho_n=r]$
1   Initiate $g$: Set $g(v) := \infty$, for all $v \in \mathcal{V}(G)$   // Minimum distance
2   Set $g(p) = 0$
3   Initiate $f$: Set $f(v) := \infty$, for all $v \in \mathcal{V}(G)$   // $f$-values
4   Set $f(p) = h(p)$
5   Set $\overline{Q} := \emptyset$   // Closed set (set of expanded nodes)
6   Set $R := \{p\}$   // Open set (candidate nodes for expansion)
7   while $(R \neq \emptyset$ && $r \notin R)$
8       $q := \text{argmin}_{q' \in R}\ f(q')$   // Vertex to expand. $R$ is maintained by a heap data-structure.
9       if $(q == r)$   // Goal vertex reached.
10          return  Reconstruct_Path $(G, g, r)$
11      $R = R - q$   // Remove $q$ from open set.
12      $\overline{Q} = \overline{Q} \cup q$   // Add $q$ to closed set.
13      for each $(\{w \mid w \in \mathcal{N}_G(q)$ and $w \notin \overline{Q}\})$   // For each neighbor of $q$ that's not in closed set
14          $R = R \cup w$   // Add $w$ to open set.
15          Set $g' := g(q) + \mathcal{C}_G([q, w])$
16          if $(g' < g(w))$
17              Set $g(w) = g'$
18              Set $f(w) = g' + h(w, r)$
19  return [ ]   // Goal vertex is not reachable.
```

Heuristic Function

The choice of the heuristic function, h, is extremely crucial in an A* search. A heuristic function is a positive scalar function of the vertices in the search graph. For A* to return an optimal solution,

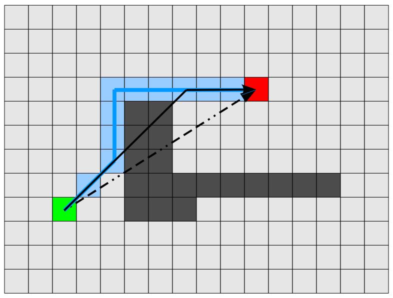


Figure 2.20: The two different types of heuristics for an 8-connected grid graph: The green is the start state and red is the goal vertex. Dashed arrow represents h_E, solid black arrow represents h_8, light blue path represents the actual least cost path. Obstacles are in dark gray.

the heuristic function needs to be an admissible one – that is, it should be such that it never overestimates the actual minimum cost for reaching the goal. However to make the search more efficient and minimize the number of vertices expanded, the heuristic must be as close as possible to the actual minimum cost to the goal. For a graph constructed by discretization of an Euclidean space (like the one in Figure 1.4(a)), one obvious and commonly used heuristic function is the Euclidean distance to the goal, *i.e.* $h_E(u, v) = \|\mathbf{u} - \mathbf{v}\|_2$ (where, $\mathbf{z}$ represent the coordinate of the vertex, z, in the original configuration space). But, for particular types of discretization one can use heuristic functions with tighter bounds. For example, for an 8-connected grid graph (one created by uniform square discretization of a plane as in Figure 2.20), one can use a more efficient heuristic given by $h_8(u, v) = \sqrt{2}\ \min(\Delta x, \Delta y) + |\Delta x - \Delta y|$, where $\Delta x = |\mathbf{u}_x - \mathbf{v}_x|$ and $\Delta y = |\mathbf{u}_y - \mathbf{v}_y|$, with $\mathbf{z}_x$ and $\mathbf{z}_y$ respectively representing the x and y coordinates of the point $\mathbf{z}$ in the original coordinate space. This is illustrated in Figure 2.20.

On-the-fly Graph Construction

Before we conclude this section, we would like to emphasize an important properties of the search algorithms (not just limited to Dijkstra's and A*, but generalizes to almost all search algorithms in general): Although in both Algorithms 2.3.1 and 2.3.3, a graph G was mentioned as an input parameter, it is in fact not necessary to construct the entire graph from before and pass it on to the algorithm. If a graph can be constructed from a set of simple rules we may just need to pass those rules to the algorithms. In that case the Dijkstra's or A* algorithm can construct the graph on-the-fly as it progresses (not very different from the way vertices are 'created' with the progress of iterations in Figures 2.18 and 2.19). In fact, the entire graph G may very well be infinite, but we still can find a path from a given start to a given goal vertex in the graph using either of the algorithms.

Chapter 3

Search-based Path Planning with Topological Constraints in 2 and 3 Dimensional Euclidean Spaces

3.1 Introduction

3.1.1 Motivation: Homotopy Classes of Trajectories

In Euclidean configuration spaces homotopy classes of trajectories arise due to presence of obstacles in an environment. Two trajectories connecting the same start and goal coordinates are in the same homotopy class if they can be smoothly deformed into one another without intersecting any obstacle in the environment, otherwise they are in different homotopy classes. In many applications, it is important to distinguish between trajectories in different homotopy classes, as well as identify the different homotopy classes in an environment (*e.g.*, trajectories that go left around a circle in two dimensions versus right). For example, in order to deploy a group of agents to explore an environment [9], an efficient strategy ought to be able to identify the different homotopy classes and deploy one robot in each homotopy class. One may also wish to determine the least cost path for each robot constrained to or avoiding specified homotopy classes. In many problems the notion of *visibility* is linked intrinsically with homotopy classes. In tracking of uncertain agents in an environment with dynamic obstacles, the ability to deal with occlusions during a certain time frame is important [89]. A knowledge of the possible homotopy classes of trajectories that a target can take in the environment when it is occluded can help more efficient belief propagation.

Despite being mostly an uncharted research area, homotopy class constraints often appear in path planning problems. For example, in multi-agent planning problems [88, 49], the trajectories often need to satisfy certain proximity or resource constraints or constraints arising due to tasks allocated to agents, which translates into restricting the solution trajectories to certain homotopy classes that respect those constraints. In exploration and mapping problems [9], agents often need to plan trajectories based on their mission or part of the environment they are assigned for mapping or exploration, and hence restrict their trajectories to certain homotopy classes.

Classification of homotopy classes in two-dimensional workspaces has been studied in the robotics literature using geometric methods [37, 41], probabilistic road-map construction [75] techniques and triangulation-based path planning [22]. However, efficient planning for least cost trajectories with homotopy class constraints is difficult using such representations even in 2-dimensions. Neither it is possible to efficiently explore/find optimal trajectories in different homotopy classes in an environment.

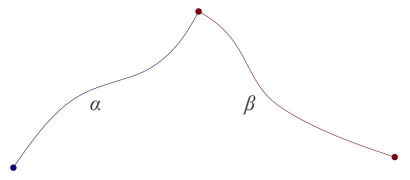
In this chapter we propose a novel way of classifying and representing homology classes, a close analog of homotopy classes, in two and three dimensional Euclidean configuration spaces, which are the types of configuration spaces we encounter most often in robot planning problems. For the 2-dimensional case we use theorems from complex analysis for developing a compact way of representing homology classes of trajectories, while for 3-dimensional configuration spaces we exploit theorems from electromagnetism. In later chapter we show that the formulae for 2 and 3 dimensional cases can in fact be generalized to arbitrary D-dimensional Euclidean configuration spaces with obstacles.

The novelty of our work lies in the fact that our proposed representation allows us to identify/distinguish trajectories in different classes and compute least-cost paths in non trivial configuration spaces with topological constraints using graph search-based planning algorithms. The representation we propose is designed to be independent of the type of the environment, the discretization scheme or cost function. Our proposed representation can also be used in configuration spaces with additional degrees of freedom that do not effect homotopy classes of the trajectories (*e.g.* for unicycle modes of mobile robots, the configuration space consists of variables x, y and θ. But the last variable, θ, does not effect the homotopy classes of trajectories. Only the projection of the $X-Y$ plane is enough to capture the topological information).

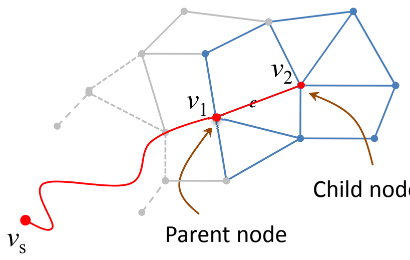
Using such a representation we show that topological constraints can be seamlessly integrated with graph search techniques for determining optimal paths subject to constraints. We also discuss how this method can be used to explore multiple homotopy classes in an environment using a single graph search.

3.1.2 Capturing Topological Information in Search-based Planning

In search-based planning algorithms one typically starts by discretizing a given environment to create a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Starting form an initial vertex, $v_s \in \mathcal{V}$, a typical graph-search algorithm *expands* the nodes of the graph by traversing the edges. Values are maintained and associated with each expanded node that capture the metric information (distance/cost) of shortest path leading to the expanded node from v_s. For example, A* search algorithm (Algorithm 2.3.3) maintains two functions $g, f : \mathcal{V} \to \mathbb{R}$. $g(v)$ is the cost of the current path from the start node to node v, and $f(v) = g(v) + h(v)$ is an estimate of the total distance from start to goal going through v. The algorithm maintains an *open set*, the set of nodes to be *expanded*. Each time it expands a node v, it updates the values of $g(v')$ for each neighbor v' of v, by adding to $g(v)$ the cost of the edge $c(\overline{vv'})$ (the update happens only if the newly computed value is lower than the previous value). This process continues until a desired vertex $v_g \in \mathcal{V}$ is reached [39].



(a) Given an arbitrary differential 1-form (or a measure) "$(\cdot)$", $\int_{\alpha\sqcup\beta}(\cdot) = \int_{\alpha}(\cdot) + \int_{\beta}(\cdot)$, where $\alpha \sqcup \beta$ is the total curve formed by α and β together.



(b) We want to design a differential 1-form, integration of which along curves, give the desired H-signature. In a graph-search setup, due to the additivity property, $\mathcal{H}(\widetilde{v_s v_2}) = \mathcal{H}(\widetilde{v_s v_1}) + \int_e \omega$ – the value for a path up to a vertex can be easily computed from the value up to one of its children.

Figure 3.1: Additivity of integration can be exploited in graph search algorithm.

The fact that the value of $g(v')$ can be computed from $g(v) + c(\overline{vv'})$ is due to the fact that the cost function is additive (Figure 3.1(a)). This is because the metric information about the underlying space is captured using a differential 1-form, namely the infinitesimal length/cost, $\mathrm{d}l$ (strictly speaking, it is a *measure*, but we will compare it with a 1-form since they share the property of integrability). The cost of an edge, e, of the graph is then computed as an integral of the form $c(e) = \int_e \mathcal{J}(l)\ \mathrm{d}l$ (with some scaling function $\mathcal{J}$). By the virtue of integration, the cost function, c, is additive (*i.e.* if α and β are two curves that share a common end point, then $c(\alpha \sqcup \beta) = c(\alpha) + c(\beta)$, where "$\sqcup$" indicates the *disjoint union*, and represent the total curve formed by the two curves together). This implies, in an arbitrary graph search algorithm, during the expansion of the vertices of the graph, the cost of the shortest path up to a vertex that is being expanded can simply be computed by adding to that of its parent (in terms of sequence of expansions) the cost of the edge connecting to it. This additive property of length/cost is key in developing such graph search algorithms.

While the quantity, $\mathcal{J}(l)\ \mathrm{d}l$, yields metric information, there are other differential 1-forms that can incorporate other information about the underlying space and can be used for guiding the search algorithm. The main idea in this thesis is to determine a differential 1-forms that encodes topological information about the space and let us guide the search accordingly. In particular, we will be trying to find generators of the De Rham's cohomology group, $H^1_{dR}(\mathbb{R}^D - \mathcal{O})$ [8], for the Euclidean space $\mathbb{R}^D$ punctured by the obstacles $\mathcal{O}$. A more detailed discussion from the algebraic topology view point will be provided in the next chapter. In this chapter we introduce the basic concepts using more familiar tools.

3.1.3 H-signature as Class Invariants for Trajectories

We consider a very general differential 1-form in a given D-dimensional configuration space $\mathcal{C}$. If $x_1, x_2, \cdots, x_D$ are the coordinate variables describing the configuration space, a general differential 1-form can be written as $\mathrm{d}h := f_1(\mathbf{x})\ \mathrm{d}x_1 + f_2(\mathbf{x})\ \mathrm{d}x_2 + \cdots + f_D(\mathbf{x})\ \mathrm{d}x_D$. Thus, for any given trajectory/curve, τ, in this configuration space, one can compute $\mathcal{H}(\tau) = \int_\tau \mathrm{d}h$. We call this the

H-signature of τ (Figure 3.1(b)).

We want to design the 1-form $\mathrm{d}h$ and the H-signature of a trajectory such that it is an invariant across trajectories in the same homotopy class. However, because we use 1-forms and their integrals along closed curves to classify trajectories, we naturally obtain invariants for homology classes of trajectories [40, 71, 8]. But in most practical robotics problems the notion of homology and homotopy of trajectories can be used interchangeably, especially when finding the least cost path. This is discussed in greater detail with examples in Sections 3.5.1 and 3.6.

3.2 Homotopy and Homology Classes of Trajectories

Let $\mathcal{C}$ be the D-dimensional configuration space (which exclude the inaccessible regions, *e.g.* the obstacles). Let $x_1, x_2, \cdots, x_D$ be the coordinate variables used to describe the configuration space. A point in $\mathcal{C}$ is thus represented by $\mathbf{x}$.

Definition 3.2.1 (Homotopic trajectories)**.** Two trajectories τ_1 and τ_2 connecting the same start and end coordinates, $\mathbf{x}_s$ and $\mathbf{x}_g$ respectively, are homotopic iff one can be continuously deformed into the other without intersecting any obstacle.

Formally, if $\tau_1 : [0,1] \to \mathcal{C}$ and $\tau_2 : [0,1] \to \mathcal{C}$ represent the two trajectories (with $\tau_1(0) = \tau_2(0) = \mathbf{x}_s$ and $\tau_1(1) = \tau_2(1) = \mathbf{x}_g$), then τ_1 is homotopic to τ_2 iff there exists a continuous map $\eta : [0,1] \times [0,1] \to \mathcal{C}$ such that $\eta(\alpha, 0) = \tau_1(\alpha)\ \forall \alpha \in [0,1]$, $\quad \eta(\beta, 1) = \tau_2(\beta)\ \forall \beta \in [0,1]$, and $\eta(0,\gamma) = \mathbf{x}_s, \eta(1,\gamma) = \mathbf{x}_s\ \forall \gamma \in [0,1]$. Alternatively, τ_1 and τ_2 are homotopic iff $\tau_1 \sqcup -\tau_2$ belongs to the trivial class of the first homotopy group of $\mathcal{C}$, denoted by $\pi_1(\mathcal{C})$. In the notation of [40], $[\tau_1 \sqcup -\tau_2] = \mathbf{0} \in \pi_1(\mathcal{C})$.

Definition 3.2.2 (Homologous trajectories)**.** Two trajectories τ_1 and τ_2 connecting the same start and end coordinates, $\mathbf{x}_s$ and $\mathbf{x}_g$ respectively, are homologous iff τ_1 together with τ_2 (the later with opposite orientation) forms the complete boundary of a 2-dimensional manifold embedded in $\mathcal{C}$ not containing/intersecting any of the obstacles.

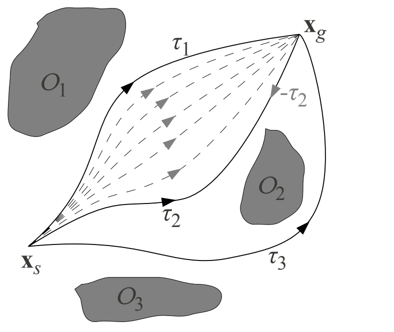
Formally, τ_1 and τ_2 are homologous iff $\tau_1 \sqcup -\tau_2$ belongs to the trivial class of the first homology group of $\mathcal{C}$, denoted by $H_1(\mathcal{C})$. In the notation of [40], $[\tau_1 \sqcup -\tau_2] = \mathbf{0} \in H_1(\mathcal{C})$.

A set of homotopic trajectories form a homotopy class, while a set of homologous trajectories form a homology class.

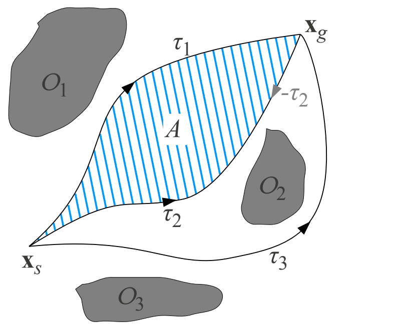
At an intuitive level the above two definitions may appear equivalent. For example, in Figure 3.2(a), τ_1 is homotopic to τ_2 since one can be continuously deformed into the other via a sequence of trajectories marked by the dashed curves. As a consequence, the area swept by this continuous deformation, A, forms a 2-dimensional region in the free configuration space whose boundary is the closed loop $\tau_1 \sqcup -\tau_2$. Indeed, the one-way implication is true as shown below.

Lemma 3.2.3. *If two trajectories are homotopic, they are homologous.*

Proof. This follows directly from the Hurewicz theorem [40] that guarantees the existence of an homomorphism from the homotopy groups to the homology groups of an arbitrary space. ■

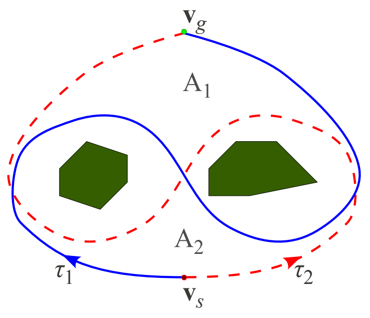


(a) τ_1 is homotopic to τ_2 since there is a continuous sequence of trajectories representing deformation of one into the other. τ_3 belongs to a different homotopy class since it cannot be continuously deformed into any of the other two.

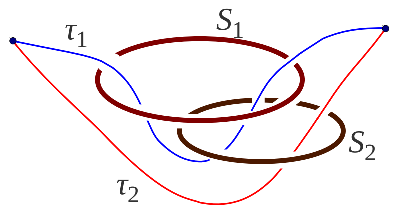


(b) τ_1 is homologous to τ_2 since there exists an area A (shaded region) such that $\tau_1 \sqcup -\tau_2$ is the boundary of A. τ_3 belongs to a different homology class since such an area does not exist between τ_3 and any of the other two trajectories.

Figure 3.2: Illustration of homotopy and homology equivalences. In this example τ_1 and τ_2 are both homotopic as well as homologous.



(a) In 2-dimensions



(b) In 3-dimensions

Figure 3.3: Examples where the trajectories are homologous, but not homotopic

The converse of Lemma 3.2.3, however, does not always hold true. There are subtle difference between homology and homotopy in spite of their similar notions, and one can create examples where two trajectories are not homotopic in spite of being homologous.

Homotopy equivalence arises naturally in many robotics problems. On the other hand, homology is less natural. However it is much simpler to compute homologies. One can establish direct correspondence between homology groups of trajectories and differential 1-forms whose integrals yield homology class invariants for trajectories via the De Rham theorem [8]. Since, according to the discussion of Section 3.1.2 we desire such differential forms, the rest of the paper will be developed with homology classes of trajectories under consideration rather than their homotopy classes. The assumption will be that in many of the practical robotics problems where homotopy classes of trajectories are of greater significance, homology classes of trajectories will serve as a fair analog. We will justify this claim in Section 3.5.1 and through experimental results (Section 3.6).

To clarify the distinction between homotopy equivalence and homology equivalence of trajectories, we present two examples where homology is not same as homotopy. The first example is in 2-dimensions. In Figure 3.3(a) we observe that the trajectories τ_1 and τ_2 are not homotopic, but they are homologous (since their H-signatures, as defined in Section 3.3.2, are equal). This is seen perhaps more easily by considering the interior defined by the union of the areas marked by A_1 and A_2 which indeed forms the boundary for $\tau_1 \sqcup -\tau_2$. In Figure 3.3(b), one can observe that the two trajectories are not homotopic. However, they are homotopic if we only consider S_1 or S_2 but not both. Hence their H-signatures are the same (*i.e.* they are homologous). Thus, if we were exploring different homotopy classes in this environment using the described method, we would be finding one trajectory for these two homotopy classes.

3.3 H-signature in 2-dimensional Euclidean Configuration Space

We consider a 2-dimensional subset of $\mathbb{R}^2$ as the configuration space. The obstacles are thus *punctures* or discontinuities in that subset. The approach for designing a H-signature for such a 2-dimensional configuration space is based on theorems from Complex Analysis, specifically the Cauchy Integral theorem and Residue theorem.

3.3.1 Background: Complex Analysis

Cauchy Integral Theorem

The Cauchy Integral Theorem states that if $f : \mathbb{C} \to \mathbb{C}$ is an holomorphic (analytic) function in some simply connected region $\mathcal{R} \subset \mathbb{C}$, and γ is a closed oriented (i.e. directed) contour completely contained in $\mathcal{R}$, then the following holds,

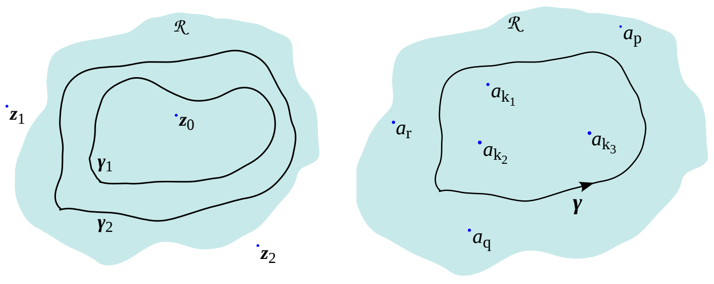
$$\oint_\gamma f(z)\ \mathrm{d}z = 0 \tag{3.3.1}$$

Moreover, if z_0 is a point inside the region enclosed by γ, which has an anti-clockwise (or positive) orientation, then for the function $F(z) = f(z)/(z - z_0)$ with a simple pole at z_0, the following holds

$$\oint_\gamma \frac{f(z)\ \mathrm{d}z}{z - z_0} = 2\pi i f(z_0) \tag{3.3.2}$$

The Residue Theorem

A direct consequence of the Cauchy Integral Theorem, the Residue Theorem, states that, if $F : \mathcal{R} \to \mathbb{C}$ is a function defined in some simply connected region $\mathcal{R} \subset \mathbb{C}$ that has simple poles at the distinct points $a_1, a_2, \cdots, a_M \in \mathcal{R}$, and holomorphic (analytic) everywhere else in $\mathcal{R}$, and say γ is a closed positively oriented Jordan curve completely contained in $\mathcal{R}$ and enclosing only the points



(a) The integrals over contours γ_1 and γ_2 are equal

(b) Only the poles enclosed by γ influence the value of the integral of F.

Figure 3.4: Cauchy Integral Theorem and Residue Theorem

$a_{k_1}, a_{k_1}, \cdots, a_{k_m}$ out of the poles of F, then the following holds,

$$\oint_\gamma F(z)\ \mathrm{d}z = 2\pi i \sum_{l=1}^{m} \lim_{\xi \to a_{k_l}} (\xi - a_{k_l}) F(\xi) \tag{3.3.3}$$

The scenario is illustrated in Figure 3.4(b).

It is important to note that in both the Cauchy Integral Theorem and the Residue Theorem the value of the integrals are independent of the exact choice of the contour γ as long as the mentioned conditions are satisfied (see Figure 3.4(a)).

At this point it is worth mentioning that the Cauchy Integral Theorem and the Residue Theorem are simply particular manifestations of a more general formulation in which one can compute linking numbers between manifolds embedded in Euclidean spaces of arbitrary dimensions. We will in fact discuss this general formulation in Chapter 4, and hence derive the aforesaid theorems as special cases of that formulation.

3.3.2 Designing a H-signature

We exploit the above theorems for designing a differential 1-form that can be used to construct a homology class invariant for 2-dimensional configuration space.

We start by representing the 2-dimensional configuration space as a subset of the complex plane $\mathbb{C}$. Thus a point in the configuration space, $(x, y) \in \mathcal{C}$, is represented as $x + iy \in \mathbb{C}$. The obstacles are assumed to be simply-connected regions in $\mathbb{C}$ and are represented by $\mathcal{O}_1, \mathcal{O}_2, \cdots, \mathcal{O}_N$.

Definition 3.3.1 (Representative points)**.** We define one "representative point" in each connected obstacle such that it lies in the interior of the obstacle. The exact location of the representative points is not of particular significance as long as they each lie inside the respective obstacles. Thus we define the points $\zeta_l \in \mathcal{O}_l,\ \forall l = 1, \cdots, N$. Figure 3.5(a) shows such representative points inside three obstacles.

As we will see in results presented in Section 3.6.1, it is not necessary that we choose representative points for all obstacles. We need to choose such points only on the larger and relevant obstacles that contribute towards the practical notion of homotopy classes. In practical scenarios, for example when constructed from sensors on-board a robot, environments often contain small obstacles and noise. It's important that we choose the ζ_l carefully only inside relevant (large) obstacles which influence our notion of homotopy class of the trajectories. This can be achieved by putting a threshold on the minimum diameter of the obstacles on which we put the *representative points*. Smaller obstacles can be disregarded.

Definition 3.3.2 (Obstacle Marker Function)**.** For a given set of "representative points", we define the "Obstacle Marker Function" function $\mathcal{F} : \mathbb{C} \to \mathbb{C}^N$ as follows,

$$\mathcal{F}(z) = \begin{bmatrix} \frac{f_1(z)}{z-\zeta_1} \\ \frac{f_2(z)}{z-\zeta_2} \\ \vdots \\ \frac{f_N(z)}{z-\zeta_N} \end{bmatrix} \tag{3.3.4}$$

where $f_l,\ l = 1, 2, \cdots, N$ are analytic functions over entire $\mathbb{C}$ such that $f_l(\zeta_l) \neq 0,\ \forall l$. Typical examples of such f_l are polynomials in z.

Thus, $\mathcal{F}$ is a complex vector function, the l^{th} component of which has a single simple pole/singularity at ζ_l.

Definition 3.3.3 (H-signature in 2-dimensional configuration space)**.** For the given configuration space and set of obstacles, we define the *obstacle marker function* as described above, and hence define the H-signature of a trajectory τ the vector function $\mathcal{H}_2 : C_1(\mathbb{C}) \to \mathbb{C}^N$

$$\mathcal{H}_2(\tau) = \int_\tau \mathcal{F}(z)\ \mathrm{d}z$$

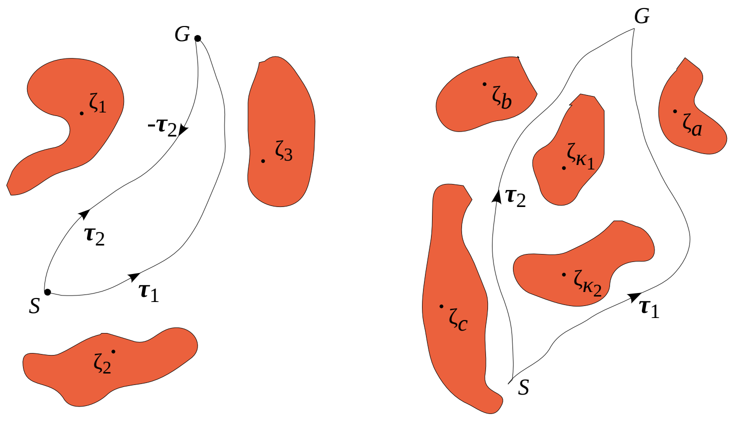
where $C_1(\mathbb{C})$ is the set of all curves/trajectories in $\mathbb{C}$.

Lemma 3.3.4. *Two trajectories τ_1 and τ_2 connecting the same points in the described 2-dimensional configuration space are homologous if and only if $\mathcal{H}_2(\tau_1) = \mathcal{H}_2(\tau_2)$*

Sketch of Proof. We note that by changing the orientation of a path over which an integration is being performed, we change the sign of the integral. If τ is a path, its oppositely oriented path is represented as $-\tau$. Thus, as we see from Figure 3.5(a), τ_1 along with $-\tau_2$ forms a positively oriented closed loop.

If τ_1 and τ_2 are in the same homology class, the area enclosed by τ_1 and τ_2 does not contain any of the "representative points", ζ_i, hence rendering the function $\mathcal{F}$ analytic in that region. Hence from the Cauchy Integral Theorem we obtain,

$$\begin{aligned} & \int_{\tau_1 \sqcup -\tau_2} \mathcal{F}(z)\ \mathrm{d}z = \mathbf{0} \\ \Rightarrow\ & \int_{\tau_1} \mathcal{F}(z)\ \mathrm{d}z + \int_{-\tau_2} \mathcal{F}(z)\ \mathrm{d}z = \mathbf{0} \\ \Rightarrow\ & \int_{\tau_1} \mathcal{F}(z)\ \mathrm{d}z - \int_{\tau_2} \mathcal{F}(z)\ \mathrm{d}z = \mathbf{0} \\ \Rightarrow\ & \int_{\tau_1} \mathcal{F}(z)\ \mathrm{d}z\ =\ \int_{\tau_2} \mathcal{F}(z)\ \mathrm{d}z \end{aligned}$$



(a) In same Homotopy class, forming a closed contour

(b) In different Homotopy classes, enclosing obstacles

Figure 3.5: Two trajectories in same and different homotopy classes

where the **0** in bold implies that it is a N-vetor of zeros.

If τ_1 and τ_2 are in different homology classes, we can easily note that the closed positive contour formed by τ_1 and $-\tau_2$ will enclose one or more of the obstacles, and hence their corresponding "representative points". This is illustrated in Figure 3.5(b). Let us assume that enclosed "representative points" are $\zeta_{\kappa_1}, \zeta_{\kappa_2}, \cdots, \zeta_{\kappa_n}$. Moreover we note that at least one component of the vector function $\mathcal{F}$ has a simple pole at ζ_l for each $l = 1, 2, \cdots, N$. Thus, by the Residue Theorem and Definition 3.3.2,

$$\int_{\tau_1} \mathcal{F}(z)\ \mathrm{d}z \ + \int_{-\tau_2} \mathcal{F}(z)\ \mathrm{d}z \ =$$

$$2\pi i \textstyle\sum_{u=1}^{n} \lim_{\xi\to\zeta_{\kappa_u}} (\xi - \zeta_{\kappa_u}) \begin{bmatrix} \frac{f_1(\xi)}{\xi-\zeta_1} \\ \frac{f_2(\xi)}{\xi-\zeta_2} \\ \vdots \\ \frac{f_N(\xi)}{\xi-\zeta_N} \end{bmatrix}$$

$$\Rightarrow \quad \int_{\tau_1} \mathcal{F}(z)dz - \int_{\tau_2} \mathcal{F}(z)dz \ = \begin{bmatrix} \cdots \\ f_{\kappa_1}(\zeta_{\kappa_1}) \\ \vdots \\ f_{\kappa_2}(\zeta_{\kappa_2}) \\ \vdots \\ f_{\kappa_n}(\zeta_{\kappa_n}) \\ \cdots \end{bmatrix} \neq \mathbf{0}$$

Hence proved. ■

We have hence shown that $\mathcal{H}_2$ gives a *homology invariant* for trajectories in 2-dimensional Euclidean configuration space with obstacles.

3.3.3 Computation for a Line Segment

As discussed earlier in Section 3.1.2, and will be discussed later in Section 3.5, we discretized the given configuration space and create a graph out of it. In many practical implementations we assume that every edge in the graph is a line segment. Thus it is for those line segments that we really need to compute the H-signatures. Thus it is important that we are able to do so efficiently. In this section we will show how to compute the H-signature for a *small* line segment in a 2-dimensional configuration space using a closed-form formula.

Given a line segment e connecting points z_1 and z_2, we can parametrize the segment using the variable $z = (1-\lambda)z_1 + \lambda z_2$, where $\lambda \in [0,1]$ is the parameter. Thus we have,

$$\begin{aligned}\mathcal{H}_2(e) &= \int_e \mathcal{F}(z)\ \mathrm{d}z \\ &= \int_0^1 \mathcal{F}\Big((1-\lambda)z_1 + \lambda z_2\Big)\,(z_2 - z_1)\ \mathrm{d}\lambda \end{aligned} \tag{3.3.5}$$

For computing the H-signature of $e = \{z_1 \to z_2\}$ analytically, we assume that f_l are chosen to be constants. Let $f_l = A_l$ (*const.*) for all $l = 1, 2, \cdots, N$.

Now, a standard integration result gives for the l^{th} component of $\mathcal{H}_2(e)$

$$\begin{aligned}&\int_0^1 \frac{A_l}{(1-\lambda)z_1 + \lambda z_2 - \zeta_l}\,(z_2 - z_1)\ \mathrm{d}\lambda \\ &\qquad = A_l\left(\ln(z_2 - \zeta_i) - \ln(z_1 - \zeta_l)\right)\end{aligned}$$

However we note that the *logarithm of a complex number* does not have an unique value. For any $z' \in \mathbb{C}$, $\ln(z') = \ln(|z'|e^{i\left(\arg(z')+2k\pi\right)}) = \ln(|z'|) + i\left(\arg(z') + 2k\pi\right)$, $\forall k = 0, \pm1, \pm2, \ldots$ (where $\arg(x+iy) = \operatorname{atan2}(y,x)$). Hence, following the assumption that e is a small line segment, we choose the *smallest* of all the possible values over different k's. Thus, the l^{th} component of $\mathcal{H}_2(e)$ is computed as,

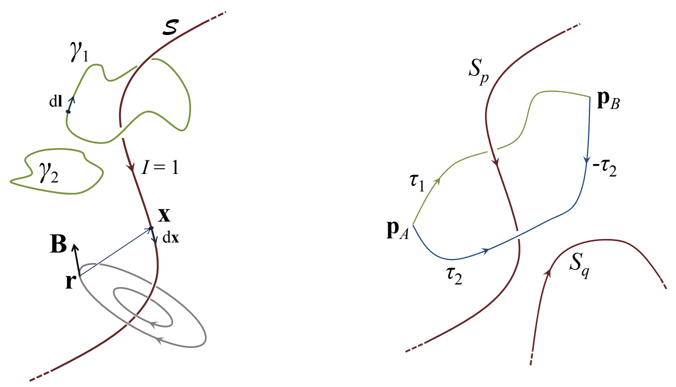
$$\begin{aligned}A_l\Big[&\ln(|z_2 - \zeta_l|) - \ln(|z_1 - \zeta_l|)\ + \\ &i\ \ \mathrm{absmin}_{k\in\mathbb{Z}}\Big(\arg(z_2 - \zeta_l) - \arg(z_1 - \zeta_l) + 2k\pi\Big)\Big]\end{aligned}$$

where $\mathrm{absmin}_{k\in\mathbb{Z}}G(k)$ returns the value of $G(k)$ that has the minimum absolute value (*i.e.* closest to 0) over all $k \in \mathbb{Z}$. Typically, we can do away with checking a few values of k around 0 and picking the local minimum, since the value of $\arg(z_2 - \zeta_l) - \arg(z_1 - \zeta_l) + 2k\pi$ is monotonic in k.

3.4 H-signature in 3-dimensional Euclidean Configuration Space

While in the two-dimensional case, theoretically any finite obstacle on the plane can induce multiple homotopy and homology classes for trajectories joining two points, the notion of homotopy/homology classes in three dimensions can only be induced by obstacles with *genus* [1] one or more, or with obstacles stretching to infinity. For example, a torus-shaped obstacle in a three-dimensional environment creates two primary homotopy classes, which can be informally described as: *i.* The trajectories passing through the "hole" of the torus, and *ii.* the trajectories passing

[1]The *genus* of an obstacle refers to the number of *handles* [63]).



(a) Magnetic field due to current in $\mathcal{S}$, & its integration along closed loop γ_i.

(b) 2 trajectories, τ_1 & τ_2, connecting the same points form a closed loop.

Figure 3.6: Application of Biot-Savart law and Ampere's law to robot path planning with topological constraints in 3-D.

outside the "hole" of the torus. Figure 3.7 shows some examples of obstacles that can or cannot induce such classes for trajectories. A sphere or a solid cube, for example, cannot induce multiple homotopy classes in an environment.

3.4.1 Background: Electromagnetism

Biot-Savart Law

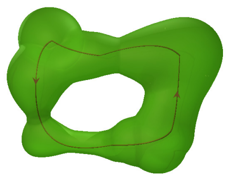
Consider a single hypothetical current-carrying *curve* (a current conducting wire) embedded in a 3-dimensional space carrying a steady current of unit magnitude (Figure 3.6(a)). There is no source for the current nor any sink - only a steady flow persisting inside the conductor due to absence of any dissipation. It is to be noted that such a steady current is possible iff the *curve* is closed (or open, but extending to infinity, where we close the curve using a loop at infinity. See Figure 3.8(a) and Construction 3.4.3). We denote the curve by $\mathcal{S}$. Then, according to the Biot-Savart Law [36], the magnetic field $\mathbf{B}$ at any arbitrary point $\mathbf{r}$ in the space, due to the current flow in $\mathcal{S}$, is given by,

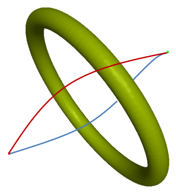
$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \int_{\mathcal{S}} \frac{(\mathbf{x} - \mathbf{r}) \times \, \mathrm{d}\mathbf{x}}{\|\mathbf{x} - \mathbf{r}\|^3} \tag{3.4.1}$$

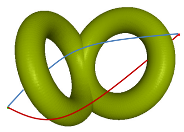
where, $\mathbf{x}$, the integration variable, represents the coordinate of a point on $\mathcal{S}$, and $\mathrm{d}\mathbf{x}$ is an infinitesimal element on $\mathcal{S}$ along the direction of the current flow.

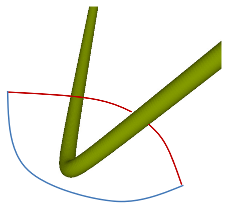
Ampere's Law

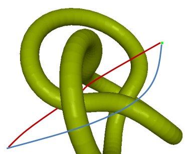
While Biot-Savart law gives a recipe for computing the magnetic field from a given current configuration, Ampere's Law [36], in a sense, gives the inverse of it. Given the magnetic field $\mathbf{B}$ at every point in the space, and a closed loop γ (Figure 3.6(a)), the line integral of $\mathbf{B}$ along γ gives the


(a) Skeleton of a generic genus 1 obstacle is modeled as a current-carrying conductor.


(b) A torus-shaped genus 1 obstacle.


(c) A genus 2 obstacle.


(d) An infinite tube is a genus 1 obstacle.


(e) A knot-shaped obstacle with genus 1.

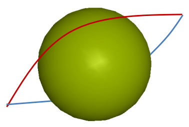

(f) A sphere **does not** induce homotopy classes and has genus 0.

Figure 3.7: Examples of obstacles in 3-D. (a-e) induce homotopy classes, (f) does not.

current *enclosed* by the loop γ. That is,

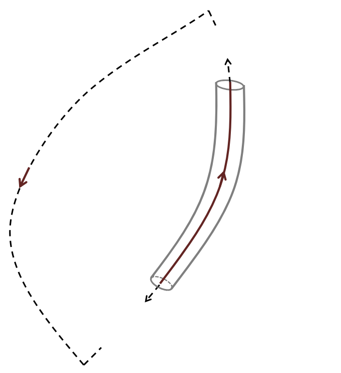
$$\Xi(\mathcal{C}) := \int_\gamma \mathbf{B}(\mathbf{l}) \cdot \ \mathrm{d}\mathbf{l} = \mu_0 I_{encl} \tag{3.4.2}$$

where, $\mathbf{l}$, the integration variable, represents the coordinate of a point on γ, and $\mathrm{d}\mathbf{l}$ is an infinitesimal element on $\mathcal{C}$.

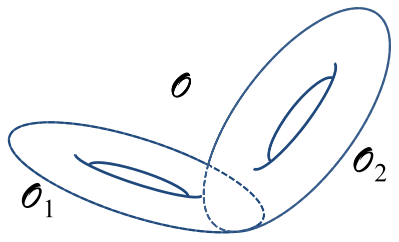
In Biot-Savart Law and Ampere's Law one can conveniently choose the constant μ_0 to be equal to 1 by proper choice of units. Moreover, by choice, the value of the current flowing in the conductor is unity. Thus, for any closed loop γ, the value of $\Xi(\gamma)$ is zero iff γ does not enclose the conductor, otherwise it is ± 1 (the sign depends on the direction of integration performed on γ). Thus in Figure 3.6(a), $\Xi(\gamma_1) = 1$ and $\Xi(\gamma_2) = 0$.

Once again, we would like to emphasize that the above mentioned laws, though appearing naturally in physical systems, are simply special cases of a general formulation in algebraic topology that we will discuss in Chapter 4. We will also explicitly derive these laws from that general formulation.

Definition 3.4.1 (Simple Homotopy-Inducing Obstacle in 3-dimensional Configuration Space)**.** A *Simple Homotopy-inducing Obstacle* (SHIO) is a bounded obstacle of *genus* 1, for example a torus (Figure 3.7(a), 3.7(b)) or a knot (Figure 3.7(e)).



(a) An unbounded obstacle and its skeleton can be closed at a large distance to create a *closed loop*.



(b) An obstacle with genus 2, $\mathcal{O}$, can be decomposed into 2 obstacles, each with genus one, $\mathcal{O}_1$ and $\mathcal{O}_2$.

Figure 3.8: Illustration of Constructions 3.4.3 and 3.4.4.

3.4.2 Designing a H-signature

For the 2-dimensional case, each obstacle on the plane that induces the notion of multiple homotopy classes was assigned a *representative point*. Analogously, for the 3-dimensional case, we need to define a *skeleton* for every SHIO. Intuitively, a skeleton of a 3-dimensional obstacle is a 1-dimensional curve that is completely contained inside the obstacle such that the surface of the obstacle can be "shrunk" onto the skeleton in a continuous fashion without altering the topology of the surface of the obstacle. Formally, we define the skeleton of an obstacle in terms of *deformation retract* [40].

Definition 3.4.2 (Skeleton)**.** A 1-dimensional manifold, S, is called a *skeleton* of a SHIO, $\mathcal{O}$, iff S is homeomorphic to $\mathbb{S}^1$ (a circle), S is completely contained inside $\mathcal{O}$, and if S is a *deformation retract* (Definition 2.1.5) of $\mathcal{O}$.

Thus, the fact that τ_1 and τ_2 are of the same or of different homotopy/homology classes is not altered by replacing $\mathcal{O}$ by S.

In the literature, algorithms for constructing skeletons of solid objects is a well-studied [6, 46]. However in the present context we have a much relaxed notion of skeleton. While we can adopt any of the different existing algorithms for automated construction of skeleton from a 3-dimensional obstacles, this discussion is out of the scope of the present work. Figure 3.7(a) demonstrates skeletons for several genus 1 obstacles.

Conversion of Generic Obstacles into SHIOs

Given a set of obstacles in a three-dimensional environment, we perform the following two constructions/reduction on the obstacles so that the only kind of obstacle we have in the environment are *Simple Homotopy-Inducing Obstacles*. The Construction 3.4.3 is mostly trivial in the sense that it can be easily automated for arbitrary obstacles. Construction 3.4.4 on the other hand is linked with the construction of *skeleton* of the obstacles (Definition 3.4.2).

Construction 3.4.3. Closing infinite, unbounded obstacles In most of the problems that we are concerned with, the domain in which the trajectories of the robots lie is finite and bounded. This gives us the freedom of altering/modifying the obstacles or parts of obstacles lying outside that domain without altering the problem. One consequence of this freedom is that we can *close* infinite and unbounded obstacles (*e.g.* Figure 3.7(d)) at a large distance from the domain of interest (Figure 3.8(a)).

Construction 3.4.4. Decomposing obstacles with genus > 1 After closing all infinite, unbounded obstacles in an environment according to Construction 3.4.3, if there is an obstacle with genus k (*e.g.* Figure 3.7(c)), we can decomposed/split it into k obstacles, possibly overlapping and touching each other, but each with genus 1 (Figure 3.8(b)). This does not change the obstacles or the problem in any way. This construction just changes the way we identify obstacles and construct their skeletons. For example in Figure 3.8(b) the original obstacle $\mathcal{O}$ with genus 2 is realized as two obstacles $\mathcal{O}_1$ and $\mathcal{O}_2$, each with genus 1 and overlapping each other. The decomposition of obstacles into SHIOs allows us define k skeletons for each obstacle of genus k and simplify computations of h-signatures of trajectories.

Note that neither of these constructions effect the original problem or the result. In the results presented in this thesis we do both constructions manually.

Skeleton of SHIOs as Current Carrying Curves for H-signature Construction

Construction 3.4.5. Modeling *skeleton* of a SHIO as a current carrying manifold Given m obstacles in an environment, $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m$, with genus $k_1, k_2, \ldots, k_m$ respectively, we can construct $M = k_1 + k_2 + \cdots + k_m$ skeletons from M SHIOs (obtained using Constructions 3.4.3 and 3.4.4), namely $S_1, S_2, \ldots, S_M$. Each S_i is a closed, connected, boundary-less 1-dimensional manifold. We model each of them as a current-carrying conductor carrying current of unit magnitude (Figures 3.7(a), 3.8(a)). The direction of the currents is not of importance, but by convention, each is of unit magnitude.

Definition 3.4.6 (Virtual Magnetic Field due to a Skeleton)**.** Given S_i, the skeletons of a Simple Homotopy-Inducing Obstacle, we define a *Virtual Magnetic Field vector* at a point $\mathbf{r}$ in the space due to the current in S_i using Biot-Savart Law as follows,

$$\mathbf{B}_i(\mathbf{r}) = \frac{1}{4\pi} \int_{S_i} \frac{(\mathbf{x} - \mathbf{r}) \times \, \mathrm{d}\mathbf{x}}{\|\mathbf{x} - \mathbf{r}\|^3} \tag{3.4.3}$$

where, $\mathbf{x}$, the integration variable, represents the coordinates of a point on S_i, and $\mathrm{d}\mathbf{x}$ is an infinitesimal element on S_i along the chosen direction of the current flow in S_i.

Definition 3.4.7 (H-signature in 3-dimensional Configuration Space)**.** Given an arbitrary trajectory, τ, in the 3-dimensional environment with M skeletons, we define the *H-signature* of τ to be the function $\mathcal{H}_3 : C_1(\mathbb{R}^3) \to \mathbb{R}^M$,

$$\mathcal{H}_3(\tau) = [h_1(\tau),\ h_2(\tau),\ \ldots,\ h_M(\tau)]^T \tag{3.4.4}$$

where, $C_1(\mathbb{R}^3)$ is the space of all curves/trajectories in $\mathbb{R}^3$, and

$$h_i(\tau) = \int_\tau \mathbf{B}_i(\mathbf{l}) \cdot \, \mathrm{d}\mathbf{l} \tag{3.4.5}$$

is defined in an analogous manner as the integral in Ampere's Law. In defining h_i, $\mathbf{B}_i$ is the *Virtual Magnetic Field* vector due to the unit current through skeleton S_i, $\mathbf{l}$ is the integration variable that represents the coordinate of a point on τ, and $\mathrm{d}\mathbf{l}$ is an infinitesimal element on τ.

Lemma 3.4.8. *Two trajectories τ_1 and τ_2 connecting the same points in the described 3-dimensional configuration space are homologous if and only if $\mathcal{H}_3(\tau_1) = \mathcal{H}_3(\tau_2)$.*

Sketch of Proof. Since τ_1 and τ_2 connect the same points, $\tau_1 \sqcup -\tau_2$, *i.e.* τ_1 and $-\tau_2$ together (where $-\tau$ indicates the same curve as τ, but with the opposite orientation) form a closed loop in the 3-dimensional environment (Figure 3.6(b)). We replace the obstacles $\mathcal{O}_1, \mathcal{O}_2, \ldots, \mathcal{O}_m$ in the environments with the skeletons $S_1, S_2, \ldots, S_M$.

Consider the presence of just the skeleton S_i. By the direct consequence of Ampere's Law and our construction in which a unit current flows through S_i, the value of

$$h_i(\tau_1 \sqcup -\tau_2) = \int_{\tau_1 \sqcup -\tau_2} \mathbf{B}_i(\mathbf{l}) \cdot \, \mathrm{d}\mathbf{l}$$

is non-zero if and only if the closed loop formed by $\tau_1 \sqcup -\tau_2$ encloses the current carrying conductor S_i (*i.e.* there does not exist a surface not intersecting S_i, the boundary of which is $\tau_1 \sqcup -\tau_2$). For example, in Figure 3.6(b), $h_p(\tau_1 \sqcup -\tau_2) = 1$ and $h_q(\tau_1 \sqcup -\tau_2) = 0$. Now, by the definition of line integration we have the following identity,

$$\begin{aligned} h_i(\tau_1 \sqcup -\tau_2) &= \textstyle\int_{\tau_1 \sqcup -\tau_2} \mathbf{B}_i(\mathbf{l}) \cdot \, \mathrm{d}\mathbf{l} \\ = \textstyle\int_{\tau_1} \mathbf{B}_i(\mathbf{l}) \cdot d\mathbf{l} - \int_{\tau_2} \mathbf{B}_i(\mathbf{l}) \cdot \, \mathrm{d}\mathbf{l} &= h_i(\tau_1) - h_i(\tau_2) \end{aligned} \tag{3.4.6}$$

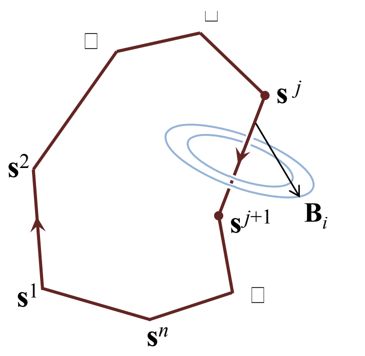
Thus, $h_i(\tau_1) = h_i(\tau_2)$ if and only if the closed loop formed by τ_1 and τ_2 does not enclose S_i (*i.e.* homologous in presence of S_i).

Now in presence of skeletons $S_1, S_2, \ldots, S_M$ the same argument extends for each skeleton individually. Thus τ_1 and τ_2 are homologous if an only if $\mathcal{H}_3(\tau_1) = \mathcal{H}_3(\tau_2)$. ■

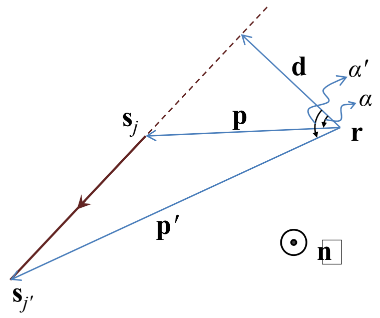
Hence we have shown that the proposed formula for H-signature is a homology class invariant for trajectories in 3-D.

3.4.3 Computation for a Line Segment

Once again, we are interested in efficient computation of the H-signature for small line segments since those are the ones that will make up edges of the graph formed by discretization of the environment. For all practical applications we assume that a skeleton of an obstacle, S_i, is made up of finite number (n_i) of line segments: $S_i = \{\overrightarrow{\mathbf{s}_i^1\mathbf{s}_i^2}, \ \overrightarrow{\mathbf{s}_i^2\mathbf{s}_i^3}, \ \ldots, \ \overrightarrow{\mathbf{s}_i^{n_i-1}\mathbf{s}_i^{n_i}}, \ \overrightarrow{\mathbf{s}_i^{n_i}\mathbf{s}_i^1}\}$ (Figure 3.9(a)).



(a) A skeleton of an obstacle can be constructed/approximated so that it is made up of n line segments.



(b) Magnetic field at $\mathbf{r}$ due to the current in a line segment $\overline{\mathbf{s}_i^j\mathbf{s}_i^{j'}}$.

Figure 3.9: Closed-form computation of magnetic field.

Thus, the integration of equation (3.4.3) can be split into summation of n_i integrations,

$$\mathbf{B}_i(\mathbf{r}) = \frac{1}{4\pi}\sum_{j=1}^{n_i}\int_{\overrightarrow{\mathbf{s}_i^j\mathbf{s}_i^{j'}}} \frac{(\mathbf{x}-\mathbf{r})\times\ \mathrm{d}\mathbf{x}}{\|\mathbf{x}-\mathbf{r}\|^3} \tag{3.4.7}$$

where $j' \equiv 1 + (j \mod n_i)$. It is to be notes that a skeleton of an unbounded obstacle created from Construction 3.4.3 can be made up of finite and few line segments. The only feature of such a skeleton might be that some of the points that make up the line segments ($\mathbf{s}_i^j$) might be located at a large distance from the domain of interest, which is used to close the skeleton.

One advantage of this representation of skeletons is that for the straight line segments, $\overrightarrow{\mathbf{s}_i^j\mathbf{s}_i^{j'}}$, the integration can be computed analytically. Specifically, using a result from [36] (also, see Figure 3.9(b)),

$$\begin{aligned}\int_{\overrightarrow{\mathbf{s}_i^j\mathbf{s}_i^{j'}}} \frac{(\mathbf{x}-\mathbf{r})\times\ \mathrm{d}\mathbf{x}}{\|\mathbf{x}-\mathbf{r}\|^3} &= \frac{1}{\|\mathbf{d}\|}\left(\sin(\alpha')-\sin(\alpha)\right)\hat{\mathbf{n}} \\ &= \frac{1}{\|\mathbf{d}\|^2}\left(\frac{\mathbf{d}\times\mathbf{p}'}{\|\mathbf{p}'\|} - \frac{\mathbf{d}\times\mathbf{p}}{\|\mathbf{p}\|}\right)\end{aligned} \tag{3.4.8}$$

where, $\mathbf{d}, \mathbf{p}$ and $\mathbf{p}'$ are functions of $\mathbf{s}_i^j, \mathbf{s}_i^{j'}$ and $\mathbf{r}$ (Figure 3.9(b)), and can be expressed as,

$$\mathbf{p}=\mathbf{s}_i^j-\mathbf{r},\quad \mathbf{p}'=\mathbf{s}_i^{j'}-\mathbf{r},\quad \mathbf{d}=\frac{(\mathbf{s}_i^{j'}-\mathbf{s}_i^j)\times(\mathbf{p}\times\mathbf{p}')}{\|\mathbf{s}_i^{j'}-\mathbf{s}_i^j\|^2} \tag{3.4.9}$$

We define and write $\mathbf{\Phi}(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, \mathbf{r})$ for the RHS of Equation (3.4.8) for notational convenience. Thus we have,

$$\mathbf{B}_i(\mathbf{r}) = \frac{1}{4\pi}\sum_{j=1}^{n_i} \mathbf{\Phi}(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, \mathbf{r}) \tag{3.4.10}$$

where, $j' \equiv 1 + (j \mod n_i)$.

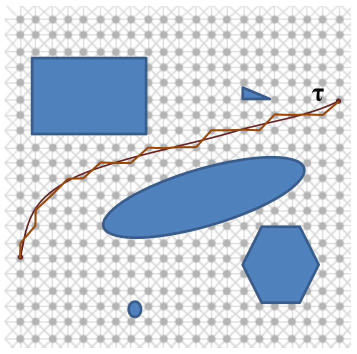


Figure 3.10: A trajectory in the original configuration space is represented by a path in the discrete graph.

Given a small line segment, e, we can now compute the H-signature, $\mathcal{H}(e) = [h_1(e),\ h_2(e),\ \ldots,\ h_M(e)]^T$, where,

$$h_i(e) = \frac{1}{4\pi} \int_e \sum_{j=1}^{n_i} \mathbf{\Phi}(\mathbf{s}_i^j, \mathbf{s}_i^{j'}, \mathbf{l}) \cdot \ \mathrm{d}\mathbf{l} \tag{3.4.11}$$

can be computed numerically. For the numerical integration, in all our experimental results we used the GSL (GNU Scientific Library), which has a highly efficient implementation of adaptive integration algorithms with desired precision. We used a cache for storing the H-signature of edges that has been computed in order to avoid re-computation.

3.5 H-signature Augmented Graph

Once we have the means of computing H-signature for each edge (small line segments), we introduce the concept of *H-signature augmented graph*. Typically, a graph, $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$, is created for the purpose of graph-search based planning by discretization of an environment, placing a vertex at each discretized cell, and by connecting the neighboring cells with edges (See Figure 1.4(a) for an example in 2-dimensional configuration space). Paths in such a graph represent trajectories in the original configuration space (Figure 3.10). However, it should be kept in mind that this representation is approximate, and any arbitrary trajectory in the continuous configuration space will not be faithfully represented at scales smaller than the discretization size.

In the following discussion we perform a construction using $\mathcal{G}$, without distinguishing between 2 and 3-dimensional configuration spaces explicitly, once we have discretized the environment, and perform a general treatment with the graph $\mathcal{G}$. For H-signature trajectories or line segments we use the generic function $\mathcal{H}$, which we understand to be $\mathcal{H}_2$ or $\mathcal{H}_3$ depending on the dimensionality of the configuration space.

Let $\mathbf{v}_s$ be the start coordinate in the configuration space, and $\mathbf{v}_g$ be the goal coordinate (by the boldface $\mathbf{v}$'s, with a slight abuse of notation, we will indicate both the vertex in the graph as well as the coordinate of the vertex in the original configuration space). By Lemma 3.3.4 or 3.4.8, any two trajectories from $\mathbf{v}_s$ to $\mathbf{v}$ that belong to the same homology class will have the same

H-signature. The H-signature can assume different, but discrete values corresponding on the class of the trajectory. We also write $\mathcal{P}(\mathbf{v}_s, \mathbf{v})$ to denote the set of all trajectories from $\mathbf{v}_s$ to $\mathbf{v}$, and $\widetilde{\mathbf{v}_s\mathbf{v}} \in \mathcal{P}(\mathbf{v}_s, \mathbf{v})$ to denote a particular trajectory in that set.

Definition 3.5.1 (Allowed and Blocked Homology Classes)**.** Suppose it is required that we restrict all our search for trajectories connecting $\mathbf{v}_s$ and $\mathbf{v}_g$ to certain homology classes, or not allow some other. We denote the set of allowed H-signatures of trajectories leading up to $\mathbf{v}_g$ by the set $\mathcal{A}$, and the set of blocked H-signatures as $\mathcal{B}$. $\mathcal{A}$ and $\mathcal{B}$ are essentially complement of each other ($\mathcal{A} \cup \mathcal{B} = \mathcal{U}$, where the universal set, $\mathcal{U}$, is the set of the H-signatures of all the classes of trajectories joining $\mathbf{v}_s$ and $\mathbf{v}_g$), and $\mathcal{B}$ can be an empty set when all classes are allowed.

Following the discussion in Appendix A, for the 3-dimensional configuration space it is also possible to restrict search to *non-looping* trajectories by putting all h-signatures that have at least one element outside $(-1, 1)$ into the set $\mathcal{B}$.

