ALMA MATER STUDIORUM · UNIVERSITÀ DI BOLOGNA

SCUOLA DI INGEGNERIA E ARCHITETTURA

Corso di Laurea in Ingegneria dell'Automazione
Curriculum Automation Engineering

Potential Fields Implementation In Path Planning For Autonomous Vehicles

Relatore: Studente:
Prof.MASSIMO FERRI LUCA DOMENICHELLI

Anno accademico 2019/2020

Ever Tried.

Ever Failed.

No Matter.

Try Again.

Fail Again.

Fail Better.

(Samuel Beckett)

Abstract

Questa tesi vuole affrontare i campi da potenziale nel contesto della pianificazione del percorso applicata a macchine automatizzate. Apre indulgendo sulla globalità dello argomento: affronta definizioni specifiche quali ad esempio quella di ambiente applicata al contesto specifico. Procede menzionando in maniera chiara e concisa diversi approcci alla pianificazione del percorso impiegati nella pratica. Si concentra dunque sull'argomento cardine: i campi da potenziale. Intraprende uno studio approfondito sulla relazione fra i due argomenti, introducendo la nozione di campi da potenziale artificiali. Questi ultimi vengono dissezionati allo scopo di essere compresi appieno, sia nel funzionamento che nella ideazione. Conclude esplorando le problematiche principali legate allo utilizzo dei campi da potenziale nella pratica di pianificazione del percorso, offrendo tuttavia possibili soluzioni mirate ad aggirare queste complicanze.

Acknowledgments

I would like to express my deepest appreciation to the esteemed professor Massimo Ferri, for his invaluable insight on the topic of this dissertation and his guidance in its drafting.

I will never thank my parents enough for everything they have done and will continue to do for me

Special thanks to my beloved sister, for being a better life partner than i could ever have wished for

I cannot begin to thank my sworn brother Raoul Andriulli enough, for his unparalleled contribution to the success of my journey

I'm deeply indebted to my sworn brothers Mattia Barigelletti e Matteo Stupazzini, for the peerless assistance and friendship they have shown me during this project, as well as in all other aspects of my life

My everlasting gratitude goes to my precious friend Viola Magnani, without whom none of this would not have been possible

I'm eternally grateful to my treasured friend Altea Magnani, who never left my side and, in doing so, helped me grow and better myself

I must also extend my sincere thanks to my cherished friend Francesca Benedettini, for her unwavering support and constant encouragement throughout my study

Contents

Introduction

Path planning is a contemporary field of study that finds several applications in computational geometry, computer animation, robotics and computer games. In layman's terms, it is the collection of all those methods through which any form of artificial intelligence can navigate its surrounding to reach a prefixed destination. When it comes to robotics, path planning aims to minimise the gap in decision making prowess present between automated machines and humans by reenacting the cognitive process we undergo when taking decisions, by means of mathematics and algorithms.

This dissertation will focus on robotic motion planning, and particularly on the use of potential fields. It looks to brief the reader in on the notion of path planning as a whole, and then single-out a distinct technique to provide a more in-depth explanation of its behaviour and performance.

The first chapter provides a broad rundown of what robotic motion planning is. It defines concepts that may be unknown to anyone not involved in the field of artificial intelligence. The second chapter recalls all and only those theoretical knowledges essential to the global understanding of the dissertation. Lastly, the third and final chapter expands on the concepts discussed in the first chapter by addressing a specific approach utilised in actual implementations of path planning: potential fields.

Chapter 1

Path planning: a general overview

Path planning, also known as motion planning, is the task of finding a continuous route which will allow a robot to move from the starting configuration $\mathbf{q}_{start}$ to the goal configuration $\mathbf{q}_{goal}$. The increasing complexity and depth of modern Artificial Intelligences (AIs) is leading robots to outperform manual labor in a variety of on-field applications: the demand for high performing path planning algorithms has never been higher.
Over the course of this chapter we will discover how AIs read the inputs received from their surroundings to plan a safe path to follow, as well as the main struggles they have to overcome to do so. The contents of this chapter are taken from [4], [7], [8] and [9].

1.1 Environment

We call environment the space in which our robot operates. Let us define the environment ξ as the set of all which surrounds the robot, without being part of the robot itself. We call:

- ξ_{obs} the subspace of ξ which the robot cannot access
- ξ_{free} the subspace of ξ devoid of obstacles ($\xi_{free} = \xi - \xi_{obs}$)
- ξ_{danger} the subspace of ξ which the robot can access, but shouldn't

Now that we have understood these concepts, we will define several aspects which a robot analyzes to distinguish between AI environments:

- **Observability**: if a robot has access to all information in the environment, at each point of time, we say that the environment is **fully observable**. Otherwise, it's **partially observable**
- **Stasis**: if the environment can change itself while the robot operates in it, we call the environment **dynamic**. Otherwise, we say it is **static**
- **Determinability** if each state of the environment can be determined by the robot configuration in the previous state, said environment is **deterministic**. Otherwise, it is **stochastic**
- **Discreteness**: if the final outcome of the task at hand is driven by a finite set of possibilities, our robot is operating in a **discrete** environment; otherwise the environment is said to be **continuous**. In general, continuous environments rely on unknown and ever-changing inputs
- **Agency**: depending on whether or not our robot operates alone in the environment, we distinguish between **single-agent** and **multi-agent** ones
- **Seriality**: if the current percept alone is sufficient to determine the next best action, the environment is **episodic**. Otherwise, we say it's **sequential**

1.2 Approaches

As we previously stated, the raison d'être of path planning is to provide the robot with a safe path to follow in order to reach $\mathbf{q}_{goal}$. To achieve this, several approaches can be followed.

