TECHNISCHE
UNIVERSITÄT
WIEN
Vienna University of Technology

Collision-Free Indoor Navigation Using an Artificial Potential Field Controller

MASTERARBEIT

eingereicht an

der Technischen Universität Wien im

MASTERSTUDIENGANG TELEKOMMUNIKATION (066 437)

TUNA TANDOGAN

Wien, März 2015

Betreuer:

Florian Meyer Univ.Ass. Dipl.-Ing.
Franz Hlawatsch Ao.Univ.Prof. Dipl.-Ing. Dr.techn.

Preface

First and foremost, I would like to thank my parents for their infinite patience in supporting my education. I would also like to thank Franz Hlawatsch who has encouraged me to strive for my best, and Florian Meyer who was always available for advice. Last but not least, I would like to thank all my friends who have made Vienna University of Technology a wonderful place to study at, and Vienna, an enjoyable city to live in.

Tuna Tandogan
March 2015

Abstract

Path planning is the problem of finding a collision-free path from a mobile agent's initial position to its goal position, and moving the agent along that path. This Master's thesis presents a control algorithm based on artificial potential fields for navigating a mobile agent in indoor environments. An artificial potential field is a scalar field defined in the environment of the agent, whose intensity relates to objects in the environment and the goal of the agent. The gradient of the scalar field is a vector field and determines the motion of the agent. The proposed algorithm produces control vectors that move the agent in an indoor environment towards its goal position. Collision with obstacles is quantified probabilistically, and the resulting quantity called the *probability of collision* is used to adapt the potential fields. Localization is a necessary step for calculating the probability of collision and the control vectors. Anchor nodes with known positions in the environment provide measurement information necessary for localization. The agent performs range measurements relative to the anchor nodes, and its position is estimated by a cubature Kalman filter from observed data. The probability of collision is used to adapt the artificial potential field such that within the framework of our approximation of the true probability of collision, the resulting control vector attempts to keep the probability of collision under a threshold. Simulations show that the proposed algorithm enables the agent to navigate indoor environments without collisions in various scenarios. In the absence of local minima between the initial and goal positions of the agent, a path is easily found with little to no adaptation of the fields. The algorithm also shows some capability for escaping from and avoiding local minima.

Table of contents

List of figures xi

Nomenclature xii

1 Introduction 1
- 1.1 Motivation . . . 1
- 1.2 Outline . . . 2
- 1.3 Configuration Space . . . 2
- 1.4 Path Planning . . . 3
- 1.5 State of the Art . . . 5
- 1.6 System Model . . . 6
  - 1.6.1 Motion Model . . . 7
  - 1.6.2 Measurement Model . . . 8

2 Estimation and Control Strategy 9
- 2.1 Sequential Bayesian Estimation . . . 9
- 2.2 Cubature Kalman Filter . . . 10
- 2.3 Control Strategy . . . 12
- 2.4 Calculation of Control Vector . . . 13

3 Artificial Potential Fields 15
- 3.1 Background . . . 15
- 3.2 Local Minima . . . 15
- 3.3 Attractive Field . . . 18
- 3.4 Repulsive Field . . . 18
- 3.5 Obstacles and Boundaries . . . 22

4 Probability of Collision 23
- 4.1 Probability of Collision . . . 23
- 4.2 Decorrelation Transform . . . 24
- 4.3 Approximate Probability of Collision . . . 25

4.4 Non-Convex Obstacles 28
4.5 Numerical Validation of the Approximation 28

5 Numerical Simulations **33**
5.1 Effects of Simulation Parameters 33
5.2 Simulation Scenarios 35
5.2.1 Indoor Scenario with a Rectangular Obstacle 35
5.2.2 Indoor Scenario with a Local Minimum 36
5.2.3 Indoor Scenario with Multiple Obstacles 37

6 Conclusion **41**

References **43**

List of figures

1.1 The path planning problem. 4
1.2 The overall signal processing system. 7

2.1 The hidden Markov model of the dynamic system used for sequential Bayesian estimation. 10

3.1 Two scenarios that produce a local minimum $\mathbf{q}_{\min}$. 16
3.2 Visualization of the attractive field. 18
3.3 Geometric underlying the integral in Equation (3.3). 20
3.4 Repulsive field around a 1D obstacle with $w = 1$ and $R = 1$. 20
3.5 Effects of varying w and R on the repulsive field $\phi_r(\mathbf{q}; R, w)$. 21
3.6 A configuration space with orientable surfaces. 22