We define the *H-signature augmented graph* of $\mathcal{G}$ as the graph $\mathcal{G}_H(\mathcal{G}) = \{\mathcal{V}_H, \mathcal{E}_H\}$, such that each node in this new graph has the H-signature of a trajectory leading up to the coordinate of the node from $\mathbf{v}_s$ appended to it. That is, each node in this augmented graph is given by $\{\mathbf{v}, \mathcal{H}(\widetilde{\mathbf{v}_s\mathbf{v}})\}$, for some $\widetilde{\mathbf{v}_s\mathbf{v}} \in \mathcal{P}(\mathbf{v}_s, \mathbf{v})$. Thus, corresponding to a given $\mathbf{v} \in \mathcal{V}$, since there are discrete homology classes of trajectories from $\mathbf{v}_s$ to $\mathbf{v}$, there are a discrete number of the augmented states, $\{\mathbf{v}, \mathbf{h}\} \in \mathcal{V}_H$, where $\mathbf{h}$ is a M-vector (M being the number of *representative points* or the number of *SHIOs* depending on whether it's a 2 or 3-dimensional configuration space) and assumes the values of the H-signatures corresponding to the discrete homology classes. Thus, we define the *H-signature augmented graph* of $\mathcal{G}$ as follows,

$$\mathcal{G}_H = \{\mathcal{V}_H, \mathcal{E}_H\}$$

where,

1. $$\mathcal{V}_H = \left\{ \{\mathbf{v}, \mathbf{h}\} \;\middle|\; \begin{array}{l} \mathbf{v} \in \mathcal{V}, \text{ and,} \\ \mathbf{h} = \mathcal{H}(\widetilde{\mathbf{v}_s\mathbf{v}}) \text{ for some trajectory} \\ \qquad\qquad \widetilde{\mathbf{v}_s\mathbf{v}} \in \mathcal{P}(\mathbf{v}_s, \mathbf{v}), \text{ and,} \\ \mathbf{h} \in \mathcal{A} \text{ (equivalently, } \mathbf{h} \notin \mathcal{B}) \\ \qquad\qquad\qquad \text{when } \mathbf{v} = \mathbf{v}_g \end{array} \right\}$$

2. An edge $\{\{\mathbf{v}, \mathbf{h}\} \to \{\mathbf{v}', \mathbf{h}'\}\}$ is in $\mathcal{E}_H$ for $\{\mathbf{v}, \mathbf{h}\} \in \mathcal{V}_H$ and $\{\mathbf{v}', \mathbf{h}'\} \in \mathcal{V}_H$, iff
   i. The edge $\{\mathbf{v} \to \mathbf{v}'\} \in \mathcal{E}$, and,
   ii. $\mathbf{h}' = \mathbf{h} + \mathcal{H}(\mathbf{v} \to \mathbf{v}')$, where, $\mathcal{H}(\mathbf{v} \to \mathbf{v}')$ is the *H-signature* of the edge $\{\mathbf{v} \to \mathbf{v}'\} \in \mathcal{E}$.

3. The cost/weight associated with an edge $\{\{\mathbf{v}, \mathbf{h}\} \to \{\mathbf{v}', \mathbf{h}'\}\}$ is same as that associated with edge $\{\mathbf{v} \to \mathbf{v}'\} \in \mathcal{E}$.

The consequence of point 3 in the above definition is that an *admissible heuristics* for search in $\mathcal{G}$ will remain admissible in $\mathcal{G}_H$. That is, if $f(\mathbf{v}, \mathbf{v}_g)$ was the heuristic function in $\mathcal{G}$, we define $f_H(\{\mathbf{v}, \mathbf{h}\}, \{\mathbf{v}_g, \mathbf{h}'\}) = f(\mathbf{v}, \mathbf{v}_g)$ as the heuristic function in $\mathcal{G}_H$ for any $\mathbf{h}' \in \mathcal{A}$.

The consequence of augmenting each node of $\mathcal{G}$ with a H-signature is that now nodes are distinguished not only by their coordinates, but also the H-signature of the trajectory followed

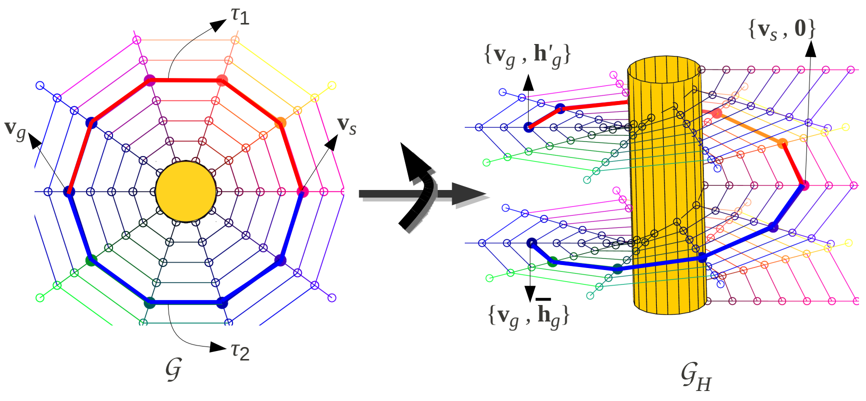

τ_1
$\mathbf{v}_g$
$\mathbf{v}_s$
τ_2
$\mathcal{G}$
$\{\mathbf{v}_g, \mathbf{h}'_g\}$
$\{\mathbf{v}_s, \mathbf{0}\}$
$\{\mathbf{v}_g, \overline{\mathbf{h}}_g\}$
$\mathcal{G}_H$

Figure 3.11: The topology of the augmented graph, $\mathcal{G}_H$ (right), compared against $\mathcal{G}$ (left), for a cylindrically discretized 2-dimensional configuration space around a circular obstacle

to reach it. Typically we use graph search algorithms like A* (or variants like D* or D*-lite) where nodes in the graph $\mathcal{G}_H$ are expanded starting from the node $\{\mathbf{v}_s, \mathbf{0}\}$ (where by $\mathbf{0}$ we mean a M-dimensional vector of zeros).

The topology of this augmented graph for a 2-dimensional case is illustrated in Figure 3.11. A goal state $\mathbf{v}_g$ is the same in $\mathcal{G}$ irrespective for the path (τ_1 or τ_2) taken to reach it. Whereas in the H-signature augmented graph, the states are differentiated by the additional value of $\mathbf{h}_g$. We can perform a graph search in the augmented graph, $\mathcal{G}_H$, using any standard graph search algorithm starting from the state $\{\mathbf{v}_s, \mathbf{0}\}$. The goal state (i.e. the state, upon expansion of which we stop the graph search) is potentially any of the states $\{\mathbf{v}_g, \mathbf{h}_g\}$ for any $\mathbf{h}_g \in \mathcal{A}$ (or $\mathbf{h}_g \notin \mathcal{B}$ if $\mathcal{B}$ is provided instead of $\mathcal{A}$). We can use the same heuristic that we would have used for searching in $\mathcal{G}$, i.e. $f_H(\mathbf{v}, \mathbf{h}) = f(\mathbf{v})$. It is to be noted that $\mathcal{G}_H$ is essentially an infinite graph, even if $\mathcal{G}$ is finite. However the search algorithm needs to expand only a finite number of states. Since for a given $\mathbf{v}$, the states $\{\mathbf{v}, \mathbf{h}\}$ can assume some discrete values of $\mathbf{h}$ (corresponding to the different homology classes). To determine if $\{\mathbf{v}, \mathbf{h}'_g\}$ and $\{\mathbf{v}, \overline{\mathbf{h}}_g\}$ are the same states, we can simply compare the values of $\mathbf{h}'_g$ and $\overline{\mathbf{h}}_g$.

3.5.1 Uses of the H-signature Augmented Graph

There are primarily two distinct but related ways we would like to use the H-signature augmented graph with search algorithms:

i. *Exploration of environment for different homotopy classes of trajectories connecting* $\mathbf{v}_s$ *and* $\mathbf{v}_g$*:* For this problem, whenever we expand a state $\{\mathbf{v}_g, \tilde{\mathbf{h}}\} \in \mathcal{V}_H$, for some $\tilde{\mathbf{h}} \notin \mathcal{B}$, we store the path up to that node, and continue expanding more states until the desired number of classes are explored. Although H-signature is a homology class invariant, and not a homotopy class invariant, by Lemma 3.2.3, two trajectories are homotopic implies that they are homologous. Thus, two trajectories that are homotopic will be in the same homology class, and hence their H-signatures will be the same. Thus, in such problems where we find least cost trajectories with different H-signatures in a configuration space using the said method, we are always guaranteed to obtain trajectories in distinct homotopy classes as well.

ii. *Planning with H-signature constraint:* For searches with H-signature constraint, we stop upon expansion of a goal coordinate $\{\mathbf{v}_g, \tilde{\mathbf{h}}\}$ for some $\tilde{\mathbf{h}} \notin \mathcal{B}$ (or equivalently, $\tilde{\mathbf{h}} \in \mathcal{A}$).

3.5.2 Theoretical Analysis

Theorem 3.5.2. *If $\mathbf{P}_H^* = \{\{\mathbf{v}_1, \mathbf{h}_1\}, \{\mathbf{v}_2, \mathbf{h}_2\}, \cdots, \{\mathbf{v}_p, \mathbf{h}_p\}\}$ is an optimal path in $\mathcal{G}_H$, then the path $\mathbf{P}^* = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p\}$ is an optimal path in the graph $\mathcal{G}$ satisfying the* H*-signature constraints specified by $\mathcal{A}$ and $\mathcal{B}$*

Proof. By construction of $\mathcal{G}_H$, the path $\{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p\}$ satisfies the given H-signature constraints. Moreover by definition, $\mathbf{P}_H^*$ is a minimum cost path in $\mathcal{G}_H$. Since the cost function in $\mathcal{G}_H$ is the same as the one in $\mathcal{G}$ and does not involve $\mathbf{h}_j$, it follows that the projection of $\mathbf{P}_H^*$ on $\mathcal{G}$ given by $\mathbf{P}^* = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_p\}$ is an optimal path in the graph $\mathcal{G}$ satisfying the constraints defined in $\mathcal{G}_H$. ■

3.6 Results

The method described in this paper was implemented in C++ and MATLAB. In the sections below we present results in 2 and 3-dimensional configuration spaces.

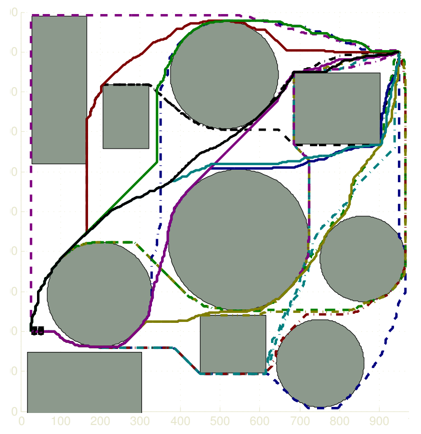
3.6.1 Two-dimensional Configuration Space

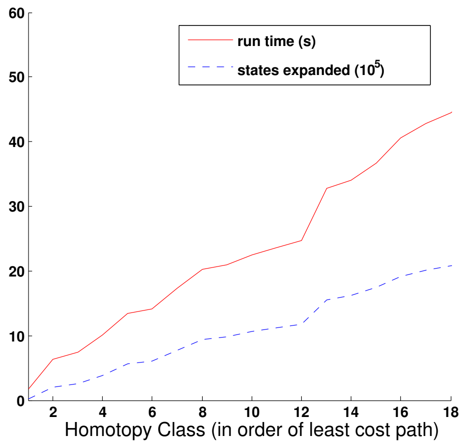
Path Prediction by Homotopy Class Exploration

Figure 3.12(a) shows a large 1000×1000 discretized environment with circular and rectangular obstacles. We explore trajectories in different classes in order of their path costs using the method 'i.' described in Section 3.5.1. The implementation was done in C++ running on an Intel Core 2 Duo processor with 2.1 GHz clock-speed and 4GB RAM. All the different trajectories in different homotopy classes were determined in a single run of graph search on $\mathcal{G}_H$ as described earlier. Figure 3.13 shows similar exploration of multiple homotopy classes in a smaller environment.

As discussed earlier, in such exploration problems, although we use the H-signature as the class invariants in the search algorithms, since non-homologous trajectories are guaranteed to be non-homotopic, we are guaranteed to obtain trajectories in different homotopy classes.

We also constructed 10 such environments using random circular and rectangular obstacles. Table 3.1 demonstrate the efficiency of the searches. The time indicates the cumulative time during the search until a shortest-path trajectory in a particular homotopy class is found. This is relevant to problems of tracking dynamic entities, such as people, where one often needs to predict possible paths in order to bias the tracker or to deal with occlusion by anticipating where the dynamic entity will appear. Since people can choose different paths to their destinations, we need to be able to predict least cost paths that lie in different homotopy classes.


(a) Paths in 20 different homotopy classes

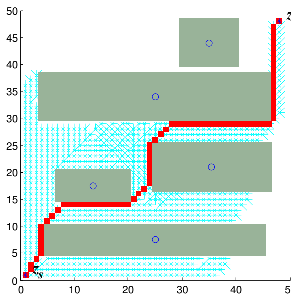


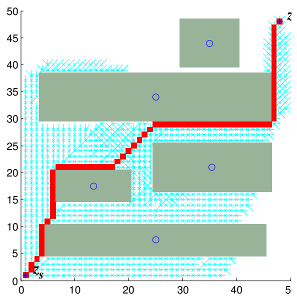
(b) Run-time & states expanded for finding the least-cost paths in a particular run

Figure 3.12: Exploring homotopy classes in 1000×1000 discretized environments to find least cost paths in each

Hmtp. class (i)	time ellapsed until i^{th} hmtp. class explored (s)			states expanded cumulative (10^6)		
	min	max	mean	min	max	mean
1	1.41	2.01	1.71	0.021	0.039	0.032
2	3.58	8.58	5.15	0.099	0.313	0.170
3	5.09	9.69	6.77	0.180	0.375	0.244
4	6.13	12.46	8.92	0.237	0.494	0.345
5	7.80	17.74	11.50	0.285	0.776	0.472
6	10.53	18.56	13.05	0.422	0.825	0.555
7	10.92	19.74	15.38	0.473	0.888	0.681
8	13.35	20.32	17.01	0.604	0.935	0.773
9	15.08	21.76	18.60	0.693	1.027	0.858
10	15.53	26.28	20.87	0.720	1.252	0.978

Table 3.1: Statistics of searching least-cost paths in first 10 homotopy classes in 10 randomly generated environments. The numbers represent the cumulative values till the i^{th} homotopy class is explored.


(a)


(b)

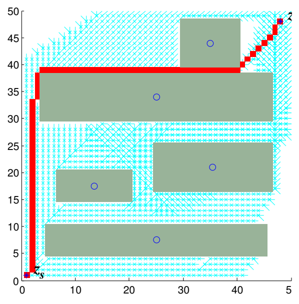
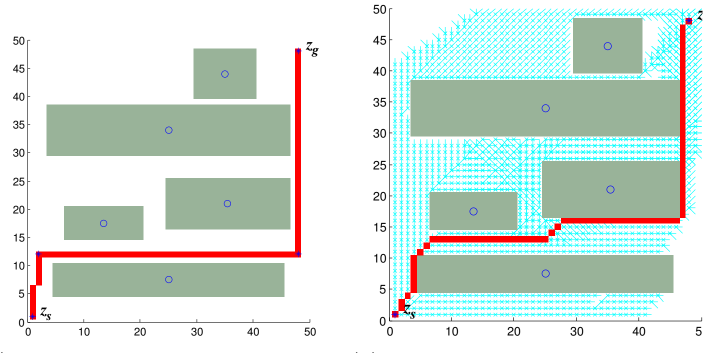

(c)

Figure 3.13: Exploring homotopy classes by blocking the class obtained from previous search. The blue shaded region shows the projection of the nodes expanded in $\mathcal{G}_H$ on to $\mathcal{G}$.



(a) Suboptimal key-point generated trajectory. (b) Optimal trajectory in same class as key-point generated trajectory.

Figure 3.14: Homotopy class constraint determined using suboptimal key-point generated trajectory.

H-signature Constraint: H-signature Defined by Key-points

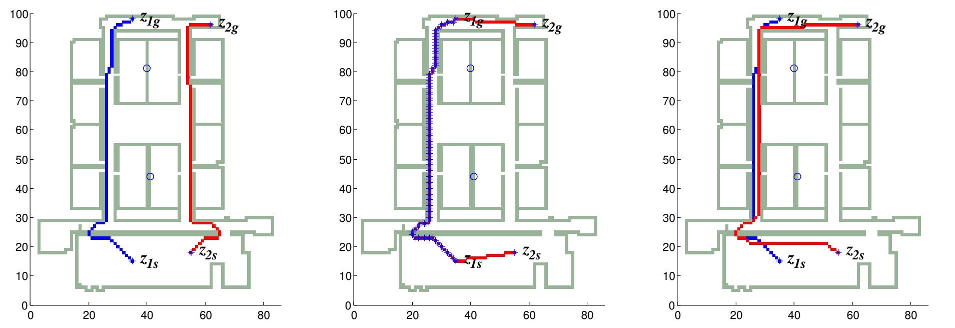
Figure 3.14(a) demonstrates an example where we define homology classes using a sample (suboptimal) trajectory specified by key-points. One can then compute the H-signature for such a trajectory. It can then be used to search $\mathcal{G}_H$ for an optimal path in the same class (or different) as the sample trajectory (Figure 3.14(b)).

Although technically we have imposed homology class constraint by imposing the H-signature constraint, we observe that the optimal trajectory that we obtain is in fact in the same homotopy class as the key-point generated trajectory. In fact we observe that in most robotics planning problems imposing H-signature constraints indeed impose the corresponding homotopy class constraint as well.

Multiple Robot Visibility Problem

The problem of path planning for multiple robots with visibility constraints can also make use of our approach. If one robot needs to plan its path such that it is never obstructed from the view of another robot by some obstacle, we can apply the technique of planning with H-signature constraint to obtain the desired trajectories. In Figure 3.15(a)-(c) two robots plan trajectories to their respective goals. The robot on the right needs to plan a trajectory such that it is in the "visibility" of the robot on the left, whose trajectory is given. Thus, in order to determine the H-signature of the desired homotology class it first constructs a suboptimal path by connecting its own start and goal points to the start and goal of the left robot, such that the trajectory of the left robot is completely contained in it (Figure 3.15(b))) as key points. The H-signature of this path gives the desired homology class, thus re-planning with that class as the only allowed class gives the desired optimal plan (Figure 3.15(c)).

The natural constraint in this situation is that of homotopy. But we once again observe that even imposing the H-signature constraint we do obtain trajectory in the desired homotopy class.

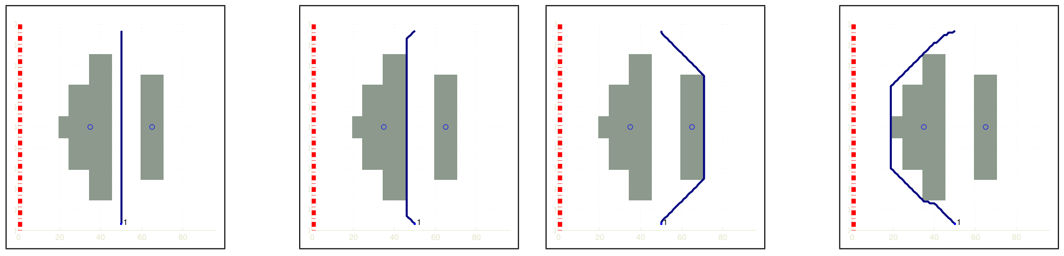


(a) Unconstrained plans of two robots.

(b) Suboptimal path constructed for robot on the right – used to determine H-signature of desired homology class.

(c) Optimal plan with *visibility constraint* satisfied.

Figure 3.15: 100×100 discretized environment with 2 *representative points* on the central large connected walls.

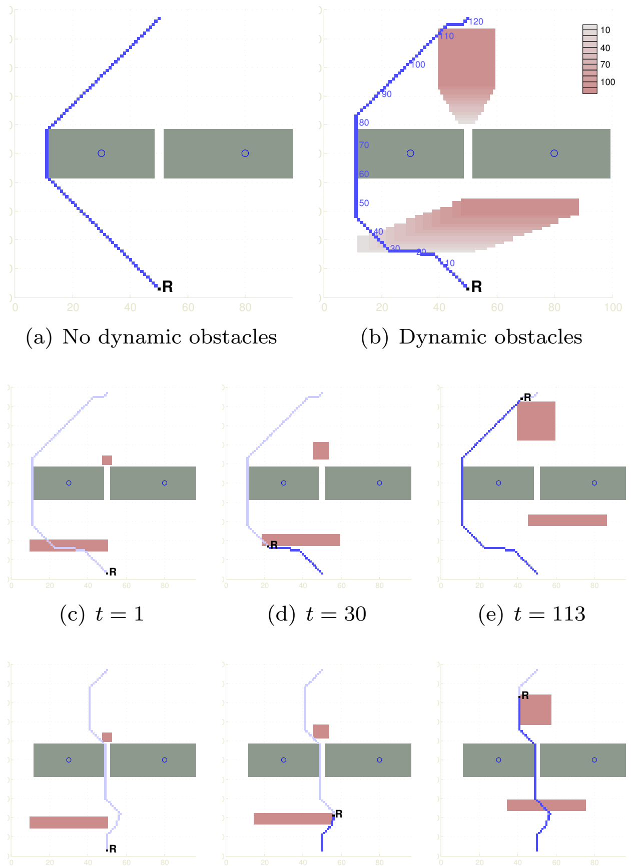


(a) $w = 0.0$, $\mathcal{B} = \{\}$ (b) $w = 0.01$, $\mathcal{B} = \{\}$ (c) $w = 0.0$, $\mathcal{B} = \{\mathbf{h}_0\}$ (d) $w = 0.01$, $\mathcal{B} = \{\mathbf{h}_0\}$

Figure 3.16: Planning with non-Euclidean length as cost as well as homotopy class constraint

Arbitrary Cost Functions

Our method is not limited to Euclidean length cost functions. It can deal with arbitrary cost functions. For example, in Figure 3.16 there are two large obstacles and a *communication base* to the left of the environment marked by the bold dotted line, $x = 0$. An agent is supposed to plan its path from the bottom to the top of the environment, while minimizing a weighted sum of the length of the trajectory and the distance of the trajectory from the *communication base*. Thus, in this case, besides the transition costs of the states in $\mathcal{G}$, each state, $z = x + iy \in \mathcal{G}$, is assigned a cost $w \cdot x$, the penalty on separation from the *communication base*. Thus the net penalized cost of the trajectory, τ, that is being minimized is of the form $c = \int_\tau ds + w \int_\tau x(s) ds$, where x is the x-coordinate of the points on the trajectory, parametrized by s, the length of the trajectory. The trajectories in figures 3.16(a) and (b) with penalty weights $w = 0$ and $w = 0.01$ respectively have H-signature of $\mathbf{h}_0$. Blocking this class, but having a small penalty over distance from *communication base* gives the trajectory in 3.16(d) that passes close to the *communication base*.



(a) No dynamic obstacles (b) Dynamic obstacles

(c) $t = 1$ (d) $t = 30$ (e) $t = 113$

(f) $t = 1$ (g) $t = 19$ (h) $t = 81$

Figure 3.17: Planning with time as an additional coordinate. The postion of the agent is denoted by **R** in figures (c)-(h).

Planning with Additional Coordinates

In Figure 3.17, besides x and y, we have used *time* as a third coordinate in $\mathcal{G}$. There are two dynamic obstacles in the environment - the one at the bottom only translates, while one near the top both translates as well as expands in size. We have two *representative points* on the two static obstacles. Figure 3.17(a) shows the solution upon blocking the first homotopy class in the environment without the dynamic obstacles. Figure 3.17(b) shows the planned trajectory in the environment with dynamic obstacles (with the color intensity representing the time coordinate). Figure 3.17(c-e) show the execution of the trajectory at different instants of time of the same. Figure 3.17(g-h) show the execution of the plan without H-signature constraint.

However note that in this example the representative points are on the static obstacles. In spite of having a third coordinate, the H-signature of trajectories are being computed on a 2-dimensional plane for the projection of the trajectories on to the plane.

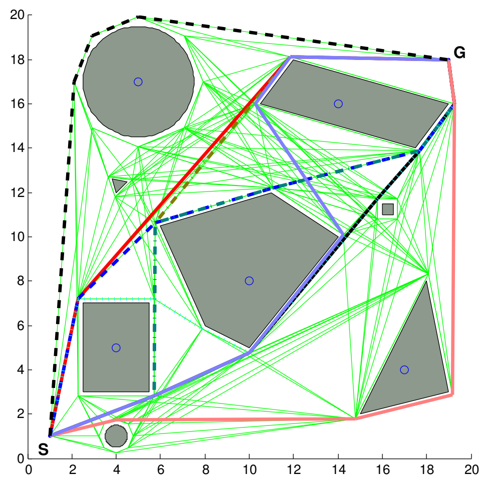


Figure 3.18: Exploring homotopy classes using a Visibility Graph

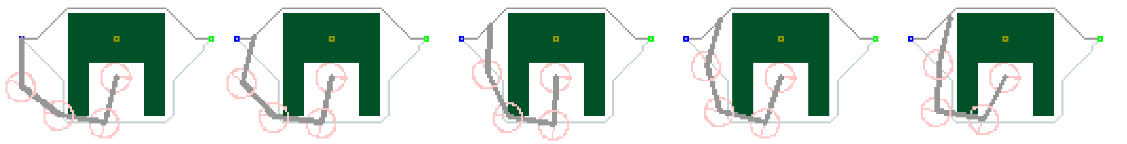
Implementation Using Visibility Graph

To demonstrate the versatility of the proposed algorithm we implemented it using a Visibility Graph as the state graph, $\mathcal{G}$. Figure 3.18 shows the visibility graph generated in an environment with polygonal obstacles and the shortest paths in 9 homotopy classes that we explore. Obstacles were *inflated* in order to incorporate collision safety and circular obstacles were approximated by polygons. *Representative points* were placed only on the large obstacles (determined by threshold on diameter and marked by blue circles in the figure) and visibility graph was constructed. A* search was used for searching the H-signatue augmented graph. The implementation was made in MATLAB. The average run-time of the search until the 9^{th} homotopy class was explored was 0.4 seconds and about 100 states were expanded.

Application to Robust and Efficient Path Planning for Robotic Arm End-effector

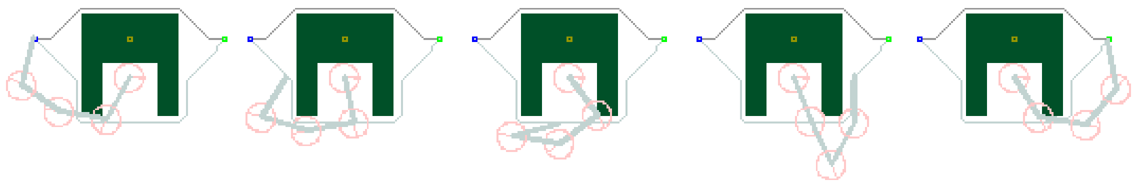
Typically, planning a trajectory in the joint space of a n-link robot arm, following which end-effector can reach the goal is quite expensive. The search needs to be performed in the n-dimensional configuration space of the arm.

However, one alternative approach is to plan a trajectory in the end-effector space (the 2-dimensional plane for a planar arm) and try to incrementally move the arm in small steps (by solving small local optimization problems or by using a simple feedback controller) to make the end effector follow the trajectory while respecting constraints. However, this approach can often fail (especially in presence of obstacles) since the trajectory may be such that the end-effector may not actually be able to traverse it due to joint angle constraints or the fact the total length of the arm is finite. We try to make this alternate approach more robust as follows: Instead of planning a single end-effector trajectory we plan multiple of them in different homotopy classes. We then simulate the arm and try to make the end-effector follow each trajectory incrementally (by solving small local optimization problems at each time step). The hope is that if the robot arm can reach the goal, at least one of those trajectories will be good for doing so. While we do not have any rigorous



(a) $t = 0s$ (b) $t = 6s$ (c) $t = 11s$ (d) $t = 14s$ (e) $t = 16s$

Figure 3.19: The arm end effector fails to follow the shorter trajectory due to limited length of the arm.



(a) $t = 0s$ (b) $t = 15s$ (c) $t = 49s$ (d) $t = 91s$ (e) $t = 106s$

Figure 3.20: The longer trajectory can be followed by the end effector.

theoretical guarantees for this approach, we will exemplify the concept using two scenarios.

In Figures 3.19 and 3.20 the end effector of a 4-link robot arm is required to follow a trajectory from the start coordinate at the left to the goal on the right. The base of the arm is fixed inside the U-shaped obstacle. As described earlier, we do not plan path in the 4-dimensional configuration space of the arm. Instead we plan trajectories in the 2-dimensional plane of the end effector's position. The arm cannot follow the shortest path (the one that goes above the obstacle) because of the limited length of the arm. This is illustrated in figures 3.19(a)-(e). However the path in the other homotopy class, though longer, can be followed by the arm (figures 3.20(a)-(e)).

In Figures 3.21 and 3.22 the 8-link robot arm has joint angle constraints (the angle limits are marked by pink sectors). In particular, all joints, except the one at the base can assume angles in $[-\pi/2, 0]$. The shortest path is the one that goes to the right of the obstacle (darker one). However, the robot arm cannot reach the goal following any path in this homotopy class due to the joint angle limitations, and thus it fails (Figure 3.21). However the longer trajectory in the other homotopy class can be followed, and hence the arm succeeds in following it (Figure 3.22).

The advantage of this method becomes apparent only when we consider robot arms with large number of links. Global path planning in such high dimensional configuration space by discretization and graph construction is very expensive. On the other hand, knowing the end-effector trajectory, incremental planning for the joint angles using local gradient-based search is almost indifferent to the dimensionality of the complete configuration space of the arm. Moreover, such local searches are free from most of the problems associated with continuous motion planning approaches (e.g. getting stuck at local minima, slow convergence or divergence). While we don't yet have a theoretical guarantee from this approach, the computational advantage is definitely significant.

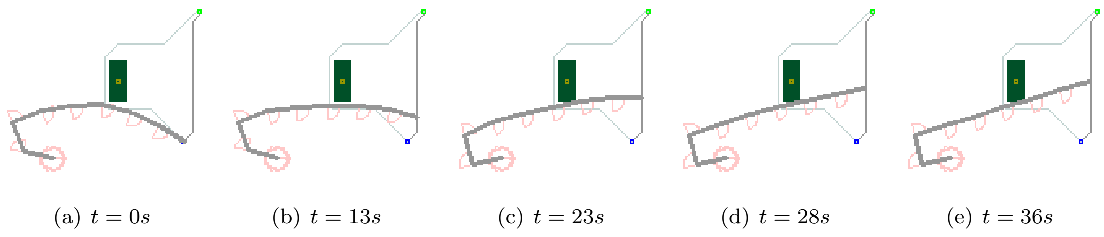


Figure 3.21: The arm end effector fails to follow the shorter trajectory due to joint angle limits.

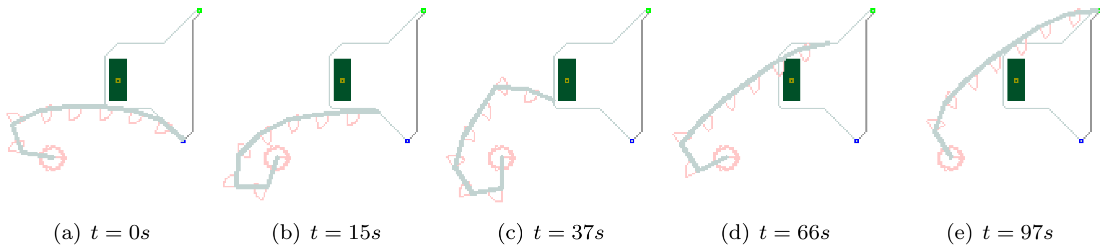


Figure 3.22: The arm end effector succeeds in following the longer trajectory.

3.6.2 Three-dimensional Configuration Space

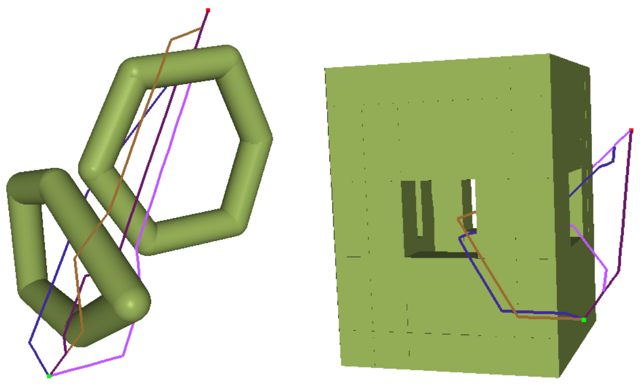
The first 3-dimensional domain in which we implement the planning algorithm is the space of 3 spatial dimensions, X, Y and Z. We also demonstrate the algorithm in the 3-dimensional space of X, Y and *time*, *i.e.* an environment with planar dynamic obstacles (Section 3.6.2).

For a problem in 3 spatial dimensions, the domain of interest is bounded by upper and lower limits of the 3 coordinates. The domain is then uniformly discretized into cubic cells and a node of $\mathcal{G}$ is placed at the center of each cell. Connectivity is established between a node and its 26 neighbors (all cells that share at least one corner, edge or face with it). Each edge is bi-directional and its cost is the Euclidean length.

Simple environments with Bounded Obstacles

Figure 3.23(a) demonstrates a simple environment, $20 \times 20 \times 18$ discretized, with two *torus-shaped* obstacles. The skeleton of each obstacle is made up of line segments passing through the central axis of the cylindrical segments. Here we restrict search to *non-looping* trajectories (See Appendix A for a precise definition). That is, we set $\mathcal{B} = \left\{ \mathbf{h} = [h_1, h_2]^T \mid |h_1| > 1 \text{ or } |h_2| > 1 \right\}$. We search for 4 homotopy classes of trajectories connecting a given start and goal coordinate. As shown in Figure 3.23(a), the algorithm finds four such trajectories: (i) going through hoops 1 and 2; (ii) going through hoop 1 but not through hoop 2; (iii) going through hoop 2 but not through hoop 1; and (iv) not going through either hoops. According to Theorem 3.5.2 each path is the least cost one in the graph and in its respective homotopy class.

Figure 3.23(b) shows the exploration of 4 homotopy classes in and around a room with windows on each wall. The skeletons for this obstacle are defined as loops around each window according to Construction 3.4.4. The trivial shortest path from the given start to goal configuration goes outside the room (the dark violet trajectory). Trajectories in other homotopy classes pass through



(a) Two hoops. (b) A room with windows.

Figure 3.23: Exploring homotopy classes in $X - Y - Z$ space.

the room.

Environment with Unbounded Pipes

Figure 3.24(a) shows a more complex environment consisting of 7 pipes stretching to infinity. The workspace of choice is $44 \times 44 \times 44$ discretized, with the start and goal coordinates at two opposite corners of the discretized space. We used Construction 3.4.3 to close the inbounded obstacles at infinity. In Figure 3.24(a) we find the least cost paths in 10 different homotopy classes.

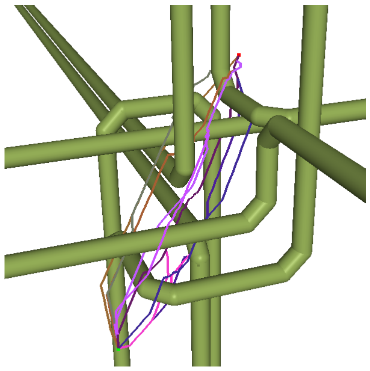
Planning with H-signature Constraint

Figure 3.24(b) demonstrates a planning problem with H-signature constraint. The darker trajectory is the global least cost path found from a search in $\mathcal{G}$ for the given start and goal coordinates. The H-signature for that trajectory was computed, and hence we computed the signature of the *complementary class* (*i.e* the class corresponding to the trajectory that passes on the *other side* of every SHIO - see Appendix A for a precise definition), and put only that in $\mathcal{A}$. The lighter trajectory is the one planned with that $\mathcal{A}$ as the set of allowed H-signature. This trajectory goes on the *opposite side* of each and every pipe in the environment as compared to the darker trajectory.

We note that in this example the notion of *complementary homology class* concurs with that of *complementary homotopy class.*

Search Speed and Efficiency

We now present the running time for the case in Figure 3.24(a). The environment, as described earlier, is $44 \times 44 \times 44$ discretized, and hence $\mathcal{G}$ contains 85184 nodes. Due to each node being connected to 26 of its neighbors, there are almost 13 times as many edges in $\mathcal{G}$. The program was run on a Intel Core 2 Duo processor with 2.1 GHz clock-speed and 3GB RAM. We first compute the values of $\mathcal{H}(e)$ for all edges $e \in \mathcal{E}$ and store them in a cache, which takes about $2273s$. Then we perform the A* search in $\mathcal{G}_H$, using the values from the cache whenever required. By doing so we eliminate the requirement of re-computing the h-signatures of the edges every time we perform


(a) Exploring 10 distinct homotopy classes.

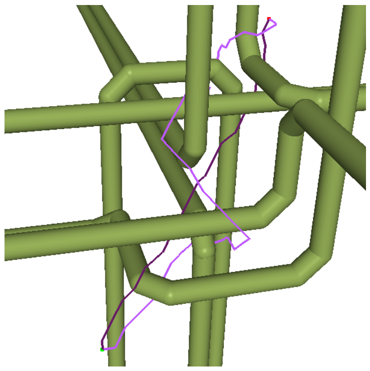

(b) Plan in the *complementary homology class* of the least cost path.

Figure 3.24: An environment with 7 unbounded pipes.

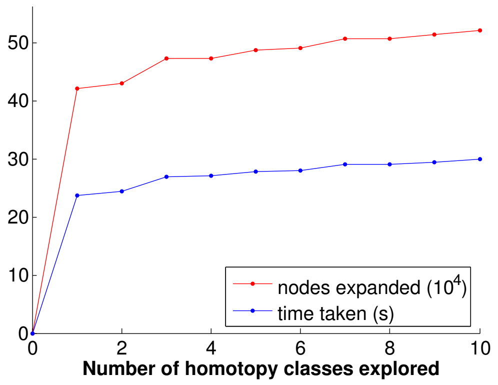
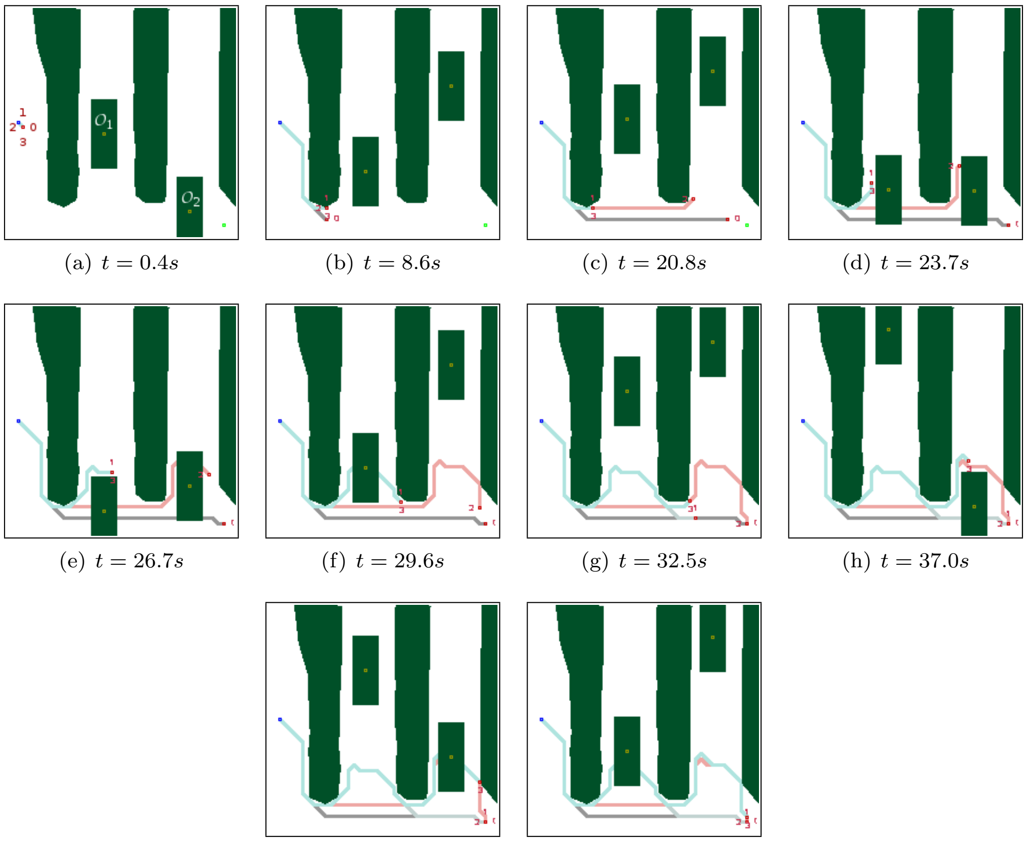


Figure 3.25: Cumulative time taken and number of states expanded while searching $\mathcal{G}_H$ for 10 homotopy classes in the problem of Figure 3.24(a).

a search, even with changed start and goal coordinates. The search for the 10 homotopy classes in Figure 3.24(a) took about $30s$ and expansion of 521692 nodes in $\mathcal{G}_H$. Figure 3.25 shows the cumulative time required and the number of nodes in $\mathcal{G}_H$ expanded.

Planning in 2-dimensional Plane with Moving Obstacles

The next 3-dimensional domain that we experiment with is that of the two-dimensional plane, but with dynamic entities. Thus the variables of interest are X, Y and *time*. The node set was formed by uniform discretization of the domain of interest. The connectivity of the graph is such that the *time* variable can increase only in the positive direction (each node connected to 9 neighboring nodes in next time step, including the same x & y). The cost of an edge, e, with differences in the coordinates of its end points $\Delta x, \Delta y$ and Δt is computed as $c(e) = \sqrt{\Delta x^2 + \Delta y^2 + \epsilon \Delta t^2}$, where ϵ is a small value for avoiding *zero cost edges* in $\mathcal{G}_H$. The skeleton of the moving obstacles are the curves traced by their centers (yellow dots on the oscillating rectangles in Online Resource 1) in



(a) $t = 0.4s$ (b) $t = 8.6s$ (c) $t = 20.8s$ (d) $t = 23.7s$

(e) $t = 26.7s$ (f) $t = 29.6s$ (g) $t = 32.5s$ (h) $t = 37.0s$

(i) $t = 40.0s$ (j) $t = 43.1s$

Figure 3.26: Screen-shots from an example with two moving obstacles ($\mathcal{O}_1$ and $\mathcal{O}_2$) showing the exploration of 4 homotopy classes in a dynamic environment. The blue trajectory (3) passes above both $\mathcal{O}_1$ and $\mathcal{O}_2$. The red trajectory (2) passes above $\mathcal{O}_2$, but not $\mathcal{O}_1$. The light blue-gray trajectory (1) passes above $\mathcal{O}_1$, but not $\mathcal{O}_2$. The dark gray trajectory (0) is the trivial shortest path.

the $X - Y - Time$ space. The skeletons are closed outside and far from the discretized domain (Construction 3.4.3). Note that in doing so, segments of the skeleton may point along negative time. However that does not effect the planning since the $X - Y - Time$ space itself can be treated no differently from $\mathbb{R}^3$.

Figure 3.26 shows the exploration of 4 homotopy classes in $X - Y - Time$ domain. The environment is 40×40 discretized in X and Y directions, and have 100 discretization cells in time. There are two dynamic rectangular obstacles, that undergo a known oscillatory motion inside a narrow passage between other static obstacles. The 4 different trajectories in the different homotopy classes are marked by different colors as well as different numbers at their current locations. The blue trajectory (3) passes above both the obstacles. The red trajectory (2) passes above the right obstacles, but not the left one. The light blue-gray trajectory (1) passes above the obstacle on the left, but not one on the right. The dark gray trajectory (0) is the trivial shortest path. The trajectories in the non-trivial homotopy classes *go behind* the obstacles, a region that would otherwise not be visited by the least cost path without any H-signature consideration.

Chapter 4

Identification of Homology Classes in Euclidean Spaces with Punctures

4.1 Introduction

The methods developed in the previous chapter relied on finding a differential 1-form, the integration of which along trajectories would give homology class invariants. Such 1-forms are elements of the de Rham cohomology group of the configuration space punctured by obstacles, $H^1_{dR}(\mathbb{R}^D - \mathcal{O})$ [8]. For D-dimensional configuration space we considered $(D-2)$-dimensional *homotopy equivalents* (which were also their *deformation retracts*) to represent the obstacle – the *representative points* for the 2-dimensional case, and the *skeletons* for the 3-dimensional case. What we then ended up computing are *linking numbers* of closed loops with the homotopy equivalents of each obstacle. Depending on whether or not the closed loop $\tau_1 \sqcup -\tau_2$ formed by two trajectories τ_1 and τ_2 has zero or non-zero linking numbers with every obstacle, we could conclude if or not they are homologous (Definition 3.2.2).

In fact it is not a surprise that linking numbers are closely related to homology groups. Using the definition of linking number from [27] we will in fact show that the integration along trajectories we compute give homology class invariants for closed loops (something that we had claimed in Chapter 3, but have not proved rigorously).

We will generalize the problem in order to take into account homology classes and linking numbers of arbitrary dimensional manifolds (not just 1-dimensional curves representing trajectories). In particular, we will consider $(N-1)$-dimensional closed manifolds as generalization of 1-dimensional curves that constituted the trajectories. Obstacles will be represented by $(D-N)$-dimensional closed manifolds (which, in many cases will be deformation retracts of the original obstacles). Thus, in light of the contents of Chapter 3, we would set $N = 2$ for trajectories, and for the 2 and 3 dimensional cases we would have $D = 2$ and $D = 3$ respectively.

The aim of this chapter are:

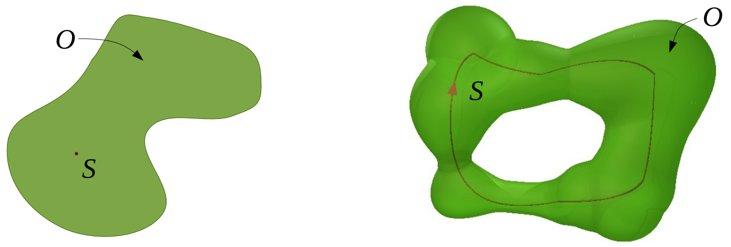


Figure 4.1: Obstacles, O, can be replaced by their equivalents, S, and that will not alter the homology class of the $(N-1)$-cycles in the complement space – an assumption that we made in Chapter 3. In either of these figures, S is a deformation retract of O. The justification for this construction is a consequence of Corollary 4.2.2.

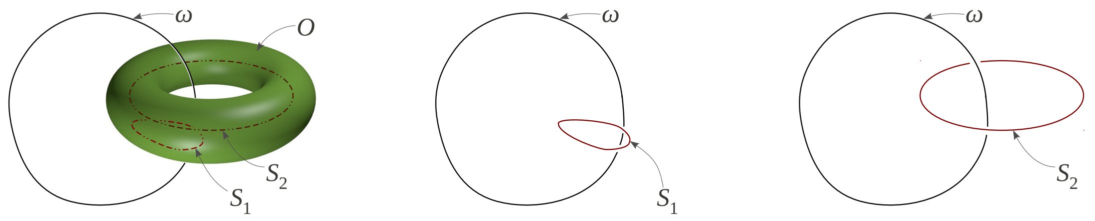
1. To find certain differential $(N-1)$-forms in the Euclidean space punctured by obstacles, and show that integration of the forms along $(N-1)$-dimensional closed manifolds give *complete invariants* for homology classes of the manifolds in the punctured space ($i.e.$ the value of the integral over two closed manifolds are equal if and only if the manifolds are homologous),
2. To generalize the tools used in Chapter 3 to arbitrary dimensional Euclidean configuration spaces punctured by obstacles.

Throughout this chapter we consider homology and cohomology with coefficients in the field $\mathbb{R}$. Also, for simplicity, we will throughout consider $N > 1$ to avoid the special treatment of the 0^{th} (co)homology groups.

4.2 Simplifying the Problem by Taking $(D-N)$-dimensional Equivalents of Obstacles

Before we delve into some of the technical details involving linking numbers, we state a few propositions related to replacement of obstacles with their $(D-N)$-dimensional representatives. The fact that we had used *representative points* to replace obstacles in 2-dimensions and *skeletons* in 3-dimensions to perform computations throughout Chapter 3 relied on the fact that replacing obstacle by their homotopy equivalents do not effect the homology groups of the complement space ($i.e.$ free space), neither does it change the homology classes of trajectories in the free space (Figure 4.1). While this assumption may appear intuitive, we need to prove it rigorously. Moreover, often one may come across obstacles which do not have a $(D-N)$-dimensional deformation retract ($e.g.$ for the $D=3, N=2$ case, a hollow torus does not have a $D-N=1$ dimensional homotopy equivalent). We will explore what $(D-N)$-dimensional equivalents we can use for such obstacles (Figure 4.3).

In the proposition and related corollaries that follow, we represent the configuration space (without obstacles) by $\mathbb{R}^D$ (the D-dimensional configuration space), an obstacle by O, and S the equivalent of the obstacle with which we replace O for computational simplicity.



(a) Both S_1 and S_2 are subsets of the solid torus, O. Moreover, each has the homotopy type of the solid torus. ω is a non-trivial cycle in $(\mathbb{R}^3 - O)$.

(b) $(\mathbb{R}^3 - S_1)$ has homology groups isomorphic to those of $(\mathbb{R}^3 - O)$. However, the cycle ω becomes trivial in $(\mathbb{R}^3 - S_1)$. Thus S_1 is not a valid replacement of O.

(c) $(\mathbb{R}^3 - S_2)$ also has homology groups isomorphic to those of $(\mathbb{R}^3 - O)$. Moreover, the cycle ω remain non-trivial in $(\mathbb{R}^3 - S_2)$. S_2 is a valid replacement of O.

Figure 4.2: A solid torus, and its valid/invalid replacements. This is an example with $D = 3, N = 2$ – the 3-dimensional case discussed in Chapter 3. The obstacle, O, need to be replaced by appropriate $(D - N) = 1$-dimensional equivalents, that we called *skeletons*. The replacement needs to be such that the inclusion map $\bar{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - S)$ induces the isomorphism.

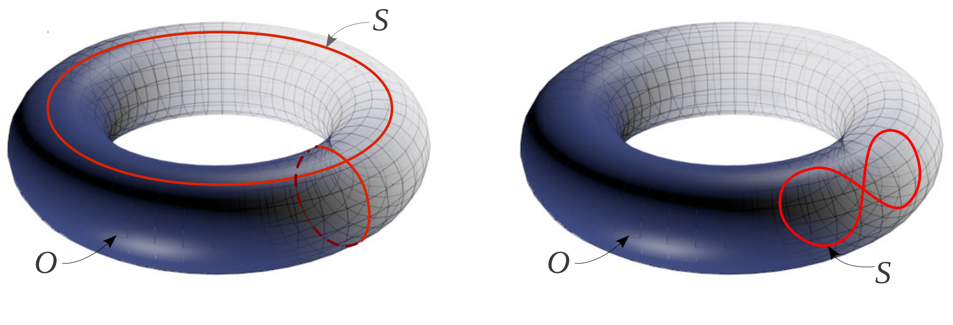
Proposition 4.2.1. *Let O be a compact, locally contractible and orientable sub-manifolds of $\mathbb{R}^D$. Let S be a $(D - N)$-dimensional compact, locally contractible and orientable sub-manifolds of O contained in the interior of O such that $H_{D-N}(S) \approxeq H_{D-N}(O)$ and $H_{D-N}(O, S) \approxeq 0$. Then the inclusion map $\bar{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - S)$ induces an isomorphism $\bar{i}_{*:N-1} : H_{N-1}(\mathbb{R}^D - O) \to H_{N-1}(\mathbb{R}^D - S)$.*

Proof. Detailed proof can be found in Appendix B.1. ■

In light of robot path planning, O in the above proposition are the solid obstacles in the environment, and S are their equivalents/replacements (in terminology of Chapter 3 they are *representative points* of obstacles on a 2-dimensional plane, and *skeletons* of obstacles in a 3-dimensional Euclidean space). The aim of the above proposition is to establish a relationship between the homology groups of the complement (or free) spaces, $(\mathbb{R}^D - O)$ and $(\mathbb{R}^D - S)$, from some known relationship between the spaces O and S. In fact, it is not just the homology groups that we are trying to establish relationship for, but homology classes of $(N - 1)$-dimensional manifolds (the *closed trajectories* in robot planning problem) in the complement space.

For example, in Figure 4.2, the solid torus, which represents an obstacle in $\mathbb{R}^3$, was previously (in Chapter 3) replaced by its 'skeleton', S_2, for computations of homology class of closed loops like ω (formed by pair of trajectories). Under such a replacement, the claim was that, the homology class of ω in the complement space remain unchanged. This replacement, according to Proposition 4.2.1, is justified by the facts that $H_1(S_2) \approxeq H_1(O)$ and $H_1(O, S_2) \approxeq 0$ (due to the fact that S_2 is a deformation retract of O). Corollary 4.2.2 simply asserts that a deformation retract (which S_2 indeed is of O) in fact satisfies the required conditions of the proposition. On the other hand, although S_1 satisfies $H_1(S_1) \approxeq H_1(O)$ it does not satisfy $H_1(O, S_1) \approxeq 0$. Thus S_1 is not a valid replacement of O.

Corollary 4.2.2. *If S and O are compact, locally contractible and orientable submanifolds of $\mathbb{R}^D$ such that S is a deformation retract of O, then the inclusion map $\bar{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - S)$ induces*



(a) The hollow torus can be replaced by the image of its generating 1-cycles, S. This replacement does not alter the $(N-1)^{th}$ homology groups of the complement space, neither does it alter the class of a $(N-1)$-cycle in the complement space, $(\mathbb{R}^D - O)$. O and S satisfy the conditions of Proposition 4.2.1.

(b) In this figure we choose a $S \subset O$ such that $H_{D-N}(S) \approxeq H_{D-N}(O)$. The replacement of O by S, as in (a), does not alter the $(N-1)^{th}$ homology groups of the complement space, $(\mathbb{R}^D - O)$. However, the replacement does not preserve the class of a $(N-1)$-cycle in the complement space. This is because $H_{D-N}(O,S) \not\approxeq 0$ in this case. Thus this is **not** a valid replacement of O.

Figure 4.3: A hollow (or thickened) torus as an obstacle in a $D=3$ dimensional space, with $N=2$ for our problem (*i.e.* we are interested in homology classes of $(N-1)=1$-dimensional manifolds, which are closed trajectories). It does not have a $(D-N)=1$-dimensional deformation retract or homotopy equivalent. However, we can replace it by its generating 1-cycles. This is the consequence of Corollary 4.2.3.

isomorphisms $\bar{i}_* : H_*(\mathbb{R}^D - O) \to H_*(\mathbb{R}^D - S)$

Proof. Deformation retract implies homotopy equivalence, and that the inclusion induces isomorphisms $i_* : H_{D-N}(S) \xrightarrow{\approxeq} H_{D-N}(O)$ for all N. Using this in the long exact sequence of homology groups for pair (O,S) we further have $H_{D-N}(O,S) \approxeq 0$ for all N (This can be concluded by observing that $\bar{i}_*$ being an isomorphism requires that for exactness of the sequence, ∂_* be a zero map, and j_* be a surjection and a zero map at the same time. Thus $H_{D-N}(O,S)$ requires to be zero for all N.). The result can then be concluded using Proposition 4.2.1. ■

Corollary 4.2.3. *Given a compact, locally contractible and orientable submanifold $O \subset \mathbb{R}^D$, let $\{S_k\}_{k=1,2,\cdots,m}$ be the images of generating cycles of $H_{D-N}(O)$ (i.e. if $\{\overline{S}_k\}$ are $(D-N)$-cycles in O such that the homology classes of $\{\overline{S}_k\}$ generate the group $H_{D-N}(O)$ freely, then $\{S_k\}$ are the images of $\{\overline{S}_k\}$), and let $\widetilde{\mathcal{S}} = \bigcup_{k=1,2,\cdots,m} S_k$. Then the inclusion map $\bar{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - \widetilde{\mathcal{S}})$ induces an isomorphism $\bar{i}_{*:N-1} : H_{N-1}(\mathbb{R}^D - O) \to H_{N-1}(\mathbb{R}^D - \widetilde{\mathcal{S}})$.*

Proof. The proof follows from the observation that the inclusion $i : \widetilde{\mathcal{S}} \hookrightarrow O$ induces an isomorphism in the $(D-N)^{th}$ homology groups of the spaces.

Clearly the $(D-N)^{th}$ homology groups of O and $\widetilde{\mathcal{S}}$ are isomorphic. The fact that the generating cycles of $\widetilde{\mathcal{S}}$ are generating cycles of O under inclusion, implies any cycle $\sigma \in C_{D-N}(O)$ can be written as $\sigma = i \circ \omega + \partial_{D-N+1} \circ \alpha$ for some $\omega \in C_{D-N}(\widetilde{\mathcal{S}})$ and $\alpha \in C_{D-N+1}(O)$. This implies that any cycle in the relative chain $C_{D-N}(O)/C_{D-N}(S)$ are trivial. Thus $H_{D-N}(O,S)$ is trivial.

The result hence follows from Proposition 4.2.1. Note that being images of cycles, each of S_k are $(D-N)$-dimensional closed (boundaryless), compact, locally contactable and orientable manifolds. ■

The consequence of the last two corollaries is that instead of computing homology in the original punctured space $(X-O)$, we can conveniently replace the obstacles with their equivalents, S, and compute homology in $(X-S)$, and yet, the results we obtain will be identical. That is exactly what we did by taking the *representative points* (Definition 3.3.1) of obstacles on a 2-dimensional plane, and *skeletons* (Definition 3.4.2) of obstacles in a 3-dimensional Euclidean space. Corollary 4.2.2 says that we can replace obstacles with such $(D-N)$-dimensional deformation retracts (Figure 4.1). However, often obstacles may not have $(D-N)$-dimensional deformation retracts (for example, a hollow torus does not have a 1 dimensional deformation retract, as illustrated in Figure 4.3). In such cases we can use Corollary 4.2.3 to replace obstacles by certain $(D-N)$-dimensional equivalents (generating cycles of $(D-N)^{th}$ homology group).