1.2.1 Roadmap

A **roadmap** Γ is a combination of curves such that:

- there is a path from $\mathbf{q}_{start} \in \xi_{free}$ to a generic configuration q'$\in \Gamma$
- there is a path from a generic configuration q" $\in \Gamma$ to $\mathbf{q}_{goal} \in \xi_{free}$
- there is a path from q' to q"

Once a roadmap has been defined, the end result will be a path in ξ_{free} connecting $\mathbf{q}_{start}$ to $\mathbf{q}_{goal}$. We distinguish between two roadmap approaches: **Visibility Graphs** and **Voronoi Diagrams**.

Visibility Graph

Visibility graphs aim to model a graph based on **observable** obstacles lying in ξ. Nodes in the graph represent geometrical vertices of said obstacles, while edges link together nodes sharing a visible connection.Once the graph is completed, Dijkstra's algorithm is employed in order to compute the shortest path.
Visibility Graphs are easier to implement when compared to Voronoi Diagrams, at the cost of being less safe (they yield paths lying in ξ_{danger}).

Voronoi Diagram

Voronoi Diagrams (in this particular conceptual setting) are defined as **the geometric locus of points farthest from the obstacles**.
The first step in developing a Voronoi Diagram is determining the distance between each point in ξ and its nearest obstacle, thus dividing the environment in regions known as **Voronoi Cells**. The borders between a cell and its surrounding cells represent the set of points farthest from the pivotal obstacles of said group of cells. It follows that the borders between all cells in ξ represent the **Voronoi Diagram** for that environment.
The computation of such diagrams is complex, but it ensures a safe path for the robot to follow (i.e., lying in ξ_{free}).

1.2.2 Cell Decomposition

The idea behind this approach is to decompose ξ_{free} in **cells**, regions of space such that:

- it's easy to compute a safe path between two configurations in the **same cell**
- it's easy to compute a safe path between two configurations in **adjacent cells**

Two cells are adjacent if they share a common boundary.
All that remains to be done is to compute a sequence of cells that connects $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$. If more than one are outputted, Dijkstra's algorithm is implemented to choose the shortest path.
It is important to note that the more cells we define, the more the algorithm will be efficient, at the cost of being slower.

Fixed Cell Decomposition

This particular approach decomposes ξ_{free} in fixed size-cells, each identified by a non-univocal number. Starting from $\mathbf{q}_{goal}$, univocally identified by the number "**0**", each adjacent cell is flagged with increasing number values, until all cells have been marked. To determine the shortest path, starting from $\mathbf{q}_{start}$, the robot moves towards $\mathbf{q}_{goal}$ following the lowest-numbered adjacent cell.

1.2.3 Fixed Graph

A graph of all feasible paths is defined a priori. Points $\mathbf{q}_{start}$ and $\mathbf{q}_{goal}$ are **always** made to coincide with vertices of the graph. Through Dijkstra's algorithm the shortest path between the two configurations is computed: it yields the list of points which indicate the shortest piecewise linear curve to follow to reach the destination.

1.2.4 Potential Fields

Potential fields represent yet another approach employed in path planning. Being the core subject matter of this thesis, I will refrain from presenting it right now, as this method will be tackled in a later chapter. Interested readers can consult **Chapter 3**, which will introduce this approach and provide an extensive study on the topic.

1.3 Criteria

In this chapter we have studied how a robot utilizes inputs it receives from its surrounding environment to navigate a path from its starting configuration $\mathbf{q}_{start}$ to its goal configuration $\mathbf{q}_{goal}$. However, we have yet to see how it understands which path is the **optimal** one. The introduction of **criteria** allows it to do so: a set of demands help

the robot narrow the choice down to the most efficient path solutions:

- **length**: the chosen path must be as short as possible
- **time**: the chosen path must be as fast as possible to travel
- **safety**: the chosen path must lie in ξ_{free}
- **shape**: the chosen path must be as straight as possible. Sharp turns should be avoided at all costs.
- **feasibility**: the chosen path must consider motion constraints.

Naturally, a path solution which perfectly fits all the criteria is almost never found. The choice will fall upon the path offering the most **efficient** compromise.

At this point the reader should have a basic understanding of what **path planning** is, what it aims to do, and how it does it. Following chapters will aid him/ her in deepening his/ her understanding of the matter. **Chapter 2** will provide the reader with the **key** tools he/ she will need to fully understand the topic discussed in this thesis, while **Chapter 3** will focus on potential fields, and how they relate to path planning.