4.1 Shortest distances to obstacle edges. 24
4.2 Probability of collision with an edge. 25
4.3 Effect of decorrelation transformation. 26
4.4 Replacing the transformed region with the original region. 26
4.5 Shortest distances to polygon edges. 26
4.6 Convex hull of a non-convex obstacle 28
4.7 Mobile agent moves on a predetermined path around a rectangular obstacle. 29
4.8 Exact and approximate probabilities of collision with $\rho = 0$. 30
4.9 Exact and approximate probabilities of collision with $\rho = 0.25$. 30
4.10 Exact and approximate probabilities of collision with $\rho = 0.5$. 31

5.1 Simulation with $\sigma_w^2 = 0.1$ and $P_{\text{th}} = 10^{-2}$. 34
5.2 Simulation with $\sigma_w^2 = 4$ and $P_{\text{th}} = 10^{-2}$. 35
5.3 Simulation with $\sigma_w^2 = 0.1$ and $P_{\text{th}} = 10^{-10}$. 36
5.4 Simulation with $\sigma_w^2 = 4$ and $P_{\text{th}} = 10^{-10}$. 37
5.5 Simple scenario with one rectangular obstacle. 38
5.6 Local minimum scenario 1. 38
5.7 Local minimum scenario 2. 39

5.8 A scenario with multiple obstacles. 39
5.9 Number of iterations per time step for the many-obstacle scenario. 40

Chapter 1

Introduction

The field of autonomous mobile robotics deals with the design of mobile robots that are capable of navigating and performing tasks in their environments without human input. A mobile agent needs to be able to perform various subtasks to achieve autonomy. These are [1]:

- *Perception*: The ability of the agent to sense its environment.
- *Localization*: The ability of the agent of determine its position in the environment.
- *Cognition*: The ability to decide how to act to achieve its goals.
- *Motion control*: The ability to generate the necessary motor outputs to achieve desired trajectory.

Examples of applications of autonomous mobile robots include: exploring hostile environments [2], long-term exploration and data collection [3], surveillance [4] and personal aide robots [5].

For autonomous mobile robots, an integral part of the cognition stage is path planning, which is the problem of finding a set of admissible positions that takes a mobile agent from its initial position to its desired final position. Path planning can be considered as the most important problem in mobile robotics. Indeed, for a mobile agent to behave autonomously, it needs to be able to safely navigate in an environment without human intervention. Localization is a necessary step to perform path planning, however perfect localization is impossible due to the presence of noise in sensor measurements. For the path planning algorithm to provide a collision free path, the agent has to take the position uncertainty of the agent into account.

1.1 Motivation

Artificial potential field (APF) methods provide an elegant solution to the path finding problem. The resulting algorithms are easy to implement and are of low complexity, which makes them ideal for real-time systems. Unfortunately, due to the existence of local minima, APF methods might fail to

find a path. In addition, uncertainty in position estimates may lead the agent to a collision since the forces due to APFs acting on the agent are position dependent. This thesis presents a probablistic method that attempts to avoid local minima while safely guiding the agent to the goal. At any time point, the agent computes its probability of collision with the obstacles. The computed probability of collision is compared with a threshold, which is a design parameter and depends on the application. For a fast-moving airborne agent where a collision can be a catastrophic, the threshold would have a low value whereas for a slow-moving wheeled agent navigating indoors, collisions might be more tolerable and thus the threshold would be higher. The agent tunes the APF to produce control vectors that will move it to its goal position while keeping the probability of collision under the specified threshold.

1.2 Outline

- The remainder of this chapter introduces the concept of configuration space. The problem of path planning is explained and our system model is presented in the following sections. Finally, an overview of the state-of-the-art methods in path planning is given in the final section.
- In **Chapter 2** the sequential Bayesian estimator is introduced and the recursive equations are presented in the context of the navigation problem. Finally, the cubature Kalman filter (CKF) is presented as a sub-optimal filter for non-linear problems.
- **Chapter 3** provides background information on APFs and an overview of the most recent developments in this area. Furthermore, the types of fields used by the controller proposed in this thesis are presented. Finally, a representation for obstacles and boundaries in introduced.
- In **Chapter 4**, the probability of collision is defined and a method of calculating an upper bound on the probability of collision is introduced. The accuracy of the approximation is compared to the exact probability of collision in numerical simulations.
- **Chapter 5** presents results of the numerical simulations in which the performance of the controller is tested in various indoor scenarios.

1.3 Configuration Space

A mobile agent moves in the *workspace* $\mathscr{W}$ which is a mathematical representation of the physical world [1, 6]. If the motion of the agent is constrained to a plane, $\mathscr{W}$ is the Euclidean-space $\mathbb{R}^2$. If the agent is able to move in three-dimensional space, then $\mathscr{W}$ is $\mathbb{R}^3$.