4.2.1 Reduced Problem Definition

Thus we have established that given any subset $\widetilde{\mathcal{O}}$ of $\mathbb{R}^D$ (which represent obstacles in robot planning problem), instead of looking at the homology classes of $(\mathbb{R}^D-\widetilde{\mathcal{O}})$, we can simply consider the homology classes of a space $(\mathbb{R}^D-\widetilde{\mathcal{S}})$ for some equivalents $\widetilde{\mathcal{S}}$ of $\widetilde{\mathcal{O}}$ (as prescribed by Corollaries 4.2.2 and 4.2.3). Of course we can decompose $\widetilde{\mathcal{S}}$ into a collection of manifolds since the types of spaces we are interested in are in fact manifolds, hence deformation retracts and images of generating cycles are manifolds as well. Thus, in the rest of the chapter, we will only consider the presence of $(D-N)$-dimensional compact, closed (boundary-less), locally contractible and orientable equivalents of the obstacles. Hence is the following reduced problem definition:

We are given $(D-N)$-dimensional $(N>1)$ compact, closed (boundaryless), locally contactable and orientable manifolds (which we call *singularity manifolds*), $S_1, S_2, \cdots, S_m$, embedded in the D-dimensional Euclidean space $\mathbb{R}^D$. Each of S_i is path-connected (*i.e.* has single component). We define the set $\widetilde{\mathcal{S}} = S_1 \cup S_2 \cup \cdots \cup S_m$ to be the set of all singularity manifolds.

We are interested in identifying homology classes of $(N-1)$-dimensional compact, closed (boundaryless), locally contactable and orientable manifolds in $(\mathbb{R}^D-\widetilde{\mathcal{S}})$ (which we call *candidate manifolds*). In order to compute that we use $(N-1)$-cycles on the *candidate manifolds* (*i.e.* top-dimensional cycles). Those, by inclusion, are $(N-1)$-cycles in $(\mathbb{R}^D-\widetilde{\mathcal{S}})$. Thus, given a candidate manifold ω, we can conveniently use a *simplicial cover* (or, equivalently, singular, cellular or cubical cover) of the manifold, $\overline{\omega}$, which is a $(N-1)$-cycle in $(\mathbb{R}^D-\widetilde{\mathcal{S}})$. However, given two cycles $\overline{\omega}_1, \overline{\omega}_2 \in Z_{N-1}(\mathbb{R}^D-\widetilde{\mathcal{S}})$, instead of checking if or not $\overline{\omega}_1-\overline{\omega}_2$ is boundary in $H_{N-1}(\mathbb{R}^D-\widetilde{\mathcal{S}})$, we would like to be able to use the action of some co-cycles in $H^{N-1}(\mathbb{R}^D-\widetilde{\mathcal{S}})$ on the cycles (specifically, integrations of differential forms over the cycles) to be able to say which homology classes they belong to.

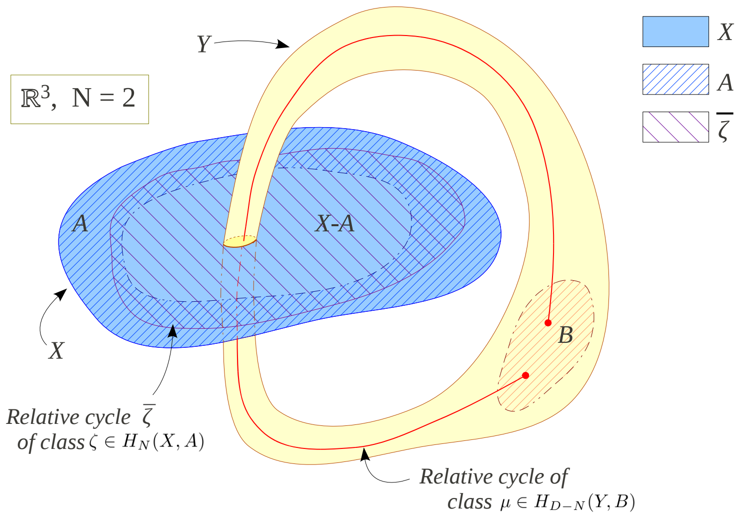

Y
$\mathbb{R}^3$, N = 2
X
A
$\overline{\zeta}$
A
X-A
X
B
Relative cycle $\overline{\zeta}$ of class $\zeta \in H_N(X, A)$
Relative cycle of class $\mu \in H_{D-N}(Y, B)$

Figure 4.4: Illustration of intersection number in $\mathbb{R}^3$ with $N = 2$ in light of Definition 4.3.1.

4.3 Preliminaries on Linking Numbers

Equipped with the notion of the $(D-N)$-dimensional replacements of the obstacles/punctures, S_i, we proceed towards computing the homology classes of $(N-1)$-cycles (in light of robot planning problem those are the closed trajectories) of $(\mathbb{R}^D - \widetilde{\mathcal{S}})$.

In this section we start by defining *intersection number*, and the related concept of *linking number*, from an algebraic topology view-point. The main purpose of this section is two fold:

i. To establish the fact that the linking numbers between the $(N-1)$-cycles in $(\mathbb{R}^D - S_i)$, and the manifolds S_i indeed tells us about the homology class of the $(N-1)$-cycles in $(\mathbb{R}^D - S_i)$ (Proposition 4.3.4),

ii. Propose a formula for computing the linking number using an integration over the $(N-1)$-cycle and a top-dimensional cycle of S_i (Proposition 4.3.5).

However, we will try to keep our initial treatment of linking/intersection numbers and related propositions fairly general from an algebraic topology consideration. We will however try to illustrate the ideas using corresponding examples from the familiar robot planning problem of Chapter 3.

4.3.1 Definitions

We first give the technical definitions of intersection number and linking number, following which we try to illustrate those using simple examples.

Technical Definitions

Recall the definition of *intersection number* [27],

Definition 4.3.1 (***Intersection Number*** – Ch. VII, Def. 4.1 of [27])**.**

If X and Y are sub-manifolds of $\mathbb{R}^D$, and $A \subset X \subset \mathbb{R}^D$, $B \subset Y \subset \mathbb{R}^D$ be such that $A \cap Y = \emptyset, X \cap B = \emptyset$ (Figure 4.4), and consider the map $p : (X \times Y, A \times Y \cup X \times B) \to (\mathbb{R}^D, \mathbb{R}^D - \{0\})$ given by $p(x, y) = x - y$ (where we used the natural vector structure of $\mathbb{R}^D$). The composition

$$H_N(X, A) \times H_{D-N}(Y, B) \xrightarrow{\times} H_D(X \times Y, A \times Y \cup X \times B) \xrightarrow{(-1)^{D-N} p_*} H_D(\mathbb{R}^D, \mathbb{R}^D - \{0\})$$

is called the *intersection pairing* (Note that the product '$\times$' for homology groups is the *homology cross product*, which is more closely related to the tensor product of the homology groups rather than the cartesian product – see p. 268 of [40]).

We write

$$\mathscr{I}(\zeta, \mu) = (-1)^{D-N} p_*(\zeta \times \mu), \quad \text{for } \zeta \in H_N(X, A),\ \mu \in H_{D-N}(Y, B)$$

and call this element of $H_D(\mathbb{R}^D, \mathbb{R}^D - \{0\}) \approxeq \mathbb{R}$ the *intersection number* of ζ and μ.

Note that the intersection number is defined for homology classes of the pairs (X, A) and (Y, B). However it can be easily extended to relative cycles $\overline{\zeta} \in Z_N(X, A)$, $\overline{\mu} \in Z_{D-N}(Y, B)$ as $\overline{\mathscr{I}}(\overline{\zeta}, \overline{\mu}) = \big[(-1)^{D-N} p(\overline{\zeta} \times \overline{\mu})\big] = \mathscr{I}([\overline{\zeta}], [\overline{\mu}])$, where $[\cdot]$ represents the homology class of a cycle. This follows from functoriality of homology (See p. 162 of [40]).

Also, once again note that the product '$\times$' between chains is more closely related to group tensor products. This can just be the group tensor product for cellular or cubical chains. But for singular or simplicial chains the product involves further operations and is called the *simplicial cross product* (See p. 277 of [40]).

Moreover, more generally, $p : X \times Y \to \mathbb{R}^D$ can be an arbitrary continuous surjective map which maps only the points $\{(x, y) \in X \times Y \ \big| x = y\}$ to $\{0\} \in \mathbb{R}^D$.

Definition 4.3.2 (***Linking Number*** – Adapted from Ch. 10, Art. 77 of [78])**.**

We borrow definitions of X, A, Y and B from Definition 4.3.1. Recall that from the *long exact sequence* of the pair (X, A) we have a map $\partial_* : H_N(X, A) \to H_{N-1}(A)$.

Now, if $\varsigma \in H_{N-1}(A)$ is such that it can be written as $\varsigma = \partial_* \zeta$ for some $\zeta \in H_N(X, A)$, and if $\mu \in H_{D-N}(Y, B)$, then the *linking number* between ς and μ is defined as $\mathscr{L}(\varsigma, \mu) = \mathscr{I}(\zeta, \mu)$.

Note that similar to the intersection number, by the functoriality of homology, linking number can be defined between a cycle, $\overline{\varsigma}$, in A and a relative cycle, $\overline{\mu}$, in (Y, B).

Simplified Description of the Definitions

Let us consider the simple case when $X = \mathbb{R}^3$, $A = \mathbb{R}^3 - S$, $Y = S$ and $B = \emptyset$, and with $D = 3, N = 2$, which arises in robot path planning in $\mathbb{R}^3$ (Figure 4.5). Let $\overline{\mu}$ be a top-dimensional cycle on the $(D - N)$-dimensional manifold, S (to be consistent with the notations used in the definition). Intersection number, as the name suggests, informally speaking, counts the number of intersections between a N-chain $\overline{\xi}$ (in light of Definition 4.3.1, it is represented by the relative cycle $\overline{\zeta}$), and a $(D - N)$-cycle $\overline{\mu}$. Thus, in Figure 4.5, informally, the intersection number between $\overline{\xi}$ and $\overline{\mu}$ is ± 1 (the sign depends on orientation).

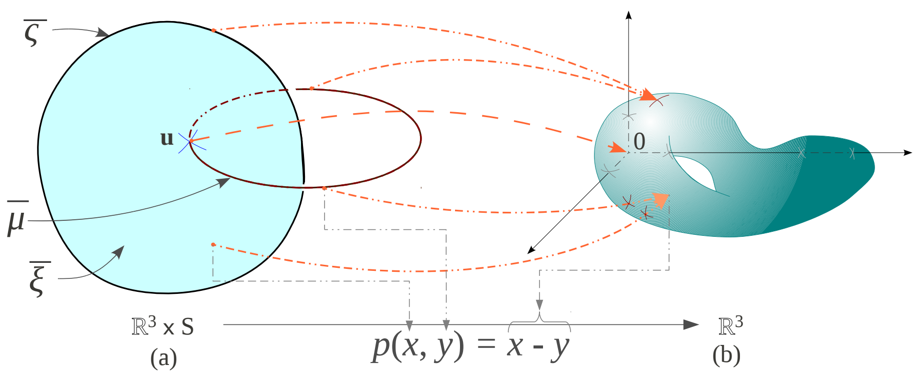
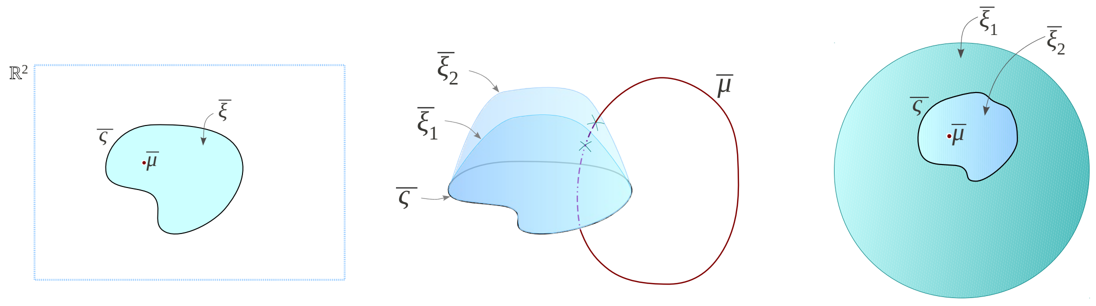


Figure 4.5: A simplified illustration (following from Figure 4.2(c)) of intersection number and linking number in $\mathbb{R}^3$ with $N = 2$. This is a special case of Definition 4.3.1 when $X = \mathbb{R}^3$, $A = \mathbb{R}^3 - S$, $Y = S$ and $B = \emptyset$.
Figure (a) on the left: The intersection number is computed between a N-chain, $\overline{\xi}$ (more precisely it is a relative cycle in (X, A) that we consider – the boundary of $\overline{\xi}$ trivialized), and the $(D-N)$-cycle, $\overline{\mu}$, that is a top-dimensional cycle on S. In this figure the said intersection number is ± 1 due to the single intersection marked by the blue 'cross' at **u**. Then, by definition, that is equal to the linking number between $\overline{\varsigma} = \partial\overline{\xi}$ and $\overline{\mu}$.
Figure (b) on the right: The precise definition requires a mapping, p, from pair of points in the original space (one point from the 2-chain, $\overline{\xi}$, embedded in the ambient space, $\mathbb{R}^3$, and another from S) to (a different copy of) $\mathbb{R}^3$. The intersection/linking number is then, informally, the number of times intersection points in the pre-image of p (points like **u**) maps to the origin, 0 (with proper sign), in the image, or equivalently, the number of times the image of $\overline{\varsigma} \times \overline{\mu}$, under the action of p, *wraps around* the origin. Thus, it is the homology class of the cycle $p(\overline{\varsigma} \times \overline{\mu})$ in the punctured Euclidean space $(\mathbb{R}^D - 0)$.

Now, if $\overline{\xi}$ has a boundary, say $\overline{\varsigma} = \partial\overline{\xi}$, the linking number between $\overline{\varsigma}$ and $\overline{\mu}$ is, by definition, the intersection number between $\overline{\xi}$ and $\overline{\mu}$.

In defining the intersection number, however, one does not talk about the N-chain $\overline{\xi}$. Instead, one talks about the corresponding relative cycle in $(\mathbb{R}^3, \mathbb{R}^3 - S)$ under the action of the quotient map $j : C_N(\mathbb{R}^3) \to C_N(\mathbb{R}^3, \mathbb{R}^3 - S)$ – that is, the part of $\overline{\xi}$ that does not intersect with S, is trivialized (which is $\overline{\zeta}$ $(= j(\overline{\xi}))$ in notation of Definition 4.3.1). This is because homology classes are not defined for chains, rather can be defined for cycles or relative cycles only. By trivializing the part of the chain that contains the boundary (*i.e.* the one lying in $(\mathbb{R}^3 - S)$), we convert it into a relative cycle, thus enabling us to talk about its homology class (ζ in Definition 4.3.1).

For constructing a precise algebraic definition of linking/intersection number, the relative cycles $\overline{\zeta}$ and $\overline{\mu}$ are then mapped to $\mathbb{R}^3$ via the map $p(x, y) = x - y$. The intersection number is then, informally, the number of times the intersection points (like **u** in Figure 4.5) map to (with proper sign) the origin in the co-domain of p. Linking number is then essentially the number of times the image of $\overline{\varsigma} \times \overline{\mu}$, under the action of p, *wraps around* the origin in the co-domain of p. In other words, it is the homology class of the cycle $p(\overline{\varsigma} \times \overline{\mu})$ in the punctured Euclidean space $(\mathbb{R}^D - 0)$.



(a) On $\mathbb{R}^2$ the linking number between a point $\overline{\mu}$ (a 0-cycle) and a 1-cycle $\overline{\varsigma}$ is uniquely determined, and is equal to the intersection number between $\overline{\mu}$ and a 2-chain $\overline{\xi}$ such that $\overline{\varsigma} = \partial\overline{\xi}$. It can be shown that the choice of $\overline{\xi}$ does not matter.

(b) Similarly, in $\mathbb{R}^3$ the linking number between a 1-cycle, $\overline{\mu}$, and a closed cycle $\overline{\varsigma}$ is uniquely determined, and is equal to the intersection number between $\overline{\mu}$ and a 2-chain $\overline{\xi}$ such that $\overline{\varsigma} = \partial\overline{\xi}$. It can be shown that the choice of $\overline{\xi}$ does not matter – that is, we could choose $\overline{\xi}_1$ or $\overline{\xi}_2$ for computation of the intersection number, and the value that we would obtain will be the same.

(c) However, if the ambient space, X, is not contractible (in this figure it is the 2-sphere, $\mathbb{S}^2$), then the linking number between $\overline{\mu}$ and $\overline{\varsigma}$ is not unambiguously defined. This is because the choices of $\overline{\xi}$ such that $\overline{\varsigma} = \partial\overline{\xi}$ can be made in ways such that its intersection number with $\overline{\mu}$ is different for the different choices. In the figure, on $\mathbb{S}^2$, the boundary of both $\overline{\xi}_1$ and $\overline{\xi}_2$ are $\overline{\varsigma}$. However the intersection number between $\overline{\xi}_1$ and $\overline{\mu}$ is zero, while that between $\overline{\xi}_2$ and $\overline{\mu}$ is ± 1.

Figure 4.6: Examples and counter-examples of uniqueness of linking number – a consequence of Proposition 4.3.3.

4.3.2 Propositions on Linking Number

In this sub-section we will state and prove two propositions, each followed by simplified explanation of the result of the propositions.

The fact that we have assumed X and Y to be embedded in $\mathbb{R}^D$ guarantees that the linking number is uniquely defined. However, one can consider a more general case where they are embedded in an arbitrary D-dimensional manifold, M. Also, the map p is replaced by a more general continuous surjective map $p : (X \times Y, A \times Y \cup X \times B) \to (M, M - \mathbf{0})$ for some base-point $\mathbf{0}$ in M. In such case the uniqueness of the linking number is non-trivial (see Figure 4.6).

Proposition 4.3.3 (Uniqueness of linking number)**.** *If $H_N(X) = H_{N-1}(X) = 0$ holds, then $\mathscr{L}(\varsigma, \mu)$ is independent of the exact choice of ζ [78]. More precisely, under the said conditions, ∂_*^{-1} exists, and thus $\mathscr{L}(\varsigma, \mu) = \mathscr{I}(\partial_*^{-1}\varsigma, \mu) = (-1)^{D-N} p_*(\partial_*^{-1}\varsigma \times \mu)$.*

Proof. From the long exact sequence for the pair (X, A), using the condition $H_N(X) = H_{N-1}(X) = 0$, it follows that $\partial_* : H_N(X, A) \to H_{N-1}(A)$ is an isomorphism (See p. 114 of [40]). Hence the result follows. [Note that the contractibility of X is sufficient for the said condition to hold.] ■

The the statement of the above proposition is about the uniqueness in the value of linking number. Intersection number, according to definition, is between a N-chain (like $\overline{\xi}$ in the example of figure 4.5 or 4.6) and a $(D - N)$-cycle (like $\overline{\mu}$ in the example of Figure 4.5). That is then, by definition, the linking number between the boundary of the N-chain, which is a $(N - 1)$-cycle (like

$\overline{\varsigma}$ in the example of Figure 4.5) and the $(D-N)$-cycle ($\overline{\mu}$). However, it may be possible that there exists another N-chain, $\overline{\xi}'$, such that $\overline{\varsigma}$ is the boundary of that N-chain as well (*i.e.* $\overline{\varsigma}$ is a common boundary between $\overline{\xi}$ and $\overline{\xi}'$ – as illustrated in Figure 4.6(b)). Then, if the intersection number between $\overline{\xi}'$ and $\overline{\mu}$ is not same as that between $\overline{\xi}$ and $\overline{\mu}$, the definition of the linking number between $\overline{\varsigma}$ and $\overline{\mu}$ becomes ambiguous. This is exemplified in Figure 4.6(c). Proposition 4.3.3 precisely gives the condition under which such ambiguity is not present.

Proposition 4.3.4 (Connection to homology of A)**.** *Consider a fixed non-zero $\mu \in H_{D-N}(Y,B)$. If, in addition to the condition of Proposition 4.3.3, we have $H_N(X,A) \approxeq H_{N-1}(A) \approxeq \mathbb{R}$, and if there exists at least one $(N-1)$-cycle in A such that its linking number with μ is non-zero, then the value of $\mathscr{L}(\varsigma,\mu)$ tells us which element of $H_{N-1}(A)$ is the chosen ς. In other words, the map $\mathcal{H} \equiv \mathscr{L}(\cdot,\mu) : H_{N-1}(A) \to H_{D-1}(\mathbb{R}^D, \mathbb{R}^D - \{0\}) \approxeq \mathbb{R}$ is an injective homomorphism.*

Proof. The map $\mathcal{H}$ is given by $\mathcal{H}(\varsigma) = (-1)^{D-N} p_*(\partial_*^{-1}\varsigma \times \mu)$. This clearly is a group homomorphism between $H_{N-1}(A)$ and $H_{D-1}(\mathbb{R}^D, \mathbb{R}^D - \{0\})$. Since by hypothesis, both the domain and the co-domain of $\mathcal{H}$ are isomorphic to $\mathbb{R}$, $\mathcal{H}$ can either be a trivial homomorphism (*i.e.* maps everything in its domain to 0 in its co-domain), or it can be an injection. The former possibility is ruled out by the hypothesis of existence of at least one $(N-1)$-cycle in A with non-zero linking number with μ. Thus the result follows. The result implies that the linking number with μ is a *complete invariant* for the homology class ς. ■

So far we have been talking about intersection number and linking number. However what we are really interested in is the homology class of $\overline{\varsigma}$ in A (in light of robot planning problems, that is the homology class of the closed trajectories in $(\mathbb{R}^D - S)$, as illustrated in figures 4.5 ans 4.6). The result of the above proposition establishes a relationship between the linking number between $\overline{\mu}$ and $\overline{\varsigma}$ (see figure 4.5 or 4.6), and the homology class of $\overline{\varsigma}$. It says that under certain conditions, the linking number will precisely tell us about the homology class of $\overline{\varsigma}$ (*i.e.* a *complete invariant*).

4.3.3 Computation of Intersection/Linking Number for Given Cycles

In this sub-section we will actually try to compute the linking number between the cycles $\overline{\varsigma}$ and $\overline{\mu}$. As described in the beginning of this chapter, we would like to be able to compute the homology class of $(N-1)$-cycles (top-dimensional cycles on $(N-1)$-dimensional manifolds) as an explicit number (or a set of numbers). Equipped with Proposition 4.3.4, that problem can be converted to the problem of computation of the linking numbers.

There can be a large variety of differential forms on a manifold. A n-form can always be integrated on a n-cycle. However, the value of the integration may not tell us anything about the homology class of the cycle. For example, in $(\mathbb{R}^2 - 0)$, *i.e.* the plane with the origin removed, $\mathrm{d}x$ is a differential 1-form. However it is *exact* and evaluates to 0 on every closed curve. On the other hand, as we saw in Chapter 3, $\mathrm{d}\theta = \frac{x\ \mathrm{d}y + y\ \mathrm{d}x}{x^2+y^2}$ $(= Im(\frac{\mathrm{d}z}{z})))$, which is *closed* but not *exact* [8] in $(\mathbb{R}^2 - 0)$, in fact tell us about the homology class of closed loops.

The purpose of the proposition below is to design a differential from, integration of which, along with the conditions of Proposition 4.3.4, captures the homology class of $(N-1)$-cycles in

A (which, in light of robot planning problem, are the punctured spaces $(\mathbb{R}^D - S)$, S being path-connected). In order to achieve this for arbitrary A (in robot planning, for example, we can arbitrary representatives, S, for the obstacles), we exploit the transformation, p, in the definition of linking numbers. Thus, the closed, non-exact differential form that we have to choose is one from the co-domain of p, namely $(\mathbb{R}^D - 0)$ (a space which is much simpler and well-known than, say, $(\mathbb{R}^D - S)$), and then *pull it back* to the original space by p. Thus we have the following proposition.

Let $\eta_0 \in \Omega_{dR}^{D-1}(\mathbb{R} - \{0\})$ be a closed but non-exact differential form in $(\mathbb{R} - \{0\})$ such that $[\eta_0] \in H_{dR}^{D-1}(\mathbb{R} - \{0\}) \approxeq \mathbb{R}$ is a generator of $H_{dR}^{D-1}(\mathbb{R} - \{0\})$.

Proposition 4.3.5. *In addition to the condition of Proposition 4.3.3, if $H_{D-N}(B) = H_{D-N-1}(B) = 0$, then the linking number between cycles $\overline{\varsigma} \in Z_{N-1}(A)$ and $\overline{\mu} \in Z_{D-N}(Y, B)$ is uniquely identified by the value of (i.e. a complete invariant for the linking number is)*

$$(-1)^{D-N} \int_{\overline{\varsigma} \times \overline{u}} p^*(\eta_0)$$

where $\overline{u} \in Z_{D-N}(Y)$ is such that $j'(\overline{u}) = \overline{\mu}$, where j' is the quotient map $Y \to Y/B$ (See Thm. 2.13 of [40] – note that for a given $\overline{\mu}$, in general, there can be many possible choices for $\overline{u}$).

Proof. Detailed proof can be found in Appendix B.2. ■

4.4 Computation in Our Specific Problem

In this section we specialize the results of the previous section to match the description of the *reduced problem definition* in Section 4.2.1. We consider the case where there is a single path-connected component of $\widetilde{\mathcal{S}}$, namely S.

For our problem, in connection to the definitions stated in Section 4.3 (see Figure 4.5), we set

$$X = \mathbb{R}^D, \ A = \mathbb{R}^D - S, \ \ Y = S \ \text{ and } B = \emptyset$$

Moreover, since $Y \equiv S$ is a $(D-N)$-dimensional compact, connected and orientable manifold, we have $H_{D-N}(S) \approxeq \mathbb{R}$. We thus choose $\overline{\mu} = \overline{S}$ to be a non-zero top dimensional cover of S such that $[\overline{S}] = \mathbf{1} \in H_{D-N}(S)$ is a generator of $H_{D-N}(S)$ (which is isomorphic to $\mathbb{R}$ since S is a closed, path-connected and orientable $(D-N)$-dimensional manifold).

Also, note that since $B = \emptyset$, the maps $j' : Y \to Y/B$ is the identity map. So in this case $[\overline{S}] \in H_{D-N}(S, B) \equiv H_{D-N}(S)$.

For this choice it is easy to verify that the conditions of Propositions 4.3.3, 4.3.4 and 4.3.5 hold.

i. *Condition for Proposition 4.3.3:* $H_N(\mathbb{R}^D) = H_{N-1}(\mathbb{R}^D) = 0$ follows from contractibility of $\mathbb{R}^D$.

ii. *Conditions for Proposition 4.3.4:*

   a. By Proposition 3.46 of [40], $H_N(\mathbb{R}^D, \mathbb{R}^D - S) \approxeq H^{D-N}(S)$. Again, using Poincar Duality for S (which is a $(D-N)$-dimensional closed, orientable manifold), $H^{D-N}(S) \approxeq H_0(S) \approxeq$

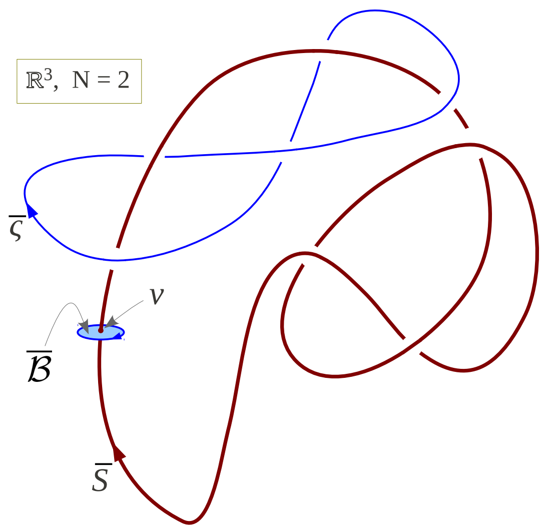


Figure 4.7: The specific problem under consideration, illustrated for $D = 3, N = 2$.

$\mathbb{R}$ (since S has a single connected component). Finally, from the long exact sequence for the pair $(\mathbb{R}^D, \mathbb{R}^D - S)$, using the contractibility of $\mathbb{R}^D$, we have, $H_N(\mathbb{R}^D, \mathbb{R}^D - S) \approxeq H_{N-1}(\mathbb{R}^D - S)$. Combining these three isomorphisms we have,

$$H_N(\mathbb{R}^D, \mathbb{R}^D - S) \approxeq H_{N-1}(\mathbb{R}^D - S) \approxeq \mathbb{R} \tag{4.4.1}$$

.

b. Consider a point $v \in S$. Since $\overline{S}$ covers S, this point is also in (the image of) $\overline{S}$. Since S is $(D-N)$-dimensional, we can choose a small N-ball, $\mathcal{B}$, centered at v such that it intersects S transversely only at v. Let $\overline{\mathcal{B}} \in C_N(\mathbb{R}^D)$ be a top-dimensional non-zero chain that covers $\mathcal{B}$. Clearly the intersection number between $\overline{S}$ and $j(\overline{\mathcal{B}})$ (where $j : \mathbb{R}^D \to \mathbb{R}^D/(\mathbb{R}^D - S)$ is the quotient map) is non-zero. Thus the linking number between $\partial\overline{\mathcal{B}}\big|_{(\mathbb{R}^D - S)}$ (which, by our construction, is a $(N-1)$-cycle in $(\mathbb{R}^D - S)$) and $\overline{S}$ is non-zero. Thus there exists at least one $(N-1)$-cycle in $(\mathbb{R}^D - S)$ that has non-zero linking number with $\overline{S}$ (see Figure 4.7).

iii. *Condition for Proposition 4.3.5:* Follows from the fact that $B = \emptyset$.

Thus, we are concerned with finding a complete invariant for homology classes of $(N-1)$-cycles, $\overline{\omega} \in Z_{N-1}(\mathbb{R}^D - S)$. Which, by Proposition 4.3.4, is the linking number between $\overline{\omega}$ and $\overline{S}$. And, finally, using Proposition 4.3.5, the complete invariant for the homology classes of such chains is given by the value of the integration

$$\begin{aligned} \phi_S(\overline{\omega}) &= (-1)^{D-N} \int_{\overline{\omega} \times \overline{S}} p^*(\eta_0) \\ &= (-1)^{D-N} \int_{\overline{\omega}} \int_{\overline{S}} p^*(\eta_0) \quad \text{[by Fubinis theorem]} \end{aligned} \tag{4.4.2}$$

4.4.1 Computation of the integral in ϕ_S

Let $\mathbf{x} \in (\mathbb{R}^D - S) \subset \mathbb{R}^D$ be the coordinate variable describing points in $(\mathbb{R}^D - S)$, and let $\mathbf{x}' \in S \subset \mathbb{R}^D$ be the one describing points in S. Thus we have $p(\mathbf{x}, \mathbf{x}') = \mathbf{x} - \mathbf{x}'$.

Again, we let $\mathbf{s} \in (\mathbb{R}^D - \{0\}) \subset \mathbb{R}^D$ be the natural coordinate variable describing points in the space $(\mathbb{R}^D - \{0\})$. A well-known result [3, 30] is that a differential closed but non-exact $(D-1)$-form in $(\mathbb{R}^D - \{0\})$ (*i.e.*, a nontrivial element of $Z_{dR}^{D-1}(\mathbb{R}^D - \{0\})$) is,

$$\eta_0(\mathbf{s}) = \sum_{k=1}^{D} \mathcal{G}_k(\mathbf{s}) \ \ (-1)^{k+1} \quad \mathrm{d}s_1 \wedge \ \mathrm{d}s_2 \wedge \cdots \wedge \ \mathrm{d}s_{k-1} \wedge \ \mathrm{d}s_{k+1} \wedge \cdots \wedge \ \mathrm{d}s_D \tag{4.4.3}$$

where

$$\mathcal{G}_k(\mathbf{s}) = \frac{1}{A_{D-1}} \ \frac{s_k}{(s_1^2 + s_2^2 + \cdots + s_D^2)^{D/2}} \tag{4.4.4}$$

where, $\mathbf{s} = [s_1, s_2, \cdots, s_D]^T \in (\mathbb{R}^D - \{0\})$, and $A_{D-1} = \frac{D\pi^{\frac{D}{2}}}{\Gamma(\frac{D}{2}+1)}$ is the surface area of the $(D-1)$-unit sphere (which acts as a normalizing factor, and can be set to any non-zero value).

Thus the pullback of η_0 under p is given by the following formula,

$$\begin{aligned} \eta(\mathbf{x}, \mathbf{x}') \ &= \ p^*(\eta_0) \ = \ \eta_0\big|_{\mathbf{s}=\mathbf{x}-\mathbf{x}'} \\ &= \sum_{k=1}^{D} \mathcal{G}_k(\mathbf{x} - \mathbf{x}') \ \ (-1)^{k+1} \quad \mathrm{d}(x_1 - x_1') \wedge \ \mathrm{d}(x_2 - x_2') \wedge \cdots \\ &\qquad \wedge \mathrm{d}(x_{k-1} - x_{k-1}') \wedge \ \mathrm{d}(x_{k+1} - x_{k+1}') \wedge \cdots \wedge \ \mathrm{d}(x_D - x_D') \end{aligned} \tag{4.4.5}$$

Now consider the quantity of our interest, $\phi(\overline{\omega}) = \int_{\mathbf{x}\in\overline{\omega}} \int_{\mathbf{x}'\in\overline{S}} \eta(\mathbf{x}, \mathbf{x}')$. On $\overline{\omega} \times \overline{S}$, at most $(N-1)$ unprimed differentials can be independent, and at most $(D-N)$ primed differentials can be independent (since $\mathbf{x}$ represents a point on the image of the $(N-1)$ chain $\overline{\omega}$ and $\mathbf{x}'$ represents a point on the image of the $(D-N)$ chain $\overline{S}$). Thus we can conveniently drop all the terms in the expansion of η (which is a $(D-1)$-differential form on $(\mathbb{R}^D - S) \times S$) that do not satisfy these conditions on maximum number of primed/unprimed differentials. Thus we obtain a simpler differential form $\tilde{\eta}$,

$$\begin{aligned} \tilde{\eta}(\mathbf{x}, \mathbf{x}') \ = \ \sum_{k=1}^{D} \Bigg(& \mathcal{G}_k(\mathbf{x} - \mathbf{x}') \ \ (-1)^{k+1+D-N} \ \cdot \\ & \sum_{\substack{\tau_i \in \{0,1\} \\ \tau_1+\cdots+\tau_D = D-N}} \mathrm{d}x_1^{(\tau_1)} \wedge \ \mathrm{d}x_2^{(\tau_2)} \wedge \cdots \wedge \ \mathrm{d}x_{k-1}^{(\tau_{k-1})} \wedge \ \mathrm{d}x_{k+1}^{(\tau_{k+1})} \wedge \cdots \wedge \ \mathrm{d}x_D^{(\tau_D)} \Bigg) \end{aligned} \tag{4.4.6}$$

[where, $x_i^{(\tau)}$ represents x_i' if $\tau = 1$, otherwise represents x_i if $\tau = 0$.]
This differential form, though simpler, has the property

$$\phi_S(\overline{\omega}) \ = \ (-1)^{D-N} \int_{\mathbf{x}\in\overline{\omega}} \int_{\mathbf{x}'\in\overline{S}} \eta(\mathbf{x}, \mathbf{x}') \ = \ (-1)^{D-N} \int_{\mathbf{x}\in\overline{\omega}} \int_{\mathbf{x}'\in\overline{S}} \tilde{\eta}(\mathbf{x}, \mathbf{x}') \tag{4.4.7}$$

Finally, we re-write the formula for $\tilde{\eta}$ using a new notation as follows,

$$\begin{aligned}\tilde{\eta}(\mathbf{x},\mathbf{x}') \quad &= \quad (-1)^{D-N} \sum_{k=1}^{D} \Bigg(\mathcal{G}_k(\mathbf{x}-\mathbf{x}') \;\; (-1)^{k+1} \cdot \\ & \qquad \sum_{\rho \in part^{D-N}(\mathcal{N}^D_{-k})} \mathrm{sgn}(\rho) \quad \mathrm{d}x'_{\rho_l(1)} \wedge \cdots \wedge \; \mathrm{d}x'_{\rho_l(D-N)} \wedge \; \mathrm{d}x_{\rho_r(1)} \wedge \cdots \wedge \; \mathrm{d}x_{\rho_r(N-1)} \Bigg)\end{aligned} \tag{4.4.8}$$

where,

- $\mathcal{N}^D_{-k} = [1, 2, \cdots, k-1, k+1, \cdots, D]$ is an ordered set,
- $part^w(\mathcal{A})$ is the set of all 2 partitions of the ordered set $\mathcal{A}$, such that the first partition contains w elements, and each of the partitions contain elements in order. [1]

Thus, the final formula for the complete invariant for homology class of $\overline{\omega} \in Z_{N-1}(\mathbb{R}^D - S)$ is,

$$\begin{aligned}\phi_S(\overline{\omega}) \quad &= \quad (-1)^{D-N} \int_{\mathbf{x}\in\overline{\omega}} \int_{\mathbf{x}'\in\overline{S}} \tilde{\eta}(\mathbf{x},\mathbf{x}') \\ &= \int_{\mathbf{x}\in\overline{\omega}} \sum_{k=1}^{D} \sum_{\rho \in part^{D-N}(\mathcal{N}^D_{-k})} \\ & \qquad \left((-1)^{k+1} \int_{\mathbf{x}'\in\overline{S}} \mathcal{G}_k(\mathbf{x}-\mathbf{x}') \;\; \mathrm{sgn}(\rho) \quad \mathrm{d}x'_{\rho_l(1)} \wedge \cdots \wedge \; \mathrm{d}x'_{\rho_l(D-N)} \right) \\ & \qquad \qquad \wedge \; \mathrm{d}x_{\rho_r(1)} \wedge \cdots \wedge \; \mathrm{d}x_{\rho_r(N-1)} \\ &= \int_{\mathbf{x}\in\overline{\omega}} \sum_{k=1}^{D} \sum_{\rho \in part^{D-N}(\mathcal{N}^D_{-k})} U^k_\rho(\mathbf{x}; S) \wedge \; \mathrm{d}x_{\rho_r(1)} \wedge \cdots \wedge \; \mathrm{d}x_{\rho_r(N-1)}\end{aligned} \tag{4.4.9}$$

where,

$$U^k_\rho(\mathbf{x}; S) = (-1)^{k+1} \;\; \mathrm{sgn}(\rho) \int_{\mathbf{x}'\in\overline{S}} \mathcal{G}_k(\mathbf{x}-\mathbf{x}') \quad \mathrm{d}x'_{\rho_l(1)} \wedge \cdots \wedge \; \mathrm{d}x'_{\rho_l(D-N)} \tag{4.4.10}$$

and by convention, $\overline{S}$ is a top-dimensional cycle covering S such that $[\overline{S}] = \mathbf{1} \in H_{D-N}(S)$.

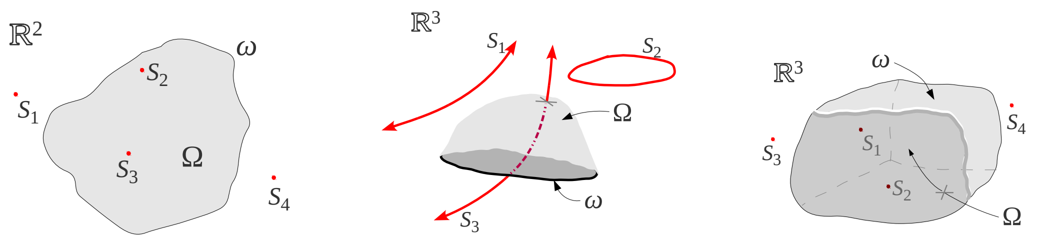
Also, note that the quantity inside the integral in the formula for ϕ_S is a differential $(N-1)$-form in $(\mathbb{R}^D - S)$. Thus we can integrate it over $\overline{\omega}$. We represent the differential $(N-1)$-form by ψ_S

$$\psi_S = \sum_{\rho \in part^{D-N}(\mathcal{N}^D_{-k})} U^k_\rho(\mathbf{x}; S) \wedge \; \mathrm{d}x_{\rho_r(1)} \wedge \cdots \wedge \; \mathrm{d}x_{\rho_r(N-1)} \tag{4.4.11}$$

[1]Let us consider an ordered set $\mathcal{A} = [a_1, a_2, \cdots, a_q]$ with $a_1 \leq a_2 \leq \cdots \leq a_q$ (where the inequality sign signifies order of arrangement and not necessarily the order of magnitude). We represent the set of all ordered 2-partitions of the set $\mathcal{A}$ into w and $q-w$ elements as $part^w(\mathcal{A})$, such that for a $\rho = [\rho_l, \rho_r] \in part^w(\mathcal{A})$, ρ_l and ρ_r are ordered sets of w and $q-w$ elements respectively, with the properties that $\rho_l \cap \rho_r = \emptyset$, $\rho_l(1) \leq \rho_l(2) \leq \cdots \leq \rho_l(w)$ and $\rho_r(1) \leq \rho_r(2) \leq \cdots \leq \rho_r(q-w)$. Then the sign of the partition, $\mathrm{sgn}(\rho)$, is defined as the permutation sign of the ordered set $\rho_l \sqcup \rho_r$. For example,

$$\begin{aligned} part^3([1,3,6,9,5]) \quad = \quad \{ \quad & [[1,3,6],[9,5]]\,, \quad [[1,3,9],[6,5]]\,, \quad [[1,3,5],[6,9]]\,, \quad [[1,6,9],[3,5]]\,, \\ & [[1,6,5],[3,9]]\,, [[1,9,5],[3,6]]\,, [[3,6,9],[1,5]]\,, [[3,6,5],[1,9]]\,, [[3,9,5],[1,6]]\,, [[6,9,5],[1,3]] \;\}. \end{aligned}$$

Then if $\rho = [[1,6,5],[3,9]] \in part^3([1,3,6,9,5])$, we write $\rho_l = [1,6,5]$ and $\rho_r = [3,9]$. Also, the j^{th} element of ρ_b, $b \in \{l, r\}$ is written as $\rho_b(j)$. Thus, in the example, $\rho_l(2) = 6$.



(a) $D = 2, N = 2$ (b) $D = 3, N = 2$ (c) $D = 3, N = 3$

Figure 4.8: Schematic illustration of some lower dimensional cases of the problem. The Cauchy Residue theorem can be applied to (a), Ampere's law to (b), and Gauss Divergence theorem to (c).

4.4.2 Incorporating Multiple Connected Components of $\widetilde{\mathcal{S}}$

So far we have worked with a single connected component of the puncture, namely, S. However, recall that the original space under consideration was $(\mathbb{R}^D - \widetilde{\mathcal{S}})$, with $\widetilde{\mathcal{S}} = \bigcup_{i=1}^m S_i$, such that each S_i is path connected, compact, closed, locally contractible and orientable. Moreover, by hypothesis, $S_i \cap S_j = \emptyset,\ \forall i \neq j$. We have the following proposition in order to compute the homology of the smaller space, $(\mathbb{R}^D - \widetilde{\mathcal{S}})$, in terms of the larger spaces, $(\mathbb{R}^D - S_k)$.

Proposition 4.4.1. $H_{N-1}(\mathbb{R}^D - \widetilde{\mathcal{S}}) \approxeq \bigoplus_{k=1}^m H_{N-1}(\mathbb{R}^D - S_k) \approxeq \mathbb{R}^m$. *Where, the first isomorphism is induced by the direct sum of the inclusion maps* $\tilde{i}_k : (\mathbb{E}^D - \widetilde{\mathcal{S}}) \hookrightarrow (\mathbb{E}^D - S_k)$.

Proof. Detailed proof can be found in Appendix B.3. ■

Thus, for any $\overline{\omega} \in Z_{N-1}(\mathbb{E}^D - \widetilde{\mathcal{S}})$, a complete invariant for the homology class of $\overline{\omega}$ is given by,

$$\phi_{\widetilde{\mathcal{S}}}(\overline{\omega}) \overset{def.}{=} \begin{bmatrix} \phi_{S_1}(\overline{\omega}) \\ \phi_{S_2}(\overline{\omega}) \\ \vdots \\ \phi_{S_m}(\overline{\omega}) \end{bmatrix} \tag{4.4.12}$$

where, ϕ_{S_i} is given by the formula in Equation (4.4.9). [Note that we have implicitly assumed a inclusion map $\tilde{i}_k : (\mathbb{E}^D - \widetilde{\mathcal{S}}) \hookrightarrow (\mathbb{E}^D - S_k)$ being applied on $\overline{\omega}$ for computation of the k^{th} component. For simplicity we don't write it explicitly, since the map is identity as far as computation is concerned.]

Thus, $[\overline{\omega}_1] = [\overline{\omega}_2]$ if and only if $\phi_{\widetilde{\mathcal{S}}}(\overline{\omega}_1) = \phi_{\widetilde{\mathcal{S}}}(\overline{\omega}_2)$, for any $\overline{\omega}_1, \overline{\omega}_2 \in Z_{N-1}(\mathbb{E}^D - \widetilde{\mathcal{S}})$.

4.5 Validations in Low Dimensions

In this section we illustrate the forms that equations (4.4.10) and (4.4.11) take under certain special cases. We compare those with the well-known formulae from complex analysis, electromagnetism and electrostatics that are known to give homology class invariants. Once again, we demonstrate all the computations using a single connected component of $\widetilde{\mathcal{S}}$.

4.5.1 $D = 2, N = 2$:

This particular case has parallels with the *Cauchy integral theorem* and the *Residue theorem* from *Complex analysis*. This formula was used in Section 3.3 for designing a H-signature in the 2-dimensional case. Here a singularity manifold, S, is a $D - N = 0$-dimensional manifold, *i.e.* a point, the coordinate of which we represent by $\mathbf{S} = [s_1, s_2]^T$ (the *representative points* according to terminology of Chapter 3 - Figure 4.8(a)).
Thus, the partitions in (4.4.11) for the different values of k are as follows,
For $k = 1$, $part^0(\{2\}) = \Big\{\{\{\}, \{2\}\}\Big\}$,
For $k = 2$, $part^0(\{1\}) = \Big\{\{\{\}, \{1\}\}\Big\}$
Thus,

$$U_1^1(\mathbf{x}) = \frac{1}{2\pi}\ (-1)^{2-2+1+1}(1)\frac{x_1 - S_1}{|\mathbf{x} - \mathbf{S}|^2} = \frac{1}{2\pi}\ \frac{x_1 - s_1}{|\mathbf{x} - \mathbf{S}|^2}$$

$$U_1^2(\mathbf{x}) = \frac{1}{2\pi}\ (-1)^{2-2+2+1}(1)\frac{x_2 - S_2}{|\mathbf{x} - \mathbf{S}|^2} = -\frac{1}{2\pi}\ \frac{x_2 - s_2}{|\mathbf{x} - \mathbf{S}|^2}$$

where the subscripts of U indicate the index of the partition used (in the lists above). Also, note that integration of a 0-form on a 0-dimensional manifold is equivalent to evaluation of the 0-form at the point.
Thus,

$$\begin{aligned} \psi_{\mathbf{S}} &= U_1^1(\mathbf{x})\ \mathrm{d}x_2 + U_1^2(\mathbf{x})\ \mathrm{d}x_1 \\ &= \frac{1}{2\pi}\ \frac{(x_1 - s_1)\ \mathrm{d}x_2 - (x_2 - s_2)\ \mathrm{d}x_1}{|\mathbf{x} - \mathbf{S}|^2} \\ &= \frac{1}{2\pi}\ Im\left(\frac{1}{z - \mathbf{S}_c}\ \mathrm{d}z\right) \end{aligned}$$

where in the last expression we used the complex variables, $z = x_1 + ix_2$ and $\mathbf{S}_c = s_1 + is_2$. In fact, from complex analysis (Residue theorem and Cauchy integral theorem) we know that $\int_\gamma \frac{1}{z - \mathbf{S}_c}\ \mathrm{d}z$ (where γ is a closed curve in $\mathbb{C}$) is $2\pi i$ if γ encloses $\mathbf{S}_c$, but zero otherwise. This is just the fact that

$$\int_\gamma \psi_{\mathbf{S}} = \int_{\mathrm{Ins}(\gamma)} \mathrm{d}\psi_{\mathbf{S}} = \begin{cases} \pm 1, \text{ if Ins}(\gamma) \text{ contains } S \\ 0, \text{ otherwise} \end{cases}$$

where $\mathrm{Ins}(\gamma)$ represents the inside region of the curve γ, *i.e.* the area enclosed by it.

4.5.2 $D = 3, N = 2$:

This particular case has parallels with the *Ampere's Law* and the *Biot-Savart Law* from *Electromagnetism*. This formula was used in Section 3.4 for designing a H-signature in the 3-dimensional case. Here a singularity manifold, S, is a $D - N = 1$-dimensional manifold, which, in light of Electromagnetism is a current-carrying line/wire (the *skeletons* according to terminology of Chapter 3 - Figure 4.8(b)).
The partitions in (4.4.11) for the different values of k are as follows,
For $k = 1$, $part^1(\{2, 3\}) = \Big\{\{\{2\}, \{3\}\}\ ,\ \{\{3\}, \{2\}\}\Big\}$,

For $k=2$, $part^1(\{1,3\}) = \Big\{\{\{1\},\{3\}\}\ ,\ \{\{3\},\{1\}\}\Big\}$,
For $k=3$, $part^1(\{1,2\}) = \Big\{\{\{1\},\{2\}\}\ ,\ \{\{2\},\{1\}\}\Big\}$,

Thus,

$$U_1^1(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+1+1}(1)\int_S \frac{x_1 - x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2' = -\frac{1}{4\pi}\ \int_S \frac{x_1 - x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2'$$
$$U_2^1(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+1+1}(-1)\int_S \frac{x_1 - x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3' = \frac{1}{4\pi}\ \int_S \frac{x_1 - x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3'$$

$$U_1^2(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+2+1}(1)\int_S \frac{x_2 - x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1' = \frac{1}{4\pi}\ \int_S \frac{x_2 - x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1'$$
$$U_2^2(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+2+1}(-1)\int_S \frac{x_2 - x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3' = -\frac{1}{4\pi}\ \int_S \frac{x_2 - x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3'$$

$$U_1^3(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+3+1}(1)\int_S \frac{x_3 - x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1' = -\frac{1}{4\pi}\ \int_S \frac{x_3 - x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1'$$
$$U_2^3(\mathbf{x}) = \frac{1}{4\pi}\ (-1)^{3-2+3+1}(-1)\int_S \frac{x_3 - x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2' = \frac{1}{4\pi}\ \int_S \frac{x_3 - x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2'$$

where, as before, the subscripts of U indicate the index of the partition used (in the lists above). Thus,

$$\begin{aligned}
\psi_S &= U_1^1(\mathbf{x})\ \mathrm{d}x_3 + U_2^1(\mathbf{x})\ \mathrm{d}x_2 + U_1^2(\mathbf{x})\ \mathrm{d}x_3 + U_2^2(\mathbf{x})\ \mathrm{d}x_1 + U_1^3(\mathbf{x})\ \mathrm{d}x_2 + U_2^3(\mathbf{x})\ \mathrm{d}x_1 \\
&= (U_2^2(\mathbf{x}) + U_2^3(\mathbf{x}))\ \mathrm{d}x_1 + (U_2^1(\mathbf{x}) + U_1^3(\mathbf{x}))\ \mathrm{d}x_2 + (U_1^1(\mathbf{x}) + U_1^2(\mathbf{x}))\ \mathrm{d}x_3 \\
&= \begin{bmatrix} U_2^2(\mathbf{x}) + U_2^3(\mathbf{x}) \\ U_2^1(\mathbf{x}) + U_1^3(\mathbf{x}) \\ U_1^1(\mathbf{x}) + U_1^2(\mathbf{x}) \end{bmatrix} \cdot\wedge \begin{bmatrix} \mathrm{d}x_1 \\ \mathrm{d}x_2 \\ \mathrm{d}x_3 \end{bmatrix} \\
&= \frac{1}{4\pi}\int_S \begin{bmatrix} -\frac{x_2-x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3' + \frac{x_3-x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2' \\ \frac{x_1-x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_3' - \frac{x_3-x_3'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1' \\ -\frac{x_1-x_1'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_2' + \frac{x_2-x_2'}{|\mathbf{x}-\mathbf{x}'|^3}\ \mathrm{d}x_1' \end{bmatrix} \cdot\wedge \begin{bmatrix} \mathrm{d}x_1 \\ \mathrm{d}x_2 \\ \mathrm{d}x_3 \end{bmatrix} \\
&= \frac{1}{4\pi}\int_S \frac{\mathrm{d}\mathbf{l}' \times (\mathbf{x}-\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|^3} \cdot\wedge \begin{bmatrix} \mathrm{d}x_1 \\ \mathrm{d}x_2 \\ \mathrm{d}x_3 \end{bmatrix}
\end{aligned}$$

where, bold face indicates column 3-vectors and the cross product "$\times$": $\mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}^3$ is the elementary cross product operation of column 3-vectors. The operation "$\cdot\wedge$" between column vectors implies element-wise wedge product followed by summation. Also, $\mathrm{d}\mathbf{l}' = [\ \mathrm{d}x_1'\ \ \mathrm{d}x_2'\ \ \mathrm{d}x_3']^T$. It is not difficult to identify the integral in the last expression, $\mathbf{B} = \frac{1}{4\pi}\int_S \frac{\mathrm{d}\mathbf{l}'\times(\mathbf{x}-\mathbf{x}')}{|\mathbf{x}-\mathbf{x}'|^3}$ with the *Magnetic Field vector* created by unit current flowing through S, computed using the *BiotSavart law.* Thus, if γ is a closed loop, the statement of the *Ampre's circuital law* gives, $\int_\gamma \mathbf{B}\cdot\ \mathrm{d}\mathbf{l} = \int_\gamma \psi_S = I_{encl}$, the *current enclodes by the loop.*

4.5.3 $D = 3, N = 3$:

This particular case has parallels with the *Gauss's law* in *Electrostatics*, and in general the *Gauss Divergence theorem*. Here a singularity manifold, S, is a $D - N = 0$-dimensional manifold, *i.e.* a point, the coordinate of which is represented by $\mathbf{S} = [S_1, S_2, S_3]^T$, which in the light of *Electrostatics*, is a point charge. The candidate manifolds are 2-dimensional surfaces (Figure 4.8(c)).
The partitions in (4.4.11) for the different values of k are as follows,
For $k = 1$, $\quad part^0(\{2,3\}) = \Big\{\{\{\}, \{2,3\}\}\Big\}$,
For $k = 2$, $\quad part^0(\{1,3\}) = \Big\{\{\{\}, \{1,3\}\}\Big\}$,
For $k = 3$, $\quad part^0(\{1,2\}) = \Big\{\{\{\}, \{1,2\}\}\Big\}$,

Here, $D - N = 0$ implies the integration of (4.4.10) once again becomes evaluation of 0-forms at $\mathbf{S}$. Thus,

$$\begin{aligned}
U_1^1(\mathbf{x}) &= \frac{1}{4\pi}\ (-1)^{3-3+1+1}(1)\frac{x_1 - S_1}{|\mathbf{x}-\mathbf{S}|^3} = \frac{1}{4\pi}\ \frac{x_1 - S_1}{|\mathbf{x}-\mathbf{S}|^3} \\
U_1^2(\mathbf{x}) &= \frac{1}{4\pi}\ (-1)^{3-3+2+1}(1)\frac{x_2 - S_2}{|\mathbf{x}-\mathbf{S}|^3} = -\frac{1}{4\pi}\ \frac{x_2 - S_2}{|\mathbf{x}-\mathbf{S}|^3} \\
U_1^3(\mathbf{x}) &= \frac{1}{4\pi}\ (-1)^{3-3+3+1}(1)\frac{x_3 - S_3}{|\mathbf{x}-\mathbf{S}|^3} = \frac{1}{4\pi}\ \frac{x_3 - S_3}{|\mathbf{x}-\mathbf{S}|^3}
\end{aligned}$$

Thus,

$$\begin{aligned}
\psi_S &= U_1^1(\mathbf{x})\ \mathrm{d}x_2 \wedge \mathrm{d}x_3 \ + \ U_1^2(\mathbf{x})\ \mathrm{d}x_1 \wedge \mathrm{d}x_3 \ + \ U_1^3(\mathbf{x})\ \mathrm{d}x_1 \wedge \mathrm{d}x_2 \\
&= \frac{1}{4\pi}\left(\frac{x_1 - S_1}{|\mathbf{x}-\mathbf{S}|^3}\ \mathrm{d}x_2 \wedge \mathrm{d}x_3 \ + \ \frac{x_2 - S_2}{|\mathbf{x}-\mathbf{S}|^3}\ \mathrm{d}x_3 \wedge \mathrm{d}x_1 \ + \ \frac{x_3 - S_3}{|\mathbf{x}-\mathbf{S}|^3}\ \mathrm{d}x_1 \wedge \mathrm{d}x_2 \ + \right) \\
&= \left(\frac{1}{4\pi}\frac{\mathbf{x}-\mathbf{S}}{|\mathbf{x}-\mathbf{S}|^3}\right) \cdot\wedge\ [\ \mathrm{d}x_2 \wedge \mathrm{d}x_3\ ,\ \ \mathrm{d}x_3 \wedge \mathrm{d}x_1\ ,\ \ \mathrm{d}x_1 \wedge \mathrm{d}x_2]^T \qquad (4.5.1)
\end{aligned}$$

The quantity $\mathbf{E} = \frac{1}{4\pi}\frac{\mathbf{x}-\mathbf{S}}{|\mathbf{x}-\mathbf{S}|^3}$ can be readily identified with the electric field created by an unit point charge at $\mathbf{S}$. If $\mathcal{A}$ is a closed surface, then $\int_{\mathcal{A}} \mathbf{E}\cdot\ \mathrm{d}\mathbf{A} = \int_{\mathcal{A}} \psi_S = Q_{encl}$, the charge enclosed by $\mathcal{A}$.

4.6 Examples and Applications

We implemented the general formula for computing $\psi_{\mathcal{S}}(\omega)$ in C++ programming language for arbitrary D and N. Singularity manifolds, S, and candidate manifold, ω, are discretized to create simplicial complexes $\overline{S}$ and $\overline{\omega}$ respectively, thus enabling us to compute the integral in equations (4.4.9) and (4.4.10) as a sum of integrals over pair of simplices. In the following section, for simplicity, we use the same notation of the manifolds to refer to their equivalent simplicial complex. We extensively used the Armadillo linear programming library [74] for all vector and matrix operations, and the GNU Scientific Library [32] for all the numerical integrations.

4.6.1 An Example for $D = 5, N = 3$

In Section 4.5 we have shown that the general formulation we proposed in Section 4.4 indeed reduces to known formulae that gives us the homology class invariants for certain low dimensional cases.