Chapter 2

Mathematical Tools

This brief chapter will not provide the reader with any knowledge regarding path planning. It solely serves as a collection of all those tools one would need in order to grasp the concepts illustrated over the course of this dissertation. If the reader is confident in his/ her understanding of the topics covered hereafter, he/ she is welcome to move on to the following chapter. The contents of this chapter are taken from [1] and [13].

2.1 Euclidean Distance

The **euclidean distance** between any two points **p** and **q** is the length of the line segment that connects them.
In **Cartesian coordinates**, the euclidean distance between $\mathbf{p} = (p_1, p_2, ..., p)$ and $\mathbf{q} = (q_1, q_2, ..., q_n)$ which is equal to the euclidean distance between **q** and **p**, is given by the **Pythagorean formula**:

$$d(p, q) = d(q, p) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \tag{2.1}$$

which is equivalent to computing the **euclidean length** of the displacement vector $||q - p||$:

$$||q - p|| = \sqrt{(q - p) \cdot (p - q)} \tag{2.2}$$

2.2 Derivability

Let $f : A \to \mathbb{R}$ be a function, with $A \subset \mathbb{R}^n$, $\overrightarrow{P} \in A$. Furthermore, consider the vectors $\overrightarrow{e}_1 = (1, 0, ..., 0)$, $\overrightarrow{e}_2 = (0, 1, ..., 0)$, ..., $\overrightarrow{e}_n = (0, ..., 0, 1)$: we call **partial derivative** of f with respect to $x_1, x_2, ..., x_n$ in $\overrightarrow{P}$:

$$\frac{\partial f}{\partial x_i}(\overrightarrow{P}) = \lim_{h \to 0} \frac{f(\overrightarrow{P} + h\overrightarrow{e}_i) - f(\overrightarrow{P})}{h} \tag{2.3}$$

Basically, we let one variable vary while the others do not.

2.3 Differentiability

Let $f : A \to \mathbb{R}$ be a function, with $A \subset \mathbb{R}^n$, $\overrightarrow{P} \in A$, f derivable in $\overrightarrow{P}$. We say that f is **differentiable** in $\overrightarrow{P}$ $\forall$ $\overrightarrow{h} \in \mathbb{R}^n$ if, and only if:

$$f(\overrightarrow{P} + \overrightarrow{h}) = f(\overrightarrow{P}) + < \overrightarrow{h}, \overrightarrow{\nabla} f(\overrightarrow{P}) > + o(\overrightarrow{h}) \tag{2.4}$$

2.4 Gradient

Let $f : A \to \mathbb{R}$ be a function, with $A \subset \mathbb{R}^n$, $\overrightarrow{P} \in \mathbb{R}$ and f derivable in $\overrightarrow{P}$. We call **gradient** of f in $\overrightarrow{P}$:

$$\overrightarrow{\nabla} f(\overrightarrow{P}) = (\frac{\partial f}{\partial x_1}(\overrightarrow{P}), \frac{\partial f}{\partial x_2}(\overrightarrow{P}), ..., \frac{\partial f}{\partial x_n}(\overrightarrow{P})) \tag{2.5}$$

Meaning the vector of its partial derivatives.

2.5 Potential

Let $\overrightarrow{F} : A \to \mathbb{R}^n$ be a vector field, with $A \subset \mathbb{R}$. If $U : A \to \mathbb{R}$ exists and is of differentiability class C^2, so that

$$\overrightarrow{\nabla} U = \overrightarrow{F} \tag{2.6}$$

we say that $\overrightarrow{F}$ is conservative, and we call U a **potential**.

2.6 Critical Points

Let $f : A \to R$ be a function, with $A \subset \mathbb{R}^n$. If all **second order derivatives** of f exist, we call **Hessian Matrix** of f:

$$Hess(f) = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & & & \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix} \tag{2.7}$$

Now let $f : A \to \mathbb{R}$ be a function of differentiability class C^1, with $A \subset \mathbb{R}^n$, $\overrightarrow{P} \in A$. If

$$\overrightarrow{\nabla} f(\overrightarrow{P}) = (0, 0, \ldots, 0) \tag{2.8}$$

we say that $\overrightarrow{P}$ is a **critical point**.
Consider the critical point $\overrightarrow{P}_0 \in A$. We have:

- $\overrightarrow{P}_0$ is a **global maximum** for f in A if, and only if, we have that $f(P) \leq f(P_0)\ \forall \overrightarrow{P} \in A$
- $\overrightarrow{P}_0$ is a **global minimum** for f in A if, and only if, we have that $f(P) \geq f(P_0)\ \forall \overrightarrow{P} \in A$
- $\overrightarrow{P}_0$ is a **local maximum** for f if we have that $\exists\ \delta > 0$ so that $f(P) \leq f(P_0)\ \forall \overrightarrow{P} \in B(\overrightarrow{P}_0, \delta)$, with $B(\overrightarrow{P}_0, \delta)$ being a closed ball