A complete description of the pose of a mobile agent with d degrees-of-freedom is given by a d-dimensional parameter vector $\mathbf{q} = (q^{(1)} \dots q^{(d)})^{\mathrm{T}}$, which is called the configuration of the mobile

agent. The space of all possible configurations of a mobile agent is called the *configuration space* $\mathscr{C}$, which is a d-dimensional manifold.

In the case of a mobile agent modeled as a two-dimensional object that can translate and rotate on a plane, three parameters are required to specify its configuration, two for its position and one for its orientation. The resulting three-dimensional manifold is $\mathbb{R}^2 \times S^1$, and the agent is said to have three degrees-of-freedom. If the mobile agent is modeled as an object that can translate in 3-D and rotate around its three axes, six parameters are required to define its configuration and its C-space is the 6-D manifold $\mathbb{R}^3 \times S^3$ [7].

Path planning of a mobile agent is usually done in $\mathscr{C}$. However, the problem becomes computationally intractable as the DOF of a robot increases. Indeed, it has been shown that path planning in $\mathbb{R}^3$ is NP-hard [8]. A common approach for simplifying path planning is to treat the agent as a point. A further simplification is obtained by assuming that the agent is holonomic. A holonomic agent is one whose *differential DOF* (DDOF) has the same dimension as its *DOF* [1]. DDOF corresponds to the number of configuration parameters that the agent can set independently. For a translating point agent moving in $\mathbb{R}^2$, this means the agent can set velocities in x and y directions independently.

This thesis treats the mobile agent as a 1D circular holonomic robot with physical radius r. With this assumption, the orientation of the mobile agent does no longer play a role in path planning. Path planning is performed in $\mathscr{C}$ where the agent is represented by a point and all obstacles are enlarged by the radius r of the agent. It is assumed that all obstacles have been enlarged appropriately. The configuration of the agent at time t is then given by the vector $\mathbf{q}_t = (x_t, y_t)^{\mathrm{T}}$.

1.4 Path Planning

Figure 1.1 depicts a typical scenario where path planning is necessary to move the agent to the goal. Path planning methods can be broadly grouped into three categories: *roadmaps* [9–12], *cell decomposition* [10–12] and *artificial potential fields* [10–12].

- *Roadmap* [9–12] approaches identify a set of routes in the free configuration space $\mathscr{C}$ which are represented by a graph G. Once the graph is constructed, a graph search algorithm finds a path from the agents initial configuration $\mathbf{q}_{\text{init}}$ to the goal configuration $\mathbf{q}_{\text{goal}}$. Examples of roadmap-based path planning methods are the *visibility graph* [13] and *Voronoi diagram* [14] approaches. The former captures the connectivity of $\mathscr{C}$ in a visibility graph whose nodes represent the vertices of polygonal obstacles as well as $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$, and the edges connect nodes which have a line-of-sight with each other. The latter method captures the connectivity of the space $\mathscr{C}$ in a Voronoi diagram which partitions the space $\mathscr{C}$ into smaller regions such that all the points on a boundary separating two regions are equidistant to the obstacles contained in the two regions.
- *Cell decomposition* [10–12] approaches divide the configuration space $\mathscr{C}$ into simple connected regions, called cells, that are either free or occupied. The relationship of neighboring

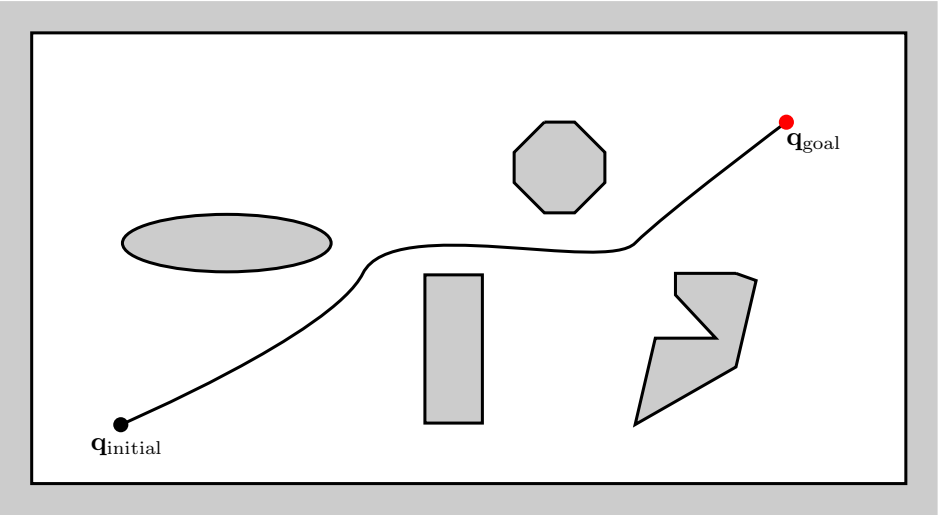


Fig. 1.1 The path planning problem.