In this section we present numerical validation for a higher dimensional case. While we want the example to be non-trivial, we would also like it to be such that the results obtained numerically can be interpreted and verified without much difficulty. Hence we consider the following example.

Consider $D = 5$ and $N = 3$. The candidate manifold hence needs to be $N - 1 = 2$-dimensional. We consider a 2-sphere centered at the origin in $\mathbb{R}^5$ as the candidate manifold. In particular, we consider a family of candidate manifolds that is described by

$$\omega(R_C) = \{\mathbf{x} \ \mid \ x_1^2 + x_2^2 + x_3^2 = R_C^2, \quad x_4 = 0, \quad x_5 = 0\} \tag{4.6.1}$$

Correspondingly, a possible ball $\Omega(R_C)$, such that $\omega(R_C) = \partial\Omega(R_C)$, is hence given by,

$$\Omega(R_C) = \{\mathbf{x} \ \mid \ x_1^2 + x_2^2 + x_3^2 \leq R_C^2, \quad x_4 = 0, \quad x_5 = 0\} \tag{4.6.2}$$

A candidate manifold, $\omega(R_C)$, can be parametrized, which in turn can be conveniently used for triangulation (see Figure 4.9(b)), using two parameters, $\theta \in [-\frac{\pi}{2}, \frac{\pi}{2}]$ and $\phi \in [0, 2\pi]$, as follows,

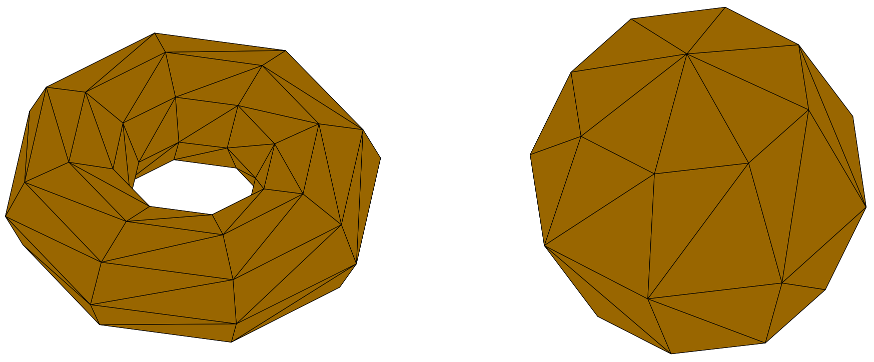
$$\begin{aligned} x_1 &= R_C \cos(\theta)\cos(\phi) \\ x_2 &= R_C \cos(\theta)\sin(\phi) \\ x_3 &= R_C \sin(\theta) \\ x_4 &= 0 \\ x_5 &= 0 \end{aligned} \tag{4.6.3}$$

We consider a single connected component as the singularity manifold, S, that is described by a 2-torus (Figure 4.9(a)) as follows,

$$\begin{aligned} x_1 &= 0 \\ x_2 &= 0 \\ x_3 &= (R_T + r\cos(\phi'))\cos(\theta') - (R_T + r) \\ x_4 &= (R_T + r\cos(\phi'))\sin(\theta') \\ x_5 &= r\sin(\phi) \end{aligned} \tag{4.6.4}$$

with $R_T > r$ and the parameters $\theta' \in [0, 2\pi]$ and $\phi' \in [0, 2\pi]$. For all examples that follow, we choose $r = 0.8, R_T = 1.6$.

Now consider the particular candidate manifold $\omega(1.0)$ (*i.e.* $R_C = 1.0$). By numerical computation of integrals in (4.4.9) and (4.4.10), the value of $\phi_S(\omega(1.0))$ that we obtain for the above example is -1. In order to interpret this result we first observe that $\omega(1.0)$ does not intersect S (*i.e.* there is no common solution for (4.6.3) and (4.6.4) with $R_C = 1.0, r = 0.8, R_T = 1.6$). However on S (Equations (4.6.4)), when $x_1 = x_2 = x_4 = x_5 = 0$, x_3 can assume the values 0, $-2r$, $-2R_T$ and $-2(R_T + r)$. Thus, if $2r > R_C$, S intersects $\Omega(R_C)$ (the ball whose boundary is $\omega(R_C)$) only at one point, *i.e.* the origin. A simple computation of the tangents revel that the intersection is transverse. Since that is a single transverse intersection with $\Omega(R_C)$, clearly the linking number between $\omega(R_C)$ and S (*i.e.* intersection number between $\Omega(R_C)$ and S according to Definition 4.3.2) is ± 1 for all $R_C < 2r$, just as indicated by the numerical analysis (*i.e.* the value of $\phi_S(\omega(1.0))$). The sign is not of importance since that is determined by our choice of orienting the manifold during triangulation.



(a) Triangulation of the singularity manifold projected on the space of x_3, x_4, x_5.

(b) Triangulation of a candidate manifold projected on the space of x_1, x_2, x_3.

Figure 4.9: A coarse triangulation using parameters θ', ϕ', θ and ϕ for creating a simplicial complex for the example in Section 4.6.1.

In fact, with different values of R_C, r and R_T, as long as $R_T > r > \frac{R_C}{2}$ is satisfied, numerically we obtain the same value of -1 for $\phi_S(\omega(R_C))$. So do we obtain by perturbation of the pose and deformation of the sphere or torus.

However with $R_C = 2.0$ for the candidate manifold, and the singularity manifold remaining the same (*i.e.* $r = 0.8, R_T = 1.6$), the value of $\phi_S(\omega(2.0))$ we obtain numerically is 0. In this case, the points at which S intersect $\Omega(2.0)$ are the origin and the point $(x_1 = x_2 = x_4 = x_5 = 0, x_3 = -0.8)$. Of course, in the family of candidate manifolds $\omega(R_C)$, $R_C \in [1.0, 2.0]$, we can easily observe that $\omega(1.6)$ indeed intersects S, thus indicating $\omega(1.0)$ and $\omega(2.0)$ possibly of different homology classes.

Next, consider the following family of candidate manifolds,

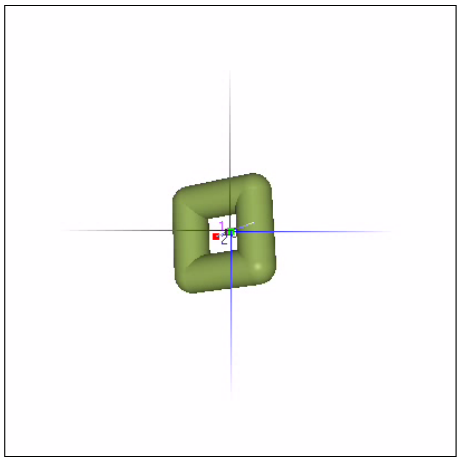
$$\omega'(T_C) = \{\mathbf{x} \mid x_1^2 + x_2^2 + x_3^2 = 2.0, \quad x_4 = 0, \quad x_5 = T_C\} \tag{4.6.5}$$

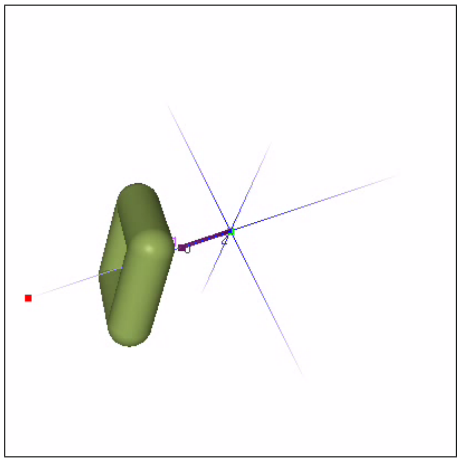
And a corresponding $\Omega'(T_C)$ such that $\omega'(T_C) = \partial\Omega'(T_C)$

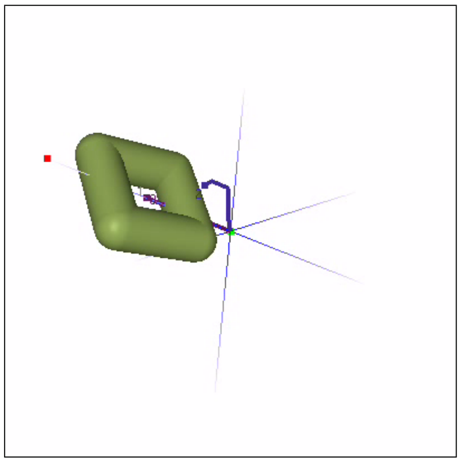
$$\Omega'(T_C) = \{\mathbf{x} \mid x_1^2 + x_2^2 + x_3^2 \leq 2.0, \quad x_4 = 0, \quad x_5 = T_C\} \tag{4.6.6}$$

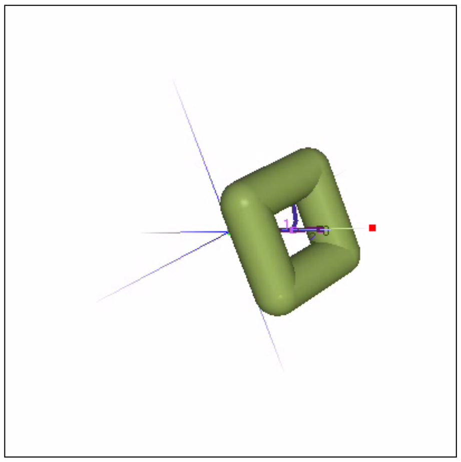
With the same S as before, if $T_C > r$, clearly there is no intersection between $\Omega'(T_C)$ and S. Thus it is not surprising that indeed by numerical computation, we found that $\phi_S(\omega'(1.0)) = 0$.

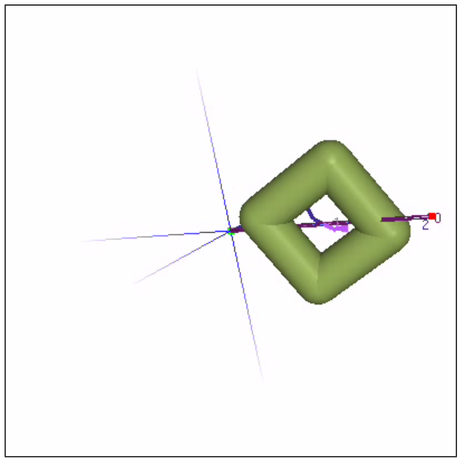
Now, since we computed $\phi_S(\omega(2.0)) = 0$ (although $\Omega(2.0)$ intersects S at 2 points) and $\phi_S(\omega'(1.0)) = 0$ (and $\Omega'(1.0)$ does not intersect S), it suggests that $\omega(2.0)$ and $\omega'(1.0)$ are in the same homology class. We will now actually try to verify that from the definition of homologous cycles (see Definition 2.1.11). It is easy to verify that none from the family of candidate manifolds $\omega'(T_C)$, $\forall T_C \in [0.0, 1.0]$ intersect S, and each is a 2-sphere. Thus ω' defines an embedding of $\mathbb{S}^2 \times I$ in $\mathbb{R}^5 - S$ such that $\omega'(0.0) \sqcup -\omega'(1.0)$ is its boundary. Taking a simplicial covering of the manifolds it follows that $\omega'(0.0)$ and $\omega'(1.0)$ are homologous. However, $\omega(2.0) = \omega'(0.0)$. Thus it follows that $\omega(2.0)$ and $\omega'(1.0)$ are homologous.


(a) $t = 1s$


(b) $t = 4s$


(c) $t = 7s$


(d) $t = 10s$


(e) $t = 13s$

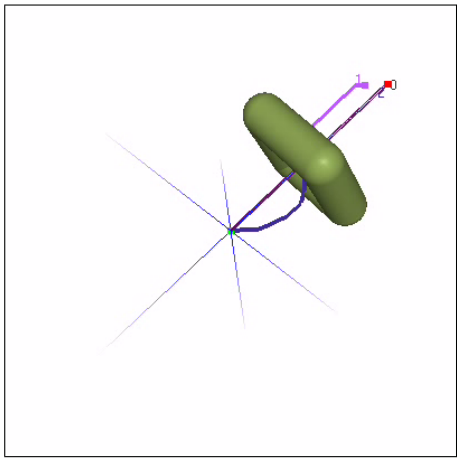

(f) $t = 16s$

Figure 4.10: Screenshots from exploration of 3 homotopy classes in a $X-Y-Z-Time$ configuration space. The loop-shaped obstacle is rotating about an axis. The blue axes are the X, Y and Z axes. Their apparent rotation is due to movement of the *camera* for viewing from different angles.

4.6.2 Exploring Paths in Different Homotopy Classes in a 4-dimensional Space

Just as we developed formulae for H-signature in the 2 and 3 dimensional cases in Chapter 3, we can now extend the formula to trajectories in higher dimensional spaces using the formula in Equation (4.4.9).

A natural extension of the example presented in Figure 3.17 would be to explore homotopy classes of trajectories in a 3-dimensional space with moving obstacles. However that makes the configuration space a 4-dimensional one consisting of the coordinates X, Y, Z and $Time$. Thus we present a result in a $X-Y-Z-Time$ configuration space where we find multiple shortest paths in different homotopy classes in the 4-dimensional space. Figure 4.10 shows the exploration of 3 homotopy classes in a 4-dimensional configuration space consisting of a dynamic obstacle in 3-dimensions. The loop-shaped obstacle is rotating about an axis. The blue axes are the X, Y and Z axes. As we observe in the sequence, trajectories numbered 0 and 1 take off from the start coordinate (green dot) and move towards the "center" of the loop. They wait there while 2 takes a different homotopy class to reach the center later. From there 0 and 2 head together towards the goal (red dot), while 1 wait to take a different trajectory to the goal. Thus the 3 trajectories are in different homotopy classes.

4.7 From Homology to Homotopy

In this section we present an informal idea without much theoretical rigor. While homotopy is significantly more difficult to deal with computationally, and it is difficult to find complete invariants for homotopy, there have been several attempts in robotics literature to construct invariants for homotopy classes of closed curves (representing trajectories) [37, 84]. We attempt to provide a relatively strong theoretical justification and generalization for such constructions. A more rigorous theoretical background is within the scope of future work.

In Proposition 4.4.1 and the following Equation (4.4.12), what we essentially had done by taking the direct sum (or *direct product* of the groups) was to take the homology groups $H_{N-1}(\mathbb{R}^D - S_k),\ k = 1, 2, \cdots, m$, and use them as generators (or basis) to create a *free abelian group*. However, we could have instead created a non-abelian group by taking the *free product* of the groups $H_{N-1}(\mathbb{R}^D - S_k)$. Let us call that group Π_{N-1} (which we will henceforth call '*Semitopy group*' for the given punctured Euclidean space). At the level of maps (*i.e.* the cycles), the group operator is the simply the free product of the cycles. Clearly, an abelianization of Π_{N-1} gives us $H_{N-1}(\mathbb{R}^D - \widetilde{\mathcal{S}})$.

From Hurewicz theorem [40] we know that an abelianization of $\pi_1(X)$ gives $H_1(X)$ for any space X. Thus, inspired by this similarity, we will mostly be looking into the case when $N = 2$. While the group Π_1 need not necessarily be isomorphic to the fundamental group of $(\mathbb{R}^D - \widetilde{\mathcal{S}})$, one may expect that it is somewhat 'closer' to it – abelianization of either gives us $H_1(\mathbb{R}^D - \widetilde{\mathcal{S}})$.

General Construction: Let $\overline{\sigma}_n = 1 \in H_n(\mathbb{S}^n),\ n > 0$ be the generator of $H_n(\mathbb{S}^n)$. Now consider a family of maps $f_j : \mathbb{S}^1 \to (\mathbb{R}^D - \widetilde{\mathcal{S}}),\ j = 1, 2, \cdots$ (image of each of which contain a base-point $\mathbf{x}_0 \in (\mathbb{R}^D - \widetilde{\mathcal{S}})$), and define corresponding n-cycles in $(\mathbb{R}^D - \widetilde{\mathcal{S}})$ as $\overline{\omega}_j := f_j \circ \overline{\sigma}_n$ (see [61], Ch. 15

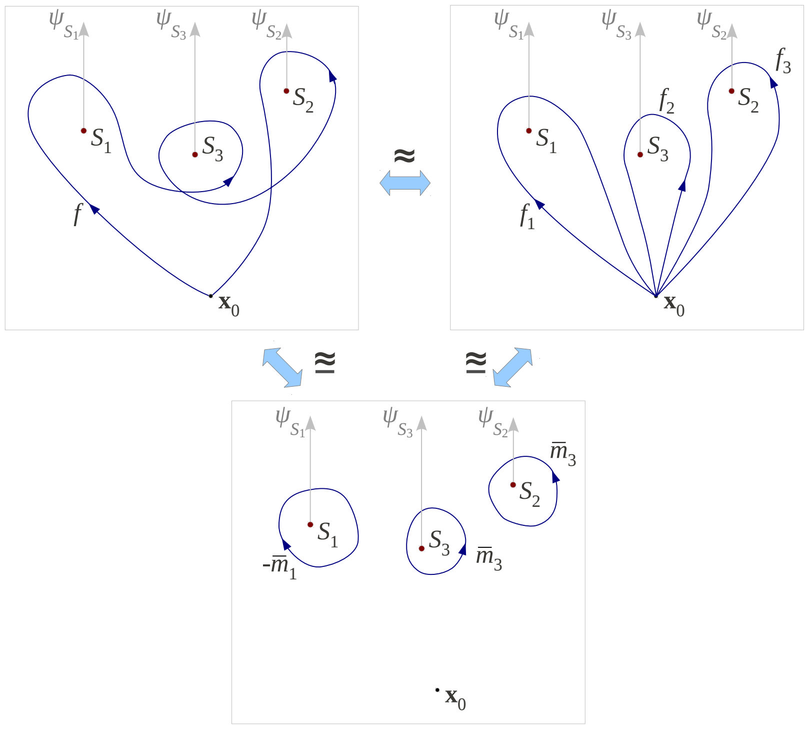


Figure 4.11: An arbitrary map $f : \mathbb{S}^1 \to (\mathbb{R}^D - \widetilde{\mathcal{S}})$ is homotopic to summation of maps, $f_1 + f_2 + f_3$, such that $\widetilde{i}_k \circ f_j$ is non-trivial for exactly one k $(= \kappa_j)$. For the f in the figure, $\kappa_1 = 1, \kappa_2 = 3, \kappa_3 = 2$. The vertical light-gray arrows show the supports of *bump* 1*-forms* in $(\mathbb{R}^D - S_k)$ – they constitute *non-intersecting support forms.*

for a similar construction).

The maps, f_j, are such that $[\widetilde{i}_k \circ \overline{\omega}_j] = \begin{cases} 0 \in H_n(\mathbb{R}^D - S_k), \text{ if } k \neq \kappa_j \\ 1 \in H_n(\mathbb{R}^D - S_k), \text{ if } k = \kappa_j \end{cases}$, where κ_j, $j = 1, 2, \cdots$, are fixed integers between 1 and m. That means $\overline{\omega}_j$ is a trivial cycle (after applying the appropriate inclusion map) in every $(\mathbb{R}^D - S_k)$ for all k, except for $k = \kappa_j$ (see Figure 4.11). Also, by construction, the image of every $\overline{\omega}_j$ contains a base-point $\mathbf{x}_0 \in (\mathbb{R}^D - \widetilde{\mathcal{S}})$.

Then we define the class (in Π_{N-1}) of any map homotopic to $f_1 + f_2 + \cdots$ (where the '+' is the addition of maps borrowed from homotopy: $w + g = (w \vee g) \circ c$, in notations of [40], Sec. 4.1) as the free product

$$[\widetilde{i}_{\kappa_1} \circ \overline{\omega}_1] * [\widetilde{i}_{\kappa_2} \circ \overline{\omega}_2] * [\widetilde{i}_{\kappa_3} \circ \overline{\omega}_3] * \cdots \in *_{k=1}^{m} H_n(\mathbb{R}^D - S_k) \approx \Pi_{N-1}$$

Henceforth, we will call this the *semitopy class* of f $(\simeq f_1 + f_2 + \cdots)$.

Definition 4.7.1 (Semitopic Decomposition, Fig.4.11)**.** Given an arbitrary map, $f : \mathbb{S}^n \to (\mathbb{R}^D - \widetilde{\mathcal{S}})$, the map is homotopic to a composition of the form $f_1 + f_2 + \cdots$, with f_j satisfying the condition mentioned above for some $\{\kappa_1, \kappa_2, \cdots\}$. This we call the semitopic decomposition of f.

We conjecture the existence of semitopic decomposition for an arbitrary f, and the uniqueness of the homotopy class of the elements in this decomposition for a particular f. That is, if $f \simeq f_1 + f_2 + \cdots \simeq f'_1 + f'_2 + \cdots$ are two semitopic decompositions, then $f_j \simeq f'_j$. If this holds, the uniqueness of the semitopy class of a semitopic decompositio is not difficult to see: If $h_k : \pi_n(\mathbb{R}^D - S_k) \rightarrow H_n(\mathbb{R}^D - S_k)$ are the Hurewicz homomorphisms [61], then, $h_{\kappa_j}([\tilde{i}_{\kappa_j} \circ f_j]) = [\tilde{i}_{\kappa_j} \circ \overline{\omega}_j]$. It is thus not difficult to see that if f_j are homotopic to maps f'_j, then $f_1 + f_2 + \cdots$ and $f'_1 + f'_2 + \cdots$ belong to the same class in Π_{N-1}. That is, $*_j\ [\tilde{i}_{\kappa_j} \circ f_j \circ \overline{\sigma}_n] = *_j\ [\tilde{i}_{\kappa_j} \circ f'_j \circ \overline{\sigma}_n]$ (where by $*_j\ q_j$ we mean the free product $q_1 * q_2 * \cdots$).

Thus by definition, if $f \simeq f'$, they are in the same semitopy class. However, the converse need not necessarily be true: Two maps f and f' may be in the same semitopy class, but in different homotopy classes. Also, it is not difficult to see that the semitopy group, Π_{N-1}, is indeed a group: If $f \simeq f_1 + f_2 + \cdots$ and $g \simeq g_1 + g_2 + \cdots$ are semitopic decompositions of f and g, then $f + g \simeq f_1 + f_2 + \cdots + g_1 + g_2 + \cdots$ is a semitopic decomposition of $f + g$.

Specialization for $n = 1$: We now specialize the above construction for $n = 1$ (in terms of our previous notations, that is $N = 2$). In this case any arbitrary closed curve is of the form $f : \mathbb{S}^1 \rightarrow (\mathbb{R}^D - \widetilde{\mathcal{S}})$. Moreover, the generators of $H_1(\mathbb{R}^D - \widetilde{\mathcal{S}})$, under the inclusion maps $\tilde{i}_{k*}$, are the generators of $H_1(\mathbb{R}^D - S_k)$ due to Proposition 4.4.1, and the image of each of these generating cycles are homeomorphic to $\mathbb{S}^1$.

It is, in general, difficult to explicitly construct a *semitopic decomposition* of an arbitrary f. However, one can indirectly compute the homology class of each element of the decomposition, $\overline{\omega}_j := f_j \circ \overline{\sigma}_1 \in Z_1(\mathbb{R}^D - S_{\kappa_j})$, without explicitly constructing the f_j's. The following discussion chalks out the basic concepts behind how this can be achieved.

In Equations (4.4.3) and (4.4.4) we had chosen a specific nontrivial element $\eta_0 \in Z_{dR}^{D-1}(\mathbb{R}^D - \{0\})$. This was a form that was symmetric about the origin. However, one can choose a different non-trivial element, even an unique one for computation of each ϕ_{S_k} in Equation (4.4.12) (say $\eta_{0,k}$), as long as $[\eta_{0,k}] \neq 0 \in H_{dR}^{(D-1)}(\mathbb{R}^D - \{0\}) \approxeq \mathbb{R}$. The type of $\eta_{0,k}$ that is of interest in the present context is as follows:

Definition 4.7.2 (Non-intersecting Support Forms with Path-connected Complement in $(\mathbb{R}^D - S_k),\ k = 1, 2, \cdots, m$)**.** Consider the maps $p_k : (\mathbb{R}^D - S_k) \times S_k \rightarrow (\mathbb{R}^D - \{0\}),\ k = 1, 2, \cdots, m$, described by $p_k(x, y) = x - y$ as previously described in Definition 4.3.1 or Section 4.4 (here we simply make the map specific for S_k instead of a general S). Let $\eta_{0,k} \in Z_{dR}^{(D-1)}(\mathbb{R}^D - \{0\})$ be closed but non-exact differential $(D-1)$-forms in $(\mathbb{R}^D - \{0\})$. We represent by $\Lambda_k \in (\mathbb{R}^D - S_k)$, the support of the 1-form, $\psi_{S_k} = \int_{\overline{S}_k} p_k^*(\eta_{0,k})$ (*i.e.* the points in $(\mathbb{R}^D - S_k)$ where the pullback of $\eta_{0,k}$ followed by integral over a top-dimensional cycle on S_k does not vanish – see Equations (4.4.11) and (4.4.2)). If $\Lambda_k \cap \Lambda_l = \emptyset,\ \forall k \neq l$, we call $\psi_{S_k},\ k = 1, 2, \cdots, m$, *non-intersecting support forms* in $(\mathbb{R}^D - S_k)$ (*i.e.* forms, whose supports do not intersect). In addition, we require that $(\mathbb{R}^D - \cup_{k=1}^m \Lambda_k)$ is path connected. We thus call ψ_{S_k} *non-intersecting support forms with path-connected complement.*

The intuition behind the above definition is that if we integrate the ψ_{S_k}'s along a 1-cycle, $\overline{\omega} \in Z_n(\mathbb{R}^D - \widetilde{\mathcal{S}})$, then at any point during the integration only one of the ψ_{S_k}'s (for a particular value of k) will be non-zero. The last condition of $(\mathbb{R}^D - \cup_{k=1}^m \Lambda_k)$ being path connected implies that if the base point, $\mathbf{x}_0$, is in $(\mathbb{R}^D - \cup_{k=1}^m \Lambda_k)$, then any point in the space of zero support can be connected by a path to $\mathbf{x}_0$ such that the path does not intersect any Λ_k.

An example of such a form can be obtained by choosing the $\eta_{0,k}$'s as the *bump* $(D-1)$-*form* [8]. It is, in particular, easy to visualize the form for $D = 2$. In Figure 4.11 the forms are represented by the light gray arrows heading upwards. In particular, if (x_k, y_k) represent the coordinates of the points S_k in the figure, the explicit formula for the forms are $\psi_{S_k}(x,y) = \delta(x-x')u(y-y')\ \mathrm{d}x\ |_{(x',y')=(x_k,y_k)} = \delta(x-x_k)u(y-y_k)\ \mathrm{d}x$, where δ is the Dirac delta function, and u is the unit step function (note that when S_k is a point, the integration over $\eta_{0,k}$ becomes a simple evaluation at the point). Clearly, the support of the form is the set $y > y_k, x = x_k$. Thus, as long as $x_k \neq x_l,\ \forall k \neq l$, the supports of these forms do not intersect. Also, it is easy to verify, by computing the integral of this ψ_{S_k} over any 1-cycle, that this form is closed but non-exact in $(\mathbb{R}^D - S_k)$. Moreover, the path-connected condition, at least for $D = 2$, is guaranteed by observing that the supports form parallel rays emanating from distinct points. A more general scenario for $D = 2$ is illustrated in Figure 4.12.

Algorithm for computing 'semitopy class': Once we are given a set of *non-intersecting support forms with path-connected complement*, and we are given a curve f, represented by the 1-cycle, $f \circ \overline{\sigma}_1 =: \overline{\omega} \in Z_n(\mathbb{R}^D - \widetilde{\mathcal{S}})$, we can propose the following algorithm for computing the class of f in Π_1: Starting from the base-point $\mathbf{x}_0$ (which we assume to lie on the image of $\overline{\omega}$), we move along the curve in a chosen direction. Due to the way we have constructed the differential forms, ψ_{S_k}, the curve will end up entering their support regions, Λ_k, and exit them in some particular sequence, one at a time. Every time it does so (say it is doing so for the j^{th} time, when it is entering and exiting $\Lambda\kappa_j$, and let $\overline{\lambda}_j$ be the part of $\overline{\omega}$ that is lying inside Λ_{κ_j} this time – see Figure 4.12), we compute the integral $v_j = \int_{\overline{\lambda}_j} \psi_{S_{\kappa_j}}$. Then the *semitopy class* of f is

$$[q_{\kappa_1}]^{v_1} * [q_{\kappa_2}]^{v_2} * [q_{\kappa_3}]^{v_3} * \cdots \quad \in \quad *_{k=1}^{m} H_n(\mathbb{R}^D - S_k)$$

where, q_k is a fixed non-trivial element (we can choose a generator) of $H_1(\mathbb{R}^D - S_k)$. Note that $[q_k]^{v_j} * [q_k]^{v_l} = [q_k]^{v_j+v_l}$.

The reason that this algorithm gives the semitopy class of f is illustrated in Figure 4.12. The point on the curve just before it enters a Λ_k, and the point on the curve just after it exists Λ_k can be joined to $\mathbf{x}_0$ using the dotted curves that do not enter any Λ_l. This is possible due to the path connectivity assumption of $(\mathbb{R}^D - \cup_{k=1}^{m}\Lambda_k)$. This gives a semitopic decomposition of $f \simeq f_1 + f_2 + \cdots$. Since the places where f differs from this semitopic decomposition $f_1 + f_2 + \cdots$ lie in regions where every ψ_{S_k} is zero, the computation using the proposed algorithm will give us the same element of $*_{k=1}^{m} H_n(\mathbb{R}^D - S_k)$ using either f or $f_1 + f_2 + \cdots$. However, by definition, the later value is the semitopy class of f. Thus we have proved that the algorithm indeed computed the semitopy class of f.

Relation to Homotopy: The condition that the semitopy group, Π_1, will be isomorphic to the fundamental group of the punctured space, $\pi_1(\mathbb{R}^D - \widetilde{\mathcal{S}})$, relies largely on the fact whether or not the Hurewicz homomorphisms, $h_k : \pi_1(\mathbb{R}^D - S_k) \to H_1(\mathbb{R}^D - S_k)$, are also isomorphisms for every k (which, in general, is not true). In addition, one needs to be able to make assertion on the existence of the *non-intersecting support forms with path-connected complement*, ψ_{S_k}. While for $D = 2$ it is

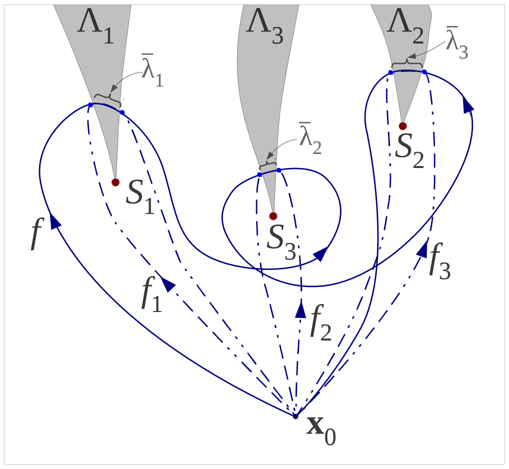


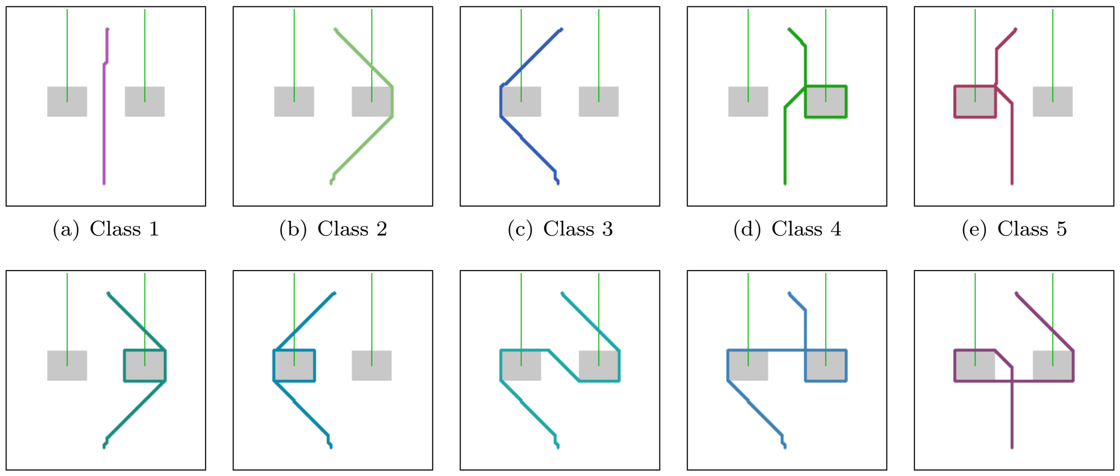
Figure 4.12: Graphical proof for the fact that the proposed algorithm computes the semitopy class of f.

definitely easy to construct the later as described, for $D \geq 3$ this definitely becomes non-trivial. For example, in $\mathbb{R}^3$ one can have two $(D - N) = 1$-dimensional manifolds, S_k and S_l, forming a link. In that case it is impossible to find the differential 1-forms that are *non-intersecting support forms with path-connected complement*. Thus, the computation prescribed in this section becomes infeasible. However, for $D = 2, N = 2$, we can observe that the semitopy group is isomorphic to the fundamental group (for which we do not provide a rigorous proof).

4.7.1 Results

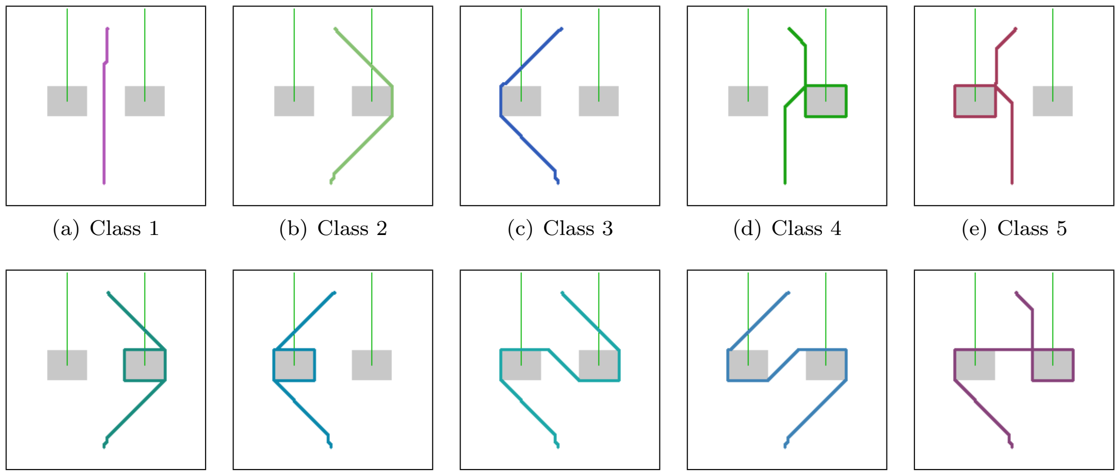
Following the constructions described above, by choosing the bump 1-form in $\mathbb{R}^2$, we can implement a robot path planning problem, similar to what we did in Chapter 3. With the choice of the bump form, $v_j = \pm 1$ depending on from which direction the trajectory crosses the support ray of $\psi_{S_{\kappa_j}}$. Instead of keeping track of the *H-signature* as we did in the *augmented graph* construction in Section 3.5, we keep track of the free product $[q_{\kappa_1}]^{v_1} * [q_{\kappa_2}]^{v_2} * [q_{\kappa_3}]^{v_3} * \cdots$ – whenever an edge of the graph crosses the support of a ψ_{S_k}, we append the corresponding element, $[q_k]^{\pm 1}$, to the free product of the parent vertex and assign it to the child vertex, followed by a possible reduction. This is, in essence, very similar to the '*word*' representation of homotopy classes of trajectories described in [37, 84].

In Figure 4.13 we explore 10 different *homology classes* of trajectories connecting a fixed pair of points (Definition 3.2.2) in a way that is very similar to the previous discussions in Chapter 3 (except that we used the bump 1-form for η_0). However, in Figure 4.14 we explore 10 different homotopy classes using the method described. Notice how they are the same until the 8^{th} class. The Class 9 in Figure 4.14(i) shows a trajectory that is in the same homology class as that of the Class 8, however it is in a different homotopy class. That is why it does not appear in Figure 4.13.


(a) Class 1 (b) Class 2 (c) Class 3 (d) Class 4 (e) Class 5

(f) Class 6 (g) Class 7 (h) Class 8 (i) Class 9 (j) Class 10

Figure 4.13: The first 10 homology classes (Definition 3.2.2) of trajectories. They are in different homotopy classes as well.


(a) Class 1 (b) Class 2 (c) Class 3 (d) Class 4 (e) Class 5

(f) Class 6 (g) Class 7 (h) Class 8 (i) Class 9 (j) Class 10

Figure 4.14: The first 10 homotopy classes of trajectories. The trajectory in the 9^{th} homotopy class was missing from Figure 4.13.

Once again, we emphasize the fact that this method works reliably for exploring homotopy classes only in $D = 2$-dimensional punctured Euclidean space, since in this dimension the *semitopy groups* are isomorphic to the homotopy groups, and we can reliably construct the required *non-intersecting support forms with path-connected complement*. For higher values of D, none of these can be guaranteed.

Chapter 5

Coverage and Exploration Using Search-based Methods

5.1 Introduction

Metric in a robot configuration space can be induced by many different problem criteria. In Chapter 3 although metric was of less concern to us, by the virtue of using an optimal search algorithm (A* algorithm), we did make use of a metric in the configuration space. Most of the times we used the Euclidean flat metric restricted to the graph to determine length of a path. However, the notion of a metric is also fundamental to much of coverage [58, 18, 17, 12] and exploration [83, 80, 79] problems in robotics. An approach towards solving coverage problems with n robots involve partitioning the configuration space into n *tessellations* (a simply connected partition). In particular, it requires a Voronoi tessellation – a metric-based tessellation. While such a tessellation is easy to achieve in a convex environment with Euclidean metric, it becomes increasingly difficult in environments with obstacles and non-Euclidean metric. But most configuration in robotics are non-convex due to presence of obstacles, and en exploration problems, as we will discuss later, non-Euclidean metric is central. In this chapter we try to develop certain tools to address those issues.

The problem of attaining good coverage of an environment is fundamental to many practical multi-robot problems. A common coverage control approach, that is efficient and can be implemented in a distributed fashion, is through the definition of feedback control laws defined with respect to the centroids of Voronoi cells resulting from the Voronoi tessellation of an environment. In [16], the authors propose gradient descent-based individual robot control laws that guarantee optimal coverage of a convex environment given a density function which represents the desired coverage distribution. The authors of [76] build upon this idea and develop decentralized control laws that position a mobile sensor network optimally with respect to a known probability distribution. In [77], this approach is extended to consider near-optimal controllers that do not require prior knowledge of a desired coverage distribution. To address the limitation of requiring a convex environment, the authors of [67] propose the use of *geodesic Voronoi tessellations* determined by the geodesic distance rather than the Euclidean distance. However such a method involves significant amount of geometric computations and work only for environments with polygonal obstacle. Besides, we would also like to be able to solve the coverage problem for non-Euclidean metric intrinsic to the configuration space. We will use a graph search based approach to develop tools for solving the coverage problem in non-Convex environments with non-Euclidean metric.

Equipped with that, by the end of this chapter we will be considering the following scenario: A team of robots enter an unknown and non-convex environment. The robots must control to explore the environment for map construction and converge to a formation in the map that disperses the robots to locations that permit them to continue to engage in activities such as persistent surveillance. This description lends itself to a broad class of robotics applications. We will hence focus on an essential component toward this scenario: The development of decentralized individual robot control laws based on uncertain estimates of the environment that drive the team of robots to explore and cover the environment. We will use a distributed implementation of the algorithm for computing generalized Voronoi tessellation of non-convex environments (using a discrete representation) in real-time for use in feedback control laws. Hence we will use an entropy-based metrics that allow for cooperative coverage control in unknown non-convex environments and at the same time achieve exploration through information gain.

5.2 Background: Coverage Functional, Voronoi Tessellation and Continuous-time Lloyd's Algorithm

In this section we discuss the basic concepts for deriving the continuous-time Lloyd's algorithm. Let $\Omega \subset \mathbb{R}^D$ be a path connected sub-manifold (with boundaries, and in general non-convex) of $\mathbb{R}^D$ that represents the environment. We assume that $Int(\Omega)$ is equipped with a metric (tensor) η which induces a distance function d in the space. Typically we will use the standard Euclidean coordinate chart inherited by Ω from $\mathbb{R}^D$ due to its embedding.

There are n mobile robots in the environment, and in particular the position of the i^{th} robot is represented by $\mathbf{p}_i \in \Omega$ and the tessellation associated with it by W_i, $\forall i = 1, 2, \ldots, n$. By definition, the tessellations are such that $Int(W_i) \cap Int(W_j) = \emptyset, \forall i \neq j$, and $\cup_{i=1}^n W_i = \Omega$. For a given set of robot positions $P = \{\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_n\}$ and tessellations $W = \{W_1, W_2, \ldots, W_n\}$ such that $\mathbf{p}_i \in W_i, \forall i = 1, 2, \ldots, n$, the *coverage functional* is defined as:

$$\mathcal{H}(P, W) = \sum_{i=1}^{n} \mathcal{H}(\mathbf{p}_i, W_i) = \sum_{i=1}^{n} \int_{W_i} f_i(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q}) \, \mathrm{d}\mathbf{q} \tag{5.2.1}$$

where $f_i : \mathbb{R} \to \mathbb{R}$ are smooth and strictly increasing functions, $\phi : \Omega \to \mathbb{R}$ is a weight or density function, and $\mathrm{d}\mathbf{q}$ represents an infinitesimal area or volume element.

The name "*coverage functional*" is indicative of the fact that $\mathcal{H}$ measures how *bad* the coverage is. In fact, for a given set of initial robot positions, P, the eventual aim of the algorithm is to devise a control law that minimizes the function $\tilde{\mathcal{H}}(P) := \min_W \mathcal{H}(P, W)$ (*i.e.* the best value of $\mathcal{H}(P, W)$ for a given P). It is easy to show [58, 67] that $\tilde{\mathcal{H}}(P) = \mathcal{H}(P, V)$, where $V = \{V_1, V_2, \cdots, V_n\}$ is the

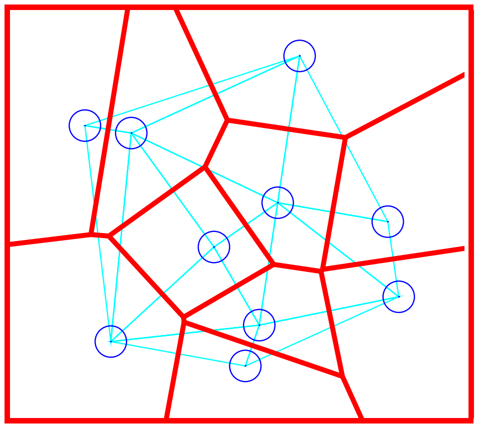


Figure 5.1: Voronoi tessellation of a convex Ω (rectangular region) with $n = 10$ robots (blue circles). The red line segments show the boundary of the tessellations. Note how a boundary segment is the perpendicular bisector of the cyan line joining the robots sharing the boundary segment.

Voronoi tessellation given by

$$V_i = \{\mathbf{q} \in \Omega \mid f_i(d(\mathbf{q}, \mathbf{p}_i)) \leq f_j(d(\mathbf{q}, \mathbf{p}_j)), \forall j \neq i\} \tag{5.2.2}$$

Whenever $\mathbf{p}_i \in Int(\Omega)$, the control law for minimizing $\tilde{\mathcal{H}}(P) = \sum_{i=1}^{n} \int_{V_i} f_i(d(\mathbf{q}, \mathbf{p}_i))\phi(\mathbf{q})\ \mathrm{d}\mathbf{q}$ can be reduced into the problem of following its gradient. Although V_i are functions of P, it can be shown using methods of differentiation under integration [67] that

$$\frac{\partial \tilde{\mathcal{H}}(P)}{\partial \mathbf{p}_i} = \int_{V_i} \frac{\partial}{\partial \mathbf{p}_i} f_i(d(\mathbf{q}, \mathbf{p}_i))\ \phi(\mathbf{q})\ \mathrm{d}\mathbf{q} \tag{5.2.3}$$

Typically one chooses $f_i(x) = x^2$ for most practical implementations. However, a variation of the problem for taking into account finite sensor footprint of the robots, constructs a *power Voronoi tessellation* [67], in which one chooses $f_i(x) = x^2 - R_i^2$, where R_i is the radius of the sensor footprint of the i^{th} robot.

Until now we haven't made any major assumption on the distance function d. However, if the space Ω is convex, and metric η is flat (Euclidean), then d is the Euclidean distance given by $d(\mathbf{x}, \mathbf{y}) = \|\mathbf{x} - \mathbf{y}\|_2$. Under this assumption, and using the form of f_i we discussed, the formula of (5.2.3) can be simplified to obtain

$$\frac{\partial \tilde{\mathcal{H}}(P)}{\partial \mathbf{p}_i} = 2(\mathbf{p}_i - \mathbf{p}_i^*) \tag{5.2.4}$$

where, $\mathbf{p}_i^* = \frac{\int_{V_i} \mathbf{q}\phi(\mathbf{q})d\mathbf{q}}{\int_{V_i} \phi(\mathbf{q})d\mathbf{q}}$, the weighted centroid of V_i. Moreover, the Euclidean distance function makes computation of the Voronoi tessellation very easy – V, due to Equation (5.2.2), can be constructed from the perpendicular bisectors of the line segments $\overline{\mathbf{p}_i\mathbf{p}_j}$, $\forall i \neq j$, thus making each V_i a convex polygon (Figure 5.1). This also enable closed-form computation of the centroid, $\mathbf{p}_i^*$ when the weight function ϕ is uniform.

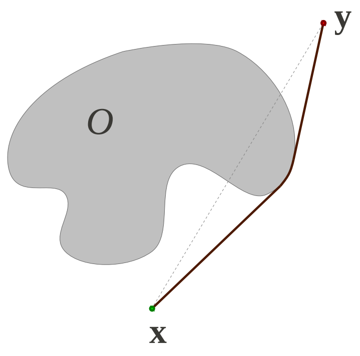


Figure 5.2: Presence of holes/punctures (due to obstacles) in the Euclidean space changes the distance function in the punctured space. For example, in this figure, $d(\mathbf{x}, \mathbf{y})$ is the length (induced by the Euclidean metric) of the thicker curve that is the shortest connecting $\mathbf{x}$ and $\mathbf{y}$ and lying entirely in $(\mathbb{R}^2 - O)$.

Equation (5.2.4) yields the simple control law in continuous-time Lloyd's algorithm: $\mathbf{u}_i = -k(\mathbf{p}_i - \mathbf{p}_i^*)$. Lloyd's algorithm [58] and its continuous-time asynchronous implementations [16] are distributed algorithms for minimizing $\mathcal{H}(P, W)$ with guarantees on completeness and asymptotic convergence to a local optimum, when Ω is convex and in an Euclidean distance setting.

5.3 Generalization to non-Euclidean Distance Function

In robot configuration spaces that are topologically Euclidean punctured by obstacles, non-Euclidean distance functions may arise in two primary ways: **i.** Due to the presence of holes/obstacles (Figure 5.2), and **ii.** A non-Euclidean metric intrinsic to the obstacle-free regions.

This makes the computation of the Voronoi tessellation in Equation (5.2.2) significantly difficult, as well as we lose the simple formula for the gradient of $\tilde{\mathcal{H}}$ as in Equation (5.2.4).

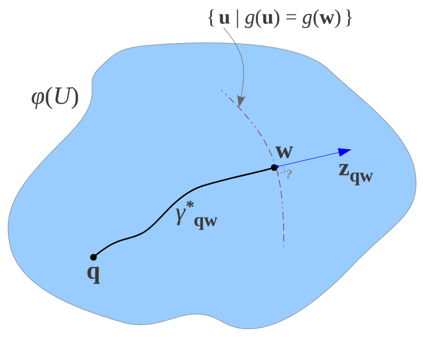
In Equation (5.2.3), with $f_i(x) = x^2 + c$, we observe that

$$\frac{\partial}{\partial \mathbf{p}_i} f_i(d(\mathbf{q}, \mathbf{p}_i)) = 2\ d(\mathbf{q}, \mathbf{p}_i)\ \frac{\partial}{\partial \mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$$

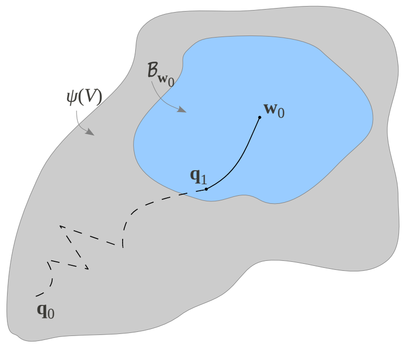
For computing $\frac{\partial}{\partial \mathbf{p}_i} d(\mathbf{q}, \mathbf{p}_i)$ we have the following proposition and corollary, which is a generalization of Proposition 4 of [67].

In the discussions that follow, we will assume summation over repeated indices, i and j, following Einstein summation convention.

Proposition 5.3.1. *Let $C = (U, \phi)$ be a coordinate chart on a open subset U of a D-dimensional manifold, Ω. Suppose U is Riemannian everywhere, equipped with a metric g. Let $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be the distance function in U in terms of the coordinate chart C, induced by the metric (i.e. $d(\mathbf{q}, \mathbf{w})$, for $\mathbf{q}, \mathbf{w} \in Img(\phi) \subseteq \mathbb{R}^D$, is the length of the shortest path connecting $\phi^{-1}(\mathbf{q})$ and $\phi^{-1}(\mathbf{w})$ in $U \subseteq \Omega$.). Suppose d is smooth everywhere inside $Img(\phi)$, and that there exists an unique geodesic*



(a) Illustration for Proposition 5.3.1. The tangent to the geodesic $\gamma^*_{\mathbf{qw}}$ at $\mathbf{w}$ is parallel to the normal to the surface $\{\mathbf{u}|g(\mathbf{u}) = g(\mathbf{w})\}$ at $\mathbf{w}$.



(b) Illustration for Corollary 5.3.2. Much of the pathologies outside $\mathcal{B}_{\mathbf{w}_0}$ do not effect the result of Proposition 5.3.1 holding for $\mathbf{q}_0$ and $\mathbf{w}_0$.

Figure 5.3: Relationship between tangent to a geodesic and the derivative of the distance function.

of length $d(\mathbf{q}, \mathbf{w})$ connecting any two points $\mathbf{q}, \mathbf{w} \in Img(\phi)$.

Then the following is true for every $\mathbf{q}, \mathbf{w} \in Img(\phi) \subseteq \mathbb{R}^D$ (Figure 5.3(a))

$$\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}, \mathbf{u})\bigg|_{\mathbf{u}=\mathbf{w}} \quad = \quad \sqrt{g_{ij}(\mathbf{w})\ z^i_{\mathbf{qw}} z^j_{\mathbf{qw}}}\ \ \mathbf{z_{qw}}$$

Where,

i. $\frac{\partial}{\partial \mathbf{u}} f = \left[\frac{\partial f}{\partial u_1}, \frac{\partial f}{\partial u_2}, \cdots, \frac{\partial f}{\partial u_D}\right]$, is the vector of derivatives of a function, $f : \mathbb{R}^D \to \mathbb{R}$, with respect to the coordinate variables of the given chart, C (note that this derivative depends on the coordinate chart and is different from the gradient operator '∇' on the metric space [47]),

ii. $\mathbf{z_{qw}}$ is a normalized (unit) vector of coefficients (in the coordinate chart C) of the tangent at $\mathbf{w}$ to the shortest geodesic connecting $\mathbf{q}$ to $\mathbf{w}$ (i.e., if the unit tangent to the geodesic is $z^i_{\mathbf{qw}} \frac{\partial}{\partial u^i}$, the coefficient vector is simply $\mathbf{z_{qw}} = \left[z^1_{\mathbf{qw}}, z^2_{\mathbf{qw}}, \cdots, z^D_{\mathbf{qw}}\right]$. The normalization condition requires $\|\mathbf{z_{qw}}\|_2 = 1$.).

Sketch of Proof. For notational convenience, let us define $g(\mathbf{u}) := d(\mathbf{q}, \mathbf{u})$, $\forall \mathbf{u} \in Img(\phi)$ (Thus, $g(\mathbf{w})$ is the length of the shortest geodesic connecting $\mathbf{q}$ to $\mathbf{w}$). Note that the notation is different from the metric g.

We start by making a graphical interpretation of the statement of the proposition. The statement of the proposition implies that the normals to the constant g surfaces in $\mathbb{R}^D$ are parallel to the tangents to the geodesics (rather the image of the geodesics in U under the action of ϕ) connecting $\mathbf{q}$ and the point at which we take the normal. This is illustrated in Figure 5.3(a).

Now consider g as a function from $\mathbb{R}^D$ to $\mathbb{R}^+$ with an unique minima at $\mathbf{q}$. Let $\gamma_{\mathbf{qw}}$ represents any arbitrary curve in $Img(\phi) \subseteq \mathbb{R}^D$ connecting $\mathbf{q}$ to $\mathbf{w}$. By the *fundamental theorem of calculus* and using the fact that $g(\mathbf{q}) = 0$, we have,

$$I(\gamma_{\mathbf{qw}}) \quad := \quad g(\mathbf{w}) \quad = \quad \int_{\gamma_{\mathbf{qw}}} \frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \cdot\ \mathrm{d}\mathbf{u} \tag{5.3.1}$$

where, $\mathrm{d}\mathbf{u}$ is the coefficient vector (in chart C) of an infinitesimal element along the tangent to the curve. (Effectively, we have simply used the Euclidean metric on $\mathbb{R}^D$ to write the expression relating gradient of an arbitrary function $g : \mathbb{R}^D \to \mathbb{R}$ to its line integral).

Now, the length of the curve $\gamma_{\mathbf{qw}}$ is given by

$$L(\gamma_{\mathbf{qw}}) \quad := \quad \int_{\gamma_{\mathbf{qw}}} \sqrt{g_{ij}(\mathbf{u}) \;\; \mathrm{d}u^i \; \mathrm{d}u^j} \tag{5.3.2}$$

By definition, the value of $L(\gamma_{\mathbf{qw}})$ is minimum when $\gamma_{\mathbf{qw}}$ is the shortest geodesic (which is unique by hypothesis – call it $\gamma^*_{\mathbf{qw}}$) connecting $\mathbf{q}$ and $\mathbf{w}$, and the minimum value is clearly $g(\mathbf{w})$ (by definition of g). Thus,

$$\begin{gathered} L(\gamma_{\mathbf{qw}}) \quad \geq \quad I(\gamma_{\mathbf{qw}}) \quad [= g(\mathbf{w}), \text{ a const. independent of } \gamma_{\mathbf{qw}}] \;\; , \\ \text{equality holds when } \gamma_{\mathbf{qw}} = \gamma^*_{\mathbf{qw}} \end{gathered} \tag{5.3.3}$$

Now, consider a family of infinitesimal elements of $\mathbb{R}^D$ represented by the coefficient vector $\mathrm{d}\mathbf{u} = [\ \mathrm{d}u_1,\ \mathrm{d}u_2, \cdots,\ \mathrm{d}u_D]$ located at an arbitrary point $\mathbf{u} \in \mathit{Img}(\phi)$ such that $\mathbf{u} +\ \mathrm{d}\mathbf{u}$ lies inside $\mathit{Img}(\phi)$. From the triangle inequality of d (since it is induced by a Riemannian metric) we have,

$$\begin{aligned} & d(\mathbf{q}, \mathbf{u} +\ \mathrm{d}\mathbf{u}) \quad \leq \quad d(\mathbf{q}, \mathbf{u}) + d(\mathbf{u}, \mathbf{u} +\ \mathrm{d}\mathbf{u}) \\ \Longrightarrow \quad & d(\mathbf{q}, \mathbf{u} +\ \mathrm{d}\mathbf{u}) - d(\mathbf{q}, \mathbf{u}) \quad \leq \quad d(\mathbf{u}, \mathbf{u} +\ \mathrm{d}\mathbf{u}) \\ \Longrightarrow \quad & \frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \bigg|_{\mathbf{u}} \cdot\ \mathrm{d}\mathbf{u} \quad \leq \quad \sqrt{g_{ij}(\mathbf{u}) \;\; \mathrm{d}u^i \; \mathrm{d}u^j} \end{aligned} \tag{5.3.4}$$

One can re-write the above inequality as

$$\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \bigg|_{\mathbf{u}} \cdot\ \widehat{\mathrm{d}\mathbf{u}} \quad \leq \quad \sqrt{g_{ij}(\mathbf{u}) \;\; \widehat{\mathrm{d}u}^i \;\; \widehat{\mathrm{d}u}^j} \tag{5.3.5}$$

where, $\widehat{\mathrm{d}\mathbf{u}}$ is the unit vector along $\mathrm{d}\mathbf{u}$.

Equality of the triangle inequality of course holds when $\mathbf{u}$ lies on the geodesic connecting $\mathbf{q}$ and $\mathbf{u} +\ \mathrm{d}\mathbf{u}$. Again, due to the property of inner product, the maximum value of $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \cdot\ \widehat{\mathrm{d}\mathbf{u}}$ would occur when $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u})$ parallel to $\widehat{\mathrm{d}\mathbf{u}}$. Thus, equality in (5.3.4) holds when $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u})$ is parallel to $\widehat{\mathrm{d}\mathbf{u}}$

Consider the case when equality holds. We let $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) = k(\mathbf{u})\, \widehat{\mathrm{d}\mathbf{u}}$. Thus, the equality condition of (5.3.5) gives $k(\mathbf{u}) = \sqrt{g_{ij}(\mathbf{u}) \;\; \widehat{\mathrm{d}u}^i \, \widehat{\mathrm{d}u}^j}$. Thus, the condition for equality is

$$\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \quad = \quad \sqrt{g_{ij}(\mathbf{u}) \;\; \widehat{\mathrm{d}u}^i \, \widehat{\mathrm{d}u}^j} \quad \widehat{\mathrm{d}\mathbf{u}} \tag{5.3.6}$$

Now consider a curve $\gamma'_{\mathbf{qw}}$ (connecting $\mathbf{q}$ and $\mathbf{w}$) such that at every point $\mathbf{u}$ on it, the tangent is parallel to $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u})$. The aforesaid maximum condition thus holds true at every point on this curve, and hence at every point on it

$$\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u}) \cdot \mathrm{d}\mathbf{u} \quad = \quad \sqrt{g_{ij}(\mathbf{u}) \ \ \mathrm{d}u^i \ \mathrm{d}u^j} \tag{5.3.7}$$

where $\mathrm{d}\mathbf{u}$ are of course infinitesimal elements at $\mathbf{u}$ on $\gamma'_{\mathbf{qw}}$ along the tangent to the curve at that point. Integrating (5.3.7) along $\gamma'_{\mathbf{qw}}$,

$$I(\gamma'_{\mathbf{qw}}) \ = \ L(\gamma'_{\mathbf{qw}}) \tag{5.3.8}$$

However, due to (5.3.3), this equality holds only when the curve is $\gamma^*_{\mathbf{qw}}$, and then its value is $g(\mathbf{w})$. By the assumption of uniqueness of shortest geodesic, $\gamma'_{\mathbf{qw}}$ should thus be the shortest geodesic.