- $\overrightarrow{P}_0$ is a **local minimum** for f if we have that $\exists\ \delta > 0$ so that $f(P) \geq f(P_0)\ \forall \overrightarrow{P} \in B(\overrightarrow{P}_0, \delta)$, with $B(\overrightarrow{P}_0, \delta)$ being a closed ball
- $\overrightarrow{P}_0$ is a **saddle** if it is neither a maximum nor a minimum

As can be seen from [13], if $f : A \to \mathbb{R}$ is a function of differentiability class C^2, with $A \subset \mathbb{R}$ and $\overrightarrow{P} \in A$, let $\overrightarrow{P}$ be a critical point and $M = Hess(F(\overrightarrow{P}))$. Then:

- if M is **positive-definite** $\Rrightarrow \overrightarrow{P}$ is a **local minimum**
- if M is **negative-definite** $\Rrightarrow \overrightarrow{P}$ is a **local maximum**
- if M is **indefinite** $\Rrightarrow \overrightarrow{P}$ is a **saddle**

Under the same assumptions:

- if M is **positive semi-definite** $\Rrightarrow \overrightarrow{P}$ is a **local minimum** or a **saddle**
- if M is **negative semi-definite** $\Rrightarrow \overrightarrow{P}$ is a **local maximum** or a **saddle**

A solid understanding of these concepts is advised, as they represent one of the core concepts around which the contents discussed in the following chapter revolve.

2.7 Conic

Let Π^2 be a euclidean plane, in which a cartesian coordinate system $\mathcal{R} = (\mathcal{O}, \overrightarrow{\mathcal{B}})$ is fixed. Consider the following generic algebraic second order equation in x and y:

$$a_{11}x^2 + a_{22}y^2 + 2a_{12}xy + 2a_{01}x + 2a_{02}y + a_{00} = 0 \tag{2.9}$$

with $a_{ij} \in \mathbb{R} \; \forall \; i, j \in \{0, 1, 2\}$. Said equation represents a **conic** in the euclidean plane Π^2, with respect to the coordinate system $\mathcal{R}$. Furthermore, given the following equations:

$$a_{11}x^2 + a_{22}y^2 + 2a_{12}xy + 2a_{01}x + 2a_{02}y + a_{00} = 0 \tag{2.10}$$

$$a'_{11}x^2 + a'_{22}y^2 + 2a'_{12}xy + 2a'_{01}x + 2a'_{02}y + a'_{00} = 0 \tag{2.11}$$

we can declare that they represent the same conic of Π^2 with respect to $\mathcal{R}$ if, and only if $\exists \; \lambda \in \mathbb{R} - \{0\}$ so that $a'_{ij} = \lambda \cdot a_{ij} \; \forall \; i, j \in \{0, 1, 2\}$. It follows that every conic $\mathcal{C}$ of Π^2 is defined, with respect to $\mathcal{R}$, by infinite algebraic second order equations, proportional between each other. Generally, we say that the equation of $\mathcal{C}$ with respect to $\mathcal{R}$ is defined up to a proportionality coefficient.

2.8 Paraboloid

Let Π^3 be a euclidean space, in which a cartesian coordinate system $\mathcal{R} = (\mathcal{O}, \overrightarrow{\mathcal{B}})$ is fixed. Consider the following generic algebraic second order equation in x, y and z:

$$\begin{aligned} &a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz \\ &+ 2a_{01}x + 2a_{02}y + 2a_{03}z + a_{00} = 0 \end{aligned} \tag{2.12}$$

with $a_{ij} \in \mathbb{R} \; \forall \; i, j \in \{0, 1, 2, 3\}$. Said equation represents a **quadric** in the euclidean space Π^3, with respect to the coordinate system $\mathcal{R}$. Furthermore, given the following equations:

$$\begin{aligned} &a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + 2a_{23}yz \\ &+ 2a_{01}x + 2a_{02}y + 2a_{03}z + a_{00} = 0 \end{aligned} \tag{2.13}$$

$$\begin{aligned} &a'_{11}x^2 + a'_{22}y^2 + a'_{33}z^2 + 2a'_{12}xy + 2a'_{13}xz + 2a'_{23}yz \\ &+ 2a'_{01}x + 2a'_{02}y + 2a'_{03}z + a'_{00} = 0 \end{aligned} \tag{2.14}$$

we can declare that they represent the same quadric of Π^3 with respect to $\mathcal{R}$ if, and only if $\exists \; \lambda \in \mathbb{R} - \{0\}$ so that $a'_{ij} = \lambda \cdot a_{ij} \; \forall \; i, j \in$

$\{0, 1, 2, 3\}$.
It follows that every quadric $\mathcal{Q}$ of Π^3 is defined, with respect to $\mathcal{R}$, by infinite algebraic second order equations, proportional between each other. Generally, we say that the equation of $\mathcal{Q}$ with respect to $\mathcal{R}$ is defined up to a proportionality coefficient.
By this point it is obvious to the reader that a quadric is nothing but the analogous of a conic, defined on a space of dimension 3. Given a quadric $\mathcal{Q}$ in Π^3, we call A the **associated matrix** of $\mathcal{Q}$ with respect to $\mathcal{R}$, where

$$A = \begin{pmatrix} a_{00} & a_{01} & a_{02} & a_{03} \\ a_{01} & a_{11} & a_{12} & a_{13} \\ a_{02} & a_{12} & a_{22} & a_{23} \\ a_{03} & a_{13} & a_{23} & a_{33} \end{pmatrix} \in \mathcal{S}_4(\mathbb{R}) \tag{2.15}$$