free cells is encoded in an adjacency graph which is then searched for a path connecting $\mathbf{q}_{\text{init}}$ and $\mathbf{q}_{\text{goal}}$. Most well-known cell decomposition methods are the grid-based approaches which can be split into two groups; *regular grids* [15] and *irregular grids* [15]. Regular grids decompose the environment into cells of equal size; as a result regular grids have the same number of nodes and edges regardless of the number of obstacles in the environment, which makes them desirable from a computational perspective. However, decomposing obstacle free regions into small cells causes unnecessary memory overhead. Irregular grids overcome this problem by first decomposing the region into large cells, and in the next iteration only the cells that contain an obstacle are further decomposed. *Quadtrees* are examples of irregular grids [16].

- *Artificial potential field (APF)* [10–12] approaches construct a potential field $\phi(\mathbf{q})$ in the configuration space $\mathscr{C}$ that directs the agent to the goal configuration $\mathbf{q}_{\text{goal}}$. The potential field $\phi(\mathbf{q})$ has two components: the attractive field $\phi_{\text{a}}(\mathbf{q})$ which moves the robot towards the goal, and the repulsive field $\phi_{\text{r}}(\mathbf{q})$ which pushes the robot away from obstacles [17]. The overall field is given by the sum of the attractive and the repulsive fields, i.e. $\phi(\mathbf{q}) = \phi_{\text{a}}(\mathbf{q}) + \phi_{\text{r}}(\mathbf{q})$. In comparison to the roadmap and cell decomposition approaches, with APFs not only path planning can be performed, they also serve as a control law for the agent [1]. If the agent is able to localize its configuration with respect to the goal and obstacle configurations, then the gradient of the potential field is the control input for the current time step. Since the path of the agent is determined with how it reacts to the potential field, this method is also called *reactive control*. [18].

The roadmap and cell decomposition approaches combine an environment modeling step with a graph search algorithm to find a path. Since they require *a priori* knowledge of the environment,

they are also called *global methods*. In contrast, the APF methods only require knowledge of goal configuration to construct the attractive field; the repulsive fields are constructed *on-line* as obstacles are sensed. Therefore, APF methods are referred to as *local* methods [19].

1.5 State of the Art

Increasing popularity of real-time systems such as autonomous mobile robots and video games motivated research into fast path planning algorithms. In the context of global methods, research has been focused on environment modeling and graph search methods. In addition, for local methods such as APFs, numerous strategies have been proposed to escape from local minima. In this section, we provide an overview of the developments in roadmap and cell decomposition based path planning methods, as well as methods involving the sampling of the workspace.

The pioneering work in the field of graph search was introduced in [20] and the method came to be known as *Dijkstra's algorithm*. The so called *A** algorithm [21] introduces a heuristic to lead the search in direction of the goal node, thereby reducing the number of graph nodes explored. Consequently the A* algorithm runs much faster than Dijkstra's algorithm and is one of the most well-known pathfinding algorithms today. However, A* is still much too slow for real-time applications; in addition, it requires prior knowledge of the environment and it is also not suitable for dynamic environments.

Several variants of the A* algorithm have been proposed to rectify its shortcomings. The *D** algorithm proposed in [22] extended the applicability of A* algorithm to dynamic environments. In [23], the *D* Lite* algorithm, which is algorithmically simpler and runs at least as fast as D* was introduced. So far, these algorithms constrained the position of the agent to the grid points, as a result the steering angle of the agent was limited to multiples of 45°. The *Field D** algorithm, proposed in [24], removes this constraint and allows the agent to move along at any angle, resulting in shorter paths. *Hierarchical Annotated A** (HAA*), proposed in 2008 [25], can be considered to be one of the most advanced path planning algorithms. HAA* satisfies real-time constraints by subdividing the problem into hierarchical subproblems and is able to deal with heterogeneous multi-terrain maps.