Thus, we have proved that the curve connecting $\mathbf{q}$ to $\mathbf{w}$ for which the tangent at every point $\mathbf{u}$ on it is parallel to $\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u})$, is in fact the geodesic connecting $\mathbf{q}$ to $\mathbf{w}$. Moreover we have also shown that when that happens, the equality of (5.3.5) or (5.3.5) holds, which in turn gives the condition of 5.3.6. Thus, by changing notations and specializing for $\mathbf{u} = \mathbf{w}$, we obtain the required result.

■

Corollary 5.3.2. *Let $C = (V, \psi)$ be a coordinate chart on a open subset V of a D-dimensional manifold, Ω. Let $d : \mathbb{R}^D \times \mathbb{R}^D \to \mathbb{R}$ be the distance function on V in terms of the coordinate chart C (i.e. $d(\mathbf{q}, \mathbf{w})$, for $\mathbf{q}, \mathbf{w} \in Img(\psi) \subseteq \mathbb{R}^D$, is the distance between $\psi^{-1}(\mathbf{q})$ and $\psi^{-1}(\mathbf{w})$ in $V \subseteq \Omega$).*

We are given $\mathbf{q}_0, \mathbf{w}_0 \in Img(\psi) \subseteq \mathbb{R}^D$. Suppose there exists a open neighborhood $\mathcal{B}_{\mathbf{w}_0} \subseteq \mathbb{R}^D$ of $\mathbf{w}_0$ (Figure 5.3(b)) such that,

a. $g(\cdot) := d(\mathbf{q}_0, \cdot)$ is smooth everywhere in $\mathcal{B}_{\mathbf{w}_0}$,

b. The distance function restricted to $\mathcal{B}_{\mathbf{w}_0} \times \mathcal{B}_{\mathbf{w}_0}$ is induced by a Riemannian metric, such that for any two $\mathbf{u}, \mathbf{v} \in \mathcal{B}_{\mathbf{w}_0}$, there is an unique shortest geodesic $\gamma_{\mathbf{uv}}$ of length $d(\mathbf{u}, \mathbf{v})$.

c. A shortest path (of length $d(\mathbf{q}_0, \mathbf{w}_0)$) is defined between $\mathbf{q}_0$ and $\mathbf{w}_0$, such that the part of the shortest path connecting $\mathbf{q}_0$ and $\mathbf{w}_0$ that lies inside $\mathcal{B}_{\mathbf{w}_0}$ is unique,

Then the following holds,

$$\left. \frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_0, \mathbf{u}) \right|_{\mathbf{u}=\mathbf{w}_0} \quad = \quad \sqrt{g_{ij}(\mathbf{w}_0) \ z^i_{\mathbf{q}_0\mathbf{w}_0} z^j_{\mathbf{q}_0\mathbf{w}_0}} \ \ \mathbf{z}_{\mathbf{q}_0\mathbf{w}_0}$$

Note that the derivative $\left.\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_0, \mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0}$ is defined due to assumption 'a.', and the tangent $\mathbf{z}_{\mathbf{q}_0\mathbf{w}_0}$ exists due to assumptions 'b.' and 'c.'.

Proof. We first start by noting that the statement of this Corollary is a significant generalization of Proposition 5.3.1. We only make assumption of a Riemannian metric in the neighborhood of $\mathbf{w}_0$ (Figure 5.3(b)). This will enable us to use the result for Riemannian manifolds with boundaries (*e.g.* locally Riemannian with holes/punctures/obstacles – the kind of spaces we are most interested in), as well as opens up possibilities for more general metric spaces that may not be Riemannian outside $\mathcal{B}_{\mathbf{w}_0}$ (e.g. Manhattan metric in $\|\mathbf{u}\|_1 \leq 1$, Riemannian metric elsewhere).

Consider a shortest path $\gamma^*_{\mathbf{q}_0\mathbf{w}_0}$ connecting $\mathbf{q}_0$ and $\mathbf{w}_0$. Let $\mathbf{q}_1(\neq \mathbf{w}_0)$ be a point on this path (between $\mathbf{q}_0$ and $\mathbf{w}_0$) that lies inside $\mathcal{B}_{\mathbf{w}_0}$ (which we can always find since $\mathcal{B}_{\mathbf{w}_0}$ is open).

From the very definition of shortest path we have $\gamma^*_{\mathbf{q}_0\mathbf{w}_0} = \gamma^*_{\mathbf{q}_0\mathbf{q}_1} \cup \gamma^*_{\mathbf{q}_1\mathbf{w}_0}$, for some shortest path, $\gamma^*_{\mathbf{q}_0\mathbf{q}_1}$, connecting $\mathbf{q}_0$ and $\mathbf{q}_1$, and the shortest geodesic, $\gamma^*_{\mathbf{q}_1\mathbf{w}_0}$, connecting $\mathbf{q}_1$ and $\mathbf{w}_0$ (which is unique by assumption '*b.*'). Thus it follows that,

$$\mathbf{z}_{\mathbf{q}_0\mathbf{w}_0} = \mathbf{z}_{\mathbf{q}_1\mathbf{w}_0} \tag{5.3.9}$$

Again, by triangle inequality, for any $\mathbf{u} \in \mathcal{B}_{\mathbf{w}_0}$

$$\begin{aligned} & d(\mathbf{q}_0, \mathbf{u}) \ \leq \ d(\mathbf{q}_0, \mathbf{q}_1) + d(\mathbf{q}_1, \mathbf{u}) \\ \Rightarrow \ & g(\mathbf{u}) \ \leq \ h(\mathbf{u}) \end{aligned} \tag{5.3.10}$$

where $h(\mathbf{u}) := d(\mathbf{q}_0, \mathbf{q}_1) + d(\mathbf{q}_1, \mathbf{u})$ and $g(\mathbf{u}) := d(\mathbf{q}_0, \mathbf{u})$.
However, equality does hold when $\mathbf{q}_0$, $\mathbf{q}_1$ and $\mathbf{u}$ lie on the same shortest path. This, in particular, is true when $\mathbf{u} = \mathbf{w}_0$ (due to our choice of $\mathbf{q}_1$). Now, by our assumptions, both g and h are smooth at $\mathbf{w}_0$ (assumptions '*a.*' and '*b.*' respectively). Thus we have $g(\mathbf{u}) \leq h(\mathbf{u})$, and at $\mathbf{u} = \mathbf{w}_0$ they satisfy equality and are smooth. This implies the derivatives of the functions at $\mathbf{w}_0$ should be same,

$$\begin{aligned} \left.\frac{\partial}{\partial \mathbf{u}} g(\mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} &= \left.\frac{\partial}{\partial \mathbf{u}} h(\mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} \\ \Rightarrow \ \left.\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_0, \mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} &= \left.\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_1, \mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} \end{aligned} \tag{5.3.11}$$

Now, $\mathcal{B}_{\mathbf{w}_0}$ satisfies the conditions for U in Proposition 5.3.1, and $\mathbf{q}_1$ and $\mathbf{w}_0$ are points inside it. Thus by Proposition 5.3.1,

$$\left.\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_1, \mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} = \sqrt{g_{ij}(\mathbf{w}_0)\ z^i_{\mathbf{q}_1\mathbf{w}_0} z^j_{\mathbf{q}_1\mathbf{w}_0}}\ \ \mathbf{z}_{\mathbf{q}_1\mathbf{w}_0} \tag{5.3.12}$$

Substituting from (5.3.9) and (5.3.11) into (5.3.12) we obtain the proposed result. ■

Notes:

1. We note that when the metric is locally Euclidean in the given chart (*i.e.* $g_{ij} = \delta_{ij}$ everywhere as was the case in [67]), the result of Corollary 5.3.2 simply reduces to

$$\left.\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_0, \mathbf{u})\right|_{\mathbf{u}=\mathbf{w}_0} = \mathbf{z}_{\mathbf{q}_0\mathbf{w}_0}$$

2. If the metric is locally isotropic in the given chart (*i.e.* if the matrix representation of the metric is a multiple of the identity matrix at every point), and can be written as $g_{ij}(\mathbf{q}) =$

$\zeta(\mathbf{q})\delta_{ij}$ for some $\zeta : \mathbb{R}^D \to \mathbb{R}$, then the result of Corollary 5.3.2 reduces to

$$\frac{\partial}{\partial \mathbf{u}} d(\mathbf{q}_0, \mathbf{u})\Big|_{\mathbf{u}=\mathbf{w}_0} \quad = \quad \zeta(\mathbf{w}_0)\ \mathbf{z}_{\mathbf{q}_0 \mathbf{w}_0}$$

3. We are mostly interested in manifolds with boundaries that are locally Riemannian (*e.g.* manifold with obstacles as in Figure 5.2). For such manifolds, the condition '*b.*' of Corollary 5.3.2 holds for every $\mathbf{w}_0$ in the interior of the manifold, and the conditions '*a.*' and '*c.*' hold for almost all $\mathbf{w}_0$ except for possibly a set of measure zero (a claim through observation, for which we do not provide any proof in this thesis).

Corollary 5.3.2, along with the assumption that $f_i(x)$ is of the form $x^2 + c$, enables us to re-write Equation (5.2.3) as follows

$$\frac{\partial \tilde{\mathcal{H}}(P)}{\partial \mathbf{p}_i} = 2 \int_{V_i} d(\mathbf{q}, \mathbf{p}_i)\ \sqrt{g_{ij}(\mathbf{p}_i)\ z^i_{\mathbf{q},\mathbf{p}_i} z^j_{\mathbf{q},\mathbf{p}_i}}\ \ \mathbf{z}_{\mathbf{q},\mathbf{p}_i}\ \ \phi(\mathbf{q})\quad \mathrm{d}\mathbf{q} \tag{5.3.13}$$

With the assumption of isotropy of the metric in the given chart, this further reduces to

$$\frac{\partial \tilde{\mathcal{H}}(P)}{\partial \mathbf{p}_i} \ = \ 2\ \zeta(\mathbf{p}_i) \int_{V_i} d(\mathbf{q}, \mathbf{p}_i)\ \ \mathbf{z}_{\mathbf{q},\mathbf{p}_i}\ \phi(\mathbf{q})\ \ \mathrm{d}\mathbf{q} \tag{5.3.14}$$

This, as we will see later, gives a computable formula for the gradient of $\tilde{\mathcal{H}}$ when d is a general distance function. And once we have the gradient of $\tilde{\mathcal{H}}$, the control law for minimizing $\tilde{\mathcal{H}}$ would simply be for each robot to move in a direction opposite to the gradient

$$\mathbf{u}_i = -k \frac{\partial \tilde{\mathcal{H}}(P)}{\partial \mathbf{p}_i} \tag{5.3.15}$$

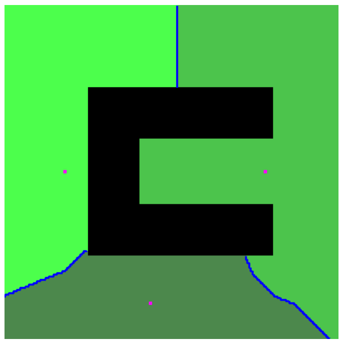
This give a generalized Lloyd's algorithm with guarantee of asymptotic stability (as easily seen by considering $\tilde{\mathcal{H}}$ a Liapunov function candidate). In the final converged solution, each robot will be at the *generalized centroid* of their respective Voronoi tessellation (since that's when the gradient of $\tilde{\mathcal{H}}$ vanishes).

Typically, since we will be computing the control commands in a discrete setup, we are mostly interested in the direction of $\mathbf{u_i}$ rather than its magnitude. Moreover, the metric in the given chart will be isotropic in most practical robot planning problems. Thus we clump the leading term $2\ \zeta(\mathbf{p}_i)$ of (5.3.14) inside k to obtain the following control law

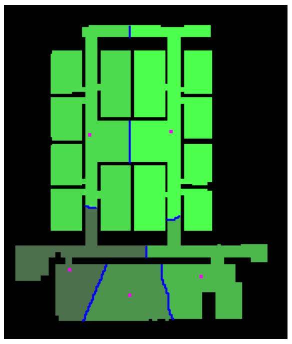
$$\mathbf{u}_i = -k \ \int_{V_i} d(\mathbf{q}, \mathbf{p}_i)\ \ \mathbf{z}_{\mathbf{q},\mathbf{p}_i}\ \phi(\mathbf{q})\ \ \mathrm{d}\mathbf{q} \tag{5.3.16}$$

5.4 Graph-search Based Lloyd's Algorithm

In order to develop a version of the Lloyd's algorithm for a general distance function, we first need to be able to compute the general Voronoi tessellation of Equation (5.2.2) for arbitrary distance function, d. We adopt a discrete graph-search based approach for achieving that. We consider a



(a) 3 robots in a simple environment (200 × 200 discretized).



(b) 5 robots in an office environment (170 × 200 discretized).

Figure 5.4: The 'geodesic Voronoi tessellation' of non-convex workspaces created using Algorithm 5.4.1 on a uniformly discretized 8-connected graph. The robot locations are marked by enlarged magenta pixels. We used a metric that is locally Euclidean. However due to non-convexity the distance function is not Euclidean.

uniform square tiling of Ω and creating a graph G out of it (See Section 2.3). The costs/weights of the edges of the graph are the metric lengths of the edges due to their embedding in Ω. It is to be noted that in doing so we end up restricting the metric of the original space to the discrete graph.

5.4.1 Graph-search Based Voronoi Tessellation

The key idea is to make a basic modification to the Dijkstra's algorithm (Algorithm 2.3.1). For creating Voronoi tessellations we initiate the *open set* with multiple start nodes from which we start propagation of the wavefronts. Thus the wavefronts emanate from multiple sources. The places where the wavefronts collide will hence represent the boundaries of the Voronoi tessellations. In addition, we can conveniently alter the distance function, the level-set of which represents the boundaries of the Voronoi tesellation. This enables us to even create *geodesic power Voronoi tessellation*. Algorithm 5.4.1 outlines the procedure. Figure 5.5 illustrates the progress of the algorithm. More examples of tessellations created using the algorithm are illustrated in Figure 5.4.

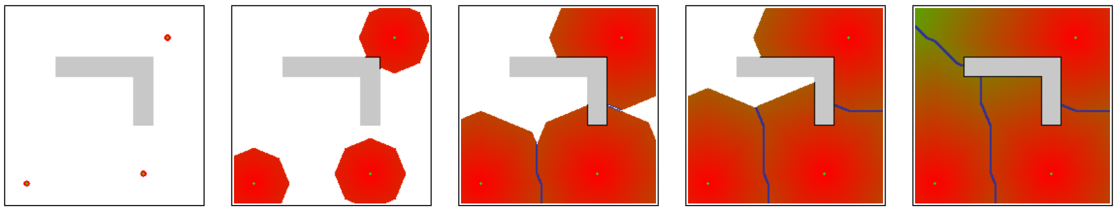

(a) *iter* = 100. The agent locations are visible. (b) *iter* = 10100. (c) *iter* = 25100. (d) *iter* = 30100. (e) *iter* = 37300.

Figure 5.5: Illustration of progress of the **Basic_Tessellation** algorithm in a environment with an L-shaped obstacle. The filled area indicates the set of expanded vertices (complement of Q). The boundaries of the tessellations are visible in blue. The graph is constructed by 200×200 uniform square discretization of the environment.

Algorithm 5.4.1

```
$\tau$ = Basic_Tessellation $(G, \{p_i\}, \{R_i\})$
Inputs:  a. Graph $G$
         b. Agent locations $p_i \in \mathcal{V}(G),\ i = 1, 2, \cdots, N$
         c. Agent weight/radius $R_i \in \mathbb{R}^+,\ i = 1, 2, \cdots, N$
Outputs: a. Tessellation map $\tau : \mathcal{V}(G) \to \{1, 2, \cdots, N\}$

Initiate $g$: Set $g(v) := \infty$, for all $v \in \mathcal{V}(G)$    // Shortest distances
Initiate $\rho$: Set $\rho(v) := \infty,\ \forall v \in \mathcal{V}(G)$    // Power distances: $f(g(v))$
Initiate $\tau$: Set $\tau(v) := -1,\ \forall v \in \mathcal{V}(G)$    // Tessellation
for each $(\{i \in \{1, 2, \cdots, N\}\})$
    Set $g(p_i) = 0$
    Set $\rho(p_i) = -R_i^2$
    Set $\tau(p_i) = i$
Set $Q := \mathcal{V}(G)$    // Set of un-expanded nodes
while $(Q \neq \emptyset)$
    $q := \text{argmin}_{q' \in Q}\ \rho(q')$    // Maintained by a heap data-structure.
    if $(g(q) == \infty)$
        break
    Set $Q = Q - q$    // Remove $q$ from $Q$
    Set $j := \tau(q)$
    for each $(\{w \in \mathcal{N}_G(q)\})$    // For each neighbor of $q$
        Set $g' := g(q) + \mathcal{C}_G([q, w])$
        Set $\rho' :=$ PowerDist$(g', R_j)$
        if $(\rho' < \rho(w))$
            Set $g(w) = g'$
            Set $\rho(w) = \rho'$
            Set $\tau(w) = j$
return $\tau$
```

where, **PowerDist**$(x, r) = x^2 - r^2$

The procedure returns the map τ, that gives for each vertex in the graph, the index of the tessellation it belongs to. Note that $g(q)$ now contains the geodesic distances (up to approximation due to restriction to the graph) of q from the location of that robot whose tessellation contains q.

5.4.2 Algorithm for Tessellation and Control Computation

In order to compute the control command for the robots (*i.e.* the action of the robot in the next time step), we use the formula in Equation (5.3.16). In a general metric setup, the vector $\mathbf{z}_{\mathbf{q},\mathbf{p}_i}$ is the unit vector along the tangent at $\mathbf{p}_i$ to the geodesic joining $\mathbf{q}$ to $\mathbf{p}_i$. In a discretized setup, the unit vectors $\mathbf{z}_{\mathbf{q},\mathbf{p}_i}$ is approximated as the unit vectors along edges of the form $[p'_i, p_i]$ for some $p'_i \in \mathcal{N}_G(p_i)$ such that the shortest path in the graph connecting p_i and q passes through p'_i. For a given q, we know that $\tau(q)$ is the index of the robot whose tessellation it belongs to, and can compute the shortest path in the graph joining the nodes $p_{\tau(q)}$ and q. The neighbor of $p_{\tau(q)}$ through which the shortest path passes is the desired $p'_{\tau(q)}$, and it is maintained in the variable $\eta(q)$ in an efficient way as described in Algorithm 5.4.2. We can also compute the integration of (5.3.16) on the fly as we compute the tessellations.

Algorithm 5.4.2

```
$\{\tau, \{p'_i\}\}$ = **Tessellation_and_Control_Computation** $(G, \{p_i\}, \{R_i\}, \overline{\phi})$
Inputs:  a. Graph $G$
         b. Agent locations $p_i \in \mathcal{V}(G),\ i = 1, 2, \cdots, N$
         c. Agent weight $R_i \in \mathbb{R}^+,\ i = 1, 2, \cdots, N$
         d. Discretized weight/density function $\overline{\phi} : \mathcal{V}(G) \rightarrow \mathbb{R}$
Outputs: a. The tessellation map $\tau : \mathcal{V}(G) \rightarrow \{1, 2, \cdots, N\}$
         b. The next position of each robot, $p'_i \in \mathcal{N}_G(p_i),\ i = 1, 2, \cdots, N$
Initiate $g$: Set $g(v) := \infty$, for all $v \in \mathcal{V}(G)$   // Shortest distances
Initiate $\rho$: Set $\rho(v) := \infty,\ \forall v \in \mathcal{V}(G)$   // Power distances
Initiate $\tau$: Set $\tau(v) := -1,\ \forall v \in \mathcal{V}(G)$   // Tessellation
Initiate $\eta$: Set $\eta(v) := \emptyset,\ \forall v \in \mathcal{V}(G)$ // Pointer to robot neighbor. $\eta : \mathcal{V}(G) \rightarrow \mathcal{V}(G)$
for each ($\{i \in \{1, 2, \cdots, N\}\}$)
    Set $g(p_i) = 0$
    Set $\rho(p_i) = -R_i^2$
    Set $\tau(p_i) = i$
    Set $\mathbf{I}_i := \mathbf{0}$   // The control integral (negative of gradient of $\tilde{\mathcal{H}}$). $\mathbf{I}_i, \mathbf{0} \in T\mathscr{C}$
    for each ($\{q \in \mathcal{N}_G(p_i)\}$)   // For each neighbor of $p_i$
        Set $\eta(q) = q$
Set $Q := \mathcal{V}(G)$   // Set of un-expanded nodes
while ($Q \neq \emptyset$)
    $q := \text{argmin}_{q' \in Q}\ \rho(q')$   // Maintained by a heap data-structure.
    if ($g(q) == \infty$)
        break
    Set $Q = Q - q$   // Remove $q$ from $Q$
    Set $j := \tau(q)$
    Set $s := \eta(q)$
    if ($s\ != \emptyset$)   // Equivalently, $q \notin \{p_i\}$
        Set $\mathbf{I}_j$ += $g(q) \times \frac{\mathbf{P}(s) - \mathbf{P}(p_j)}{\|\mathbf{P}(s) - \mathbf{P}(p_j)\|_2} \times \overline{\phi}(q)$   // Integral in Eq. (5.3.16)
    for each ($\{w \in \mathcal{N}_G(q)\}$)   // For each neighbor of $q$
        Set $g' := g(q) + \mathcal{C}_G([q, w])$
        Set $\rho' :=$ **PowerDist**$(g', R_j)$
        if ($\rho' < \rho(w)$)
            Set $g(w) = g'$
            Set $\rho(w) = \rho'$
            Set $\tau(w) = j$
            if ($s\ != \emptyset$)   // Equivalently, $q \notin \{p_i\}$
                Set $\eta(w) = s$
for each ($\{i \in \{1, 2, \cdots, N\}\}$)
    Set $p'_i := \text{argmax}_{u \in \mathcal{N}_G(p_i)}\ \frac{\mathbf{P}(u) - \mathbf{P}(p_i)}{\|\mathbf{P}(u) - \mathbf{P}(p_i)\|_2} \cdot \mathbf{I}_i$   // Choose action best aligned along $\mathbf{I}_i$.
return $\{\tau, \{p'_i\}\}$
```

where, $T\mathscr{C}$ is the action space of each robot (generally the tangent space - a normed vector space, assumed to be the same for all robots). Typically, $T\mathscr{C} = \mathbb{R}^D$. Also, the function $\mathbf{P} : \mathcal{V}(G) \rightarrow \mathscr{C}$ is such that $\mathbf{P}(q)$ gives the coordinate of the node at q. Thus, $\mathbf{P}(p_i) = \mathbf{p}_i$ and $\mathbf{P}(q) = \mathbf{q}$ in relation to Equation (5.3.16). In an uniform discretization setting we take $\overline{\phi}(q) = \kappa\ \phi(\mathbf{P}(q))$ for an arbitrary positive constant κ.

5.4.3 Overall Algorithm: Adapted Lloyd's Algorithm

The overall algorithm consists of iterating over "**Tessellation_and_Control_Computation**" and updating the positions of the robots (so that they move along the approximate negative gradient of $\tilde{\mathcal{H}}$) at each iteration.

Algorithm 5.4.3

```
$\{\tau^f, \{p_i\}^f\}$ = Adapted_Lloyds $(G, \{p_i\}, \{R_i\}, \overline{\phi})$
Inputs:  a. Graph $G$
         b. Initial agent locations $p_i \in \mathcal{V}(G),\ i = 1, 2, \cdots, N$
         c. Agent weight $R_i \in \mathbb{R}^+,\ i = 1, 2, \cdots, N$
         d. Discretized density function $\overline{\phi} : \mathcal{V}(G) \to \mathbb{R}$
Outputs: a. Final tessellation map $\tau : \mathcal{V}(G) \to \{1, 2, \cdots, N\}$
         b. Robot final position, $p_i' \in \mathcal{V}(G),\ i = 1, 2, \cdots, N$
1   Initiate $H^0 := \{\mathbf{P}(p_1), \mathbf{P}(p_2), \cdots, \mathbf{P}(p_N)\}$   // Initiate history of robot positions
2   Set $t := 0$
3   while $(t < m$  OR  $\mathbf{er}(H^{(t-m):t}) > \epsilon)$ // not converged
4       Set $\{\tau, \{p_i'\}\}$ := Tessellation_and_Control_Computation $(G, \{p_i\}, \{R_i\}, \overline{\phi})$
5       for each $(\{i \in \{1, 2, \cdots, N\}\})$
6           Move $i^{\text{th}}$ robot from $\mathbf{P}(p_i)$ to $\mathbf{P}(p_i')$
7           Set $p_i = p_i'$   // Update robot positions
8       Set $t$ += 1
9       Set $H^t := \{\mathbf{P}(p_1), \mathbf{P}(p_2), \cdots, \mathbf{P}(p_N)\}$   // Append current position to history
10  return $\{\tau, \{p_i\}\}$   // Latest tessellation & positions
```

where,

$$\mathbf{er}(H^{a:b}) = \frac{1}{1+b-a} \max_{i \in \{1,2,\cdots,N\}} \left(\sum_{h=a}^{b} \left| H_i^h - \tfrac{1}{1+b-a} \sum_{l=a}^{b} H_i^l \right|^2 \right)$$

is a measure of the variation in the position of the sensors over the most recent $1+b-a$ time steps. Convergence is hence detected by checking if the variation in the positions over the most recent m time-steps is less than a desired threshold, ϵ.

5.4.4 Results

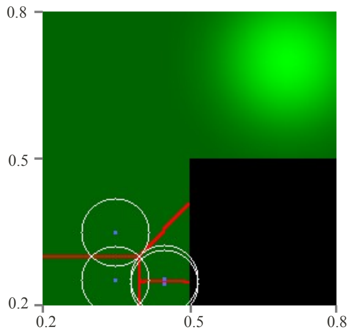
We implemented the above algorithm in C++ programming language. The 2-dimensional configuration space of the robots was uniformly discretized into square cells and the graph G was created out of it by placing a node in each free cell. Nodes were not placed inside obstacles.

L-shaped Environment

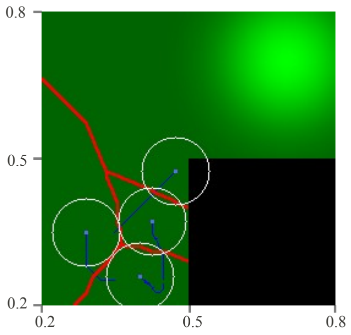
The first set of results demonstrate the coverage of the L-shaped environment described in Figure 5.6 by 4 homogeneous robots. The environment is discretized into 240×240 cells. The density function is given by $\phi(\mathbf{q}) = k(\|\mathbf{q} - \mathbf{q}_c\|_2 / h)$, where,

$$k(\kappa) = \begin{cases} 1 - \frac{3}{2}\kappa^2 + \frac{3}{4}\kappa^3 & \text{if} \quad 0 \le \kappa \le 1\,, \\ \frac{1}{4}(2-\kappa)^3 & \text{if} \quad 1 \le \kappa \le 2\,, \\ 0 & \text{otherwise}\,, \end{cases}$$

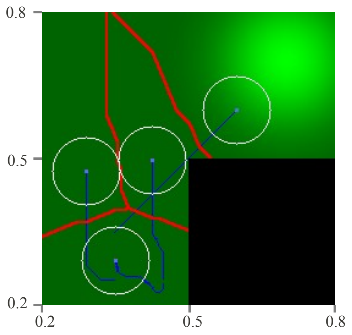
with $\mathbf{q}_c = [0.7, 0.7]^T$, $h = 0.15$ and the value of ϕ is normalized between 0.1 and 2.0. Figures 5.6 shows snapshots of the robot configuration at different iterations. The intensity of green represents the weight function. The curves in red are the boundaries of the tessellations, and the white circles represent the radii of the robots' sensor footprints. The robots start near the bottom of the environment. Note how in the final configuration the distribution of the robots is biased toward



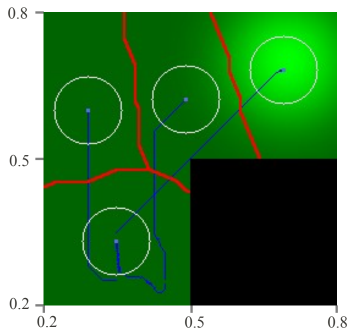
(a) $t = 0$.



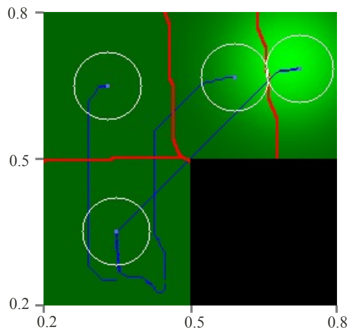
(b) $t = 50$.



(c) $t = 100$.



(d) $t = 150$.



(e) $t = 200$ (*Converged*).

Figure 5.6: Adapted Lloyd's algorithm in a L-shaped environment. The environment is 0.6×0.6 units in size, discretized into 240×240 cells, and have an obstacle occupying the lower right quadrant. Intensity of green represent magnitude of the weight function. The red lines are the boundaries of the tessellations. The robots start off from the lower left corner of the environment, follow the **Adapted_Lloyds** algorithm, and finally attain convergence with good coverage of the environment.

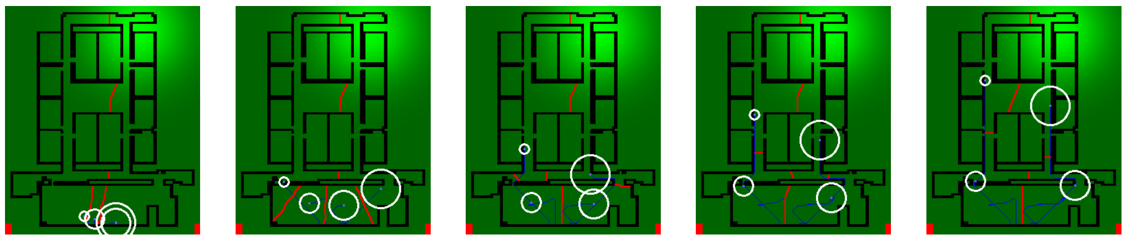
the center of the density function, $\mathbf{q}_c$. The **Adapted_Lloyds** algorithm along with the plotting procedures run at a rate of 17 Hz on a single processor (2.1 GHz, 3Gb RAM) machine.

In a Complex Indoor Environment

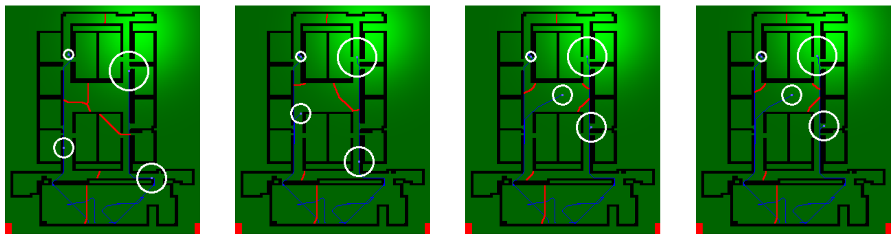
Next we test the algorithm on a more complex office-like indoor environment of dimension 2.84×3.33 units (284×333 discretized) and the origin at the center of the environment. There are 4 heterogeneous robots. The robots' sensor foot-print radii are chosen as $0.1, 0.2, 0.3$ and 0.4 units as illustrated in Figure 5.8. Figure 5.7 demonstrates the results. The weight/density is chosen to be the Gaussian function $\phi(\mathbf{q}) = 0.135\ e^{2.0|\mathbf{q}-\mathbf{q}_c|^2}$, with $\mathbf{q}_c = [0.5, 1.3]^T$.

Distributed Implementation

In a distributed implementation, each robot is assumed to have a copy of the graph G, discretized density $\overline{\phi}$, and have knowledge about the weight and location of neighbor robots. This information may be available assuming communication among neighbor robots. By neighbors we mean robots that share boundaries of the computed tessellation. Hence each robot is able to run locally



(a) $t = 10$. (b) $t = 60$. (c) $t = 110$. (d) $t = 160$. (e) $t = 210$.



(f) $t = 260$. (g) $t = 310$. (h) $t = 360$. (i) $t = 410$ (*Converged*).

Figure 5.7: Adapted Lloyd's algorithm in a real indoor environment. The environment is 2.84×3.33 units in size, discretized into 284×333 cells, so that each cell is 0.01×0.01 units in size. The radii of the robot footprints are $0.1, 0.2, 0.3$ and 0.4 units. The robots start off in the big room in the bottom of the environment, and follow the **Adapted_Lloyds** algorithm to attain coverage. Note that the final distribution of the robots is biased towards regions of high density function.

Figure 5.8: Radii of the sensor footprints of the robots in simulation of Fig. 5.7.

the "**Tessellation_and_Control_Computation**" algorithm on its own local processor using only local information. In the distributed version of the "**Adapted_Lloyds**" algorithm, each robot is interested in computing its own control command and issuing it. The control commands for neighbor robots are not computed on the local processor or are discarded. After every time step the robots broadcast their new positions (from a ground truth or any other localization method) and receive updated positions of neighbors. This communication closes the control loop and accounts for errors in the execution of computed controls.

5.5 Application to Simultaneous Coverage and Exploration Problem

So far the metric in Ω we have considered was Euclidean (flat). The non-Euclidean distance function arose due to non-convexity of Ω. In this section we will present an example from robot exploration problem where non-Euclidean metric arise quite naturally.

We consider the problem of deploying n mobile robots in an unknown or partially known environment, which upon collaborative exploration of the environment, will converge to an optimal or near-optimal coverage.

5.5.1 Entropy as Density Function

In order to address this problem each mobile robot maintains and communicates a probability map for the discretized environment such that $p(\mathbf{q})$ is the probability that the vertex $\mathbf{q}$ is inaccessible (*i.e.* occupied or represents an obstacle), for all $\mathbf{q} \in \mathcal{V}(G)$. A threshold on the value of probability determines whether a particular node in $\mathcal{V}(G)$ is occupied/inaccessible for computation of the Geodesic Voronoi tessellations as well as control. Moreover the *Shannon entropy* for each cell can be computed as follows,

$$e(\mathbf{q}) = p(\mathbf{q}) \ln(p(\mathbf{q})) + (1 - p(\mathbf{q})) \ln(1 - p(\mathbf{q})). \tag{5.5.1}$$

[Note that for simplicity we used the same notation for a vertex and its coordinate in the original space.] This gives us an *Entropy map*, *i.e.* a value of entropy associated with each vertex $\mathbf{q}$ - a map that represents uncertainty or the need to gather information within the environment. The Shannon entropy is such that it assumes high values for vertices for which the uncertainty is high (i.e. probability is close to 0.5), whereas it is low for known or visited vertices. Thus, we identify the weight or density function $\phi(\cdot)$ with the entropy $e(\cdot)$.

$$\phi(\mathbf{q}) = e(\mathbf{q}) \tag{5.5.2}$$

This, by the construction of the control laws described before, will drive the mobile robots towards regions of high entropy within the robot's own tessellation, hence resulting in exploration of the environment.

5.5.2 Entropy-Based Metric

For exploration and for environments with uncertainty it is desired that tessellation boundaries be such that they "bisect" the uncertainty (or entropy) among the adjacent robots for cooperative exploration. This notion is illustrated in Fig. 5.9, where a high entropy region is placed asymmetrically between two robots in a convex environment without obstacles. The dashed line shows the boundary of a Voronoi tessellation created using the standard Euclidean metric. However, one mobile robot has a larger unexplored region than the other. An alternate division is depicted with a solid line that splits the unexplored region equally. This division is a result of weighting the metric with the entropy.

In particular, the matrix representation of the metric tensor in the standard coordinate chart of the plane is given by,

$$\eta(\mathbf{q}) = e(\mathbf{q}) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{5.5.3}$$

In terms of finding the Voronoi tessellations and control commands using Algorithm 5.4.2, the

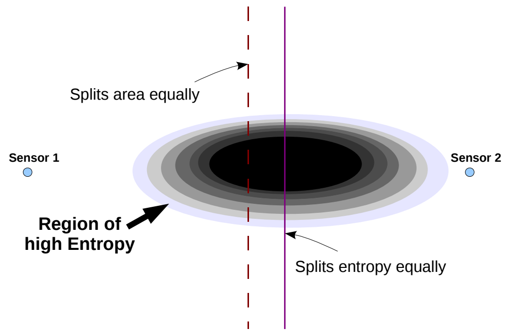


Figure 5.9: Entropy-weighted Voronoi tessellation.

only required change is to weigh the edges of the graph G by entropy in those regions instead of the Euclidean length of the edges. In particular, we define the cost of an edge $[\mathbf{v}_s, \mathbf{v}_t] \in \mathcal{V}(G)$ using the following approximation

$$\mathcal{C}_G(\varepsilon) = \left(\frac{e(\mathbf{v}_s) + e(\mathbf{v}_t)}{2} + \epsilon \right) \|\mathbf{v}_s - \mathbf{v}_t\|_2 \tag{5.5.4}$$

where ϵ is a small positive integer chosen to compensate for noise in near-zero values of entropy and to make sure that the cost of an edge doesn't vanish. We can hence use the "**Tessellation_and_Control_Computation**" algorithm on this graph to compute the tessellations and control commands.

5.5.3 Time Dependence of Entropy, Coverage, and Convergence

We now detail how the probability map is updated based on the sensor readings. For the discussion that follows, $p^t(\mathbf{q})$ represents the estimated probability of occupancy of $\mathbf{q}$ at the t^{th} iteration based on all measurements.

Inter-robot Communication

As discussed earlier, for the distributed architecture, each robot maintains its own copy of probability and entropy maps. Each updates its own maps based on readings from its own on-board sensor as well as information acquired from its neighboring robots about parts of their copies of their probability maps. A sensor fusion model (described in next section) is used to aggregate the data. For communicating its own probability map to other robots, each robot broadcasts the *new* information acquired by its own sensor over a *time window* or *phase*. Essentially the broadcasting of probability maps by each robot is done in *phases*. During a phase, a robot broadcasts a constant message (part of its own probability map) with a fixed timestamp over and over (repeatedly). This is to make sure that other robots receive this message. The robot also broadcasts its unique identity along with the message. Also, instead of broadcasting the whole probability map in each phase, each robot broadcasts only whatever new it has sensed during the previous broadcast phase. Thus the broadcasted information actually comprises of a small window in the whole probability map as

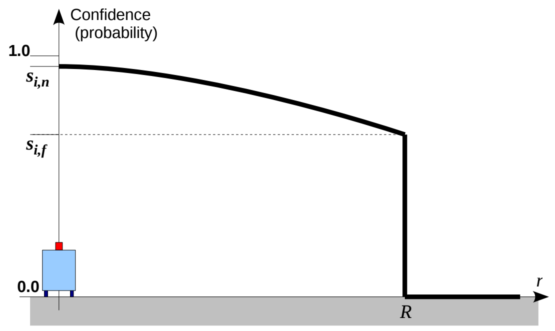


Figure 5.10: The sensor model.

well as in time, inside which the probability readings have changed. This makes each broadcast messages rather small. Essentially each robot maintains two buffers: The *current sensing buffer*, and the *broadcast buffer*. New readings from a robot's own laser sensor are added to the current sensing buffer, while things in the broadcast buffer are broadcasted. At the end of a broadcast *phase* the content of the broadcast buffer is pushed into the main probability map maintained by the robot, the content of the current sensing buffer is copied into the broadcast buffer, and the current sensing buffer is cleared for new sensor data. The information received from other robots about their map are directly added to the main probability map. This *differential* approach of communication significantly reduces the communication overhead required for sharing map data.

Sensor Model

We use a sensor model for each robot, $s_i(r)$, which gives the probability that the i^{th} robot's sensor measures the state of a grid cell located at a distance r from it correctly. In particular, in our simulations we use,

$$s_i(r) = \begin{cases} s_{i,n} + \frac{r^2}{R_i^2}(s_{i,f} - s_{i,n}) & \text{if } r \leq R_i \\ 0 & \text{otherwise,} \end{cases}$$

where R_i is the sensor range, and $0 \leq s_{i,f} \leq s_{i,n} \leq 1$ gives the far and near values of the confidence of the sensor.

Thus, if at time-step t the sensor of the i^{th} robot receives a measurement $z_i^t(\mathbf{q})$ (which is 1 for occupied, and 0 for unoccupied) for the cell $\mathbf{q}$, the probability that the cell is occupied based only on this measurement is given by $u_i^t(\mathbf{q}) = z_i^t(\mathbf{q})\ s_i(\|\mathbf{q} - \mathbf{p}_i^t\|) + (1 - z_i^t(\mathbf{q}))(1 - s_i(\|\mathbf{q} - \mathbf{p}_i^t\|))$. We use a sensor fusion model to compute the net probability of occupancy for the cells based on the individual measured probabilities. In particular, one can compute $p^t(\mathbf{q}) = g^{-1}\left(\frac{\sum_{i,t'} g(u_i^{t'}(\mathbf{q}))}{\sum_{i,t'} 1}\right)$, where g is a strictly increasing function in $[0, 1]$, and the summations are taken over all the measurements by all sensors over all time instants [83]. For our experiments we choose $g(x) = x^m,\ m > 0$. We note that by choosing $m \to \infty$, the value of $p^t(\mathbf{q})$ essentially becomes the *supremum* of all the measurements for $\mathbf{q}$. Alternatively, choosing $g(\cdot) = \log(\cdot)$ gives the geometric mean of the measurements, which has also been used in [83].

In order to compensate for the sensor noise the entropy map is smoothed by passing it through a min-filter. The smoothed entropy map is consequently used for computing the density function.

Time-varying Density Function

A consequence of updating the probability map is that the entropies, and hence the metric, d, and the weight function, ϕ, becomes a function of time. This, in general, is inconsistent with the formulation of the generalized Lloyd's algorithm prescribed by Equation (5.3.13) since the function $\tilde{\mathcal{H}}$ is not a Liapunov candidate any more since it varies with time. Thus we lose the guarantee of convergence. However, if we choose the weight function in Equations (5.5.2 - 5.5.4) with a slight modification, and with some assumption on the sensor model, we can obtain guarantees on stability.

Lemma 5.5.1 (Exploration and Convergence Guarantee)**.** *Suppose there exists an ϵ radius around each mobile robot such that it is able to sense the occupancy and reduce the entropy of the cells within that radius below the value of τ' in a permanent manner. Then there exists a $\tau \geq \tau'$, and a small value ι, such that instead of the weight/density function mentioned in Equations (5.5.2), if we choose*

$$\phi(\mathbf{q}, t) = \begin{cases} \iota, & \text{if } e(\mathbf{q}, t) < \tau \\ e(\mathbf{q}, t), & \text{otherwise} \end{cases},$$

and instead of entropy, e, in the Equation (5.5.4) for metric, if we use $e'(\mathbf{q}, t) = \max(e(\mathbf{q}, t), \tau)$, then we can guarantee complete exploration of the environment (entropy of every cell being reduced below the value of τ) and convergence of the algorithm.

Proof. The small value ι is chosen as a representative of zero for numerical stability. Ideally, it should be several orders of magnitude smaller than τ'.

The main idea of the proof lies in the observation that the control law of Equation (5.3.15) drives the agents towards the instantenious *weighted generalized centroids* [67] of their respective Voronoi tessellations. However, by the above construction of the weight function, ϕ, the weighted generalized centroid of V_i at time t will always lie inside the set $\{\mathbf{q} \in V_i \mid e(\mathbf{q}, t) \geq \tau\}$. Thus the robot will be driven towards a region where the entropy is greater than or equal to τ. In doing so it will eventually reduce the entropy of the region below τ'. This will continue until the entropy of every cell in Ω gets reduced below τ'.

Once that is attained, we have $\phi(\mathbf{q}, t) = \iota$ and $e'(\mathbf{q}, t) = \tau$, which are constant for all t. The metric and weight/density function become independent of time, thus ensuring convergence. ∎

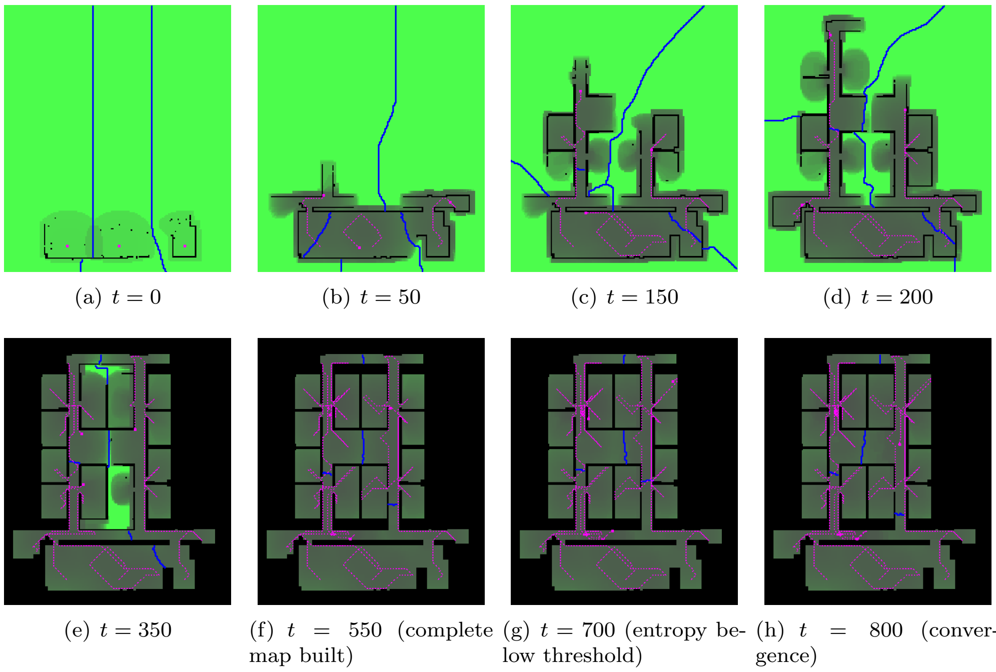
The Overall Algorithm

So far we have described the various components of the algorithm. To put those in perspective, the steps below are what goes on at a higher level on each robot in sequence while exploring and covering an unknown or partially known environment in a distributed fashion.

i. Each robot maintains its own probability, entropy and obstacle maps.

ii. Each robot use sensor data as well as communicate with its neighbors to update the maps. They also communicate their locations.

iii. Each robot computes its own entropy-weighted Voronoi tesellation and the corresponding control commands, and take a step accordingly.


(a) $t = 0$ (b) $t = 50$ (c) $t = 150$ (d) $t = 200$

(e) $t = 350$ (f) $t = 550$ (complete map built) (g) $t = 700$ (entropy below threshold) (h) $t = 800$ (convergence)

Figure 5.11: Exploration and coverage of an unknown environment

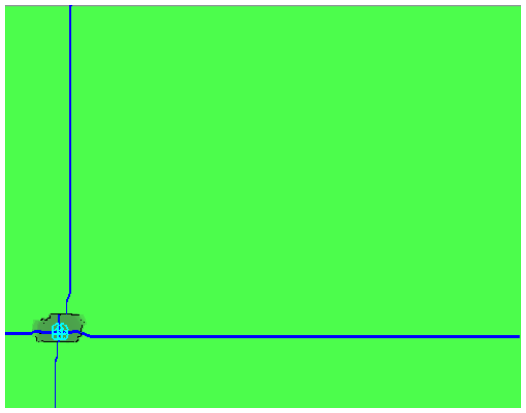
5.5.4 Results

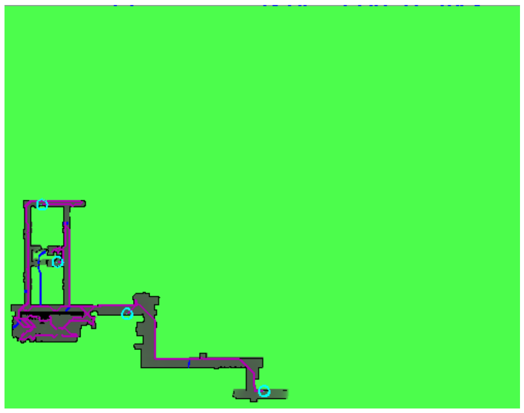
Single Thread Implementation

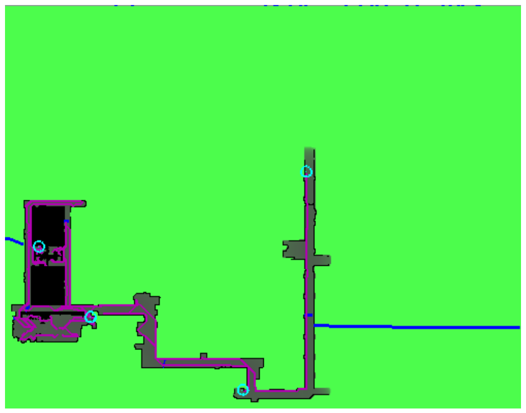
Our first implementation is a standalone C++ implementation that was mostly single-threaded. Figure 5.12 shows the screenshots of three robots exploring a cluttered environment. The boundaries of the tessellations are shown by the bold blue lines. The dotted lines show the robot trajectories. The intensity of the pixes in the environment represent the entropy. The mobile sensors start off with absolutely no prior knowledge about the environment, hence highest vale of entropy ln(0.5) assigned to each cell. They then collaboratively explore the environment and attain full exploration, coverage and convergence.

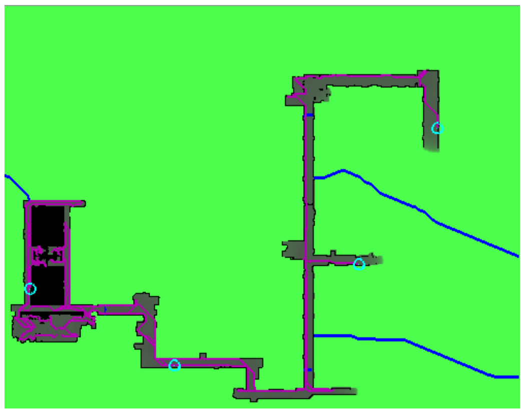
Larger Environment

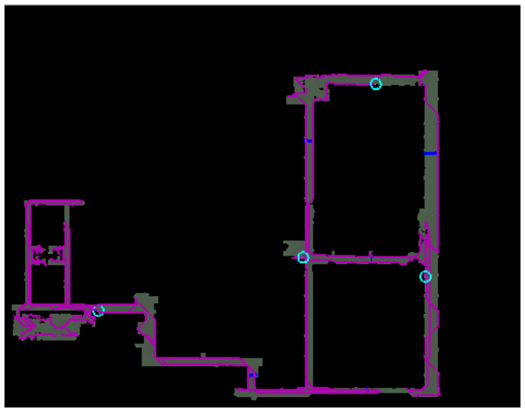
Figure 5.12 shows the screenshots from a simulation of four robots exploring a large (1000×783 uniformly discretized) cluttered environment. The boundaries of the tessellations are shown by the bold blue lines. The robot positions are encircled by cyan circles. The dark lines show the robot trajectories. The intensity of the pixels in the environment represent the entropy, and the unreachable regions are colored in black. The mobile robots begin at the room in the lower left with no prior knowledge about the environment, hence the highest value of entropy, ln(0.5), is assigned to each cell. Besides collaboratively exploring the environment the robots distribute themselves in such a way that they maintain proper coverage of the explored environment both during exploration and after completely building the map. The mobile robots attain full exploration, coverage, and convergence within $t = 2750$ iterations. Each iteration, which involves computing the voronoi


(a) $t = 0$


(b) $t = 500$


(c) $t = 1000$


(d) $t = 1600$


(e) $t = 2600$ (complete map built)

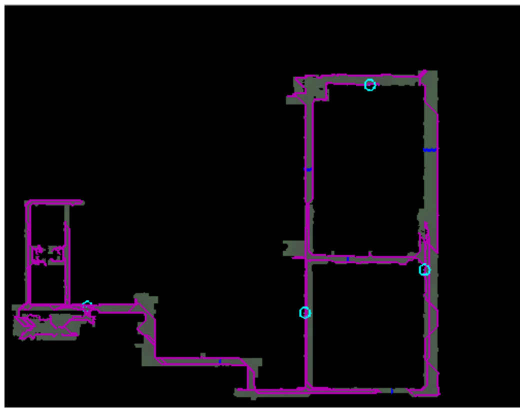
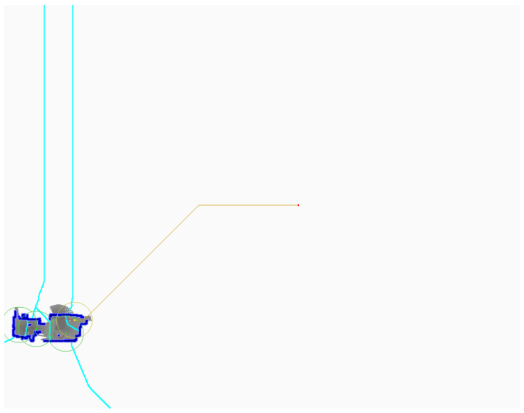
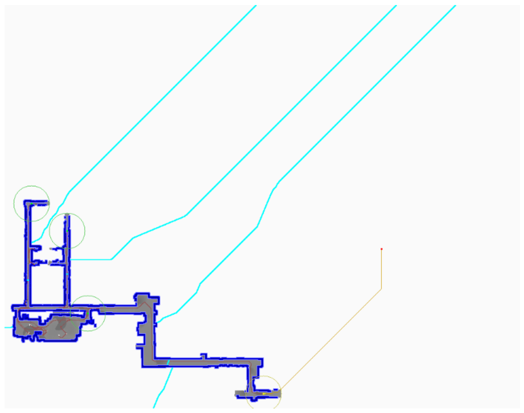
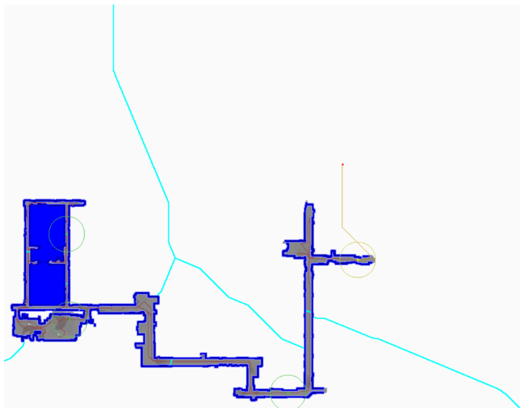

(f) $t = 2750$ (convergence)

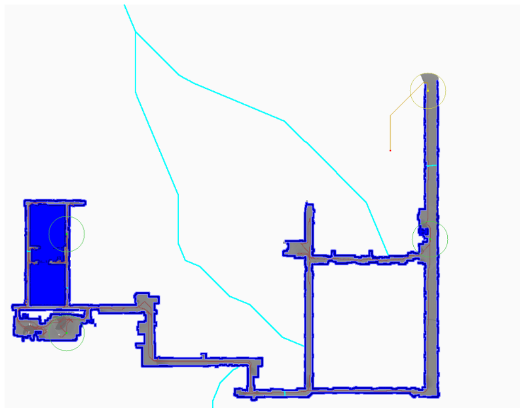
Figure 5.12: Exploration and coverage of a large unknown environment. Green indicates uncertainty.

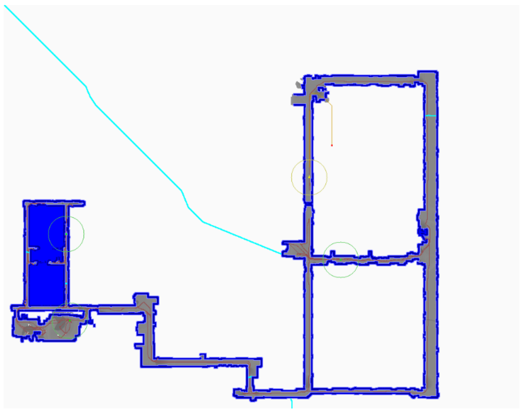
tessellations as well as the control commands for all the robots, takes about $1.7s$ running on a single processor as described in earlier results.


(a) *iter* = 101


(b) *iter* = 1501


(c) *iter* = 3951


(d) *iter* = 5701


(e) *iter* = 7101 (complete map built)

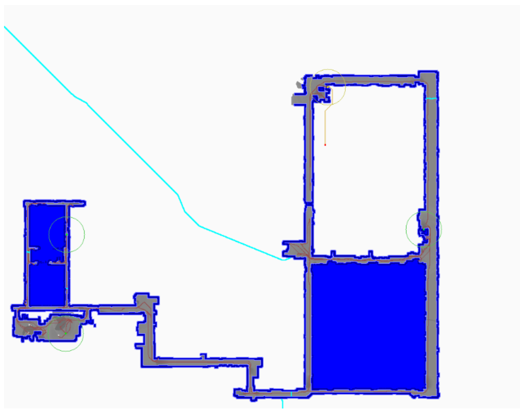

(f) *iter* = 8501 (convergence)

Figure 5.13: ROS implementation of coverage and exploration.

ROS Implementation

We have made a preliminary implementation of our algorithm in ROS (Robot operating system). The features that are different from the previous implementation are:

i. In ROS the simulations are really decentralized. Each robot runs its own individual thread and needs to communicate with each other.

ii. The sensors (footprint and noise) as well as robot kinematics are realistically simulated. We use a feedback linearization technique for controlling the non-holonomic robots.

iii. We aggressively use multi-thread implementations wherever possible, thus enabling large amount of parallel computation for each robot. This makes the program modular as well.

iv. Since the robots need to communicate, we had to implement efficient communication techniques. In particular, in every time steps the robots broadcast only new information that it got from the sensors since previous broadcast.

v. The ROS implementation is perfectly suited for implementation in real robots for real experiments.

The sequence of images in Figure 5.13 show the same benchmarking environment as the previous one being explored by 4 robots, but this is in the ROS implementation. Running on a single processor, the main thread for each robot runs at 1 Hz. This rate of computation is significantly faster than what is required in real robot experiments.