If $detA(\mathcal{Q}) \neq 0$, $\mathcal{Q}$ is said to be non-degenerate. In which case, depending on its behaviour with respect to the improper plane π_∞, $\mathcal{Q}$ is:

- a **paraboloid** if π_∞ is **tangent** to $\mathcal{Q}$
- an **ellipsoid** if π_∞ is **external** to $\mathcal{Q}$
- a **hyperboloid** if $\pi_\infty$is **secant** $\mathcal{Q}$

Let $\mathcal{Q}$ be a quadric of Π^3, with $A \in \mathcal{S}(\mathbb{R})$ being its associated matrix with respect to $\mathcal{R}$. We have:

- $\mathcal{Q}$ is a paraboloid if and only if $A_{00} = 0$
- $\mathcal{Q}$ is an ellissoid if and only if $A_{00} \neq 0$ and concordant with a_{33}, and its minor $\begin{pmatrix} a_{22} & a_{23} \\ a_{23} & a_{33} \end{pmatrix}$ has positive determinant
- $\mathcal{Q}$ is a hyperboloid if and only if $A_{00} \neq 0$, but it does not simultaneously satisfy the two conditions required to be an ellipsoid

with A_{00} being the algebraic complement of the element a_{00} in the matrix A.

2.9 Algorithms

Algorithms are defined as a finite sequence of instructions to be followed in order to solve a problem or perform a computation. They are widely used in robotics, as they represent the most efficient way of translating a specific set of tasks the robot is required to perform to a language it can understand: mathematics.
This section will explore those algorithms the understanding of which will greatly aid the reader in thoroughly grasping the concepts of this dissertation. Before delving into them, however, it bears importance to illustrate the following concept:

In mathematics, a **graph** is an ordered pair comprised by a set of **vertices** (also called nodes) and a set of **edges** (also called lines) connecting them.

The previous notion should prove helpful in visualizing the inner workings of the algorithms investigated hereafter.

2.9.1 Dijkstra's Shortest Path Algorithm

Dijkstra's algorithm is employed to find the **shortest path** between nodes in a graph. Since it relies on a starting node from which the algorithm spreads, it proves to be a suitable and efficient tool to be applied to robotic motion planning. Dijkstra's algorithm follows the best-first approach, and revolves around the concept that any subpath $B \rightarrow C$ of the shortest path $A \rightarrow C$ is also the shortest path between B and C. In theory, Dijkstra's algorithm is able to find the shortest path between a set starting node and all the other vertices in the graph. However, practical applications often employ it in order to compute the shortest route between two specific vertices, defined a priori: the starting node and the destination node.
The algorithm is as follows:

- define a set of all the visited nodes, and initialize it as empty
- define a set of all the unvisited notes, and fill it with all the nodes in your graph
- initialize the distance of your starting node A as 0
- initialize the distance between every pair of nodes as ∞

then

1. set the unvisited node with the shortest distance as the current node $\mathcal{C}$
2. consider all the unvisited neighbours of $\mathcal{C}$, and compute their distances from the current node
3. if the calculated distance between two nodes is shorter than the previously known distance, update it with the new distance
4. set the current node as visited, and update the new current node as the node with the shortest distance from the previous current node
5. if the current node is the destination node, stop. You have reached your goal and computed the shortest distance to it from your starting node
6. if the current node is not the destination node, and there are are unvisited nodes neighbouring $\mathcal{C}$, go back to step 2

The computational complexity of the algorithm is:

$$\mathcal{O}((|\mathbf{E}| + |\mathbf{V}|)log(|\mathbf{V}|))$$

where

- |**V**| denotes the number of nodes
- |**E**| denotes the number of edges

2.9.2 Gradient Descent Algorithm

This particular algorithm is the backbone supporting the whole of chapter three. It stems from calculus, thanks to which we know that if a function f is differentiable in $\overrightarrow{P}$, its **gradient** points towards the direction of **maximum growth**. Conversely, if one where to follow the direction indicated by the **negative gradient** of f, he would trail the path of **maximum descent**, which is what we want. To understand this parallelism, let's consider the function $f(q)$: since $\overrightarrow{\nabla} f(q)$ points towards the direction of maximum growth, we want to move against it, as we are trying to reach the global minimum of f. We have:

$$q_{n+1} = q_n - \overrightarrow{\nabla} f(q_n) \tag{2.16}$$

Clearly, $f(q_n) \geq f(q_{n+1})$, which means we can repeat this process to reach progressively lower values of $f(q)$. Simply put:

- the robot starts in its initial configuration $\mathbf{q}_{goal}$
- the robot acquires informations about its surroundings in the form of outputs of $f(q)$
- the robot computes $\overrightarrow{\nabla} f(q)$ and moves towards the configuration pointed by $-\overrightarrow{\nabla} f(q)$

It logically follows that, in the absence of local minima, by iterating this set of instructions for a finite amount of times, the robot will descend until it reaches the global minimum of $f(q)$. The presence of local minima does indeed represent a major problem when employing gradient descent: solutions will be discussed in **Chapter 3**.

Chapter 3

Potential Fields

In this final chapter we will explore the main topic of this dissertation: **Potential Fields**, and how they can be exploited to guide a robot to its goal configuration.
The core idea around which this approach revolves is rather straightforward: the robot should be **attracted** towards its **goal configuration** while being **repelled** by **obstacles**. The contents of this chapter are taken from [2], [3], [4], [5], [6], [10], [11] and [12].

3.1 Artificial Potential Fields

In order to gain a clear understanding of the matter at hand, the reader is invited to picture the robot as a marble, positioned in an uneven environment. As any closed system tends to the state of **lowest potential**, the marble will roll until its gravitational potential **U(q)** = **mgh** is lowest: the global minimum of the environment.
Knowing that, we want to create a function which will allow our robot to reproduce the same behaviour in any environment ξ it will operate in. To accomplish this, we first have to define the properties of the function we want to create:

- the **global minimum** of the function has to correspond to $\mathbf{q}_{goal}$
- the value of the function has to **increase** as we move **farther away** from $\mathbf{q}_{goal}$
- the value of the function has to **decrease** as we **approach** $\mathbf{q}_{goal}$

Also, we need to remember the fundamental concept our function must follow: attracting the robot towards $\mathbf{q}_{goal}$ and repelling it from obstacles. With these criteria clear, we can now create a function to describe an **Artificial Potential Field**.
First, we want our function to be the superposition of two smaller functions, which illustrate the crucial aspects of our field:

- the **attractive potential $\mathbf{U_{att}}$ (q)**
- the **repulsive potential $\mathbf{U_{rep}}$ (q)**

Thus, our differentiable potential function $U : \xi_{free} \rightarrow \mathbb{R}$ is described as

$$\mathbf{U(q) = U_{att}(q) + U_{rep}(q)} \tag{3.1}$$

Now that we have modelled the main frame of our function, we are going to delve into each member separately, to better understand their inner nuances.

3.1.1 Attractive Potential

This first member of our equation serves as the guide of our robot: it directs it towards the local minima of our function. In terms of potential, this translates to a well. The function describing a **parabolic well** is:

$$U_{\text{att}}(q) = \frac{1}{2} K \rho^2_{goal}(q) \tag{3.2}$$

where

- $\mathbf{K}$ is a positive scaling factor
- $\rho_{\mathbf{goal}}(\mathbf{q})$ is the euclidean distance $||\mathbf{q} - \mathbf{q}_{goal}||$, which is differentiable everywhere in ξ

We know that every conservative force $\overrightarrow{F}$ equals the negative gradient of its potential U, so we can calculate the attractive force generated by our potential well:

$$\begin{aligned} \overrightarrow{F}_{att}(q) &= -\overrightarrow{\nabla} U_{att}(q) \\ &= -\xi(q - q_{goal}) \end{aligned} \tag{3.3}$$

The problem with the parabolic well attractive potential is that it tends to infinity when ρ_{goal}(q)$\to\infty$, thus generating forces too large to be controlled. A solution to this problem is to consider a different potential well: the **conic well**, defined as

$$U_{\text{att}}(q) = K' \rho_{goal}(q) \tag{3.4}$$

The attractive force of this potential field is:

$$\begin{aligned} \overrightarrow{F}_{att}(q) &= -\overrightarrow{\nabla} U_{att}(q) \\ &= \frac{-K'(q - q_{goal})}{||q - q_{goal}||} \end{aligned} \tag{3.5}$$

For us to benefit from the advantages of both forces without suffering due to their shortcomings, it's best to combine them in a piecewise equation that varies based on the distance from $\mathbf{q}_{goal}$. Having called $\mathbf{q}_p$ the pivotal point beyond which $\overrightarrow{F}_{att}(q)$ generated by the parabolic well potential spirals out of control, forcing us to adopt a conic well potential, our attractive potential field will be defined as:

$$U_{att}(q) = \begin{cases} \frac{1}{2} K \rho^2_{goal}(q) & \text{if } q \leq q_p \\ K' \rho_{goal}(q) & \text{if } q > q_p \end{cases} \tag{3.6}$$

Now that we have found an attractive potential field that suits our needs, we'll move on to the second member of our equation: the repulsive potential.