Chapter 6

Dimensional Decomposition for Efficient Planning

6.1 Motivation

So far we have mostly used graph search based planning techniques as our primary tools. We have incorporated certain metric and topological information in the search algorithms and have been able to use the versatility of graph search based method to generalize some of the continuous approaches. However the biggest challenge that one encounters in discretization, graph creation and using search algorithms is that the number of vertices, average degree and hence the complexity of the search algorithm increases exponentially with the dimensionality of the configuration space. The challenge becomes evident in multi-robot planning problems where the configuration spaces of the individual robots cannot be decoupled due to presence of complex inter-robot constraints.

In this chapter we try to exploit certain structures in the high dimensional configuration space and the constraints that prevent complete decoupling of the problem into lower dimensional planning problems, and hence use graph search techniques in concert with gradient ascent type of techniques in order to solve a class of constrained optimization problems. This particular class of optimization problem turns out to be well-suited for solving multi-robot path planning problem in cluttered non-convex environments with pair-wise constraints on their trajectories.

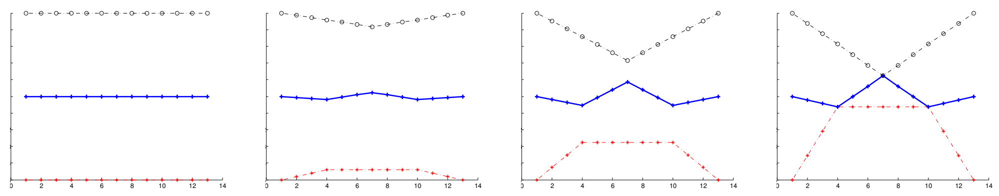
Distributed implementation of optimization problems is an important field of research in distributed systems [73, 5]. In many optimization problems the joint state space of all the search variables is too big or too complex for the optimization problem to be solved centrally. At other times the complete information about all the state variables is not available to any central processor. Thus distributed implementation of such problems become indispensable. One example of particular interest to us is multi-robot planning problems. For instance, robots navigating towards their respective goals while staying within the communication range is one of the common planning problems in multi-robot robotics. Here the search variables are the robot trajectories, each of which theoretically lies in an infinite dimensional Hilbert space. Planning in the joint state-space of all the robots in such a case while satisfying complex constraints may be very expensive, if not practically impossible. Multi-robot path planning suffers from the inherent complexity resulting from the necessity of operating in Cartesian products of configuration and state spaces [23]. The continuous path planning problem is even more difficult to solve in a centralized setting [59] unless the problem is solved sequentially for each robot [88]. Open loop trajectory planning problems can be reduced to optimization problems. While completeness results are often possible [1] for simple problems with no constraints, it is difficult to respect more complex multi-robot constraints. Another instance of distributed optimization problem is task allocation for multiple robots [33]. In these methods, one can impose rendezvous constraints at intermediate time points as tasks and reformulate the path planning problem as a task allocation problem. This then lends itself to auction-based solutions [25] for the team. However, these methods can produce highly sub-optimal solutions in the environments with obstacles without guarantees on convergence.

Closely related to this class of problems is *separable optimization problems* [5] (optimization problems that can be split up into simpler sub-problems involving only certain partitions of the variable set) with linear constraints have been studied extensively in the past and solved in a distributed fashion using techniques based on dual decomposition [73, 5]. Augmented Lagrangian type methods have been used for solving similar problems more efficiently [4, 66]. However such methods are limited to problems with linear constraints and rely on convexity of cost functions.

In this chapter we investigate a distributed implementation of a separable optimization problem with *non-linear* constraints arising from coupling between pairs of robots. We do not make any assumption on the convexity of the cost or the constraint functions. Our theoretical analysis shows that the algorithm converges to an optimal solution under certain conditions. As a demonstration of the implementation of our algorithm we will mostly concentrate on solving the problem of path planning for teams of robots coupled with constraints on the distances between pairs of robots. Continuous motion planning is possible for such problems [2], but only practical in environments with moderate complexity. We explore discrete path planning algorithms for solving the individual simpler optimization problems in a large environment with obstacles and solve the global problem using our distributed optimization algorithm. We show that our approach is able to find efficiently optimal paths with complex cost functions, in arbitrarily complex environments, and with non-linear pair-wise constraints. We therefore demonstrate the utility and versatility of the proposed algorithm by solving large scale optimization problems in a distributed fashion, which otherwise would have been intractable.

Brief Description of the Solution Approach

The intuitive concept behind the main algorithm developed in this chapter is that we start off by solving the global unconstrained problem, which is completely decoupled and hence can be solved as a bunch of lower dimensional problems. Then we gradually increase the penalty weights for violation of the constraints which are modeled as soft constraints, in a way not unlike dual and Lagrangian decomposition methods. We show that in every iteration of the algorithm, if we increase the penalty weights along certain specific directions (*Separable Optimal Flow Direction* and *Ascent Direction*) we are guaranteed to attain optimality and convergence in the limit (Figure 6.1). In order to deal with obstacles/punctures in the configuration space of individual robots, we need to do an exhaustive search in the different homotopy classes of trajectories. The tools developed in



(a) Unconstrained plans, $k = 0$ (b) $k = 50$ (c) $k = 150$ (d) Converged solution, $k = 216$

Figure 6.1: Demonstration of convergence towards global optimal solution with progress of iterations. In this example there are 3 robots planning trajectories from left to right of the environment. The unconstrained trajectories, (a), are 12 units long, parallel, and separated by 5 units. Trajectories are defined by displacements along Y direction of 11 unit-spaced points on each. The final trajectories, (b), satisfy rendezvous constraints as described in Section 6.6.1.

Chapters 3 and 4 can be used for that purpose.

6.2 Problem Definition

6.2.1 The Optimization Problem

We consider the following general problem:

Find

$$\{\pi_1^*, \ldots, \pi_N^*\} = \text{argmin}_{\pi_1 \ldots \pi_N} \sum_{j=1 \ldots N} c_j(\pi_j) \tag{6.2.1}$$

subject to the pairwise constraints

$$\Omega_{ij}(\pi_i^*, \pi_j^*) = 0, \qquad i, j = 1 \cdots N \tag{6.2.2}$$

We call each π_i a *partition* of the set of search variables. In the context of multi-robot planning problem π_i will represent the path of robot i, while $c_j(\pi_j)$ will be the cost of path π_j. Ω_{ij} will represent the violation of the constraint between the trajectories of robot i and j.

6.2.2 Problem Assumptions

The following assumptions are made about the variables and functions appearing in the problem definition. Note that the smoothness of c_i and Ω_{ij} and the conditions involving derivatives of these functions (all the assumptions except Assumption 1) are used only for proving the final theorem involving computation (Theorem 6.4.5). We do not need any of those assumptions for the basic problem definition, algorithm or for proving Theorems 6.4.2 or 6.4.4.
Assumptions:

1. The optimization variables π_i lie in abstract vector spaces that are continuous, differentiable and simply connected. In the general case they may be considered as vectors in a finite

dimensional Euclidean space, but they can lie (as we'll discuss later) in more complex spaces like infinite dimensional Hilbert spaces. We represent the space in which π_i lies as $\mathbb{H}$

2. The cost functions $c_i : \mathbb{H} \to \mathbb{R}_+$ are assumed to be continuous and smooth. In context of path planning for mobile robots, an example of such a cost function is the Euclidean length of the trajectories in an environment without obstacles. However, in our implementation we will later relax the condition. Note that we don't make any immediate assumption on the convexity of the cost functions.

3. The functions $\Omega_{ij} : \mathbb{H} \times \mathbb{H} \to \mathbb{R}$, defined for the unordered pair $\{i, j\}$, are continuous. Also, we require that the second derivatives of Ω_{ij} with respect to each of its parameters as well as the first mixed derivative are defined. That is, $\Omega_{ij}^{(0,2)}, \Omega_{ij}^{(2,0)}$ and $\Omega_{ij}^{(1,1)}$ are defined, where the superscripts denote the order of partial derivatives w.r.t. the respective parameters. A simple example of such a constraint function in context of multi-robot path planning is constraint on the distance between trajectories of two robots.

4. The functions Ω_{ij} are assumed to have the following properties

   i. Ω_{ij} is symmetric in its two parameters (i.e. $\Omega_{ij}(\pi_i, \pi_j) = \Omega_{ij}(\pi_j, \pi_i)$), and

   ii. $\Omega_{ij}^{(1,0)}(\pi_i, \pi_j) = -\Omega_{ij}^{(0,1)}(\pi_i, \pi_j)$.

   It is easy to note that these properties will be true if Ω_{ij} has the functional form $\Omega_{ij}(\pi_i, \pi_j) = G_{ij}(\pi_i - \pi_j)$, where $G_{ij} : \mathbb{H} \to \mathbb{R}$ is a continuous, smooth even function. Note that we don't make any immediate assumption on the convexity of Ω_{ij}.

5. We assume that besides c_r and Ω_{ij}, the quantities $c_r^{(2)}$, $\Omega_{ij}^{(1,0)}$, $\Omega_{ij}^{(2,0)}$ and $\Omega_{ij}^{(1,1)}$ are readily computable for a given set of input variables. This may be achieved either by knowing the expressions of the derivatives in closed form, or by computing them numerically.

If in a particular problem the cost and constraint functions are not smooth, they can always be approximated by smooth functions at the non-smooth regions using one of the many Mollification techniques [64]. We also note that inequality constraints can be converted to equality constraints (as required by (6.2.2)) by taking max or min with zero, followed by Mollification treatments.

It is to be noted that in spite of the conditions imposed on Ω_{ij}, they can represent a wide variety of constraints, especially in robotics applications. The proposed functional form of the constraints can model constraints on communication, visibility, rendezvous, convoying, collision avoidance, and other more complex coordination between pairs of robots involving their trajectories as well as their derivatives (say, to incorporate dynamic and kinematic constraints).

6.3 The Algorithm

A pseudo-code of our algorithm is presented in Algorithm 6.3.1. The global optimization problem is decomposed into a series of lower-dimensional unconstrained optimization problems, each of which is solved in a single *partition*, π_r, of the search variables. In each iteration the penalty weights on the violation of constraints, W^k, are incremented along the direction given by V^k. The intuitive concept behind the algorithm is that we start off by solving the global unconstrained problem, which is completely decoupled (line 1). Then we gradually increase the penalty weights (line 6) for the constraints which are modeled as soft constraints. The directions in which we can change the weights to guarantee optimality and convergence are described later in the *Theoretical Analysis* section. With the new set of weights we solve a sub-problem, which is an unconstrained optimization problem on only a single partition, namely π_r (line 7). Thus we note that in each iteration only the variables in a single partition are changed, while the others remain unchanged.

An application of the algorithm in goal-directed navigation of multiple robots with rendezvous constraints at various points on the trajectory is illustrated in Figure 6.1. We note that the algorithm starts off with unconstrained trajectories at $k = 0$ and in each iteration only a single robot plans its trajectory. As the iterations progress, the robots gradually change their trajectories due to increase in penalty weights and try to satisfy the rendezvous constraints. Eventually they reach the global solution. It is to be noted that the gradual increment in the weights play a key role in ensuring that the solution is optimal. One-shot increase in the weights to large values to satisfy the constraints would have resulted in suboptimal trajectories since the robots are planning sequentially. This particular example is described in more details in Section 6.6.1.

Algorithm 6.3.1

```
$g$ = **Distributed_Optimization** $(\{c_i\}, \{\Omega_{ij}\}, \mathcal{S})$
  Inputs:    a. Cost functions $c_i,\ i = 1, 2, \cdots, N$
             b. Constraint functions $\Omega_{ij},\ \{i, j\} \in \mathcal{P}^N$
             c. The set $\mathcal{S} = \{r_0, r_1, r_2, \cdots\},\ \ r_k \in \{1, 2, \cdots, N\}$
  Outputs:   a. Converged solutions $\pi_i,\ i = 1, 2, \cdots, N$
1   Compute $\pi_i^0 = \text{argmin}_{\pi_i} c_i(\pi_i),\ \ \forall i \in \mathcal{N}^N$     // The unconstrained plans.
2   Initiate $W_{ij}^0 = 0$ for all unordered $\{i, j\} \in \mathcal{P}^N$     // Weights on constraints.
3   Set $r = r_0 \in \mathcal{S}$ and $k = 0$
4   while $\left(\left(\Omega_{ij}(\pi_i^k, \pi_j^k) \neq 0\ \forall \{i, j\} \in \mathcal{P}^N\right)\right)$
5       Set $V^k = ComputeStepDirection(W^k, \{\pi\}^k, r)$     // Choose a direction from intersection
                                                            of Separable Optimal Flow Direction
                                                            and Ascent Direction.
6       Set $W^{k+1} = W^k + \epsilon^k V^k$     // Update weights.
7       Compute $\pi_r^{k+1} = \text{argmin}_{\pi_r}\left[c_r(\pi_r) + \sum_{\{ir\} \in \mathcal{P}_r^N} W_{ir}^{k+1} \Omega_{ir}(\pi_i^k, \pi_r)\right]$  // Update $r^{th}$ part'n.
8       Set $\pi_j^{k+1} = \pi_j^k$ for all other $j \neq r$     // Keep other partitions unchanged.
9       Set $k = k + 1$
9       Set $r = r_k \in \mathcal{S}$
13  return $\{\pi_i\}_{i=1,2,\cdots,N}$
```

The sets $\mathcal{N}^N$ is the set of all natural numbers from 1 to N, and $\mathcal{P}^N$ is the set of all unordered pairs of numbers in $\mathcal{N}^N$. The set $\mathcal{S}$ gives a sequence of partitions for which the individual sub-problems are solved in each iteration. $\mathcal{S} = \{r_0, r_1, r_2, \cdots\}$, where $r_i \in \{1, 2, \cdots, N\}\ \forall i \in \mathbb{N}$ can be constructed such that the partitions appear almost at equal frequency in the sequence. As we will see later, the sequence does not influence the convergence or optimality of the solution, but may influence the speed of convergence (number of iterations required to converge).

Also, the step-sizes, ϵ^k, are small and determine the precision of the solution. Theorem 6.4.5 is guaranteed to hold when the step-sizes are infinitesimal. But for practical purpose we choose finite and small step-sizes. The sequence $\mathcal{E} = \{\epsilon^0, \epsilon^1, \epsilon^2, \cdots\}$ may be predefined, set to a fixed constant step-size, or may be adaptive.

The procedure "*ComputeStepDirection*" in the Algorithm computes the direction in which to increase the penalty weights, W. We call this direction a *Step Direction.* In the following section (and especially Theorem 6.4.5) we will discuss the ways of computing such a direction.

6.4 Theoretical Analysis

In this section we investigate the conditions under which the proposed algorithm will converge to an optimal solution. Theorem 6.4.4 along with Theorem 6.4.2 proves that under certain conditions the algorithm is guaranteed to converge to an optimal solution. Theorem 6.4.5 gives a prescription, using which will ensure that the said conditions hold.

6.4.1 Notations and Preliminaries

Unordered pair

We define the set of unordered pairs of natural numbers from 1 to N, and its subsets as follows:

$$\mathcal{P}^N = \{\{1,2\},\{1,3\},\cdots,\{1,N\},\{2,3\},\{2,4\},\cdots,\{N-1,N\}\ \}$$

and, $\mathcal{P}^N_r = \{\{1,r\},\cdots,\{r-1,r\},\{r+1,r\},\cdots,\{N,r\}\}$

In the following discussions, the subscripts "ij" of quantities (like W) or functions (like Ω) are elements from $\mathcal{P}^N$. Thus the order of the subscripts does not matter. When we write W (or W_1, or W_2), we mean the vector of length $\frac{1}{2}N(N-1)$ of all the W_{ij}'s. Also, often we will write $\{i,j\} \equiv \{q,r\} \in \mathcal{P}^N_r$ to indicate that $\{i,j\}$ is such that exactly one of i or j is equal to r, while the other one, which is not equal to r, is denoted by k.

Other Notations

i. For notational convenience we define the sets $\mathcal{N}^N = \{1,2,\cdots,N\}$ and $\mathcal{N}^N_{-r} = \{1,2,\cdots,r-1,r+1,\cdots N\}$

ii. We denote the set $\{\pi_1,\pi_2,\cdots,\pi_N\}$ as $\{\pi\}$. On similar lines, if there are arbitrary functions $\Upsilon_i : \mathbb{A} \to \mathbb{B}$, $i = \{1,2,\cdots N\}$, we denote the collection of all these functions as $\{\Upsilon\} : \mathbb{A} \to \mathbb{B}\times\mathbb{B}\times\cdots\times\mathbb{B}$. Conversely, the j^{th} element of $\{\Upsilon\}$ is denoted as $[\{\Upsilon\}]_j$ or Υ_j

iii. The subset of $\{\pi\}$ without the r^{th} element is denoted by $\{\pi\}_{-r}$. Thus, $\{\pi\}_{-r} = \{\pi_1,\pi_2,\cdots,\pi_{r-1},\pi_{r+1},\cdots,\pi_N\}$

iv. If we have a smooth function $f : \mathcal{A}_1 \times \mathcal{A}_2 \times \cdots \mathcal{A}_n \to \mathbb{R}$, its derivatives are represented by $f^{(a_1,a_2,\cdots,a_n)}$. Thus, for a single-variable function, $f^{(1)} \equiv \nabla f$, $f^{(2)} \equiv \nabla^2 f$, and for a two-valued function, $f^{(1,1)} \equiv \nabla_x\nabla_y f$, etc.

v. We define the Lagrangian of the global problem $\overline{U}$, and the Dual function $\overline{\Psi}$,

$$\overline{U}(\{\pi\},W) := \sum_{k\in\mathcal{N}^N} c_k(\pi_k) + \sum_{\{kl\}\in\mathcal{P}^N} W_{kl}\Omega_{kl}(\pi_k,\pi_l)$$

$$\begin{aligned}\{\overline{\Pi}\}(W) &:= \text{argmin}_{\{\pi\}}\left[\overline{U}(\{\pi\},W)\right]\\ \overline{\Psi}(W) &:= \min_{\{\pi\}}\left[\overline{U}(\{\pi\},W)\right] = \overline{U}(\{\overline{\Pi}\}(W),W)\end{aligned} \tag{6.4.1}$$

In other words, $\{\overline{\Pi}\}(W)$ is the global optimum for the penalized objective function with W as penalty weights, and $\overline{\Psi}(W)$ is the optimal value.

vi. Similarly, we define for each *partition* π_r,

$$U_r(\pi_r, W_1, W_2) := c_r(\pi_r) + \sum_{\{kr\}\in\mathcal{P}_r^N} W_{1,kr}\Omega_{kr}(\overline{\Pi}_k(W_2),\pi_r)$$

$$\Pi_r(W_1,W_2) := \text{argmin}_{\pi_r}\left[U_r(\pi_r,W_1,W_2)\right]$$

$$\begin{aligned}\Psi_r(W_1,W_2) &:= \min_{\pi_r}\left[U_r(\pi_r,W_1,W_2)\right]\\ &= U_r(\Pi_r(W_1,W_2),W_1,W_2)\end{aligned} \tag{6.4.2}$$

That is, for a given value, $\{\overline{\Pi}\}_{-r}(W_2)$, of the partitions (except the r^{th} one), $\Pi_r(W_1,W_2)$ gives the optimum for an individual sub-problem with W_1 as penalty weights.

vi. Finally, we define the following,

$$\begin{aligned}\mathbf{M}_r(W) &= c_r^{(2)}(\overline{\Pi}_r(W))\\ &\quad + \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W))\\ \mathbf{N}_{lr}(W) &= \Omega_{lr}^{(1,1)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W))\end{aligned} \tag{6.4.3}$$

Also, we note that the r^{th} component of $\{\overline{\Pi}\}(W)$ is given by,

$$\begin{aligned}&\overline{\Pi}_r(W)\\ &= \text{argmin}_{\pi_r}\Big(c_r(\pi_r) + \textstyle\sum_{\{kr\}\in\mathcal{P}_r^N} W_{kr}\Omega_{kr}(\overline{\Pi}_k(W),\pi_r)\\ &\qquad + \textstyle\sum_{k\in\mathcal{N}_{-r}^N} c_k(\overline{\Pi}_k(W))\\ &\qquad + \textstyle\sum_{\{kl\}\in\mathcal{P}^N/\mathcal{P}_r^N} W_{kl}\Omega_{kl}(\overline{\Pi}_k(W),\overline{\Pi}_l(W))\Big)\\ &= \text{argmin}_{\pi_r}\Big(c_r(\pi_r) + \textstyle\sum_{\{kr\}\in\mathcal{P}_r^N} W_{kr}\Omega_{kr}(\overline{\Pi}_k(W),\pi_r)\Big)\\ &= \Pi_r(W,W)\end{aligned} \tag{6.4.4}$$

Thus,

$$\overline{\Pi}_r^{(1)}(W) = \Pi_r^{(1,0)}(W,W) + \Pi_r^{(0,1)}(W,W) \tag{6.4.5}$$

6.4.2 Theorems

Definition 6.4.1. [*Separable Optimal Flow*] Given the functions c_r, Ω_{ir} $\forall\{ir\}\in\mathcal{P}_r^N$ we call V a *Separable Optimal Flow Direction* and ϵ a corresponding *Separable Optimal Flow Step* at W for the r^{th} partition if and only if the following holds,

$$\begin{aligned}\Psi_r(W+\epsilon V,W) - \Psi_r(W,W) &\le \Psi_r(W+\epsilon V,W+\epsilon V) - \Psi_r(W,W+\epsilon V)\\ \text{and,}\quad V_{ij} = 0,\ &\forall\{i,j\}\text{ such that } r\notin\{i,j\}\end{aligned} \tag{6.4.6}$$

Together, V and ϵ are said to define a *Separable Optimal Flow* at W for Ψ_r.

Theorem 6.4.2. *[Optimality at each Iteration] If the Step Direction, V^k, returned by procedure ComputeStepDirection at the k^{th} iteration in Line 5 of the Algorithm 6.3.1, along with the chosen Step Size, ϵ^k, define a Separable Optimal Flow at W^k for Ψ_{r_κ}, $\forall$ k, then $\forall$ k:*

$$\{\pi_1^k, \dots, \pi_N^k\} = \text{argmin}_{\{\pi\}} \left[\sum_{i \in \mathcal{N}^N} c(\pi_i) + \sum_{\{ij\} \in \mathcal{P}^N} W_{ij}^k \cdot \Omega_{ij}(\pi_i, \pi_j) \right]$$

Proof. Detailed proof can be found in Appendix C.1. ■

The result of the Theorem 6.4.2, in brief, can be stated as

$$\pi_i^k = \overline{\Pi}_i(W^k), \quad \forall i, k \tag{6.4.7}$$

The implication of the result is that there are specific directions (which we call *Separable Optimal Flow Directions*) in which we can increment the penalty weight vector W, such that the global optimum for the new set of penalty weights differs from the previous global optimum (i.e. optimum for the previous set of weights) in only one partition of the optimization variables, namely π_r. Thus, by moving along such a direction in k^{th} iteration, we only need to change π_{r_k}, and still remain at an optimum of the penalized net cost.

Definition 6.4.3. [*Ascent Flow*] We call V an *Ascent Direction* and ϵ a corresponding *Ascent Step* at W if and only if the following holds,

$$\epsilon \sum_{\{ij\} \in \mathcal{P}^N} V_{ij} \Omega_{ij}(\overline{\Pi}_i(W + \epsilon V), \overline{\Pi}_j(W + \epsilon V)) \quad > \quad 0 \tag{6.4.8}$$

Together, V and ϵ are said to define a *Ascent Flow* at W.

Theorem 6.4.4. *[Convergence of the Algorithm] If the Step Direction, V^k, returned by the procedure ComputeStepDirection at the k^{th} iteration in Line 5 of the Algorithm 6.3.1, along with the chosen Step Size, ϵ^k, is an Ascent Flow as well as a Separable Optimal Flow (for the r_k^{th} partition) at W^k, for every k, then the Algorithm converges to an optimal solution, if one exists.*

Proof. Detailed proof can be found in Appendix C.2. ■

The result of Theorem 6.4.3 implies that if we always increment the penalty weights along directions that are both *Ascent Direction*s and *Separable Optimal Flow Direction*s, we will eventually converge to the global optimum, if it exists. Knowing the π_i^k's from previous iterations, it is easy to choose such a direction V^k for the current iteration as one that has positive inner product with the vector of all constraint violations at k^{th} iteration (i.e., the vector made of $\Omega_{ij}(\pi_i^k, \pi_j^k)$'s).

As it is apparent from the proofs of the last two Theorems (Appendix C.1 and C.2), so far we have made no assumption on the convexity or even the smoothness of the functions c_i and Ω_{ij}. All our required assumptions are encoded in the conditions of *Separable Optimal Flow* and *Ascent Flow*. However, in the theorem that follows, we will be focusing more on some computational aspects.

This would require that we be able to differentiate these functions, and make certain assumptions about *small steps*.

Theorem 6.4.5. *[Computation of Separable Optimal Flow Direction] If the functions c_r and Ω_{ir} $\forall\{ir\} \in \mathcal{P}_r^N$ abide by the Problem Assumptions, we can find an Ascent Flow and a Separable Optimal Flow for the ${r_k}^{th}$ partition at W^k, if it exists, along with a small enough Step Size, ϵ^k, at Lines 5 and 6 of the Algorithm 6.3.1. This can be achieved using only the following quantities that are readily known or computable:*

$$W^k, \quad \pi_i^k \equiv \overline{\Pi}_i(W^k), \quad \forall i \in \mathcal{N}^N \quad \textit{(known from previous iteration)},$$

$$c_i^{(2)}(\pi_i^k), \quad \forall i \in \mathcal{N}^N,$$

$$\Omega_{ij}^{(0,2)}(\pi_i^k, \pi_j^k), \quad \Omega_{ij}^{(1,1)}(\pi_i^k, \pi_j^k) \quad \textit{and} \quad \Omega_{ij}^{(1,0)}(\pi_i^k, \pi_j^k), \quad \forall\{ij\} \in \mathcal{P}^N,$$

In general we get to choose from a large set of possible Separable Optimal Flows and a large set of possible Ascent Flows, thus giving us the opportunity to find a common "flow" (V^k and ϵ^k) from the two sets.

Proof. Detailed proof can be found in Appendix C.3. ■

The theorem implies that we can compute a *Separable Optimal Flow Directions* which is also an *Ascent Direction*, if such a direction exists exists, in terms of quantities that can be easily computed using the variables from the previous iteration, within the limits of the error introduced by the finite step-size. It is important to note that in general there is much freedom in choosing from a large set of possible *Separable Optimal Flow Direction* (set of linear combinations of the eigenvectors such that (C.3.9) is satisfied). Moreover an *Ascent Direction* can be chosen just by choosing the sign of ϵ correctly. Although we do not discuss the existence of such directions as the iterations progress, we can expect that such directions will exist almost always.

6.5 Discrete Solution for the Sub-problems

Except for the result presented in Section 6.6.1, all the other implementations that we will present use an approximate discrete approach for solving the sub-problems in Line 7 of Algorithm 6.3.1. This was, in the first place, the motivation behind developing this method, due to which we hoped to be able to use graph search-based planning in lower dimensional subspace of the variable space (*i.e.* each π_r), yet ensuring that we move towards the global optimum.

In all the problems solved using discrete approach, the type of constraints that we will consider is time-parameterized constraints on the distances between pairs of robots. This particular type of constraint has a broad scope in multi-robot coordination problems, which includes, but is not limited to, the problem of rendezvousing in order to exchange information and the problem of maintaining communication while executing tasks. The constraints are defined between pairs of robots and are modeled as the minimum distance of separation between the robots as a function of time.

The way we define such constraints and solve the sub-problems is that we take Cartesian product of the configuration space of each robot with time ($\mathcal{C} \times [0, T]$), thus making time a state variable for the robot. This type of discrete implementation is illustrated in Section 6.6.2.

In addition we introduce the notion of *tasks* as an additional level of complexity in the graph. The tasks are in the form of points in the original configuration space (that is, before augmenting with time) that the robots need to visit. We assume that each robot has been assigned an unordered set of tasks, each of which it needs to visit at some point in time. Results in Section 6.6.3 illustrates this in details.

6.5.1 Basic Graph Construction

Planning for goal-directed navigation for an individual robot $R_i, 1 \leq i \leq N$ is, as usual, modeled as computing a least-cost path through a graph G_i formed by discretization of the configuration space (as described in Section 2.3). In addition, we augment each state in the graph G_i with an additional variable - time index t, such that a state in the augmented graph becomes $\{\mathbf{s}, t\}$.

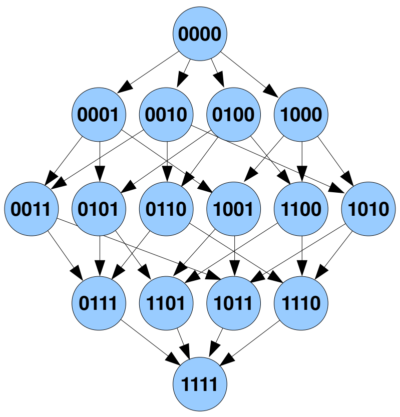
The augmented graph, H_i, is essentially a graph product between the graph G_i and the time graph, $\{0 \rightarrow 1 \rightarrow \cdots \rightarrow T\}$. It is precisely defined as follows: The vertex set of the augmented graph is $\mathcal{V}(H_i) = \mathcal{V}(G_i) \times \{0, 1, \cdots, T\}$, and edges are defined such that a particular node $\{\mathbf{s}, t\} \in \mathcal{V}(H_i)$ connects to the node $\{\mathbf{s}', t+1\} \in \mathcal{V}(H_i)$ if and only if $\mathbf{s}' = \mathbf{s}$ or $[\mathbf{s}, \mathbf{s}'] \in \mathcal{E}(G_i)$. Planning in such an H_i ensures that the planning is done both in space and time, and the time parametrized trajectory returned by the planner is consistent with the fact that the robots can move only forward in time. Such planning enables us to incorporate time-paramertized distance constraints.

We assume that all trajectories of interest to us are at most T timesteps. Thus, the solution to a typical planning problem for a single robot is a T-step path in H_i from $\{Start_i, 0\}$ to $\{Goal_i, T\}$, and can be represented by the ordered set $\pi_i = \{s_0 = Start_i, s_1, \ldots, s_T = Goal_i\}$. Thus, $\pi_i(t)$ refers to the coordinate s_t (in other words, the robot R_i is at this location at time t when following path π_i). The cost of a path is given as $c(\pi_i) = \sum_{j=1 \ldots T} c(\mathbf{s}_{j-1}, \mathbf{s}_j)$.

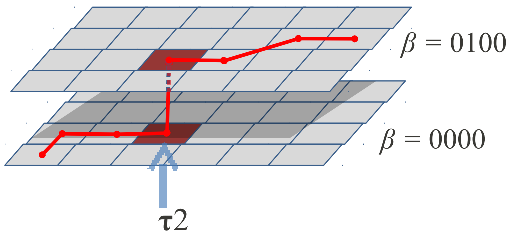
6.5.2 Product with Task Graph

The robot R_i has M_i tasks assigned to it denoted by $\tau_i^0, \tau_i^1, \cdots, \tau_i^{M_i - 1}$, where each task is a node in the graph G_i. Thus, $\tau_i^j \in \mathcal{V}(G_i)$. The planning needs to be done in such a way that the planned trajectory of robot R_i passes through each of $\tau_i^0, \tau_i^1, \cdots, \tau_i^{M_i - 1}$, *i.e.* $\tau_i^j \in \pi_i \forall j = 0, 1, 2, \cdots, M_i - 1$. The order of execution of the tasks is obtained as part of the solution to the planning problem.

To model the tasks and in order to incorporate them in the search graph we first define a new state coordinate (besides $\mathbf{s}$ and t) and call it "task indicator", β, which is essentially a variable that we will represent as a binary number consisting of M_i bits (for robot R_i), each bit being the flag or indicator of whether the corresponding task has been completed. For notational convenience we define a function B such that $\beta = B_M(\{j_1, j_2, \cdots, j_k\})$ is a M-bit long binary number with 1's at the positions $j_1, j_2, \cdots, j_k$, and 0's at the rest. Thus $B_7(\{2, 4, 5\}) = 0110100$ and $B_3(\{0\}) = 001$. Note that $B_M(\{j_1, j_2, \cdots, j_k\})$ in fact represents the state in which only the tasks $\tau^{j_1}, \tau^{j_2}, \cdots, \tau^{j_k}$ have been executed. We define the inverse function of B to be the one that returns the indices of the non-zero bits in a given β, i.e., $B^{-1}(\beta) = \{j_1, j_2, \cdots, j_k\}$



(a) The task graph Υ_i showing the possible transitions of the task indicator value, β for robot R_i with four tasks.



(b) A transition from one node in K_i to another where β gets changed. The bit-2 of β gets flipped to 1 when τ_2 is visited.

Figure 6.2: The task graph Υ_i, and its product with H_i.

We define a task graph Υ_i for robot R_i such that the vertices of Υ_i are M_i-bit binary numbers. Connection between two vertices of Υ_i exist iff the vertices differ by exactly one bit, and the edge points from the vertex with lower value to the one with higher value. An example of such a graph for $M_i = 4$ is shown in Figure 6.2(a). For a robot R_i with M_i tasks to execute, the starting value of "task indicator" will be $B_{M_i}(\{\}) = 00\cdots0$, which represents the state where no task has been executed, and its goal value will be $B_{M_i}(\{0, 1, 2, \cdots, M_i - 1\}) = 11\cdots1$, which represents the state where all the tasks have been executed. Since the tasks are executed one at a time, the two vertices β_1 and β_2 can be connected if and only if β_1 and β_2 differ by a single bit.

However we also note that the j_l^{th} bit in the state coordinate β is to be turned on for robot R_i only when the robot executes the j_l^{th} task assigned to it, i.e. when the spatial coordinate of the robot is $\mathbf{s} = \tau_i^{j_l}$. Using this fact we construct the product graph K_i for robot R_i such that $\{\mathbf{s}, t, \beta\} \in \mathcal{V}(K_i)$ are the vertices of the graph representing the full states of the robot. The graph K_i is defined such that,

i. $$\begin{aligned} \mathcal{V}(K_i) &= \mathcal{V}(H_i) \times \mathcal{V}(\Upsilon_i) \\ &= \mathcal{V}(G_i) \times \{0, 1, \cdots, T\} \times \mathcal{V}(\Upsilon_i) \end{aligned}$$

ii. An edge from vertex $\kappa_1 = \{\mathbf{s}_1, t_1, \beta_1\} \in \mathcal{V}(K_i)$ to vertex $\kappa_2 = \{\mathbf{s}_2, t_2, \beta_2\} \in \mathcal{V}(K_i)$ exists iff exactly one of the following holds

  a. $[\{\mathbf{s}_1, t_1\}, \{\mathbf{s}_2, t_2\}] \in \mathcal{E}(H_i)$,
  and $\mathbf{s}_2 \notin \{\tau_i^l \mid l^{th} \text{ bit of } \beta_1 \text{ is } 0\}$,
  and $\beta_1 = \beta_2$.

  b. $[\{\mathbf{s}_1, t_1\}, \{\mathbf{s}_2, t_2\}] \in \mathcal{E}(H_i)$,
  and $\mathbf{s}_2 \in \{\tau_i^l \mid l^{th} \text{ bit of } \beta_1 \text{ is } 0\}$ with $\mathbf{s}_2 = \tau_i^\lambda$,
  and $[\beta_1, \beta_2] \in \mathcal{V}(\Upsilon_i)$ such that the λ^{th} bit of β_2 is 1.

The connection condition 'ii.a.' corresponds to the robot navigating without visiting a task, while condition 'ii.b.' corresponds to the transition in the graph when the robot visits a task (illustrated in Figure 6.2(b)).

The cost of an edge $[\{\mathbf{s}_1, t_1, \beta_1\}, \{\mathbf{s}_2, t_2, \beta_2\}] \in \mathcal{E}(K_i)$ is same as the cost of the edge $[\{\mathbf{s}_1, t_1\}, \{\mathbf{s}_2, t_2\}] \in \mathcal{E}(H_i)$. For avoiding zero-cost edges, we assign a small ϵ cost to edges where $\{\mathbf{s}_1, t_1\} = \{\mathbf{s}_2, t_2\}$. Any possible state of the robot R_i defined by its spatial coordinates, time and tasks executed can be represented by a state in the graph K_i. The connection between the states of K_i define how the transition from one state to another can take place.

6.6 Results

6.6.1 An Exact Implementation

We demonstrate the Algorithm 6.3.1 using a MATLAB implementation of an idealized planning problem. The specific problem under consideration may be considered as a multi-robot goal-directed path planning in an environment without obstacles, where the trajectories of the robots are defined by displacements of unit-spaced points on the trajectories along vertical direction. Thus the trajectory, of a robot is given by $\pi_r = [start_r, y_{r2}, y_{r3}, \cdots, y_{rL}, goal_r]^T$, where $y_{ri}, i = 2, \cdots, L$ are the search variables and represent displacement of a point at x_i of the r^{th} robot's trajectory. The constraints between robots a and b are rendezvous constraints defined by

$$\Omega_{ab}(\pi_a, \pi_b) = \left((y_{ac_1} - y_{bc_1})^2 + (y_{ac_2} - y_{bc_2})^2 + \cdots\right)^{1/2} = 0$$

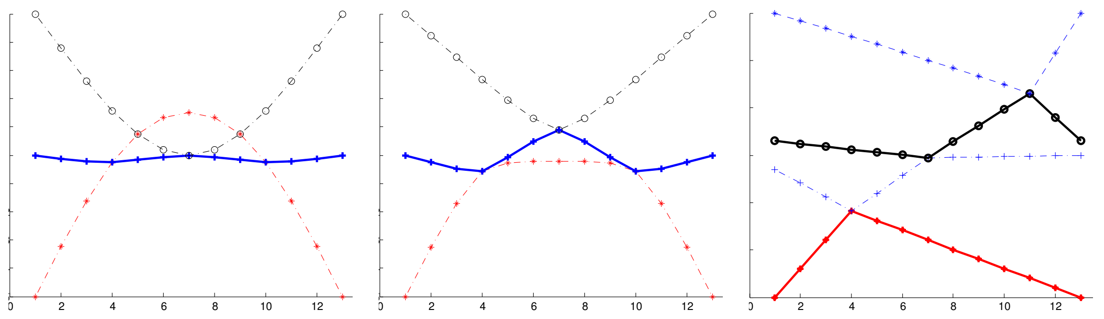
where $c_1, c_2, \cdots$ are the points where a and b need to rendezvous. There are two components to the costs functions c_r - the length of the trajectory, and the integral of the square of accelerations over the trajectory. Assuming the robots have constant X-velocity, the net cost is thus given by a weighted sum of the two components,

$$\begin{aligned} c_r(\pi_r) = \alpha &\left((y_{r2} - start_r)^2 + (y_{r3} - y_{r2})^2 + \cdots\right)^{1/2} \\ &+ \beta\left(((y_{r4} - y_{r3}) - (y_{r3} - y_{r2}))^2 + ((y_{r5} - y_{r4}) - (y_{r4} - y_{r3}))^2 + \cdots\right) \end{aligned}$$

The individual unconstrained optimization problems in Line 7 of the Algorithm were solved using MATLAB's *fminunc*.

Figure 6.1 demonstrates how the algorithm approaches the optimal feasible solution with progress of iterations. In that example we chose $\alpha = 1, \beta = 0$ and a constant step size of $\epsilon^k = 0.01$. The constraints were that the top robot had to rendezvous with the middle one at the middle point (i.e. at x_7), while the bottom and middle ones had to rendezvous at two points (y_4 and y_{10}). Since it is a symmetric case with only trajectory length as cost, it's easy to compute the optimal solution separately by optimizing over just 2 variables. The optimal cost that way is found to be 43.946, while our algorithm terminates at a cost of 43.962 for the said step size. Figures 6.3(a) and (b) illustrate the results of the same problem, but for other values of α and β, demonstrating the ability of the algorithm in dealing with complex cost functions. Figure 6.3(c) demonstrates an asymmetric case with 4 robots.

We attempted to solve these problems in a centralized fashion using MATLAB's Pseudo-Newton



(a) Symmetric case: $\alpha = 0, \beta = 1$ (b) Symmetric case: $\alpha = 1, \beta = 0.2$ (c) An asymmetric case: $\alpha = 1, \beta = 0$

Figure 6.3: Converged Solutions

search method implemented in *fmincon*. Even when the gradient and Hessian functions were explicitly provided, it failed to find a solution satisfying the required tolerance with computation time limit of 20 minutes and 5000 as the limit on number of iterations. In contrast, our algorithm solved these problems in about a minute and a few hundred iterations.

6.6.2 A Discrete Implementation

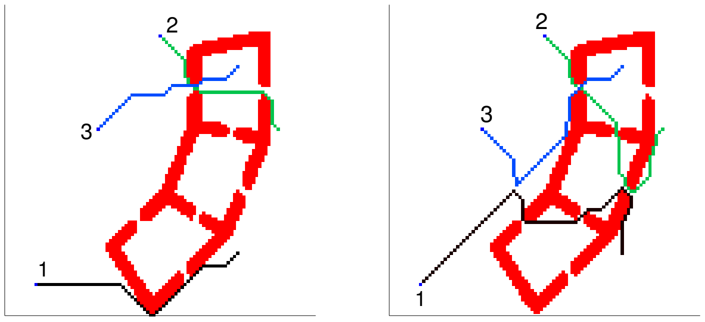
In the following examples we perform a more realistic multi-robot path planning with pair-wise distance constraints in a complex environment with large number of obstacles. Thus, even the individual sub-problems that we need to solve at Line 7 of Algorithm 6.3.1 become significantly complex. We employ the discrete method as was described in Section 6.5.

The three-dimensional state-space of each robot, $X - Y - Time$, was discretized into uniform cells to construct the hraph H_r, and an A* graph search technique was employed to obtain an optimal path in the graph as an approximation of the trajectory of r^{th} robot, π_r. The connectivity of the graph, as described in Section 6.5, is such that a cell in the x, y, t space connected to its 8 neighboring cells and itself in x, y but with the time step incremented by one. This implies that any path in the graph will be discrete approximation of an element of $\mathbb{H}$. While an 8-connected grid is quick and efficient to perform search in, it confines the motion of the robot to 8 directions (45° orientations). A consequence of this is that some seemingly sub-optimal solutions are actually optimal (minimum cost paths) in the 8-connected graph.

The cost function, c, is the Euclidean length of the trajectory. The constraints between the pairs of robots are defined by maximum distance, $\phi_{ij}(t)$, that the robots i and j can be apart at a given instant of time. Thus, the constraints are defined by,

$$\Omega_{ij}(\pi_i, \pi_j) = \int_0^T \varpi(\pi_i(t), \pi_j(t), \phi_{ij}(t)) = 0$$

where, $\varpi(\mathbf{s}, \mathbf{s}', p) = \max(0, d(\mathbf{s}, \mathbf{s}') - p)$ and d is a distance function.



(a) Unconstrained plans (b) Converged feasible solution

Figure 6.4: Planning in environment with three interconnected rooms

Dealing with Obstacles

Much of the theoretical analysis described so far relies on the fact that the values of all the variables change in small steps in each iterations. Smoothness condition is essential for Theorem 6.4.5. Unfortunately the presence of obstacles in the environment violates these condition. However the Optimality condition of Theorem 6.4.2 still holds in a single Homotopy class of all the trajectories. This means that as long as a trajectory changes in small steps and does not make big jump from one side of an obstacle to the other, Theorem 6.4.2 and the related results from Theorem 6.4.5 hold. Thus, whenever we get stuck in a particular homotopy class for all the trajectories, or whenever one of the trajectories makes a jump to a new homotopy class, we block this homotopy class as invalid and restart the planning. It is worth noting here that we use only *large* connected obstacles to characterize homotopy classes and avoid unnecessary creation of homotopy classes by small obstacles that do not effect optimality significantly. We primarily employed two different methods of blocking homotopy classes:

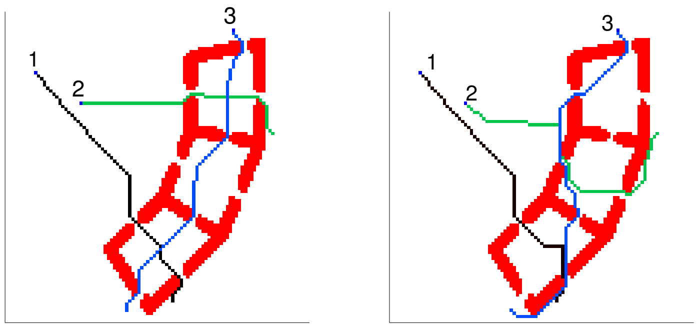
i. *Use of Blacklists* - We maintain a list of blocked regions (or balls) in the joint state-space of the robots violating one or more constraints. Every time we need to restart the planning as above, we update the "Blacklist" with the latest violation information in the homotopy class without a feasible solution. Thus, when we restart the planning we eventually will avoid the homotopy class, provided our choice of blocking ball was proper.

ii. *Systematic identification and blocking of Homotopy class* - Homotopy classes can be systematically identified using methods described in Chapter 3. We integrated such a method with our A* searches which enabled us to restrict searches in specific homotopy classes or block some homotopy classes.

In the following subsections we will present the performances of our algorithm in different environments.

Three Interconnected Rooms

The environment shown in Figure 6.4 consists of 3 interconnected rooms and 3 robots. The environment is made of discretized cells 89 × 94 in space and 200 time-steps. This results in the



(a) Unconstrained plans (b) Converged feasible solution

Figure 6.5: Extended rendezvous in environment with three interconnected rooms

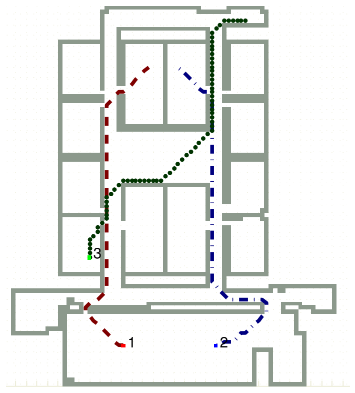
joint state-space of all the robots to have 10^{14} states. Our planner avoids planning in this huge state-space, but rather breaks up the problem into a series of plans in individual state-spaces with $< 2 \times 10^6$ states. The robots start from the left side of the environment and need to reach their goals on the right end. Figure 6.4(a) shows the unconstrained optimal plan. A constraint is defined between R_1 and R_3 such that they need to be within a distance of 2 discretization units at $t = 40$. And the constraint between R_1 and R_2 is such that they need to be within a distance of 2 discretization units at $t = 120$. The solution shown in Figure 6.4(b) was found in about one minute and 72 iterations on a computer running on 1.2GHz processor.

Extended Rendezvous

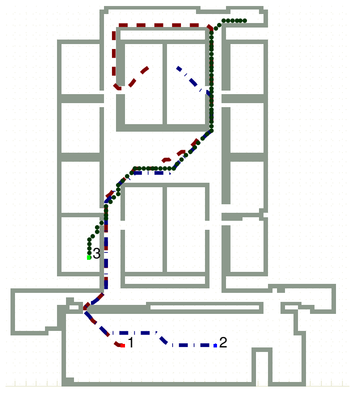
The environment shown in Figure 6.5 is similar to the previous example in size and discretization. In this example R_1 and R_2 need to traverse the environment from left to right, while R_3 needs to traverse it from top to bottom. Figure 6.5(a) shows the unconstrained optimal plan. Constraints between R_2 and R_3 is that they need to be within 2 discretization units distance from $t = 32$ to $t = 68$, and the constraints between R_1 and R_3 is that they need to be within 2 discretization units distance from $t = 84$ to $t = 120$ The solution in Figure 6.5(b) was found after 75 iterations with $\epsilon = 0.1$ as the fixed step-size.

Extended Rendezvous in a Real Environment

Figure 6.6 shows a part of the 4^{th} floor of Levine hall in University of Pennsylvania. The original map is 35 meters by 35 meters, discretized into 100×100 cells. There are 170 discretization steps in time. There are 3 robots and the unconstrained objectives, resulting in the solution in Figure 6.6(a), are that both Robot 1 (dashed trajectory) and Robot 2 (dash-dot trajectory) need to start at $t = 0$ inside the big room at the bottom and need to reach their respective goals by $t = 170$. Robot 3 (dotted trajectory) needs to start at $t = 45$ inside the small lower cubicle on the left side of the map and needs to reach the small storage space at the top right by $t = 120$. Figures 6.6(b) shows the converged feasible solution that satisfy the following constraints: A. Robot 2 needs to stay within 3 discretization units of robot 3 from $t = 60$ to $t = 80$; and B. Robot 1 then needs to stay within 3 discretization units of robot 3 from $t = 90$ to $t = 110$. Looking at the converged



(a) Unconstrained plans



(b) Converged feasible solution

Figure 6.6: Extended rendezvous in a real environment

Min	Max	Average
28 s	169 s	59.7 s

Table 6.1: Performance of the algorithm tested for extended rendezvous in a real environment (the example in Figure 6.6). The table shows the min, max and average time required for finding solution over 54 runs.

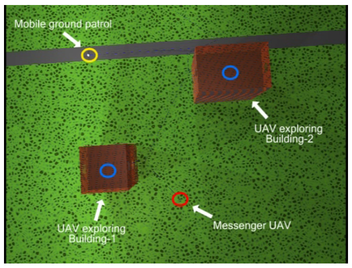
solution we observe that in order to satisfy constraint A, robot 2 loops it's trajectory around the central and lower cubicles, while to satisfy constraint B, robot 1 loops its trajectory around the central and upper cubicles.

Rendezvous in Three-Dimensional Environment

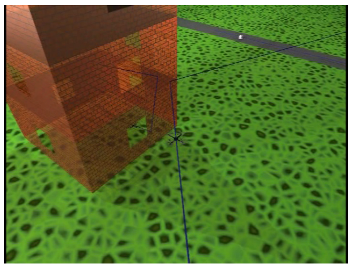
We implemented the algorithm for robots in 3 dimensional configuration space (*i.e.* $X - Y - Z$), which, along with time, makes a 4-dimensional state space for each robot in which we had to construct the graphs, H_i, and perform the search for the sub-problems.

Figure 6.7 shows screen-shots of a visualization made out of the solution obtained from the algorithm. In this example there are four robots exploring an environment with two buildings (Figure 6.7(a)). There are two explorer aerial vehicles (marked by blue circles in the figure), each of which explore one of the buildings in the environment. Each of the explorer UAVs gather a large amount of data about the interior of the buildings. The third aerial vehicle is a messenger (marked by the red circle). Its task is to visit each of the explorer UAVs, gather the data they have generated, and pass on the data to a mobile ground patrol (marked by yellow circle) before going to a base.

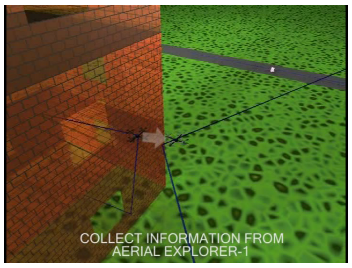
Performing such a complex collaborative task in an optimal fashion is non-trivial. The planner needs to generate plans for all four vehicles, three of which operate in 3D spaces. The joint state-space of the robots is 12 ($= 3 \times 3 + 2 + 1$) dimensional. Centralized planning for optimal trajectory satisfying all the constraints in such a huge state-space is infeasible. On the other hand, the constraints on communication and data transfer between the vehicles promptly translate into the constraints between the distances of their trajectories. As a result, we can use the algorithm



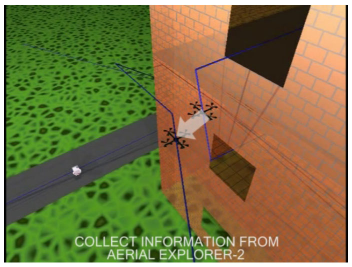
(a) Problem overview



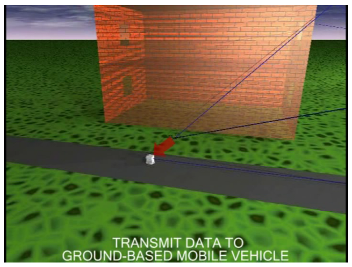
(b) $t = 5$



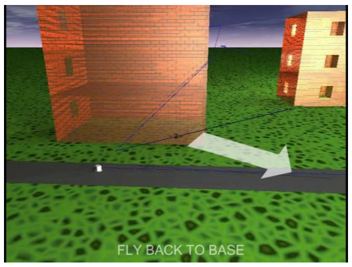
(c) $t = 12$



(d) $t = 23$



(e) $t = 37$



(f) $t = 41$

Figure 6.7: Rendezvous to exchange information in a 3-dimensional environment.

proposed in this chapter to find the optimal solution satisfying these constraints in an efficient (distributed) way. At each iteration, the algorithm plans a trajectory for a single robot in no more than a 4-dimensional (discretized) configuration space. It achieves the convergence for this problem in about 30 iterations (within error limits defined by chosen step-size). In Figures 6.7(c)-(d), the messenger UAV comes in close proximity to the explorer UAVs in order to collect the data the later have gathered, and in Figure 6.7(e), the collected data is transmitted to the ground patrol.

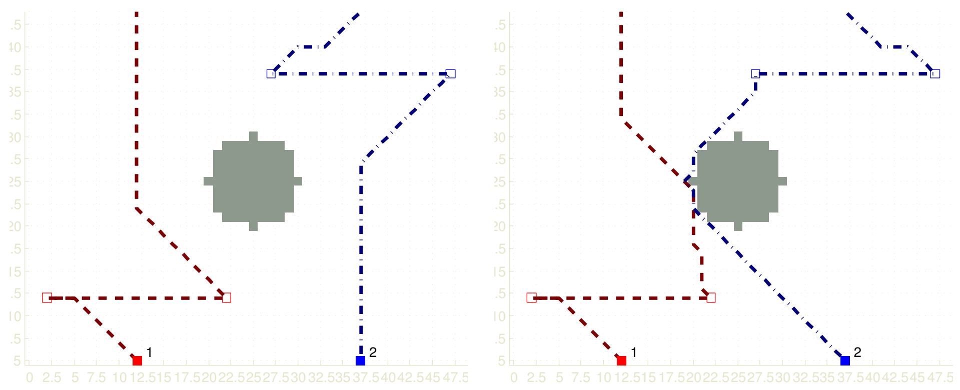
Performance

We tested the performance of our algorithm by running the example in Section 6.6.2 (*Extended rendezvous in a real environment*) multiple times, but with randomized initial & goal states and randomized time spans for the constraints. The implementation of the problem was made in C++ and was run on a computer with Intel 1.2 GHz processor. Table 6.1 gives a summary of the run-time from 54 runs.

The joint state-space of the three robots contains $\sim 170 \times (100 \times 100)^3 \simeq 1.7 \times 10^{14}$ states. Planning in such a huge graph is highly expensive, if not impossible. However our distributed planning technique was able to find the optimal solution up to the desired precision consistently and sufficiently fast, thus demonstrating the ability of the algorithm to solve large problems.

6.6.3 Additional Complexity with Tasks

As described in Section 6.5.2, we can have an additional level of complexity by introducing *tasks*. Thus, in the following examples, in each sub-problem (line 7 of Algorithm 6.3.1), we had to search in the graph K_r.



(a) Unconstrained trajectories (b) Solution satisfying constraints

Figure 6.8: Planning for two robots each with two tasks and a constraint to meet during their travel. It is interesting to note how in the final converged solution the robot on the right switches the order of execution of its tasks in order to satisfy the imposed constraint.

Example of Reordering of Tasks to Satisfy Constraints

In the simple example of Figure 6.8, we have two robots having to plan their paths from a start to a goal location. The environment contains one central large obstacle. Each of the robots has been assigned two tasks (marked by the empty boxes in color). The first figure shows the case where there is no constraint between the robots. Now we impose a constraint that the robots need to be within a distance of $0.08m$ at $t = 17.5s$ during their journey. It is interesting to note how in the final converged solution the second robot (the one in blue) switches the order of execution of its tasks in order to satisfy the imposed constraint. The environment consisted of 50 discretizations along each spatial direction and 40 discretizations in time. The joint state-space of the robots would hence have $40 \times (50 \times 50 \times 2^2)^2 = 4$ billion states. Even in this simple environment the constrained optimal planning in joint state-space would become very difficult. However our algorithm converges to a solution in about 10 iterations and in less than a minute time. The converged solution is optimal with respect to an 8-connected grid.

Exploration

Consider the example in Figure 6.9 where each robot needs to explore the inside of certain rooms in the 4th floor of Levine hall (University of Pennsylvania) assigned to them in the environment, and possibly create a map of the environment. They need to meet intermediately at $t = 120s$ to exchange information about each other's explorations so as to build a global map in a decentralized fashion. This problem perfectly fits our paradigm. We see how the robots explore the assigned rooms and also meet to exchange information. Each robot has been assigned 4 tasks. The environment consisted of 100 discretizations along each spatial direction and 250 discretizations in time. This makes a total of $250 \times (100 \times 100 \times 2^4)^2 = 6.4 \times 10^{12} = 6.4$ trillion states in the joint state-space!

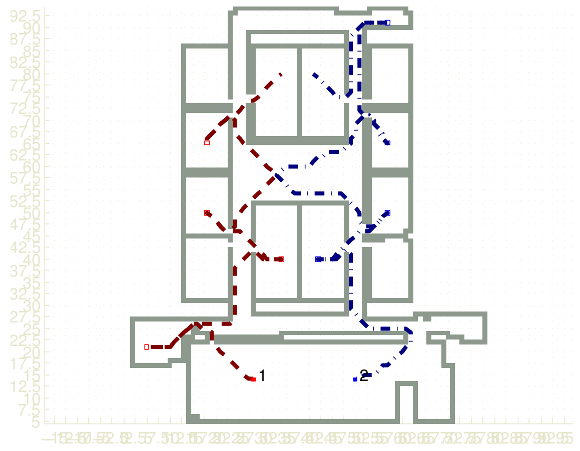


Figure 6.9: Two robots exploring certain rooms and rendezvousing to exchange information.

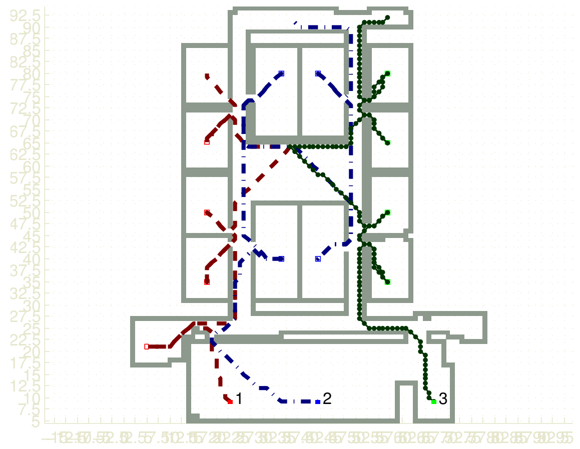


Figure 6.10: Three robots exploring certain rooms and rendezvousing to exchange information.