3.1.2 Repulsive Potential

If the attractive field is tasked with leading the robot to its goal configuration, the repulsive fields guarantees that it will reach its destination safely, without colliding against obstacles. Ideally, the repulsive potential field should not affect our robot when it is sufficiently far away from obstacles. It is possible to define a repulsive potential field able to fulfil both of these conditions:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\zeta(\frac{1}{\rho(q)} - \frac{1}{\rho_0})^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) > \rho_0 \end{cases} \tag{3.7}$$

where:

- $\rho(q)$ is the minimum euclidean distance between $\mathbf{q}$ and ξ_{obs}
- ζ is a positive scaling factor
- ρ_0 is a positive constant called **distance of influence** of the obstacles

As for the attractive potential field, we want to know how effective our repulsive field is: we want to calculate the repulsive force $\overrightarrow{F}_{\mathbf{rep}}(\mathbf{q})$. However, we are met with an issue: in order for $\overrightarrow{F}(q) = -\overrightarrow{\nabla} U(q)$ to stand, the function $\rho(q)$ must be differentiable everywhere in ξ_{free}. That is only true under the assumption that ξ_{obs} is a convex region with a piecewise differentiable boundary. Such a claim is unrealistic. In fact, while $\rho(q)$ remains differentiable almost everywhere in ξ_{free}, there exist $\mathbf{q}$ configurations for which several $\mathbf{q}_\mathrm{c}$ points are closest to ξ_{obs}, in which case $\rho(q)$ is no longer differentiable in ξ_{free}.
One way around this complication is to partition ξ_{obs} into convex components ξ_{obs_k}, with $k = 1, 2, ..., n$, and to associate to each component its respective potential field $\mathbf{U_{rep_k}}$. The total repulsive po-

tential field is simply the superposition of each specific $\mathbf{U_{rep_k}}$ associated to each component ξ_{obs_k}:

$$U_{rep}(q) = \sum_{k=1}^{n} U_{rep_k}(q) \tag{3.8}$$

We can now calculate the repulsive force $\overrightarrow{F}_{\mathbf{rep}}(\mathbf{q})$:

$$\overrightarrow{F}_{rep}(q) = \sum_{k=1}^{n} \overrightarrow{F}_{rep_k}(q) \tag{3.9}$$

with:

$$\begin{aligned} \overrightarrow{F}_{rep_k}(q) &= -\overrightarrow{\nabla} U_{rep_k}(q) \\ &= \begin{cases} \zeta(\frac{1}{\rho_k(q)} - \frac{1}{\rho_0})\frac{1}{\rho_k^2(q)}\overrightarrow{\nabla}\rho_k(q) & \text{if } \rho_k(q) \leq \rho_0 \\ 0 & \text{if } \rho_k(q) > \rho_0 \end{cases} \end{aligned} \tag{3.10}$$

Having computed both the attractive and repulsive potential field, we effectively defined an artificial potential field able to safely guide our robot to its goal configuration, independently of its starting position. Thanks to the **gradient descent** algorithm, our robot will always follow the direction of steepest descent until it reaches its goal configuration at the global minimum of the function, while simultaneously staying clear of any obstacle. The reader is invited to recall the analogy he was introduced to at the beginning of the chapter, and apply it to a functional example in light of the knowledge accumulated up to this point:

- the **robot** is the **spherical marble**
- the **environment** is the **artificial potential field**
- the **obstacles** are **protrusions**
- the environment is **plain**, the artificial potential field is a **slope**

- the **goal configuration** is the **global minimum** of the field

As the marble rolls downhill, the robot moves forward. Whenever the marble circumvents a protrusion, the robot avoids an obstacle. The iteration of these actions ultimately leads the marble to reach the bottom of the slope, at which point the robot will have attained its goal configuration.
Now that the reader understands how potential fields are employed in real motion planning applications, it is time to address realistic scenarios. Most notably, which issues are routinely incurred upon when applying this approach to real life scenarios.

3.2 Drawbacks

The potential field method was developed as an online collision avoidance approach for applications in **partially observable** environments. Its reliance on real-time informations make it susceptible to complications: the **Local Minima** problem and the **GNRON** problem.

3.2.1 Local Minima

By far the biggest drawback associated to this method, the local minima trap can easily be understood by considering the rolling marble example: in an ideal situation, the artificial potential field will present a single, global minimum which the marble will be attracted to and roll towards. However, such a consummate circumstance is seldom encountered in practice. Usually, the potential field will include several local minima, which the robot is likely to head to and be stuck in depending on its initial position. They are often associated with the presence of obstacles or the action of the repulsive force. Such a feature constitutes a grave drawback of this approach, also because it is laborious to work around. Possible solutions will be discussed henceforth.

navigation functions

A function $f : \xi_{free} \rightarrow [0, \infty]$ is called a **navigation function** if:

- it is smooth
- is uniformly maximal on the boundary of free space
- is a Morse function (every point x_c of f is isolated, where $\overrightarrow{\nabla} f(x_c) = 0$)
- **has a unique minimum at** q_{goal}

as can be seen, navigation functions have the unique characteristic of accepting a single minimum, which logically avoids the local minima trap by default. As effective as they are, however, navigation functions are only computable for a limited class of systems. Furthermore, they are more suitable for planning in partially observable, continuous environments; making them rarely compatible with potential field-guided path planning.

repulsive and vortex fields

Repulsive Fields and **Vortex Fields** follow the same principle: reworking the original function to have any existing local minima repel the robot.
The **repulsive field** approach focuses on the existing repulsive potential function, and modifies it by adding an external force. This ensures that whenever the robot approaches a local minima, it is pushed away by it.
The **vortex field** approach, on the other hand, replaces the repulsive force of the artificial field with an action forcing the robot to go around the obstacle by means of a force field, thus eliminating the source of the problem. These proposed techniques are more suited

to partially observable environments planning, as they do not require total access to its informations.
Other solutions Include:

- heuristics
- backtracking from the local minimum and implementing a different approach in order to avoid it

Having introduced and discussed the local minima issue, as well as the most commonly employed techniques for handling it, we will address a second limitation inherent to the potential field approach: the **GNRON**.

3.2.2 GNRON

The **GNRON** problem, while seemingly different, is actually a particular case of the local minima issue. It stands for **G**oal **N**on **R**eachable with **O**bstacle **N**earby. This complication arises when our goal configuration q_{goal} is in the vicinity of an obstacle. In fact, as we have previously discussed, our repulsive potential function pushes the robot away when it nears an obstacle. We also understand that the attractive potential function yields lower values the closer we are to q_{goal}. Thus, in the aforementioned configuration, the high repulsive value of the potential function given by the vicinity of an obstacle will best its attractive value, low due to the proximity of q_{goal}. In other words, the goal configuration will no longer correspond to the global minimum of the function whenever it is within the distance of influence of an obstacle.
The most effective solution to solve this problem requires a suitable adjustment of the repulsive potential field, which will now be investigated.

Adjusting the repulsive potential field

Ideally, we would want to reshape $\mathbf{U}_{rep}$ so that it does not affect our robot if it is nearing $\mathbf{q}_{goal}$. However, we have already placed a distance-related restriction on the repulsive potential, and adding a second one would complicate the function greatly, introducing further issues: we have to find a different approach. Let us consider:

$$U_{rep}(q) = \begin{cases} \frac{1}{2}\zeta(\frac{1}{\rho(q,q_{obs})} - \frac{1}{\rho_0})^2\rho^n(q, q_{goal}) & \text{if } \rho(q, q_{obs}) \leq \rho_0 \\ 0 & \text{if } \rho(q, q_{obs}) > \rho_0 \end{cases} \tag{3.11}$$

Where:

- $\rho(q, q_{obs})$ is the shortest euclidean distance between the robot and an obstacle
- $\rho(q, q_{goal})$ is the euclidean distance between the robot and the goal
- ρ_0 is the distance of influence of the obstacle
- ζ is a positive constant

The introduction of the new term $\rho^n(q, q_{goal})$ ensures that the potential function reaches its global minimum if and only if $\mathbf{q} = \mathbf{q}_{\text{goal}}$, thus eliminating the issue.

Conclusions

This dissertation explored the relatively new and fast growing field of path planning, and studied how potential fields are employed in its practical applications. At first, the reader was briefly introduced to the world of robotic motion planning via a quick tour through the foundations of this field. Subsequently, the dissertfation delved into its core subject: potential fields. It began by illustrating how they are designed and tailored in order to suit the robot's designated application, and concluded by delineating those impediments which are frequently faced when exercising this approach.

Bibliography

[1] Maria Rita Casali, Carlo Gagliardi, Luigi Grasselli. *GEOMETRIA*. ESCULAPIO, San Giovanni in Persiceto (BO), 2016.

[2] Jean-Claude Latombe. *Robot Motion Planning*. Springer US, Secaucus, New Jersey, 1990.

[3] Hendri Himawan Triharminto. *A Novel of Repulsive Function on Artificial Potential Field for Robot Path Planning*. IJECE, 6(6): 3262-3275, 2016.

[4] Howie Choset,
`http://voronoi.sbp.ri.cmu.edu/~motion`.
`http://voronoi.sbp.ri.cmu.edu/~choset`.

[5] Camillo J. Taylor. *Computational Motion Planning*. Penn State University.

[6] Camillo J. Taylor. *Artificial Potential Fields*. Penn State University.

[7] Gregor Klančar, Andrej Zdešar, Sašo Blažič, Igor Škrjanc. *Wheeled Mobile Robotics*. Butterworth-Heinemann Elsevier, 2017.

[8] Jesus Rodriguez. *6 Types of Artificial Intelligence Environments*.
`https://medium.com/@jrodthoughts/6-types-of-artificial-intelligence-environments-825e3c47d998`.

[9] Giuseppe Oriolo. *Motion Planning I: Retraction and Cell Decomposition*. Sapienza, University of Rome.

[10] Giuseppe Oriolo. *Motion Planning 3: Artificial Potential Fields*. Sapienza, University of Rome.

[11] Nancy Amato. *Potential Fields Methods*. University of Padova, 2004.

[12] S.S.Ge, Y.J.Cui. *New Potential Functions for Mobile Robot Path Planning*. IEE Transactions on Robotics and Automation, 16(5), 2000.

[13] Matteo Franca. *lezioni su funzioni in più variabili*. Alma Mater Studiorum, University of Bologna, 2020.