Our algorithm finds the solution in 1311 seconds in 17 iterations. The solution is optimal with respect to an 8-connected grid.

Exploration with Three Robots

We once again do the exploration of the Levine 4th floor, but now with 3 robots. Figure 6.10 shows the result. The constraints are between pairs of robots. Thus we used two constraints: The first one between robots 1 and 2 is to meet at $t = 120s$, and second one between robots 2 and 3 is to meet at $t = 120s$. The joint state-space had $250 \times (100 \times 100 \times 2^4)^3 = 1.024 \times 10^{18}$ states. Our algorithm finds a solution to the problem in 3003 seconds and 40 iterations.

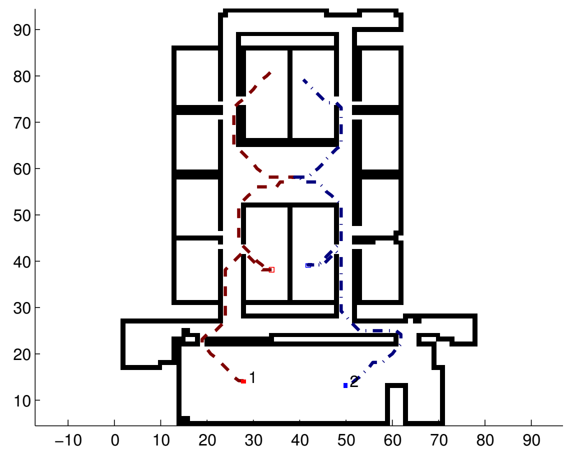


Figure 6.11: Two robots with one task each. This is the solution from one of the randomized runs.

Min	**Max**	**Average**
11 s	66 s	33.9 s

Table 6.2: Performance of the algorithm tested in the scenario of Figure 6.11. The table shows the min, max and average time required for finding solution over 10 runs.

Computation time Statistics

We chose a relatively simple scenario with two robots navigating from start to goal configuration, one task assigned to each robot, and one rendezvous constraint (Figure 6.11). The environment was same as before (Levine hall 4th floor), with 100×100 spatial discretization and 150 temporal discretization. We ran our algorithm several time by randomizing the initial positions, goal position and the constraint. Table 6.2 gives a summary of the run-time from 10 runs with randomized values.

Chapter 7

Coordinate Transformation for Efficient Optimal Planning in Environments with Non-Euclidean Metric

7.1 Introduction and Motivation

In this chapter we present a few preliminary ideas and propose some directions of research without providing much significant result. The overall question that we will be trying to address here is rather involved and we are still in the process of investigation.

In all the methods that we have discussed so far, graph search based planning has turned out to be highly robust, efficient and indispensable in practical robotics problems. Among the few drawbacks that such a method has, we have attempted to counter the problem of dimensionality by suggesting a dimensional decomposition technique in Chapter 6, where we compensated for the exponential increase in complexity of search algorithms by synthesis with a continuous gradient-based approach. In this chapter we address the other major drawback of graph search based approaches: The suboptimality induced in the final solution obtained from searching the graph, simply because of the fact that the metric on the original continuous space has been restricted to the discrete graph.

Graph-search based methods are, in general, well-suited for planning in spaces with a non-uniform Riemannian metric. In a particular coordinate chart, a non-Euclidean metric is simply translated to the costs of edges in the graph, *i.e.* weighted according to the location (and direction) of the edges. Thus, if a small segment, $\Delta\mathbf{l}$, represents an edge in the graph, and if g is the metric at a point on the 'center' of the edge, one would take $\sqrt{g_{ij}\Delta l^i \Delta l^j}$ as an approximate cost/weight of the edge (where we assumed the *Einstein summation convention* – that we will use throughout this chapter). However, in order to capture the non-Euclidean metric of the environment suitably, the

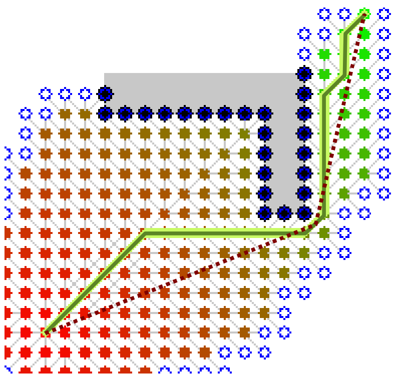


Figure 7.1: Suboptimality due to discretization. The green line is the optimal path in the graph from the given start to goal vertex in the graph. However the shortest path in the graph is not the shortest in the configuration space. The dotted line shows the shortest path in the original metric space.

discretization needs to be sufficiently fine. For example, one may use an uniform or unstructured fine discretization (*e.g.* Figure 3.12(a)), but not a visibility graph (*e.g.* Figure 3.18). But this condition eventually fires back since the least cost path in the graph created by fine discretization of an environment is most often not the least cost path in the continuous metric space. For example, an A* search with an uniform 8-connected discretization scheme (an uniform discretization into square cells, with the nodes of the graph placed at the centroid of each cell, and then each node connected to their 8 neighbors) gives an optimal path in the graph, which in the original space is a trajectory with segments constrained to head in directions that are multiples of 45° (Figure 7.1). Thus, the trajectories we obtain, although least cost paths in the graphs, are not the shortest in the original metric space.

In an Euclidean metric setting (*i.e.* a metric space, along with a coordinate chart, in which $g_{ij} = \delta_{ij}$ everywhere) one can use certain post-processing and smoothening methods to *straighten* the paths using notions of visibility (*e.g.* the dotted path in Figure 7.1 can be obtained by a simple visibility-based post-processing of the path restricted to the graph). Likewise, in Euclidean metric one can employ a visibility graph, hence obtaining paths that are truly least cost, as demonstrated in Figure 3.18. But visibility graphs or post-processing and smoothening cannot be employed easily in non-Euclidean metric spaces with punctures.

7.1.1 Problem Definition

Let us delve into the property of a metric space that allows us to use the notion of 'visibility' in an efficient way. In an Euclidean metric setting with holes (obstacles) it is the ease with which a 'line of sight' (or 'straight lines') between two points in the space can be constructed. This lets us construct a visibility graph by joining the 'corners' of the obstacles with such lines, or 'post-process' a path in the graph by checking pairs of points on the path that can be joined by such lines without intersecting obstacles. The need of checking intersection with polygonal obstacles also brings in the need of finding intersection between two 'straight lines' with ease.

In a general metric space the notion of 'line of sight' has to be replaced by *geodesics*. Typically in an arbitrary metric space the computation of the geodesic passing through two given points is highly non-trivial. One can employ a method like *shooting method* for solving the *geodesic equation* posed as a boundary value problem. However, in general, such methods are expensive and often practically infeasible.

The primary aim of this chapter is to investigate the types of metric spaces and the use of certain methods that would let us construct geodesics between two points in the space with ease.

7.1.2 A Motivating Example in Two Dimensions

Consider the following given matrix representation of metric in terms of the coordinate variables x and y,

$$g = \begin{bmatrix} x^2+y^2 & 0 \\ 0 & x^2+y^2 \end{bmatrix} \tag{7.1.1}$$

This is analogous to a cost function in a robot path planning problem (Figure 7.2(a)). Typically there would be obstacles in the space, and one would like to plan a trajectory, γ, between two given points (x_i, y_i) and (x_e, y_e), avoiding obstacles, so as to minimize $\int_\gamma \,\mathrm{d}s$, where $\mathrm{d}s = \sqrt{g_{ij}\,\mathrm{d}x^i\,\mathrm{d}x^j}$ $(= \sqrt{x^2+y^2}\,\sqrt{\mathrm{d}x^2+\mathrm{d}y^2}$, for this example – $\sqrt{x^2+y^2}$ being analogous to the cost function).

Now consider the following coordinate transformations

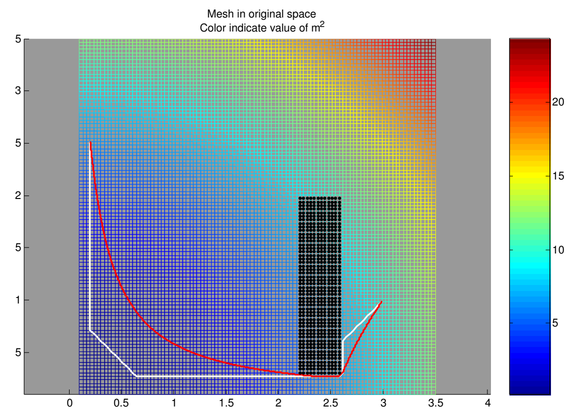
$$\begin{aligned} \overline{x}(x,y) &= xy \\ \overline{y}(x,y) &= (y^2-x^2)/2 \end{aligned} \tag{7.1.2}$$

One can now easily check, using the transformation rule $g_{kl} = \frac{\partial \overline{x}_p}{\partial x_k}\frac{\partial \overline{x}_q}{\partial x_l}\overline{g}_{pq}$, that in the *barred* coordinates the matrix representation of the metric tensor indeed becomes the identity matrix,

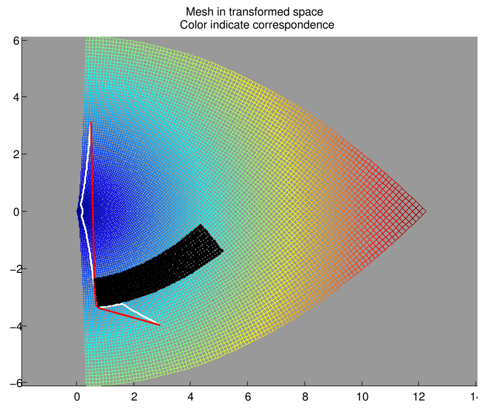
$$\overline{g} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.1.3}$$

The existence of such a transformation for an arbitrary given matrix representation of a metric, $g = \begin{bmatrix} m_{11}(x,y) & m_{12}(x,y) \\ m_{21}(x,y) & m_{22}(x,y) \end{bmatrix}$, is one of the questions that we would like to address. But before that we would like to complete the motivational example we started in this section in order to illustrate how this can be helpful in robot planning problems.

Figure 7.2(a) shows a uniform mesh color coded with the value of the square of the cost function, x^2+y^2. The labels on the axes represent the values of x and y. This embedding of the given metric space on the plane of the paper is hence *not isometric* (the plane of the paper has an Euclidean metric). The mesh is then transformed into the *barred* coordinate system to obtain the figure in 7.2(b). The color of the mesh is left same as the original one for ease of comparison. An 8-connected uniform grid is laid down in the original coordinates (Figure 7.2(a)) and a graph is



(a) A uniform mesh in original (*un-barred*) coordinates. Axes show values of (x, y). This is a non-isometric embedding of the part of the given metric space in $\mathbb{R}^2$ (the plane of the paper). Color indicates the value of $x^2 + y^2$ – the either diagonal elements of the metric g_{ij} (the square of 'cost' – red is higher cost, blue is lower).



(b) The same mesh in the transformed (*barred*) coordinates. Axes show values of $(\overline{x}, \overline{y})$. This is an isotropic embedding of the given metric space. Color of the original mesh is preserved to visualize the correspondence of points with the original mesh.

Figure 7.2: The black dots indicate obstacles. The planning problem: *Figure (a):* White path is the least cost path in the 8-connected grid graph planned in the *un-barred* coordinates. *Figure (b):* The same white path is shown in the *barred* coordinates. *Figure (b):* The red path in the *barred* coordinates is obtained by greedy post-processing of the white path. *Figure (a):* The red path in *un-barred* coordinates is obtained by inverse transformation.

hence formed. The white trajectory in Figure 7.2(a) demonstrate the least cost path for a given start and end coordinates in the 8-connected grid graph. As one can observe, the direction of the tangents on this path are restricted to multiples of 45°, which invariably leads to suboptimality. In an Euclidean metric setting one way of reducing the sub-optimality is to post-process the path by replacing portions of the path by line segments such that the segments do not intersect obstacles. However in a non-uniform metric like this, line segments are no more the geodesics. Hence this post-processing scheme is not possible in the *unbarred* coordinates. However once we have transformed the white path to the *barred* coordinates (Figure 7.2(b)), the metric is Euclidean here (an *isometric embedding* of the given metric space). Hence we can now use the said post-processing technique in this coordinate. Thus, as indicated by the red path in Figure 7.2(b), we obtain a piece-wise straight segments in the *barred* coordinates, which is the least cost path. Performing an inverse transformation on this red path gives the corresponding least cost path in the *un-barred* coordinates (red path in Figure 7.2(a)).

Generalization

We consider the problem in a 2-dimensional metric space. Let us denote a subset of this space (which is of interest to us) by $\mathbf{S}$, and assume it is equipped with the metric tensor g. The matrix representation of metric g in a given chart, C (with coordinate variables x and y), is isotropic and is given by,

$$g(x,y) = \begin{bmatrix} m^2(x,y) & 0 \\ 0 & m^2(x,y) \end{bmatrix} = m^2(x,y) \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.1.4}$$

where the function m is given/known. Thus the length/cost of a differential element is given by $\mathrm{d}s = m(x,y)\sqrt{\mathrm{d}x^2 + \mathrm{d}y^2}$. Thus, m can be interpreted as a cost map which acts as a scaling on the Euclidean length cost of an Euclidean metric.

The problem under consideration may now be posed as the problem of finding a chart $\overline{C}$ that is related to C in the following way,

$$\overline{x} = \overline{x}(x,y) \qquad , \qquad \overline{y} = \overline{y}(x,y) \tag{7.1.5}$$

such that the matrix representation of the metric tensor g in $\overline{C}$ is the identity matrix. That is,

$$\overline{g}_{kl} = \frac{\partial x_p}{\partial \overline{x}_k}\frac{\partial x_q}{\partial \overline{x}_l} g_{pq} = \delta_{kl} \tag{7.1.6}$$

where, in the second term in the above equation, summation is implied over repeated subscripts (*Einstein summation convention*) and the subscripts 1 and 2 are to indicate x and y respectively for notational convenience. δ_{kl} is the *Kronecker delta*.

It is a known fact [72] and not difficult to show that if we can find a holomorphic (complex analytic) function, f, such that $\|f(x+iy)\| = m(x,y)$ everywhere in $\mathbf{S}$, then the transformation defined by

$$\begin{aligned} \overline{x}(x,y) &= Im\left(\int f(z)dz\right)\Big|_{z=x+iy} + \kappa_0 \\ \overline{y}(x,y) &= \mp\, Re\left(\int f(z)dz\right)\Big|_{z=x+iy} + \kappa_0 \end{aligned} \tag{7.1.7}$$

for some arbitrary integration constant κ_0, satisfies the condition in (7.1.6).

However, given an arbitrary function m, it is in fact not possible to find a holomorphic function f as above. The condition that such a function would exist is closely related to the intrinsic curvature of the underlying metric space. In fact a direct computation of Ricci scalar curvature (Equation (2.2.7)) using $g = \begin{bmatrix} x^2+y^2 & 0 \\ 0 & x^2+y^2 \end{bmatrix}$ (as used in our last example) gives us a value of 0 identically for all (x,y), thus revealing that the metric space, in the first place, was a flat one. Similar computation of curvature with m as described above would reveal that such spaces all have zero curvature. In fact the described 2-dimensional metric spaces and transformations between them constitute a special class of metrics called *conformally flat metrics* and maps called *conformal maps*. It is thus not surprising that we can in fact an isometric embedding of the metric space in $\mathbb{R}^2$.

7.2 Embeddings in Euclidean Spaces

Any chart, $C = (U, \phi)$, defines an embedding of the subset, U, of a N-dimensional Riemannian manifold in an Euclidean space, $\phi : U \to \mathbb{R}^N$. However, such a chart may not induce an isometric embedding. That means,

i. the matrix representation of the metric tensor on U, in this coordinate chart, may not be the identity matrix,

ii. geodesics in U may not map to geodesics in $\mathbb{R}^N$ (which we will informally call the 'straight lines'), as we saw in the example of Figure 7.2(a).

A 'straight line' segment in the D-dimensional flat Euclidean space (with Euclidean metric δ_{ij}) between two points, $\mathbf{p}, \mathbf{q} \in \mathbb{R}^D$, has a simple representation using the vector-space structure that is natural to $\mathbb{R}^D$: It is the set of points $\{\mathbf{v} \mid \mathbf{v} = \lambda\mathbf{p} + (1-\lambda)\mathbf{q},\ \lambda \in [0,1]\}$. This is the property that we have so far sought to exploit. Hence we have tried to find transformations that would map a given metric to the Euclidean metric – more importantly it will map geodesics to 'straight' lines in the Euclidean space.

If we can find a chart $\overline{C} = (U, \psi)$ such that the matrix representation of the metric tensor on U in this coordinate chart is the identity matrix (the Euclidean metric), we will be able to exploit the vector-space structure as above. We call the embedding due to such a chart an *isometric embedding in Euclidean space.* However, there is a weaker condition that may be satisfied – we may be able to find a chart such that the geodesics in U map to geodesics in the Euclidean space (*i.e.* to the straight lines), although not isometrically. This would still allow us to exploit the vector-space structure of the Euclidean space for finding the geodesic between two points in the original space. We call the embedding due to such a chart a *orthogeodesic embedding in Euclidean space.*

7.2.1 Isometric Embedding in Euclidean Space

The discussion of Section 7.1.2 is reminiscent of the fact that any metric space (or subset of a metric space) that is flat (*i.e.* with zero intrinsic curvature) has a locally isometric embedding in an Euclidean metric space of same dimension. That means, if g_{ij} is the matrix representation of metric tensor on a manifolds using a coordinate chart $C = (U, \phi)$, and if in this coordinates we compute the Ricci scalar curvature (Equation (2.2.7)) to obtain 0 everywhere in $Img(\phi)$, then we can always find a different coordinate chart $\overline{C} = (U, \psi)$ such that the new coordinate variables are $\overline{\mathbf{x}} = \psi(\phi^{-1}(\mathbf{x}))$ and the matrix representation of metric tensor in this coordinates is $\overline{g}_{ij} = \delta_{ij}$.

Isometric embedding of an arbitrary N-dimensional Riemannian manifold is always achievable in an Euclidean space. However the minimum dimensionality of the embedding (ambient) Euclidean space where the isometric embedding is possible is, in general, much higher than N (*e.g.* Local isometric embedding due to Jant-Cartan Theorem is possible in $\mathbb{R}^{N(N+1)/2}$ [43], while that for global embedding requires $\mathbb{R}^{N(3N+11)/2}$ for compact manifolds and $\mathbb{R}^{N(N+1)(3N+11)/2}$ for non-compact manifolds due to the Nash Isometric Embedding Theorem [65]). For such high dimensional embeddings, the geodesics on the embedded manifolds are rarely intersection of hyperplanes in the embedding space with the manifolds (*e.g.* for an unit 2-sphere embedded in $\mathbb{R}^3$ it is, but not for a

2-torus embedded in $\mathbb{R}^3$). This is unlike isometric embedding that is achievable in Euclidean space of same dimension. Thus the computation of geodesics between two points remains non-trivial even if we manage to achieve such isometric embeddings in higher dimensional Euclidean spaces.

7.2.2 Non-isometric Embedding with Geodesics Mapping to 'Straight Lines'

Definition 7.2.1 (Orthogeodesic Embedding in Euclidean Space)**.** A coordinate chart, $C = (U, \phi)$, on a subset U of a N-dimensional Riemannian manifolds is said to *induce an orthogeodesic embedding* if ϕ maps every geodesic on U to a geodesic of $\mathbb{R}^N$ (with its usual Euclidean metric). Then $\phi : U \to \mathbb{R}^N$ is called an *orthogeodesic embedding.*

Note that a coordinate chart that induces an *isometric* embedding in $\mathbb{R}^N$ also induces an orthogeodesic embedding. But the converse is not always true.

A well-known non-trivial example of such an embedding is the one due to the Beltrami-Klein model of the hyperbolic space [68]. In that model, points in the N-dimensional hyperbolic space are represented by points inside an unit N-ball in $\mathbb{R}^N$. One property of this model is that geodesics on the hyperbolic space map to 'straight lines' inside the ball. Along similar lines, the Gnomonic projection of a N-sphere maps geodesics on a half-sphere (the subset U) to straight lines in $\mathbb{R}^D$ [19]. In the following section we will present some discussions and results using these two models.

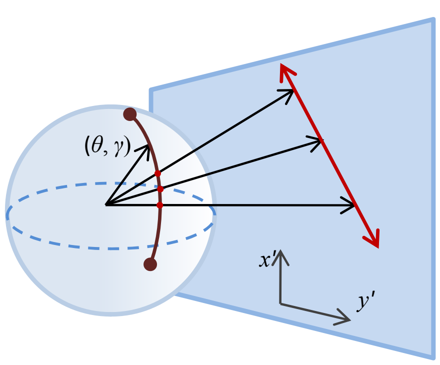
7.3 Orthogeodesic Embedding of Spaces of Constant Intrinsic Curvatures

There can be three types of metric spaces with constant intrinsic curvature everywhere: ones with positive curvature, ones with zero curvature, and ones with negative curvature. Any constant non-zero curvature space can just be scaled to obtain a space with curvature ± 1. However, two spaces can have the same constant curvature, and yet can be topologically different.

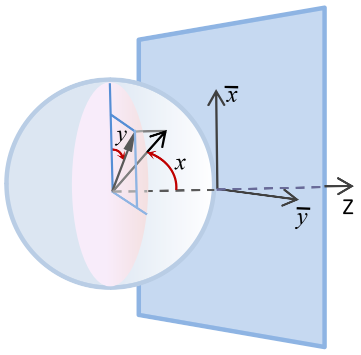
Previously in this chapter we have discussed an isometric embedding of a subset of a space with zero curvature into the Euclidean space. In this section we will construct orthogeodesic embedding (see previous section, Definition 7.2.1) of subsets of 2-dimensional spaces with constant curvatures in $\mathbb{R}^2$, and will hence see how it can be efficiently used for optimal path planning in such metric spaces.

7.3.1 Motivating Example: Gnomonic Projection of Half-sphere

Consider the unit 2-sphere embedded in $\mathbb{R}^D$, centered at the origin. The metric induced by the ambient Euclidean space gives it a constant positive curvature. One can use a coordinate chart consisting of the polar angles as the coordinate variables: $(x, y) \equiv (\theta, \gamma)$ (thus x is the latitude – the angle measured from the Z axis, and y is the longitude – the angle measured from the X axis on the XY plane. Note that the lower case and upper case letters imply completely different things: X, Y, Z are the coordinate axes in $\mathbb{R}^3$, while x, y are the spherical coordinates), that describe every point on the sphere except the poles. We are however interested in the half-sphere where $Z > 0$.



(a) Gnomonic projection maps points on the sphere to points on a plane. Geodesics are mapped to straight lines. However the map does not preserve distances. In this figure, three evenly separated points on the sphere maps to three points on the plane that are not evenly separated. However since they are on a great circle, their images lie on a straight line.



(b) The gnomonic projection hence gives an orthogeodesic embedding of half of the 2-sphere, which has a constant positive curvature everywhere, into the Euclidean 2-plane, which has zero curvature. This figure illustrates the physical interpretation of the coordinate variables introduced in Equation (7.3.2).

Figure 7.3: Gnomonic projection gives an orthogeodesic embedding of the half sphere.

The matrix representation of the metric using these coordinate variables is the well-known round metric,

$$g = \begin{bmatrix} 1 & 0 \\ 0 & \sin^2(x) \end{bmatrix} \tag{7.3.1}$$

One can perform a direct computation of Ricci scalar curvature (Equation (2.2.7)) using the above matrix representation of g, and would obtain a constant value of 2 (constant positive curvature). This immediately implies that it is impossible to find a coordinate chart such that the matrix representation of the metric will be the identity matrix even in a small open subset of the sphere – that is, it is impossible to isometrically embed any part of the 2-sphere in $\mathbb{R}^2$.

Now consider the plane $Z = 1$. The line passing through the center of the sphere and a point on the positive-Z half of the sphere intersects the sphere at an unique point. This is called the gnomonic projection of the point on the sphere (Figure 7.3(a)). Now, any geodesic on the sphere can be represented as the intersection of the sphere with a plane passing through the center of the sphere. This plane would intersect the plane $Z = 1$ in a straight line. Thus, under the gnomonic projection, geodesics on the half-sphere maps to straight lines on the $Z = 1$ plane (an orthogeodesic embedding). However, it is to be noted that this map is not an isometry.

Using a suitable Euclidean coordinate on the plane $Z = 1$, one obtains the following coordinate transformation for describing the gnomonic projection (Figure 7.3(b)),

$$\begin{aligned} \overline{x} &= \tan(x)\cos(y) \\ \overline{y} &= \tan(x)\sin(y) \end{aligned} \tag{7.3.2}$$

One can then compute the matrix representation of the metric tensor in $(\overline{x}, \overline{y})$ (which we will henceforth call the barred coordinates) to obtain the following,

$$\overline{g} \;=\; \begin{bmatrix} \dfrac{1+\overline{y}^2}{1+\overline{x}^2+\overline{y}^2} & \dfrac{-\overline{x}\;\overline{y}}{1+\overline{x}^2+\overline{y}^2} \\ \dfrac{-\overline{x}\;\overline{y}}{1+\overline{x}^2+\overline{y}^2} & \dfrac{1+\overline{x}^2}{1+\overline{x}^2+\overline{y}^2} \end{bmatrix} \tag{7.3.3}$$

Once again, a direct computation of Ricci scalar curvature using this matrix unsurprisingly gives a constant value of 2. However, this barred coordinate chart induces an orthogeodesic embedding of the half-sphere in $\mathbb{R}^2$.

One can also easily compute the inverse transformation of (7.3.2) for $x \in [0, \pi/2],\ y \in [-\pi, \pi]$,

$$\begin{aligned} x &= \arctan\left(\sqrt{\overline{x}^2+\overline{y}^2}\right) \\ y &= \operatorname{arctan2}(\overline{y},\ \overline{x}) \end{aligned} \tag{7.3.4}$$

An Example in Robot Planning

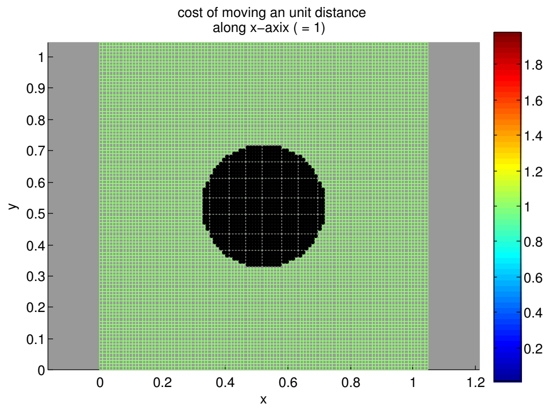
We follow an example similar to the one in Section 7.1.2. We assume that we are given the matrix representation of the metric tensor exactly as the one in Equation (7.3.1) in the *un-barred* coordinates (although this can be generalized to a much larger family of matrix representations – ones that can be reduced to (7.3.1) vis some non-singular coordinate transformation, or equivalently, ones that can be obtained from (7.3.1) via some non-singular coordinate transformation). Unlike before, this does not represent an isotropic cost function. Moreover the scalar curvature due to this metric is positive as we just discussed.

We consider the region bounded by $x \in [0,\ 1.05],\ y \in [0,\ 1.05]$ as our region of interest (Figures 7.4(a)-(b), Figure 7.5(b)). A robot needs to plan its trajectory from some point inside this region to some other, avoiding a circular obstacle at the center. Figures 7.4(a)-(b) shows the 'cost' of moving an unit distance along the x and y directions respectively in this environment (as prescribed by the metric in Equation (7.3.1)).

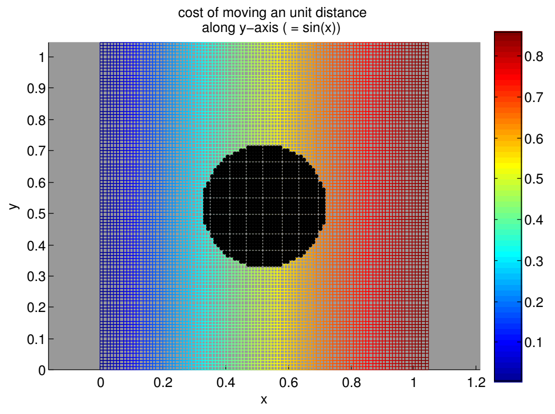
In order to solve this problem we transform the obstacle and the given start & end coordinates to the barred coordinates, and create an uniform grid in the barred coordinates. Since this coordinate system described by coordinate variables $(\overline{x}, \overline{y}$ induces an orthogeodesic embedding, we can use visibility-based techniques. Thus, after a simple graph-search based planning (with metric $\overline{g}$ – *i.e.* an edge $\overline{e}$ in the graph will have cost $\int_{\overline{e}} \overline{g}_{ij}\ \mathrm{d}\overline{x}^i\ \mathrm{d}\overline{x}^j$), we use visibility-based post-processing to obtain a 'piece-wise linear' curve (Figure 7.5(a)). Inverse-transforming this trajectory in the original unbarred coordinates gives us the desired result trajectory that minimizes the length $L = \int g_{ij}\ \mathrm{d}x^i\ \mathrm{d}x^j = \int \overline{g}_{ij}\ \mathrm{d}\overline{x}^i\ \mathrm{d}\overline{x}^j$ (where subscripts 1 and 2 in x_* respectively indicate the coordinate variables x and y. Same is implied for the barred coordinates.).

7.3.2 Klein-Beltrami Model of the Hyperbolic Plane

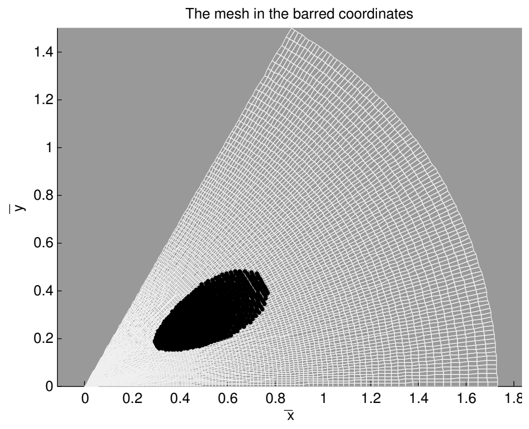
The N-dimensional hyperbolic space is a space that has a constant curvature of -1 (or a constant negative curvature after scaling). That means at every points on the manifold it locally looks like



(a) The cost for moving along the direction of x is constant everywhere. Black indicate an obstacle.

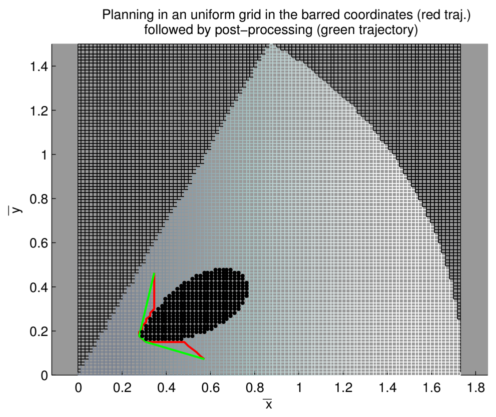


(b) The cost for moving along the y direction varies with x. Black indicate an obstacle.

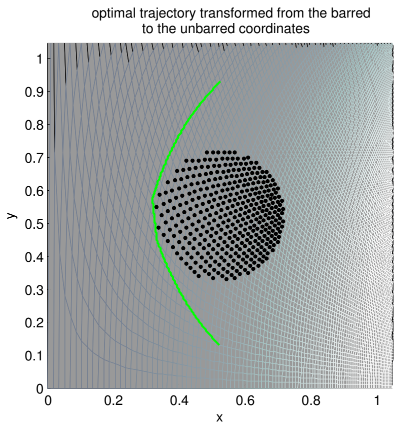


(c) The mesh in (a) and (b) transformed to the barred coordinates according to Equation (7.3.2). The axes show values of $\overline{x}$ and $\overline{y}$.

Figure 7.4: (a),(b): Anisotropic cost for robot motion as a function of position indicated by color. (c): Gnomonic projection of the mesh. Black indicates the inaccessible points (obstacles) in both the coordinates.

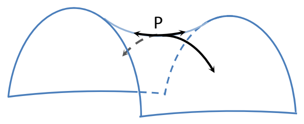


(a) An uniform mesh laid on the barred (transformed) coordinates for constructing a graph. The shortest path in the graph (red) is constrained to be oriented at multiples of 45°. A post-processing gives the piece-wise liner trajectory (green). Each linear piece is a part of a geodesic since this is an orthogeodesic embedding.

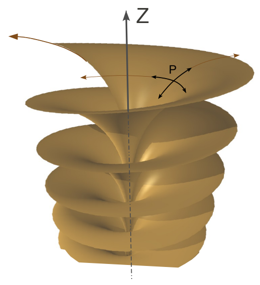


(b) The optimal trajectory in the un-barred (the original/given) coordinates is obtained by transforming the optimal trajectory in (a) back to the un-barred coordinates. This is the required least cost trajectory in the given metric space (using the inverse transformation of Equation (7.3.4)). The mesh from the barred coordinates is also transformed to show correspondence.

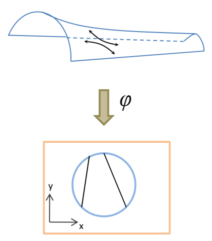
Figure 7.5: Robot planning problem in a metric space with constant positive curvature. The problem of planning a least cost trajectory from $(x_s, y_s) = (0.524, 0.126)$ to $(x_e, y_e) = (0.524, 0.922)$ in the un-barred coordinates (figure on the right) is translated to finding the least cost trajectory in the barred coordinates from $(\overline{x}_s, \overline{y}_s) = (0.5728, 0.0724)$ to $(\overline{x}_g, \overline{y}_g) = (0.3491, 0.4599)$ (figure on the left). Since the barred coordinates induce an orthogeodesic embedding, the geodesics are straight lines on the plane. The intensity of the color of the mesh is just to establish correspondence and does not indicate cost. The anisotropic cost function is properly described in Figures 7.4(a)-(b).



(a) An isolated saddle point.



(b) Dini's surface is an isometric immersion of part of the hyperbolic plane in $\mathbb{R}^3$. Every point on it resembles a saddle point.



(c) The Klein-Beltrami Model of the hyperbolic plane maps geodesics in $\mathbb{H}^2$ to chords of the unit circle in $\mathbb{R}^2$. This coordinate chart induces a orthogeodesic embedding of $\mathbb{H}^2$ in $\mathbb{R}^2$. However, the embedding is not isometric.

Figure 7.6: The hyperbolic plane.

a saddle point (Figure 7.6(a)). In this section we will mostly discuss the case when $N = 2$, when it is known as the *hyperbolic plane* (represented as $\mathbb{H}^2$). There are several possible local isometric embeddings of $\mathbb{H}^2$ in $\mathbb{R}^3$, which are typically known as pseudo-spheres (Figure 7.6(b)). However, unlike a 2-sphere, it is not possible to achive a global embedding of $\mathbb{H}^2$ (which is topologically diffeomorphic to $\mathbb{R}^2$) in $\mathbb{R}^3$ [43].

Consider the following matrix representation of a metric on a 2-dimensional manifold in a coordinate chart consisting of variables (x, y),

$$g = \begin{bmatrix} \dfrac{1-y^2}{(1-x^2-y^2)^2} & \dfrac{xy}{(1-x^2-y^2)^2} \\ \dfrac{xy}{(1-x^2-y^2)^2} & \dfrac{1-x^2}{(1-x^2-y^2)^2} \end{bmatrix} \tag{7.3.5}$$

A direct computation of the Ricci scalar curvature (Equation (2.2.7)) using this g reveals that the curvature is identically equal to -2. Thus, by definition, this coordinate chart describes a part of the hyperbolic plane. This metric, in particular, is known as the Cayley-Klein-Hilbert Metric [86], and this coordinate chart is known as the 'Klein-Beltrami Model' of hyperbolic geometry [68].

The interesting property of this particular coordinate chart on the hyperbolic plane is that geodesics passing through points (x, y) lying inside the unit disk of the embedding space ($\mathbb{R}^2$) form 'straight' chords of unit circle (Figure 7.6(c)). This coordinate chart hence clearly induces an orthogeodesic embedding of the hyperbolic plane in $\mathbb{R}^2$. However this embedding is not isometric since the lengths of the chords on the Euclidean plane do not correspond to length of the geodesics they represent.

While there are various other coordinate charts that can describe the hyperbolic place, and hence there are the corresponding matrix representations of the metric tensor (*e.g.* Poincar disc model, Poincar half-plane model, hyperboloid model), one can always find a transformation from those coordinate charts to the coordinate chart of the Klein-Beltrami model described above.

7.3.3 Coordinate Charts that Induce Orthogeodesic Embedding

We would now like to investigate the property of a coordinate chart and the corresponding matrix representation of the metric that induces an orthogeodesic embedding. Consider the geodesic equation in a coordinate chart (U, ϕ) with variables $\mathbf{x} = (x^1, x^2, \cdots, x^N)$, and g being the matrix representation of the metric tensor, on a particular Riemannian manifold,

$$\frac{\mathrm{d}^2 x^i}{\mathrm{d}t^2} + \Gamma^i_{jk} \frac{\mathrm{d}x^j}{\mathrm{d}t} \frac{\mathrm{d}x^k}{\mathrm{d}t} = 0 \tag{7.3.6}$$

The condition that the geodesic maps to a straight line in $\mathbb{R}^N$ (*i.e.* a geodesic in $\mathbb{R}^N$ under the standard Euclidean metric) is that the $[\frac{\mathrm{d}^2x^1}{\mathrm{d}t^2}, \frac{\mathrm{d}^2x^2}{\mathrm{d}t^2}, \cdots]$ (obtained from Equation (7.3.6)) be parallel to $[\frac{\mathrm{d}x^1}{\mathrm{d}t}, \frac{\mathrm{d}x^2}{\mathrm{d}t}, \cdots]$ at every point in $\mathbb{R}^N \times T_*\mathbb{R}^N$. That is,

$$\left[\frac{\mathrm{d}^2x^1}{\mathrm{d}t^2}, \frac{\mathrm{d}^2x^2}{\mathrm{d}t^2}, \cdots\right] \quad = \quad \Theta(\mathbf{x}, \dot{\mathbf{x}}) \; \left[\frac{\mathrm{d}x^1}{\mathrm{d}t}, \frac{\mathrm{d}x^2}{\mathrm{d}t}, \cdots\right]$$

for some $\Theta : \mathbb{R}^N \times T\mathbb{R}^N \to \mathbb{R}$. Using (7.3.6), this condition becomes

$$\Gamma^i_{jk}(\mathbf{x}) \;\; e^j \; e^k \quad = \quad \theta(\mathbf{x}, \mathbf{e}) \; e^i \tag{7.3.7}$$

for every $\mathbf{x} \in Img(\phi) \subseteq \mathbb{R}^N$ and every $[e^1, e^2, \cdots] \in \mathbb{R}^N$(*i.e.* coefficients of $\mathbf{e} \in T\mathbb{R}^N$), for some $\theta : \mathbb{R}^N \times \mathbb{R}^N \to \mathbb{R}$.

Given the matrix representation of the metric tensor in terms of the coordinate variables $\mathbf{x}$, the Christoffel symbols, Γ^i_{jk}, are simply functions of $\mathbf{x}$. Moreover, it is clear from (7.3.7) that θ, when expressed in the particular coordinate chart, needs to be linear in e_i.

Multiplying both sides of (7.3.7) by g_{ip} and taking sum over p we obtain,

$$\begin{aligned} \frac{1}{2}(g_{pj,k} + g_{pk,j} - g_{jk,p}) \;\; e^j \; e^k \quad &= \quad (\theta_r e^r) \; g_{ip} e^i \\ &= \quad \theta_r g_{ip} \;\; e^r \; e^i \end{aligned} \tag{7.3.8}$$

where, due to linearity we could write $\theta(\mathbf{x}, \mathbf{e}) = \theta_r e^r$, where each θ_r is a function of $\mathbf{x}$ only. Replacing the repeated indices r and i on the right hand side of the last equation by j and k respectively, and then equating the coefficients of $e^j \; e^k$ the condition becomes,

$$g_{pj,k} + g_{pk,j} - g_{jk,p} \quad = \quad \theta_j g_{kp} + \theta_k g_{jp}, \qquad \forall p, j, k \tag{7.3.9}$$

The equations are symmetric in j and k. Thus we have $N^2(N+1)/2$ equations. There are N functions, $\theta_\bullet$. Thus upon eliminating the θ's, we obtain $N(N^2 + N - 2)/2$ equations that the components of g need to satisfy.

The conditions in (7.3.9), upon elimination of the θ's, constitute the conditions for orthogeodesic embedding being induced by the specific coordinate chart. Thus, the matrix representation of the metric in equations (7.1.1) (7.3.3) and (7.3.5) would satisfy (7.3.9).

Open Question: Existence of and Finding an Orthogeodesic Embedding

The more fundamental and significantly more difficult question that one would like to answer is that of finding a coordinate chart that induces orthogeodesic embedding. This question, in essence, has some similarity with Hilbert's fourth problem [35]. With the definition of *orthogeodesic embedding* at hand, we can now look forward to put forth a rigorous statement for the problem. A given coordinate chart may not induce an orthogeodesic embedding. However, one can possibly find a coordinate transformation such that the transformed coordinates induce an orthogeodesic embedding. The existence of such a coordinate system for an arbitrary metric space is also a question worth investigation. If such coordinate charts exist only for some specific types of metric spaces, we would also like to be able to make assertions about the properties of such spaces. Besides the questions of existence of such a coordinate chart, and how to find one, one may attempt to solve the partial differential equations described by (7.3.9), thus obtaining the exact functional forms of the components of the metric tensor, given some boundary conditions.

Chapter 8

Conclusion

In this thesis we have tried to demonstrate how graph search-based algorithms, which are extremely powerful tools widely used for solving many practical robot planning problems, can be used synergistically with ideas from topology, algebraic topology, Riemannian geometry and gradient decent type algorithms. In doing so, we have been able to make some of the richer information about the underlying configuration space available to the search-based algorithms, which otherwise would have got lost due to the process of discretization and graph construction. We have kept as one of our top priorities the efficiency of the search algorithms and designed our approaches such that we do not compromise on it in order to achieve our goals. In the last two chapters we have proposed some ideas that help compensate for two of the main drawbacks of search-based algorithms - the exponential growth of complexity with increase in the dimensionality of the configuration space, and suboptimality of the shortest path in the graph because of restriction of the metric in the original configuration space to the graph. We have demonstrated the applicability of the proposed algorithms by solving some realistic example problems from robotics. Moving forward, we would like to look into more practical robotics problems and applications that require reasoning about topology and metric of the underlying configuration space, and hence be able to make use of the proposed methods in conjunction with graph search-based techniques to solve those problems.

In chapters 3 and 4 we have proposed a novel and efficient way of representing topological information of trajectories for robot motion planning. We have shown that this representation is well suited for use with graph search techniques for finding least cost paths respecting given homology class constraints as well as for exploring different homotopy classes in an environment. The method is independent of the discretization scheme, cost function or the search algorithm used. We have demonstrated the efficiency, applicability and versatility of the method in our results with examples from practical problems in robotics involving two and three dimensional configuration spaces. The analysis in Chapter 4, using tools from algebraic topology, not only provides strong theoretical justifications behind the proposed constructions and design of homology class invariants, but also help us generalize the proposed method to dimensions greater than 3. In particular, we demonstrate an example of exploration of homotopy classes in a 4 dimensional configuration space.

In Chapter 5 we presented a search-based algorithm for computing a geodesic Voronoi tessellation in non-convex environments and with non-Euclidean metric. We hence addressed the problem of deriving optimal control laws to deploy heterogeneous robotic networks in realistic environments. This has the advantage of formulating coverage as a feedback control policy that can be easily decentralized. We also presented a distributed algorithm which allows for efficient computation of the proposed control law. This algorithm is based on wavefront propagation in the same spirit of the Dijkstra Algorithm. By doing these we end up achieving a generalization of the Lloyd's algorithm to non-Euclidean metric spaces and non-convex environments. This enabled us achieve simultaneous coverage and exploration of an unknown or partially known environment in a distributed manner. Uncertainty in the environment description is introduced through the development of an entropy-based metric; enabling the computation of the instantaneous geodesic Voronoi tessellation given an uncertain environment. We provide exploration and convergence guarantees resulting from this approach and present results that demonstrate through simulation its application in real indoor environments environment.

In Chapter 6 we developed an algorithm for solving large optimization problems with pair-wise nonlinear constraints and arbitrary objective functions in a distributed fashion. We have successfully implemented the algorithm to solve the problem of multi-robot path planning with distance constraints between pairs of robots. Our theoretical analysis gives the conditions under which the algorithm converges to an optimal solution and a prescription for making sure that those conditions are satisfied. We have successfully implemented the algorithm to solve a large multi-robot path planning problem in complex environment with arbitrary shaped obstacles and distance constraints using discrete graph searches for individual sub-problems. We even extended the method to support an unordered set of tasks that the robots need to execute, while obeying the inter-robot distance constraints. We integrated the task order into the search graph of each robot and as a result obtained the order of execution of the tasks as part of the natural solution to the problem. The simulations demonstrate the efficiency of the algorithm in solving very large problems. The algorithms we proposed has therefore the potential of solving extremely complex planning problems with complicated constraints in a relatively short period of time. In addition the algorithm provides strong theoretical guarantees on solution quality.

In Chapter 7 we proposed certain ideas that we hope will inspire a new direction of research. Our preliminarily analysis shows the existence of certain metric spaces (for example, the metric on a flat space as described in Equation (7.1.1), the round metric of Equation (7.3.1), or the metric on a hyperbolic space) that allow coordinate charts which induce *orthogeodesic embedding*. We have shown, with concrete examples, how such embeddings can be useful for efficient optimal path planning in the original metric space. We have derived an explicit set of equations that describe the conditions under which the given representation of a metric induces an orthogeodesic embedding. However, the question of existence of such a coordinate chart and finding such a chart for a given metric space is still under investigation. At the moment, we are able to apply this approach to solve problems in a very limited class of metric spaces. Although many fundamental questions remain to be answered, the preliminary work suggests that we may be able to apply this method to solve practical problems. This is a nascent research direction and more thorough analysis of the conditions as well as application to realistic problems are within the scope of future work.

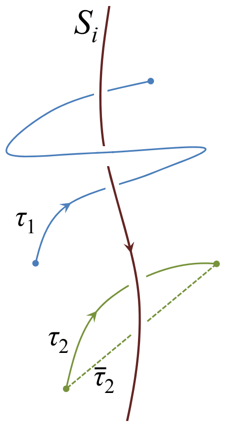
Appendix A

Looping and Non-looping trajectories in 3-dimensional Configuration Space

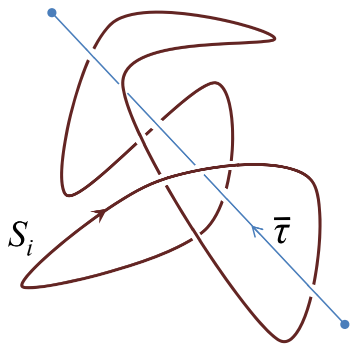
"Looping" of a trajectory around an obstacle in 3 dimensions (Figure A.1(a)) is similar in essence to *non-Jordan* curves on two-dimensional planes. However in three dimensions their precise and universal definition is more difficult. One way of identifying one of the homotopy classes of trajectories (joining a given start and an end coordinate) that do not loop around a skeleton S_i is by joining the start and the end coordinates using a straight line segment (call it $\overline{\tau}$). Then the trajectories that are homotopic to $\overline{\tau}$ form a particular homotopy class of *non-looping trajectories* w.r.t. S_i (for example, in Figure A.1(a), the class to which $\overline{\tau}_2$, and hence τ_2, belong are non-looping). However, for more complex obstacles (like knots), the notion of a *non-looping* trajectory being a straight line segment breaks down (See Figure A.1(b)). In fact the notions of *looping* and *non-looping* is imprecise in such cases. In this appendix we show that for the special simple case when S_i is an infinitely long line, the component of the H-signature $h_i(\overline{\tau})$ for a line segment $\overline{\tau}$ lies between -1 and 1. We hence propose the following mathematical definition of a *non-looping* trajectory,

Definition A.0.1 (Non-looping trajectory w.r.t. S_i)**.** In a 3-dimensional configuration space, a trajectory τ is said to be *non-looping* w.r.t. S_i if $h_i(\tau) \in (-1, 1)$ (as defined in Equation (3.4.11)). The value is in $[0, 1)$ if the trajectory goes around S_i in accordance with the *right-hand rule* with thumb pointing along the direction of the current in S_i. If the direction is opposite, the value lies in $(-1, 0]$.

This definition, in many cases, conform to our general intuition of *non-looping* trajectories. If another trajectory, τ', connecting the same start and end points as a non-looping trajectory τ, goes on the "other side of the obstacle" without looping around it, then $\tau \sqcup -\tau'$ forms a closed loop that encloses S_i. Then, $h_i(\tau \sqcup -\tau') = \pm 1 = \text{sign}(h_i(\tau \sqcup -\tau'))$. But since, τ and $\tau \sqcup -\tau'$ goes around S_i in the same orientation, we have $\text{sign}(h_i(\tau \sqcup -\tau')) = \text{sign}(h_i(\tau))$. Again by property of line integration, $h_i(\tau \sqcup -\tau') = h_i(\tau) - h_i(\tau')$. Thus, $h_i(\tau') = h_i(\tau) - \text{sign}(h_i(\tau))$. Thus we have the



(a) Trajectory that loops around a skeleton & one that doesn't. In this figure $h_i(\tau_1) > 1$ and $0 < h_i(\tau_2) = h_i(\overline{\tau}_2) < 1$.



(b) In the most general case, it is difficult to precisely identify a *non-looping* homotopy class.

Figure A.1: Looping and non-looping trajectories.

following definition.

Definition A.0.2 (Complementary Homology Class)**.** Given a trajectory τ that is *non-looping* w.r.t. all the skeletons in the environment (*i.e.* $h_i(\tau) \in (-1,1)\ \forall\ i = 1,2,\ldots,M$), we define the *Complementary Homology Class* of the class of τ to be the one for which the H-signature is $\mathcal{H}(\tau) - \text{sign}(\mathcal{H}(\tau))$, where $\text{sign}(\cdot)$ gives the vector of signs of the elements of its input vector.

We now show that when S_i is an infinitely long line, and $\overline{\tau}$ is a line segment, the component of the H-signature $h_i(\overline{\tau})$ lies between -1 and 1. Making use of the result from Equation (3.4.8), if the current carrying line segment stretches to infinity in both direction (*i.e.* it becomes a line), we have $\alpha' = \frac{\pi}{2}$ and $\alpha = -\frac{\pi}{2}$. The virtual magnetic field due to S_i at a point $\mathbf{r}$ becomes

$$\mathbf{B}_i = \frac{1}{4\pi}\frac{2\ \hat{\mathbf{n}}}{\|\mathbf{d}\|} = \frac{1}{2\pi}\frac{\hat{\mathbf{n}}}{\|\mathbf{d}\|} \tag{A.0.1}$$

Note that the contribution of the closing curve at infinity (Construction 3.4.3) becomes zero in the above quantity.

Now consider the straight line segment trajectory $\overline{\tau} = \overline{\mathbf{r}_A\mathbf{r}_B}$. Let the line containing $\overline{\tau}$ (*i.e.* formed by extending $\overline{\tau}$ to infinity in both directions) be T (Figure A.2). Consider the shortest distance between S_i and T and let it be D. Assuming S_i and T are not parallel, there is a unique point on each of these line ($\mathbf{p}$ and $\mathbf{q}$ respectively) that are closest and are separated by the distance D. The line segment joining the closest points, $\overline{\mathbf{pq}}$, is perpendicular to both S_i and T. The main diagram of Figure A.2 shows the projection of S_i and T on a plane perpendicular to $\overline{\mathbf{pq}}$. Note that this plane (the plane of the paper) is parallel to both S_i and T, since it is perpendicular to $\overline{\mathbf{pq}}$.

We define an orthonormal coordinate system with unit vectors $\hat{\mathbf{i}}$ pointing along S_i in the direction of the current, and unit vector $\hat{\mathbf{k}}$ pointing along $\overline{\mathbf{pq}}$. Using these, and referring to Figure A.2, we

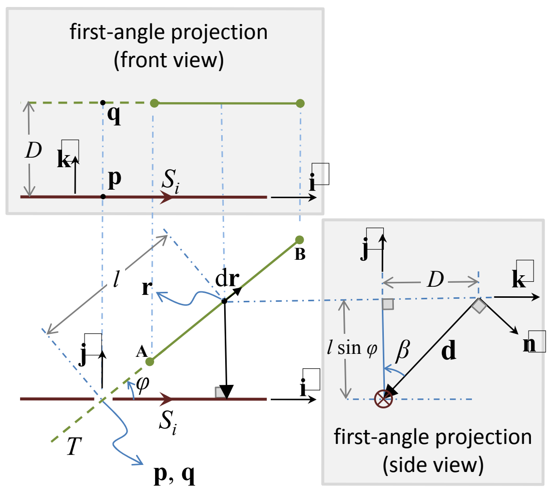


Figure A.2: An infinitely long skeleton and h-signature of a straight line segment.

now can write the following equations,

$$\begin{gathered}\|\mathbf{d}\|^2 = D^2 + l^2 \sin^2 \phi \\ \hat{\mathbf{n}} = \cos\beta\ \hat{\mathbf{k}} - \sin\beta\ \hat{\mathbf{j}}\ , \quad \mathrm{d}\mathbf{r} = (\cos\phi\ \hat{\mathbf{i}} + \sin\phi\ \hat{\mathbf{j}})\ \ \mathrm{d}l\end{gathered} \tag{A.0.2}$$

where, ϕ is a constant angle between S_i and T on the plane of the paper, $\cos\beta = \frac{l \sin\phi}{\|\mathbf{d}\|}$, $\sin\beta = \frac{D}{\|\mathbf{d}\|}$, and l is the length parameter along T starting at $\mathbf{q}$. Thus from (A.0.1),

$$\mathbf{B}_i \cdot\ \mathrm{d}\mathbf{r} = -\frac{1}{2\pi}\frac{\sin\beta\ \sin\phi}{\|\mathbf{d}\|}\ \mathrm{d}l = \ -\frac{D\sin\phi}{2\pi}\frac{\mathrm{d}l}{D^2 + l^2\sin^2\phi} \tag{A.0.3}$$

Thus,

$$\begin{aligned}\int_{\mathcal{T}} \mathbf{B}_i \cdot\ \mathrm{d}\mathbf{r} &= -\frac{D\sin\phi}{2\pi}\int_{l_A}^{l_B}\frac{\mathrm{d}l}{D^2 + l^2\sin^2\phi} \\ &= -\frac{1}{2\pi}\left(\arctan\left(\frac{l_B}{D/\sin\phi}\right) - \arctan\left(\frac{l_A}{D/\sin\phi}\right)\right)\end{aligned} \tag{A.0.4}$$

An arctangent of a quantity, with consideration for proper quadrants, can assume values between $-\pi$ and π. Thus the quantity within the outer brackets of Equation (A.0.4), that is the difference of two arctangents, can assume values between -2π and 2π. Thus the integral $\int_{\mathcal{T}} \mathbf{B}_i \cdot\ \mathrm{d}\mathbf{r}$, can assume values between -1 and 1. Thus, a straight line segment trajectory indeed has the value of $h_i(\tau)$ in $(-1, 1)$ for this simple case of infinitely long line S_i.

Appendix B

Proofs Related to Homology of Punctured Euclidean Spaces

B.1 Proof of Proposition 4.2.1

Statement of the Proposition: *Let O be a compact, locally contractible and orientable submanifolds of $\mathbb{R}^D$. Let S be a $(D-N)$-dimensional compact, locally contractible and orientable submanifolds of O contained in the interior of O such that $H_{D-N}(S) \approxeq H_{D-N}(O)$ and $H_{D-N}(O,S) \approxeq 0$. Then the inclusion map $\overline{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - S)$ induces an isomorphism $\overline{i}_{*:N-1} : H_{N-1}(\mathbb{R}^D - O) \to H_{N-1}(\mathbb{R}^D - S)$.*

Proof. One consequence of O and S being orientable manifolds, and our coefficients being in field $\mathbb{R}$, is that the homology groups of the spaces are freely and finitely generated (see pp. 198 of [40]).

Given a compact, locally contractible and orientable subset $K \subset \mathbb{R}^D$, such that the homology groups of K are finitely and freely generated, consider the following isomorphism,

$$\begin{aligned} H_{D-n}(K) &\approxeq H^{D-n}(K) \text{ [Using Corollary 3.3 of [40] and noting that the torsion} \\ &\qquad\qquad \text{subgroups of freely generated groups are trivial.]} \\ &\approxeq H_n(\mathbb{R}^D, \mathbb{R}^D - K) \text{ [By Theorem 3.46 of [40].]} \\ &\approxeq H_{n-1}(\mathbb{R}^D - K) \text{ [Using the long exact sequence of homology groups} \\ &\qquad\qquad \text{for pair } (\mathbb{R}^D, \mathbb{R}^D - K)\text{, and using contractibility of } \mathbb{R}^D.] \end{aligned} \tag{B.1.1}$$

Thus, for $n = N$, setting $K = O$ and $K = S$, we respectively obtained

$$H_{D-N}(O) \approxeq H_{N-1}(\mathbb{R}^D - O) \quad \text{and} \quad H_{D-N}(S) \approxeq H_{N-1}(\mathbb{R}^D - S)$$

This, along with the given condition, $H_{D-N}(O) \approxeq H_{D-N}(S)$, gives the following isomorphism,

$$H_{N-1}(\mathbb{R}^D - O) \approxeq H_{N-1}(\mathbb{R}^D - S) \tag{B.1.2}$$

Now we need to prove that this isomorphism is induced by the inclusion $\bar{i} : (\mathbb{R}^D - O) \hookrightarrow (\mathbb{R}^D - S)$.

Consider the following sequence of isomorphisms,

$$\begin{aligned}
H_N(\mathbb{R}^D - S, \mathbb{R}^D - O) &\approxeq H^{D-N}(O-S) \text{ [By setting } M = \mathbb{R}^D - S,\ K = O - S \text{ in} \\
&\qquad\qquad \text{Theorem 3.46 of [40].]} \\
&\approxeq H_c^{D-N}(O-S) \text{ [Since } O-S \text{ is compact, its cohomology groups are isomorphic} \\
&\qquad\qquad \text{to cohomology with compact support (pp. 242 of [40])]} \\
&\approxeq \varinjlim_U H^{D-N}(O-S, U-S) \text{ [By definition of cohomology with compact support.]} \\
&\approxeq \varinjlim_U H^{D-N}(O, U) \text{ [By excision.]} \\
&\approxeq H^{D-N}(O, S) \text{ [Due to local contractibility of } S,\ U \text{ approaches a local} \\
&\qquad\qquad \text{neighborhood of } S \text{ in the limit, which deformation retracts to } S \\
&\qquad\qquad \text{(see Theorem A.7 and 3.44 of [40]).]} \\
&\approxeq H_{D-N}(O, S) \text{ [Due to trivial torsion sub-groups by hypothesis} \\
&\qquad\qquad \text{(Corollary 3.3 of [40]).]}
\end{aligned} \tag{B.1.3}$$

In the third isomorphism, U represents an open set in O containing S (thus $U - S$ are open sets in $O-S$). The limit is taken over a sequence of such sets, $\{U_\alpha\}$, approaching S. The arguments behind the third to fifth isomorphisms are similar in essence to the ones used in the proof of Alexander Duality (Theorem 3.44) in [40].

Thus, due to the isomorphism in (B.1.3) along with the given condition $H_{D-N}(O, S) \approxeq 0$, we have

$$H_N(\mathbb{R}^D - S, \mathbb{R}^D - O) \approxeq 0 \tag{B.1.4}$$

Consider the following part of the long exact sequence of homology groups for pair $(\mathbb{R}^D - S, \mathbb{R}^D - O)$,

$$\cdots \longrightarrow H_N(\mathbb{R}^D - S, \mathbb{R}^D - O) \xrightarrow{\bar{\partial}_*} H_{N-1}(\mathbb{R}^D - O) \xrightarrow{\bar{i}_{*:N-1}} H_{N-1}(\mathbb{R}^D - S) \xrightarrow{\bar{j}_*} H_{N-1}(\mathbb{R}^D - S, \mathbb{R}^D - O) \longrightarrow \cdots$$

We have just shown that $H_N(\mathbb{R}^D - S, \mathbb{R}^D - O) \approxeq 0$. Thus the image of $\bar{\partial}_*$ is 0 in $H_{N-1}(\mathbb{R}^D - O)$. By exactness of the sequence, that is the kernel of $\bar{i}_{*:N-1}$.

Now $H_{N-1}(\mathbb{R}^D - O)$ and $H_{N-1}(\mathbb{R}^D - S)$ are isomorphic by (B.1.2). Moreover they are freely and finitely generated. Thus, a homomorphism between them that has a zero kernel must be an isomorphism. Hence $\bar{i}_{*:N-1}$ is an isomorphism. ■

B.2 Proof of Proposition 4.3.5

Statement of the Proposition: *If $H_N(X) = H_{N-1}(X) = 0$, and $H_{D-N}(B) = H_{D-N-1}(B) = 0$, then the linking number between cycles $\overline{\varsigma} \in Z_{N-1}(A)$ and $\overline{\mu} \in Z_{D-N}(Y,B)$ is uniquely identified by the value of (i.e. a complete invariant for the linking number is)*

$$(-1)^{D-N} \int_{\overline{\varsigma}\times\overline{u}} p^*(\eta_0)$$

where $\overline{u} \in Z_{D-N}(Y)$ is such that $j'(\overline{u}) = \overline{\mu}$, with j' the quotient map $Y \to Y/B$ (See Thm. 2.13 of [40] – note that for a given $\overline{\mu}$, in general, there can be many possible choices for $\overline{u}$), and $\eta_0 \in \Omega^{D-1}_{dR}(\mathbb{R} - \{0\})$ is a closed but non-exact differential form in $(\mathbb{R} - \{0\})$ such that $[\eta_0] \in H^{D-1}_{dR}(\mathbb{R} - \{0\}) \approxeq \mathbb{R}$ is a generator of $H^{D-1}_{dR}(\mathbb{R} - \{0\})$. .

Proof. By the condition of Proposition 4.3.3, we have already seen using the long exact sequence for the pair (X, A), that the map $\partial_* : H_N(X,A) \to H_{N-1}(A)$ is invertible.

Similarly, from the long exact sequence for the pair (Y, B), using the condition $H_{D-N}(B) = H_{D-N-1}(B) = 0$, the map ${j'}_* : H_{D-N}(Y) \to H_{D-N}(Y,B)$ is an isomorphism.

Finally, the contractibility of $\mathbb{R}^D$ gives us the isomorphism $\partial''_* : H_{D-1}(\mathbb{R}^D, \mathbb{R}^D - \{0\}) \to H_{D-1}(\mathbb{R}^D - \{0\})$.

Thus we have the following diagram,

$$\begin{tikzcd}
H_N(X,A) \arrow[r, shift left, "\partial_*"] & H_{N-1}(A) \arrow[l, shift left, "\partial_*^{-1}"] \\
\times & \times \\
H_{D-N}(Y,B) \arrow[r, shift left, "{j'}_*^{-1}"] \arrow[d, "\times"] & H_{D-N}(Y) \arrow[l, shift left, "{j'}_*"] \arrow[d, "\times"] \\
H_D(X\times Y, A\times Y \cup X\times B) \arrow[d, "(-1)^{D-N}p_*"] & H_{D-N}(A\times Y) \arrow[d, "(-1)^{D-N}p_*"] \\
H_D(\mathbb{R}^D, \mathbb{R}^D - \{0\}) \arrow[r, shift left, "\partial''_*"] & H_{D-1}(\mathbb{R}^D - \{0\}) \arrow[l, shift left, "{\partial''}_*^{-1}"]
\end{tikzcd}$$

Clearly, this diagram commutes since all the horizontal arrows are isomorphisms, and the vertical arrows represent the same sequence of morphisms for the two columns.

From the diagram, considering the two possible representations for the map $H_{N-1}(A) \times H_{D-N}(Y,B) \to H_D(\mathbb{R}^D, \mathbb{R}^D - \{0\})$, we have the following expressions for linking number of $\varsigma \in H_{N-1}(A)$ and $\mu \in H_{D-N}(Y,B)$,

$$\begin{aligned}
\mathscr{L}(\varsigma,\mu) &= (-1)^{D-N}\ p_*\ (\partial_*^{-1}\varsigma \times \mu) && \text{[By following the sequence marked in red]} \\
&= (-1)^{D-N}\ {\partial''}_*^{-1} \circ p_*\ (\varsigma \times {j'}_*^{-1}\mu) && \text{[By following the sequence marked in green]}
\end{aligned}$$

Now, due to the isomorphism ∂''_*, a linking number $l \in H_D(\mathbb{R}^D, \mathbb{R}^D - \{0\})$ is uniquely identified by its corresponding element $\partial''_* l \in H_{D-1}(\mathbb{R}^D - \{0\})$. Thus the linking number between ς and μ is uniquely identified by $\mathcal{L}(\varsigma,\mu) \overset{def.}{=} \partial''_*\mathscr{L}(\varsigma,\mu) = (-1)^{D-N}\ p_*\ (\varsigma \times {j'}_*^{-1}\mu)$.

Again, by the *Universal Coefficient Theorem*, $H^{D-1}(\mathbb{R}^D - \{0\}) \approxeq \mathrm{Hom}(H_{D-1}(\mathbb{R}^D - \{0\}), \mathbb{R}) \approxeq$

$\mathbb{R}$. This, along with the *De Rham theorem*, implies that the homology class (with coefficients in $\mathbb{R}$) of any cycle $\overline{m} \in Z_{D-1}(\mathbb{R}^D - \{0\})$ is uniquely identified by the value of the integral $\int_{\overline{m}} \eta_0$ (*i.e.* the value of the integral is a complete invariant for the homology class of $\overline{m}$).

Thus, using the functoriality of homology, the linking number between cycles $\overline{\varsigma} \in Z_{N-1}(A)$ and $\overline{\mu} \in Z_{D-N}(Y, B) = Z_{D-N}(Y)$ is uniquely identified by

$$\int_{(-1)^{D-N} p(\overline{\varsigma} \times \overline{u})} \eta_0$$

where $\overline{u} \in Z_{D-N}(Y)$ is such that ${j'}_*([\overline{u}]) = [\overline{\mu}]$. Upon a pullback via p, this integral becomes equal to

$$(-1)^{D-N} \int_{\overline{\varsigma} \times \overline{u}} p^*(\eta_0)$$

Hence proved. ■

B.3 Proof of Proposition 4.4.1

Statement of the Proposition: *If $\widetilde{\mathcal{S}} = \bigcup_{i=1}^m S_i$, where each S_i is path connected, compact, closed, locally contractible and orientable $(D-N)$-dimensional manifold such that $S_i \cap S_j = \emptyset$, $\forall i \neq j$, then*

$$H_{N-1}(\mathbb{R}^D - \widetilde{\mathcal{S}}) \quad \approxeq \quad \bigoplus_{k=1}^{m} H_{N-1}(\mathbb{R}^D - S_k) \quad \approxeq \quad \mathbb{R}^m$$

The first isomorphism is induced by the direct sum of the inclusion maps $\tilde{i}_k : (\mathbb{E}^D - \widetilde{\mathcal{S}}) \hookrightarrow (\mathbb{E}^D - S_k)$.

Proof. We start by observing that for any p,

$$\begin{aligned} (\mathbb{R}^D - S_p) \ \cup \ (\mathbb{R}^D - \cup_{i=p+1}^m S_i) \quad &= \quad \mathbb{R}^D \\ (\mathbb{R}^D - S_p) \ \cap \ (\mathbb{R}^D - \cup_{i=p+1}^m S_i) \quad &= \quad \mathbb{R}^D - \cup_{i=p}^m S_i \end{aligned}$$

We thus use the Mayer-Vietoris sequence [40] for the triads $(\mathbb{R}^D,\ \mathbb{R}^D - S_p,\ \mathbb{R}^D - \bigcup_{i=p+1}^m S_i)$ for $p = 1, 2, \cdots, m-1$. For an arbitrary p, using the contractibility of $\mathbb{R}^D$, the part of the sequence that is of interest to us is,

$$0 \longrightarrow H_{N-1}(\mathbb{R}^D - \cup_{i=p}^m S_i) \xrightarrow{(\tilde{u}_{p*}, \tilde{v}_{p*})} H_{N-1}(\mathbb{R}^D - S_p) \oplus H_{N-1}(\mathbb{R}^D - \cup_{i=p+1}^m S_i)) \longrightarrow 0 \tag{B.3.1}$$

Due to exactness, the middle map (which is induced by the respective inclusion maps) is an isomorphism. Note that $\tilde{u}_{1*} = \tilde{i}_{1*}$ and, $\tilde{v}_{1*} \circ \tilde{v}_{2*} \circ \cdots \tilde{v}_{(p-1)*} \circ \tilde{u}_{p*} = \tilde{i}_{p*}$.

Starting from $p = 1$, if we successively apply the isomorphism to the second homology group in sequence (B.3.1), we obtain the following sequence of isomorphisms,

$$\begin{aligned}
H_{N-1}(\mathbb{R}^D - \widetilde{\mathcal{S}}) \quad & \underset{\approxeq}{\xrightarrow{(\tilde{i}_{1*},\tilde{v}_{1*})}} \quad H_{N-1}(\mathbb{R}^D - S_1) \oplus H_{N-1}(\mathbb{R}^D - \cup_{i=2}^m S_i)) \\
& \underset{\approxeq}{\xrightarrow{(\tilde{i}_{1*},\tilde{i}_{2*},\tilde{v}_{2*})}} \quad H_{N-1}(\mathbb{R}^D - S_1) \oplus H_{N-1}(\mathbb{R}^D - S_2) \oplus H_{N-1}(\mathbb{R}^D - \cup_{i=3}^m S_i)) \\
& \qquad \cdots \qquad\qquad \cdots \\
& \underset{\approxeq}{\xrightarrow{\oplus_{k=1}^m \tilde{i}_{k*}}} \quad \bigoplus_{k=1}^m H_{N-1}(\mathbb{R}^D - S_k)
\end{aligned} \tag{B.3.2}$$

The fact that this is isomorphic to $\mathbb{R}^m$ follows from Equation (4.4.1), where we showed $H_{N-1}(\mathbb{R}^D - S_k) \approxeq \mathbb{R}$. ■

Appendix C

Proofs Related to Distributed Optimization

Please refer to Section 6.4.1 for meaning of the notations used in the proves.

C.1 Proof of Theorem 6.4.2

Statement of the Theorem: *If the Step Direction, V^k, returned by procedure ComputeStepDirection at the k^{th} iteration in Line 5 of the Algorithm 6.3.1, along with the chosen Step Size, ϵ^k, define a Separable Optimal Flow at W^k for Ψ_{r_κ}, $\forall\ k$, then $\forall\ k$:*

$$\{\pi_1^k, \ldots, \pi_N^k\} = \mathrm{argmin}_{\{\pi\}} \left[\sum_{i \in \mathcal{N}^N} c(\pi_i) + \sum_{\{ij\} \in \mathcal{P}^N} W_{ij}^k \cdot \Omega_{ij}(\pi_i, \pi_j) \right]$$

Proof. The theorem clearly holds when for $k = 0$ since $W_{ij}^0 = 0$ for all $\{ij\} \in \mathcal{P}^N$.

We prove the theorem by induction. Assume it holds for $k = 0$ through $k = \kappa$. We will now prove that it continues to hold for the $k = \kappa + 1$. By inductive assumption we have:

$$\{\pi_1^\kappa, \pi_2^\kappa, \cdots, \pi_N^\kappa\} \ = \ \mathrm{argmin}_{\pi_1, \pi_2, \cdots, \pi_N} \left(\sum_{i \in \mathcal{N}^N} c_i(\pi_i) + \sum_{\{ij\} \in \mathcal{P}^N} W_{ij}^\kappa \Omega_{ij}(\pi_i, \pi_j) \right)$$

That is,

$$\pi_i^\kappa = \overline{\Pi}_i(W^\kappa), \quad \forall i \in \mathcal{N}^N \tag{C.1.1}$$

We also note that, by Line 8 of the Algorithm,

$$\pi_j^{\kappa+1} = \pi_j^\kappa = \overline{\Pi}_j(W^\kappa), \quad \forall \ \ j \in \mathcal{N}_{-r_\kappa}^N \tag{C.1.2}$$

We prove by contradiction. Let us assume that

$$\{\pi_1^{\kappa+1}, \pi_2^{\kappa+1}, \cdots, \pi_N^{\kappa+1}\} \neq \text{argmin}_{\pi_1,\pi_2,\cdots,\pi_N} \left(\sum_{i\in\mathcal{N}^N} c_i(\pi_i) \;+\; \sum_{\{ij\}\in\mathcal{P}^N} W_{ij}^{\kappa+1}\Omega_{ij}(\pi_i,\pi_j) \right)$$

This implies that there exist $\pi_1', \pi_2', \cdots, \pi_N'$ given by

$$\{\pi_1', \pi_2', \cdots, \pi_N'\} = \text{argmin}_{\pi_1,\pi_2,\cdots,\pi_N} \left(\sum_{i\in\mathcal{N}^N} c_i(\pi_i) \;+\; \sum_{\{ij\}\in\mathcal{P}^N} W_{ij}^{\kappa+1}\Omega_{ij}(\pi_i,\pi_j) \right)$$

such that,

$$\pi_i' = \overline{\Pi}_i(W^{\kappa+1}), \quad \forall i \in \mathcal{N}^N \tag{C.1.3}$$

and

$$\sum_{i\in\mathcal{N}^N} c_i(\pi_i^{\kappa+1}) + \sum_{\{ij\}\in\mathcal{P}^N} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i^{\kappa+1},\pi_j^{\kappa+1}) \;>\; \sum_{i\in\mathcal{N}^N} c_i(\pi_i') + \sum_{\{ij\}\in\mathcal{P}^N} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i',\pi_j') \tag{C.1.4}$$

Again, by the algorithm, it holds that:

$$\pi_{r_\kappa}^{\kappa+1} = \text{argmin}_{\pi_{r_\kappa}} \left(c_{r_\kappa}(\pi_{r_\kappa}) \;+\; \sum_{\{ir_\kappa\}\in\mathcal{P}_{r_\kappa}^N} W_{ir_\kappa}^{\kappa+1}\cdot\Omega_{ir_\kappa}(\pi_i^\kappa,\pi_{r_\kappa}) \right) \tag{C.1.5}$$

Noting that $\{ir_\kappa\} \in \mathcal{P}_{r_\kappa}^N \Rightarrow i \in \mathcal{N}_{-r_\kappa}^N$, and using equation (C.1.2), we get from (C.1.5),

$$\begin{aligned}
\pi_{r_\kappa}^{\kappa+1} &= \text{argmin}_{\pi_{r_\kappa}} \Big(c_{r_\kappa}(\pi_{r_\kappa}) \;+\; \textstyle\sum_{\{ir_\kappa\}\in\mathcal{P}_{r_\kappa}^N} W_{ir_\kappa}^{\kappa+1}\cdot\Omega_{ir_\kappa}(\pi_i^{\kappa+1},\pi_{r_\kappa}) \Big) \\
&= \text{argmin}_{\pi_{r_\kappa}} \Big(c_{r_\kappa}(\pi_{r_\kappa}) \;+\; \textstyle\sum_{\{ir_\kappa\}\in\mathcal{P}_{r_\kappa}^N} W_{ir_\kappa}^{\kappa+1}\cdot\Omega_{ir_\kappa}(\overline{\Pi}_i(W^\kappa),\pi_{r_\kappa}) \Big) \\
&= \Pi_{r_\kappa}(W^{\kappa+1}, W^\kappa)
\end{aligned}$$

$$\Rightarrow \qquad \Psi_{r_\kappa}(W^{\kappa+1}, W^\kappa) = c_{r_\kappa}(\pi_{r_\kappa}^{\kappa+1}) \;+\; \sum_{\{ir_\kappa\}\in\mathcal{P}_{r_\kappa}^N} W_{ir_\kappa}^{\kappa+1}\cdot\Omega_{ir_\kappa}(\pi_i^{\kappa+1},\pi_{r_\kappa}^{\kappa+1}) \tag{C.1.6}$$

Also, from (C.1.3) and (6.4.4),

$$\pi_{r_\kappa}' = \overline{\Pi}_{r_\kappa}(W^{\kappa+1}) = \Pi_{r_\kappa}(W^{\kappa+1}, W^{\kappa+1})$$

Thus,

$$\Psi_{r_\kappa}(W^{\kappa+1}, W^{\kappa+1}) = c_{r_\kappa}(\pi_{r_\kappa}') + \sum_{\{ir_\kappa\}\in\mathcal{P}_{r_\kappa}^N} W_{ir_\kappa}^{\kappa+1}\cdot\Omega_{ir_\kappa}(\pi_i',\pi_{r_\kappa}') \tag{C.1.7}$$

Thus from (C.1.4), (C.1.6) and (C.1.7), upon rearrangement,

$$\begin{aligned}\sum_{i\in\mathcal{N}^N_{-r_\kappa}} c_i(\pi_i^{\kappa+1}) \quad &+ \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i^{\kappa+1},\pi_j^{\kappa+1}) \quad + \quad \Psi_{r_\kappa}(W^{\kappa+1},W^\kappa)\\ &> \sum_{i\in\mathcal{N}^N_{-r}} c_i(\pi_i') \quad + \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i',\pi_j') \quad + \quad \Psi_{r_\kappa}(W^{\kappa+1},W^{\kappa+1})\end{aligned}$$

$$\begin{aligned}\Rightarrow \quad \sum_{i\in\mathcal{N}^N_{-r_\kappa}} \big(c_i(\pi_i^{\kappa+1})-c_i(\pi_i')\big) \quad &+ \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\big(\Omega_{ij}(\pi_i^{\kappa+1},\pi_j^{\kappa+1})-\Omega_{ij}(\pi_i',\pi_j')\big)\\ &> \quad \Psi_{r_\kappa}(W^{\kappa+1},W^{\kappa+1})-\Psi_{r_\kappa}(W^{\kappa+1},W^\kappa)\end{aligned} \tag{C.1.8}$$

On the other hand, according to our inductive assumption, $\{\pi_1^\kappa,\pi_2^\kappa,\cdots,\pi_N^\kappa\}$ is a minimum for the global objective function with W^κ. Thus,

$$\overline{U}(\{\pi_1^\kappa,\pi_2^\kappa,\cdots,\pi_N^\kappa\},W^k) \quad\le\quad \overline{U}(\{\pi_1',\cdots,\pi_{r_\kappa-1}',\Pi_{r_\kappa}(W^\kappa,W^{\kappa+1}),\pi_{r_\kappa+1}',\cdots,\pi_N'\},W^k) \tag{C.1.9}$$

Now, since V^κ is a *Separable Flow Direction Direction* (Definition 6.4.1) according to our hypothesis, it holds that $W_{ij}^{\kappa+1}-W_{ij}^\kappa \;=\; V_{ij}^\kappa=0\;\;\forall\{ij\}\in\mathcal{P}^N-\mathcal{P}^N_{r_\kappa}$. Again from (C.1.2) we have $\pi_j^{\kappa+1} = \pi_j^\kappa\;\forall j\in\mathcal{N}^N_{-r_\kappa}$. Using these, along with (C.1.1) and (C.1.3), we get from (C.1.9),

$$\begin{aligned}\sum_{i\in\mathcal{N}^N} c_i(\pi_i^\kappa) \quad &+ \sum_{\{ij\}\in\mathcal{P}^N} W_{ij}^\kappa\cdot\Omega_{ij}(\pi_i^\kappa,\pi_j^\kappa)\\ &\le \sum_{i\in\mathcal{N}^N_{-r_\kappa}} c_i(\pi_i') \quad + \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^\kappa\cdot\Omega_{ij}(\pi_i',\pi_j')\\ &\qquad + \quad c_i(\Pi_{r_\kappa}(W^\kappa,W^{\kappa+1})) \quad + \sum_{\{ir_\kappa\}\in\mathcal{P}^N_{r_\kappa}} W_{ir_\kappa}^\kappa\cdot\Omega_{ir_\kappa}(\pi_i',\Pi_{r_\kappa}(W^\kappa,W^{\kappa+1}))\end{aligned}$$

$$\begin{aligned}\Rightarrow \quad \sum_{i\in\mathcal{N}^N_{-r_\kappa}} c_i(\pi_i^\kappa) \quad &+ \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i^\kappa,\pi_j^\kappa) \quad + \quad \Psi_{r_\kappa}(W^\kappa,W^\kappa)\\ &\le \sum_{i\in\mathcal{N}^N_{-r_\kappa}} c_i(\pi_i') \quad + \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\Omega_{ij}(\pi_i',\pi_j') \quad + \quad \Psi_{r_\kappa}(W^\kappa,W^{\kappa+1})\end{aligned}$$

$$\begin{aligned}\Rightarrow \quad \sum_{i\in\mathcal{N}^N_{-r_\kappa}} \big(c_i(\pi_i^{\kappa+1})-c_i(\pi_i')\big) \quad &+ \sum_{\{ij\}\in(\mathcal{P}^N-\mathcal{P}^N_{r_\kappa})} W_{ij}^{\kappa+1}\cdot\big(\Omega_{ij}(\pi_i^{\kappa+1},\pi_j^{\kappa+1})-\Omega_{ij}(\pi_i',\pi_j')\big)\\ &\le \quad \Psi_{r_\kappa}(W^\kappa,W^{\kappa+1}) \;-\; \Psi_{r_\kappa}(W^\kappa,W^\kappa)\end{aligned} \tag{C.1.10}$$

Thus from (C.1.8) and (C.1.10),

$$\begin{aligned}\Psi_{r_\kappa}(W^\kappa+\epsilon^\kappa V^\kappa,W^\kappa+\epsilon^\kappa V^\kappa) \;&-\; \Psi_{r_\kappa}(W^\kappa+\epsilon^\kappa V^\kappa,W^\kappa)\\ &-\; \Psi_{r_\kappa}(W^\kappa,W^\kappa+\epsilon^\kappa V^\kappa) \;+\; \Psi_{r_\kappa}(W^\kappa,W^\kappa) \quad<\quad 0\end{aligned} \tag{C.1.11}$$

However this is a contradiction to our assumption that the V^κ and ϵ^κ defines a *Separable Optimal Flow* at W^κ for Ψ_{r_κ}. Hence our assumption of the existence of $\{\pi_1',\pi_2',\cdots,\pi_N'\}$ was incorrect. This proves Theorem 6.4.2. ■

C.2 Proof of Theorem 6.4.4

Statement of the Theorem: *If the Step Direction, V^k, returned by the procedure ComputeStepDirection at the k^{th} iteration in Line 5 of the Algorithm 6.3.1, along with the chosen Step Size, ϵ^k, is an Ascent Flow as well as a Separable Optimal Flow (for the r_k^{th} partition) at W^k, for every k, then the Algorithm converges to an optimal solution, if one exists.*

Proof. Due to Theorem 6.4.2, the *Separable Optimal Flow* condition implies that $\pi_i^k = \overline{\Pi}_i(W^k), \quad \forall i, k$. That means, as we change W, at every iteration we are at the minimum of $\overline{U}(\{\pi\}, W^k)$ for the chosen W^k of that iteration. However that does not guarantee that we will be gradually minimizing the global objective $\overline{\Psi}(W) := \min_{\{\pi\}} \overline{U}(\{\pi\}, W)$. The *Ascent Flow* condition tries to guarantee that.

Due to definition of $\overline{\Psi}$, we have,

$$\overline{U}(\{\overline{\Pi}(W + \epsilon V)\}, W) \quad \geq \quad \overline{\Psi}(W)$$

Again,

$$\overline{\Psi}(W + \epsilon V) \quad = \quad \overline{U}(\{\overline{\Pi}(W + \epsilon V)\}, W) \quad + \quad \epsilon \sum_{\{ij\} \in \mathcal{P}^N} V_{ij} \Omega_{ij}(\overline{\Pi}_i(W + \epsilon V), \overline{\Pi}_j(W + \epsilon V))$$

Thus, combining the above two,

$$\overline{\Psi}(W + \epsilon V) - \overline{\Psi}(W) \quad \geq \quad \epsilon \sum_{\{ij\} \in \mathcal{P}^N} V_{ij} \Omega_{ij}(\overline{\Pi}_i(W + \epsilon V), \overline{\Pi}_j(W + \epsilon V))$$

Thus we have,

$$\epsilon \sum_{\{ij\} \in \mathcal{P}^N} V_{ij} \Omega_{ij}(\overline{\Pi}_i(W + \epsilon V), \overline{\Pi}_j(W + \epsilon V)) \ > \ 0 \quad \Longleftrightarrow \quad \overline{\Psi}(W + \epsilon V) \ > \ \overline{\Psi}(W) \tag{C.2.1}$$

The last result implies that if V^k, along with ϵ^k is an *Ascent Flow*, then in every iteration we take a step such that $\overline{\Psi}$ increases.

Again, since $\overline{U}(\{\pi\}, W)$ is linear in W, from [10] we know that $\overline{\Psi}(W) := \min_{\{\pi\}} \overline{U}(\{\pi\}, W)$ is concave in W. Moreover we do not have any constraints on the weights, W. Thus $\overline{\Psi}$ can have only an unique bounded optimum, which is a maximum. Hence taking steps towards the maximum means that we will eventually reach the feasible point, if it exists (within errors defined by the *Step Size*). ∎

C.3 Proof of Theorem 6.4.5

Statement of the Theorem: *If the functions c_r and Ω_{ir} $\forall \{ir\} \in \mathcal{P}_r^N$ abide by the Problem Assumptions, we can find an Ascent Flow and a Separable Optimal Flow for the ${r_k}^{th}$ partition at W^k, if it exists, along with a small enough Step Size, ϵ^k, at Lines 5 and 6 of the Algorithm 6.3.1.*

This can be achieved using only the following quantities that are readily known or computable:

$$W^k, \quad \pi_i^k \equiv \overline{\Pi}_i(W^k), \quad \forall i \in \mathcal{N}^N \quad \textit{(known from previous iteration)},$$

$$c_i^{(2)}(\pi_i^k), \quad \forall i \in \mathcal{N}^N,$$

$$\Omega_{ij}^{(0,2)}(\pi_i^k, \pi_j^k), \quad \Omega_{ij}^{(1,1)}(\pi_i^k, \pi_j^k) \quad \textit{and} \quad \Omega_{ij}^{(1,0)}(\pi_i^k, \pi_j^k), \quad \forall \{ij\} \in \mathcal{P}^N,$$

In general we get to choose from a large set of possible Separable Optimal Flows and a large set of possible Ascent Flows, thus giving us the opportunity to find a common "flow" (V^k and ϵ^k) from the two sets.

Proof. Since we are concentrating in computations in k^{th} iteration only, for convenience we will drop the superscripts and write $W \equiv W^k$ for the weights in the previous iteration, and thus due to Theorem C.1, $\overline{\Pi}_i(W) \equiv \pi_i^k$ will be the trajectories in the previous iteration.

From the definitions of U_r, Π_r and Ψ_r, by optimality condition, $U_r^{(1,0,0)}(\Pi_r(W_1, W_2), W_1, W_2) = 0$,

$$\Rightarrow \quad c_r^{(1)}(\Pi_r(W_1, W_2)) \; + \sum_{\{lr\} \in \mathcal{P}_r^N} W_{1,lr} \Omega_{lr}^{(0,1)}(\overline{\Pi}_l(W_2), \Pi_r(W_1, W_2)) \;=\; 0 \tag{C.3.1}$$

Differenciating (C.3.1) w.r.t. W_1, and using *Problem Assumption* 4.ii.,

$$c_r^{(2)}(\Pi_r(W_1, W_2)) \cdot \left[\Pi_r^{(1,0)}(W_1, W_2)\right]_{ij} + \sum_{\{lr\} \in \mathcal{P}_r^N} W_{1,lr} \Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W_2), \Pi_r(W_1, W_2)) \cdot \left[\Pi_r^{(1,0)}(W_1, W_2)\right]_{ij}$$

$$= \begin{cases} \Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W_2), \Pi_r(W_1, W_2)), & \text{when } \{i,j\} \equiv \{q,r\} \in \mathcal{P}_r^N \\ 0, & \text{when } i \neq r, j \neq r \end{cases}$$

Finally, setting $W_1 = W_2 = W$, and using (6.4.4) & (6.4.5),

$$\left(c_r^{(2)}(\overline{\Pi}_r(W)) + \sum_{\{lr\} \in \mathcal{P}_r^N} W_{lr} \Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W), \overline{\Pi}_r(W))\right) \cdot \left[\Pi_r^{(1,0)}(W, W)\right]_{ij} = \begin{cases} \Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W), \overline{\Pi}_r(W)), & \text{when } \{i,j\} \equiv \{q,r\} \in \mathcal{P}_r^N \\ 0, & \text{when } i \neq r, j \neq r \end{cases}$$

$$\Longrightarrow \quad \mathbf{M}_r(W) \cdot \left[\Pi_r^{(1,0)}(W, W)\right]_{ij} \;=\; \begin{cases} \Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W), \overline{\Pi}_r(W)), & \text{when } \{i,j\} \equiv \{q,r\} \in \mathcal{P}_r^N \\ 0, & \text{when } i \neq r, j \neq r \end{cases} \tag{C.3.2}$$

Similarly, differentiating (C.3.1) w.r.t. W_2,

$$\begin{aligned} & c_r^{(2)}(\Pi_r(W_1, W_2)) \cdot \left[\Pi_r^{(0,1)}(W_1, W_2)\right]_{ij} \\ & \quad + \quad \sum_{\{lr\} \in \mathcal{P}_r^N} W_{1,lr} \Big(\Omega_{lr}^{(1,1)}(\overline{\Pi}_l(W_2), \Pi_r(W_1, W_2)) \cdot \left[\overline{\Pi}_l^{(1)}(W_2)\right]_{ij} \\ & \qquad\qquad + \quad \Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W_2), \Pi_r(W_1, W_2)) \cdot \left[\Pi_r^{(0,1)}(W_1, W_2)\right]_{ij} \Big) \\ & \quad = \quad 0 \end{aligned}$$

followed by setting $W_1 = W_2 = W$, and using (6.4.4) & (6.4.5),

$$\begin{aligned}
&c_r^{(2)}(\overline{\Pi}_r(W)) \cdot \Big[\Pi_r^{(0,1)}(W,W)\Big]_{ij} + \\
&\quad \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr} \Big(\Omega_{lr}^{(1,1)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W)) \cdot \Big[\overline{\Pi}_l^{(1)}(W)\Big]_{ij} \\
&\qquad + \Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W)) \cdot \Big[\Pi_r^{(0,1)}(W,W)\Big]_{ij}\Big) \quad = \quad 0
\end{aligned}$$

$$\begin{aligned}
\Longrightarrow \quad &\Big(c_r^{(2)}(\overline{\Pi}_r(W)) \; + \; \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W))\Big)\,\mathbf{M}_r(W) \cdot \Big[\Pi_r^{(0,1)}(W,W)\Big]_{ij} \\
&\qquad + \; \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\Omega_{lr}^{(1,1)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W)) \cdot \Big[\overline{\Pi}_l^{(1)}(W)\Big]_{ij} \\
&= 0
\end{aligned} \tag{C.3.3}$$

Adding (C.3.3) and (C.3.2), and using (6.4.5),

$$\begin{aligned}
&\Big(c_r^{(2)}(\overline{\Pi}_r(W)) + \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\Omega_{lr}^{(0,2)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W))\Big) \cdot \Big[\overline{\Pi}_r^{(1)}(W)\Big]_{ij} \\
&\qquad + \; \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\Omega_{lr}^{(1,1)}(\overline{\Pi}_l(W),\overline{\Pi}_r(W)) \cdot \Big[\overline{\Pi}_l^{(1)}(W)\Big]_{ij} \\
&= \begin{cases} \Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\overline{\Pi}_r(W)), & \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N \\ 0, & \text{when } i\neq r, j\neq r \end{cases}
\end{aligned}$$

$$\begin{aligned}
\Rightarrow \quad &\mathbf{M}_r(W) \cdot \Big[\overline{\Pi}_r^{(1)}(W)\Big]_{ij} + \textstyle\sum_{\{lr\}\in\mathcal{P}_r^N} W_{lr}\mathbf{N}_{lr}(W) \cdot \Big[\overline{\Pi}_l^{(1)}(W)\Big]_{ij} \\
&= \begin{cases} \Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\overline{\Pi}_r(W)), & \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N \\ 0, & \text{when } i\neq r, j\neq r \end{cases}
\end{aligned} \tag{C.3.4}$$

Equation (C.3.4) can be written as follows,

$$\begin{bmatrix} \mathbf{M}_1(W) & W_{12}\mathbf{N}_{12}(W) & W_{13}\mathbf{N}_{13}(W) & \cdots \\ W_{21}\mathbf{N}_{21}(W) & \mathbf{M}_2(W) & W_{23}\mathbf{N}_{23}(W) & \cdots \\ W_{31}\mathbf{N}_{31}(W) & W_{32}\mathbf{N}_{32}(W) & \mathbf{M}_3(W) & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{bmatrix} \Big[\overline{\Pi}_1^{(1)}(W)\Big]_{mn} \\ \Big[\overline{\Pi}_2^{(1)}(W)\Big]_{mn} \\ \Big[\overline{\Pi}_3^{(1)}(W)\Big]_{mn} \\ \vdots \end{bmatrix} = \begin{array}{[c]l} \cdots & \\ 0 & \\ \Omega_{mn}^{(1,0)}(\overline{\Pi}_m(W),\overline{\Pi}_n(W)) & \to m^{th} \text{ row} \\ 0 & \\ \cdots & \\ 0 & \\ \Omega_{mn}^{(1,0)}(\overline{\Pi}_n(W),\overline{\Pi}_m(W)) & \to n^{th} \text{ row} \\ 0 & \\ \cdots & \end{array} \tag{C.3.5}$$

For all $\{m,n\} \in \mathcal{P}^N$.

Note: We emphasize that equations (C.3.2) and (C.3.5) gives prescriptions for computing

a. the derivatives of Π_r, and,

b. the derivatives of $\overline{\Pi}_r$

in terms of

i. $\overline{\Pi}_r$,

ii. the derivatives of c_r, and,

iii. the derivatives of Ω_{ij}.

It is important to note that while the later (**i.**, **ii.** and **iii.**) are readily available or computable, the former (**a.** and **b.**) aren't.

Now we will compute $\Psi_r^{(1,1)}$ using what we obtained from equations (C.3.2) and (C.3.5).

Differentiating (6.4.2) w.r.t. W_1 and using the optimality condition that $U_r^{(1,0,0)}(\Pi_r(W_1,W_2),W_1,W_2)=0$,

$$\begin{aligned}\Psi_r^{(1,0)}(W_1,W_2) &= U_r^{(1,0,0)}(\Pi_r(W_1,W_2),W_1,W_2)\cdot\Pi_r^{(1,0)}(W_1,W_2) \;+\; U_r^{(0,1,0)}(\Pi_r(W_1,W_2),W_1,W_2)\\ &= U_r^{(0,1,0)}(\Pi_r(W_1,W_2),W_1,W_2)\end{aligned}$$

Thus,

$$\left[\Psi_r^{(1,0)}(W_1,W_2)\right]_{ij} = \begin{cases}\Omega_{qr}(\overline{\Pi}_k(W_2),\Pi_r(W_1,W_2)), & \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N\\ 0, & \text{when } i\neq r, j\neq r\end{cases} \tag{C.3.6}$$

Differentiating (C.3.6) w.r.t. W_2,

$$\left[\Psi_r^{(1,1)}(W_1,W_2)\right]_{ij,mn} = \begin{cases}\Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W_2),\Pi_r(W_1,W_2))\cdot\left[\overline{\Pi}_k^{(1)}(W_2)\right]_{mn} + \\ \quad \Omega_{qr}^{(0,1)}(\overline{\Pi}_k(W_2),\Pi_r(W_1,W_2))\cdot\left[\Pi_r^{(0,1)}(W_1,W_2)\right]_{mn} \\ \qquad\qquad \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N \\ \\ 0, \qquad \text{when } i\neq r, j\neq r\end{cases} \tag{C.3.7}$$

From (C.3.7), using *Problem Assumptions* 4.ii., then setting $W_1=W_2=W$, using (6.4.4) & (6.4.5), we get,

$$\left[\Psi_r^{(1,1)}(W,W)\right]_{ij,mn} = \begin{cases}\Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\Pi_r(W,W))\cdot\left[\overline{\Pi}_k^{(1)}(W)\right]_{mn} \\ \quad -\,\Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\Pi_r(W,W))\cdot\left[\Pi_r^{(0,1)}(W,W)\right]_{mn}, \\ \qquad\qquad \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N \\ 0, \quad \text{when } i\neq r, j\neq r\end{cases}$$

$$= \begin{cases}\Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\overline{\Pi}_r(W))\cdot\left[\overline{\Pi}_k^{(1)}(W)-\overline{\Pi}_r^{(1)}(W)\right]_{mn} \\ \quad +\,\Omega_{qr}^{(1,0)}(\overline{\Pi}_k(W),\overline{\Pi}_r(W))\cdot\left[\Pi_r^{(1,0)}(W,W)\right]_{mn}, \\ \qquad\qquad \text{when } \{i,j\}\equiv\{q,r\}\in\mathcal{P}_r^N \\ 0, \quad \text{when } i\neq r, j\neq r\end{cases} \tag{C.3.8}$$

Now using (C.3.2) and (C.3.5), we can compute $\left[\Psi_r^{(1,1)}(W,W)\right]_{ij,mn}$ in terms of the quantities mentioned in the statement of the theorem. It is important to note that the sole purpose of this treatment was to be able to express the derivatives of Π and $\overline{\Pi}$ in terms of computable quantities.

Computing Separable Optimal Flow Direction (Definition 6.4.1): Once we obtain the complete matrix for $\Psi_r^{(1,1)}(W,W)$, we can choose a *Separable Optimal Flow Direction* as a linear combination of the right eigenvectors corresponding to the non-negative eigenvalues of $\Psi_r^{(1,1)}(W,W)$, as it is clear from the first order approximation described below. The step size, ϵ, is chosen according to a desired accuracy. Thus, if we choose $V=a_1\mathbf{e}_1+a_2\mathbf{e}_2+\cdots$, where $\mathbf{e}_1,\mathbf{e}_2,\cdots$ are the eigenvectors corresponding to non-negative eigenvalues $\lambda_1, \lambda_2, \cdots$, we get,

$$\begin{aligned}
&\Psi_r(W+\epsilon V, W+\epsilon V) - \Psi_r(W, W+\epsilon V) \\
&\qquad - \Psi_r(W+\epsilon V, W) + \Psi_r(W, W) \\
&\Rightarrow (\epsilon V)^T \left[\Psi_r^{(1,1)}(W,W)\right](\epsilon V) \geq 0 \\
&\simeq \; (\epsilon V)^T \left[\Psi_r^{(1,1)}(W,W)\right](\epsilon V) \\
&= \; \epsilon^2 \;\; (a_1\mathbf{e}_1 + a_2\mathbf{e}_2 + \cdots)^T (a_1\lambda_1\mathbf{e}_1 + a_2\lambda_2\mathbf{e}_2 + \cdots) \\
&= \; \epsilon^2 \;\; (\lambda_1 a_1^2 + \lambda_2 a_2^2 + \cdots) \;\; \geq \;\; 0
\end{aligned} \tag{C.3.9}$$

We note that all the V_{ij} for which $i \neq r, j \neq r$ does not influnce the condition (C.3.9) since $\left[\Psi_r^{(1,1)}(W,W)\right]_{ij,mn} = 0$ when $i \neq r, j \neq r$. Thus we can always choose the V such that $V_{ij} = 0,\ i \neq r, j \neq r$. Also, we note that it is even possible to choose the negative eigenvalues provided the final combination given in (C.3.9) is non-negative. Thus, within the limits of the error introduced by the finite step-size ϵ (which can be either positive or negative), such a choice of V is a *Separable Optimal Flow Direction* (if one exists). It is easy to see that if no positive real eigenvalue exists, then there does not exist a *Separable Optimal Flow* at W for that particular Ψ_r. Under such circumstances we can choose to skip to the next element in the sequence $\{r_k\}$ and retry.

Computing Ascent Direction (Definition 6.4.3): Computing an Ascent Direction is much simpler and wouldn't require any of the described computations. The simple assumption of small ϵ makes $W + \epsilon V \simeq W$. Then the approximate condition becomes

$$\epsilon \sum_{\{ij\}\in\mathcal{P}^N} V_{ij}\Omega_{ij}(\overline{\Pi}_i(W), \overline{\Pi}_j(W)) \quad > \quad 0$$

Thus, after choosing a *Separable Optimal Flow Direction*, all we need to do is choose the sign of the step, ϵ, correctly so that the above condition is satisfied. If the quantity on the left becomes zero, we choose a different vector from the set of possible Separable Optimal Flow Direction. Thus finally, V, along with ϵ, becomes both a Separable Optimal Flow as well as an Ascent Flow. ■

Bibliography

[1] Ali Ahmadzadeh, Gilad Buchman, Peng Cheng, Ali Jadbabaie, Jim Keller, Vijay Kumar, and George Pappas. Cooperative control of uavs for search and coverage. *Proceedings of the AUVSI Conference on Unmanned Systems*, 2006.

[2] Ali Ahmadzadeh, James Keller, George J. Pappas, Ali Jadbabaie, and Vijay Kumar. Critical cooperative surveillance and coverage with unmanned aerial vehicles. In Vijay Kumar Oussamma Khatib and Daniela Rus, editors, *International Symposium on Experimental Robotics*, STAR, Rio de Janeiro, July 2006. Springer-Verlag.

[3] William E. Baylis. *Clifford (Geometric) Algebras With Applications in Physics, Mathematics, and Engineering*. Birkhuser Boston, 1 edition, 1996.

[4] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*. Athena Scientific, 2007.

[5] Dimitri P. Bertsekas and John N. Tsitsiklis. *Parallel and Distributed Computation:Numerical Methods*. Prentice Hall, 1989.

[6] Harry Blum. A Transformation for Extracting New Descriptors of Shape. In Weiant W. Dunn, editor, *Models for the Perception of Speech and Visual Form*, pages 362–380. MIT Press, Cambridge, 1967.

[7] J.A. Bondy and U.S.R. Murty. *Graph theory*. Graduate texts in mathematics. Springer, 2007.

[8] R. Bott and L.W. Tu. *Differential forms in algebraic topology*. Graduate texts in mathematics. Springer-Verlag, 1982.

[9] Frederic Bourgault, Alexei A. Makarenko, Stefan B. Williams, Ben Grocholsky, and Hugh F. Durrant-Whyte. Information based adaptive robotic exploration. In *in Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS*, pages 540–545, 2002.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex Optimization*. Cambridge University Press, 2009.

[11] R.L. Boylestad and L. Nashelsky. *Electronic devices and circuit theory:*. Prentice Hall, 1996.

[12] F. Bullo, J. Cortés, and S. Martínez. *Distributed Control of Robotic Networks: A Mathematical Approach to Motion Coordination Algorithms*. Applied Mathematics Series. Princeton University Press, 2009.

[13] H.M. Choset. *Principles of robot motion: theory, algorithms, and implementation.* Intelligent robotics and autonomous agents. MIT Press, 2005.

[14] Benjamin Cohen, Sachin Chitta, and Maxim Likhachev. Search-based planning for manipulation with motion primitives. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2010.

[15] David C Conner, Alfred Rizzi, and Howie Choset. Composition of local potential functions for global robot control and navigation. In *Proc. Int'l Conf. on Intelligent Robots and Systems (IROS)*, pages 3546–3551, 2003.

[16] J. Cortes, S. Martinez, T. Karatas, and F. Bullo. Coverage control for mobile sensing networks. *IEEE Trans. Robot. Autom.*, 20(2):243–255, April 2004.

[17] Jorge Cortez, S. Martinez, and Francesco Bullo. Spatially-distributed coverage optimization and control with limited-range interactions. *ESIAM: Control, Optimisation and Calculus of Variations*, 11:691–719, 2005.

[18] Jorge Cortez, Sonia Martinez, Timur Karatas, and Francesco Bullo. Coverage control for mobile sensing networks. *IEEE Trans. Robot. and Automat.*, 20(2):243–255, 2004.

[19] H. S. M. Coxeter. *Introduction to Geometry.* Wiley, New York, 2nd edition, 1969.

[20] B. d'Andrea Novel, G. Campion, and G. Bastin. Control of nonholonomic wheeled mobile robots by state feedback linearization. *The International Journal of Robotics Research*, 14(6), Sept. 1995.

[21] K. Daniel, A. Nash, S. Koenig, and A. Felner. Theta*: Any-Angle Path Planning on Grids. *Journal of Artificial Intelligence Research*, 39:533–579, 2010.

[22] Douglas Demyen and Michael Buro. Efficient triangulation-based pathfinding. In *AAAI'06: Proceedings of the 21st national conference on Artificial intelligence*, pages 942–947. AAAI Press, 2006.

[23] J. Desai, J. Ostrowski, and V. Kumar. Modeling and control of formations of nonholonomic mobile robots. *IEEE Transactions on Robotics and Automation*, 17(6):905–908, December 2001.

[24] Jaydev P. Desai and Vijay Kumar. Motion planning for cooperating mobile manipulators. *Journal of Robotic Systems*, 16:557–579, 1999.

[25] M Bernardine Dias, Robert Michael Zlot, Nidhi Kalra, and Anthony (Tony) Stentz. Market-based multirobot coordination: a survey and analysis. *Proc. of the IEEE*, 94(7):1257 – 1270, 2006.

[26] Edsger W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271, 1959.

[27] A. Dold. *Lectures on algebraic topology.* Classics in mathematics. Springer, 2nd edition, 1995.

[28] David Ferguson, Christopher R. Baker , Maxim Likhachev, and John M Dolan. A reasoning framework for autonomous urban driving. In *Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2008)*, pages 775–780, Eindhoven, Netherlands, June 2008.

[29] Jonathan Fink, M. Ani Hsieh, and Vijay Kumar. Multi-robot manipulation via caging in environments with obstacles. In *IEEE International Conference on Robotics and Automation (ICRA)*, Pasedena, CA, May 2008.

[30] Harley Flanders. *Differential Forms with Applications to the Physical Sciences.* Dover Publications, New York, 1989.

[31] S.V. Fomin and R.A. Silverman. *Calculus of variations.* Dover Books on Mathematics. Dover Publications, 2000.

[32] Mark Galassi, Jim Davies, James Theiler, Brian Gough, Gerard Jungman, Michael Booth, and Fabrice Rossi. *Gnu Scientific Library: Reference Manual.* Network Theory Ltd., February 2003.

[33] Brian Gerkey and Maja Mataric. A formal analysis and taxonomy of task allocation in multi-robot systems. *Int'l. J. of Robotics Research*, 23(9):939–954, 2004.

[34] Robert Ghrist. *Elementary Applied Topology.* From http://www.math.upenn.edu/∼ghrist/notes.html.

[35] Jeremy J. Gray. The hilbert challenge. 2000.

[36] David J. Griffiths. *Introduction to Electrodynamics (3rd Edition).* Benjamin Cummings, 1998.

[37] D. Grigoriev and A. Slissenko. Polytime algorithm for the shortest path in a homotopy class amidst semi-algebraic obstacles in the plane. In *ISSAC '98: Proceedings of the 1998 international symposium on Symbolic and algebraic computation*, pages 17–24, New York, NY, USA, 1998. ACM.

[38] Eric A. Hansen and Rong Zhou. Anytime heuristic search. *Journal of Artificial Intelligence Research (JAIR)*, 28:267–297, 2007.

[39] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems, Science, and Cybernetics*, SSC-4(2):100–107, 1968.

[40] Allen Hatcher. *Algebraic Topology.* Cambridge University Press, 2001.

[41] John Hershberger and Jack Snoeyink. Computing minimum length paths of a given homotopy class. *Comput. Geom. Theory Appl*, 4:331–342, 1991.

[42] Dennis W. Hong, Shawn Kimmel, Rett Boehling, Nina Camoriano, Wes Cardwell, Greg Jannaman, Alex Purcell, Dan Ross, and Eric Russel. Development of a Semi-Autonomous Vehicle Operable by the Visually Impaired. pages 455–467. 2009.

[43] J.X. Hong. *Isometric embedding of Riemannian manifolds in Euclidean spaces.* Mathematical surveys and monographs. American Mathematical Society, 2006.

[44] M. A. Hsieh and V. Kumar. Pattern generation with multiple robots. In *IEEE International Conference on Robotics and Automation*, Orlando, FL, May 2006.

[45] M. A. Hsieh, S. Loizou, and V. Kumar. Stabilization of multiple robots on stable orbits via local sensing. In *IEEE International Conference on Robotics and Automation*, Rome, Italy, May 2007.

[46] Anil K. Jain. *Fundamentals of digital image processing.* Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[47] J. Jost. *Compact Riemann Surfaces.* Springer-Verlag, 1997.

[48] J. Jost. *Riemannian Geometry and Geometric Analysis.* Springer, 2008.

[49] Nejat Karabakal and James C. Bean. A multiplier adjustment method for multiple shortest path problem. Technical report, The University of Michigan, June 1995.

[50] Lydia E. Kavraki, Petr Svestka, Lydia E. Kavraki Petr Vestka, Jean claude Latombe, and Mark H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12:566–580, 1996.

[51] Pradeep Khosla and Richard Volpe. Superquadric artificial potentials for obstacle avoidance and approacb. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, Philadelphia, Apr 1988.

[52] J. O. Kim and P. K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *Robotics and Automation, IEEE Transactions on*, 8(3):338–349, Jun 1992.

[53] A.G. Kovalev. Vector bundles. 2007. From http://www.dpmms.cam.ac.uk/∼agk22/.

[54] Andrea L'Afflitto and Cornel Sultan. Calculus of variations for guaranteed optimal path planning of aircraft formations. pages 1972–1977, May 2010.

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

[56] Steven M. Lavalle. Rapidly-exploring random trees: A new tool for path planning, 1998.

[57] Maxim Likhachev and Anthony Stentz. R* search. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2008.

[58] Stuart P. Lloyd. Least squares quantization in PCM. *IEEE Trans. Inf. Theory*, 28:129–137, 1982.

[59] Savvas G. Loizou and Kostas J. Kyriakopoulos. Closed loop navigation for multiple holonomic vehicles. In *Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, pages 2861–2866, 2002.

[60] Gabriel A. D. Lopes and D. E. Koditschek. Navigation functions for dynamical, nonholonomically constrained mechanical systems. *Springer Advances in Robot Control*, 2006.

[61] J. P. May. *A Concise Course in Algebraic Topology.* The University of Chicago Press, 1999.

[62] Michael Montemerlo, Jan Becker, Suhrid Bhat, Hendrik Dahlkamp, Dmitri Dolgov, Scott Ettinger, Dirk Haehnel, Tim Hilden, Gabe Hoffmann, Burkhard Huhnke, Doug Johnston, Stefan Klumpp, Dirk Langer, Anthony Levandowski, Jesse Levinson, Julien Marcil, David Orenstein, Johannes Paefgen, Isaac Penny, Anna Petrovskaya, Mike Pflueger, Ganymed Stanek, David Stavens, Antone Vogt, and Sebastian Thrun. Junior: The stanford entry in the urban challenge. *J. Field Robot.*, 25(9):569–597, September 2008.

[63] James Munkres. *Topology.* Prentice Hall, 1999.

[64] Diego A. Murio. *The Mollification Method and the Numerical Solution of Ill-Posed Problems.* Wiley-Interscience, 1993.

[65] J. F. Nash. The imbedding problem for riemannian manifolds. *Annals of Mathematics*, 63(1):20–63, 1956.

[66] I. Necoara and J.A.K. Suykens. Interior-point lagrangian decomposition method for separable convex optimization. *Journal of Optimization Theory and Applications*, 143:567–588, 2009.

[67] L. C. A. Pimenta, V. Kumar, R. C. Mesquita, and G. A. S. Pereira. Sensing and coverage for a network of heterogeneous robots. In *Proc. of the IEEE Conf. on Decision and Control*, pages 3947–3952, Cancun, Mexico, December 2008.

[68] Arlan Ramsay and Robert D. Richtmyer. *Introduction to Hyperbolic Geometry.* Springer, 1995.

[69] E. Rimon and D.E. Koditschek. The construction of analytic diffeomorphisms for exact robot navigation on star worlds. *Trans. of the American Mathematical Society*, 327(1), Sept. 1991.

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

[71] Joseph J. Rotman. *An Introduction to Algebraic Topology.* Springer, 1988.

[72] Walter Rudin. *Real and complex analysis, 3rd ed.* McGraw-Hill, Inc., New York, NY, USA, 1987.

[73] Sikandar Samar, Stephen Boyd, and Dimitry Gorinevsky. Distributed estimation via dual decomposition. 2007.

[74] Conrad Sanderson. Armadillo: An open source c++ linear algebra library for fast prototyping and computationally intensive experiments. Technical report, NICTA, 2010.

[75] E. Schmitzberger, J.L. Bouchet, M. Dufaut, D. Wolf, and R. Husson. Capture of homotopy classes with probabilistic road map. In *International Conference on Intelligent Robots and Systems*, volume 3, pages 2317–2322, 2002.

[76] M. Schwager, J. McLurkin, and D. Rus. Distributed coverage control with sensory feedback for networked robots. In *Proc. of Robot.: Sci. and Syst.*, Philadelphia, PA, August 2006.

[77] M. Schwager, J. E. Slotine, and D. Rus. Decentralized, adaptive control for coverage with networked robots. In *Proc. of the IEEE Intl. Conf. on Robot. and Autom.*, pages 3289–3294, Rome, Italy, April 2007.

[78] H. Seifert, W. Threlfall, J.S. Birman, and J. Eisner. *Seifert and Threlfall, A textbook of topology.* Pure and applied mathematics. Academic Press, 1980.

[79] C. Stachniss. *Exploration and Mapping with Mobile Robots.* PhD thesis, University of Freiburg, Freiburg, Germany, April 2006.

[80] C. Stachniss, G. Grisetti, and W. Burgard. Information gain-based exploration using rao-blackwellized particle filters. In *Proc. of Robot.: Sci. and Syst.*, pages 65–72, Cambridge, MA, June 2005.

[81] A. Stentz. The focussed D* algorithm for real-time replanning. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1652–1659, 1995.

[82] J. Stoer and R. Bulirsch. *Introduction to numerical analysis.* Texts in applied mathematics. Springer, 2002.

[83] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents).* The MIT Press, 2005.

[84] Benjamn Tovar, Fred Cohen, and Steven M. LaValle. Sensor beams, obstacles, and possible paths. In *Workshop on the Algorithmic Foundations of Robotics*, pages 317–332, 2008.

[85] Christopher Urmson, Joshua Anhalt, Hong Bae, J. Andrew (Drew) Bagnell, Christopher R. Baker , Robert E Bittner, Thomas Brown, M. N. Clark, Michael Darms, Daniel Demitrish, John M Dolan, David Duggins, David Ferguson, Tugrul Galatali, Christopher M Geyer, Michele Gittleman, Sam Harbaugh, Martial Hebert, Thomas Howard, Sascha Kolski, Maxim Likhachev, Bakhtiar Litkouhi, Alonzo Kelly, Matthew McNaughton, Nick Miller, Jim Nickolaou, Kevin Peterson, Brian Pilnick, Ragunathan Rajkumar, Paul Rybski, Varsha Sadekar, Bryan Salesky, Young-Woo Seo, Sanjiv Singh, Jarrod M Snider, Joshua C Struble, Anthony (Tony) Stentz, Michael Taylor , William (Red) L. Whittaker, Ziv Wolkowicki, Wende Zhang, and Jason Ziglar. Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics Special Issue on the 2007 DARPA Urban Challenge, Part I*, 25(1):425–466, June 2008.

[86] Eric W. Weisstein. *Cayley-Klein-Hilbert Metric.* MathWorld–A Wolfram Web Resource.

[87] Milos Zefran. *Continuous methods for motion planning.* PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1996. Supervisor-Vijay Kumar.

[88] H. Zhang, V. Kumar, and J. Ostrowski. Motion planning under uncertainty. In *IEEE International Conference on Robotics and Automation*, Leuven, Belgium, May 16-21 1998. IEEE.

[89] Yan Zhou, Bo Hu, and Jianqiu Zhang. Occlusion detection and tracking method based on bayesian decision theory. In Long-Wen Chang and Wen-Nung Lie, editors, *Advances in Image and Video Technology*, volume 4319 of *Lecture Notes in Computer Science*, pages 474–482. Springer Berlin / Heidelberg, 2006.