Another class of methods that received widespread attention is the class of sampling-based planners. Sampling-based planners are probabilistic methods that exploit the fact that it is cheap to check if a point in $\mathscr{C}$ lies in the free configuration space or the obstacle configuration space [12]. By randomly sampling points in the free configuration space, a roadmap can be efficiently constructed and then searched for a path. One of the earliest examples of sampling-based planners are the *Probabilistic Roadmaps* (PRM) [26]. PRM uses coarse sampling to obtain nodes in the $\mathscr{C}_{\text{free}}$ and fine sampling to construct collision free edges between the nodes. RPM was conceived as a multi-query planner, i.e. it doesn't only consider paths from $\mathbf{q}_{\text{initial}}$ to $\mathbf{q}_{\text{goal}}$, as a result it may not be the fastest approach. This motivated the development of effective single-query planners such as the *Rapidly-exploring Random Tree* (RRT) planner [27]. RRT planners produce a path by randomly growing a tree from

$\mathbf{q}_{\text{initial}}$ to $\mathbf{q}_{\text{goal}}$ in an iterative procedure. Similar to APF methods, RRTs do not require an initial environment modeling step, instead the environment is explored as the planner searches for a path. In [28], an asymptotically-optimal variation of RRT, called RRT*, whose complexity is a constant factor of the complexity of RRT is proposed. The more recent *RRT*-Smart* algorithm, proposed in [29], performs an informed exploration of the search space instead of randomly sampling from it and thus reduces the execution time.

Despite the problem of local minima, APF methods remain relevant today and they are, in fact, considered one of the best methods for on-line path planning. APF methods are explored in more detail in Chapter 3. For a more detailed review of path planning methods, the interested reader is referred to [15, 30].

1.6 System Model

The *state-space-model* (SSM) of a dynamic agent can be written as a system of equations

$$\mathbf{q}_t = \mathbf{g}\big(\mathbf{q}_{t-1},\, \mathbf{u}_t\big) + \mathbf{v}_t, \qquad \text{(Motion model)} \tag{1.1}$$

$$\mathbf{z}_t = \mathbf{h}(\mathbf{q}_t) + \mathbf{w}_t. \qquad \text{(Measurement model)} \tag{1.2}$$

where $\mathbf{q}_t \in \mathbb{R}^2$ is the state of the mobile agent at discrete time t, $\mathbf{v}_t$ is the white process noise, $\mathbf{w}_t$ is the white measurement noise; $\mathbf{g}(\cdot)$ and $\mathbf{h}(\cdot)$ are known, possibly nonlinear, functions. The control input to the system $\mathbf{u}_t$ is known. Let us denote by $\mathbf{q}_{0:t-1} \triangleq \big[\mathbf{q}_0^{\mathrm{T}}, \ldots, \mathbf{q}_{t-1}^{\mathrm{T}}\big]^{\mathrm{T}}$, i.e. a stacked vector containing the history of agent positions. Similarly, let $\mathbf{z}_{0:t-1} \triangleq \big[\mathbf{z}_0^{\mathrm{T}}, \ldots, \mathbf{z}_{t-1}^{\mathrm{T}}\big]^{\mathrm{T}}$ and $\mathbf{u}_{0:t} \triangleq \big[\mathbf{u}_0^{\mathrm{T}}, \ldots, \mathbf{u}_t^{\mathrm{T}}\big]^{\mathrm{T}}$.

Due to the presence of noise terms, $\mathbf{q}_t$ and $\mathbf{z}_t$ are random vectors with the following probability densities:

$$\mathbf{q}_t \sim\; p\big(\mathbf{q}_t \big| \mathbf{q}_{0:t-1}, \mathbf{z}_{1:t-1}; \mathbf{u}_{1:t}\big), \tag{1.3}$$

$$\mathbf{z}_t \sim\; p\big(\mathbf{z}_t \big| \mathbf{q}_{0:t}, \mathbf{z}_{1:t-1}; \mathbf{u}_{1:t}\big). \tag{1.4}$$

A state-space model with random variables is also referred to as a *probabilistic state-space model* [31]. The overall signal processing system is shown in Figure 1.2 and consists of four blocks: Sensor, Estimator, Controller and Dynamics.

The sensor block is responsible for generating range measurements and is further discussed in Subsection 1.6.2. The estimator block estimates the current position of the agent using the current measurements which are performed with respect to the anchors at positions $\mathbf{q}_{\text{anchor}}^{(i)},\ \big(i = 1, \ldots, N\big)$ and control vector executed at the previous time step, and the position information from the previous time step. The estimation problem is expanded upon in Chapter 2. The controller block calculates the probability of collision with the obstacles $\mathbf{O}^{(j)},\ j = \big(j = 1, \ldots, M\big)$ and constructs appropriate APFs around the obstacles and goal position $\mathbf{q}_{\text{goal}}$. The negative gradient of the overall APF provides

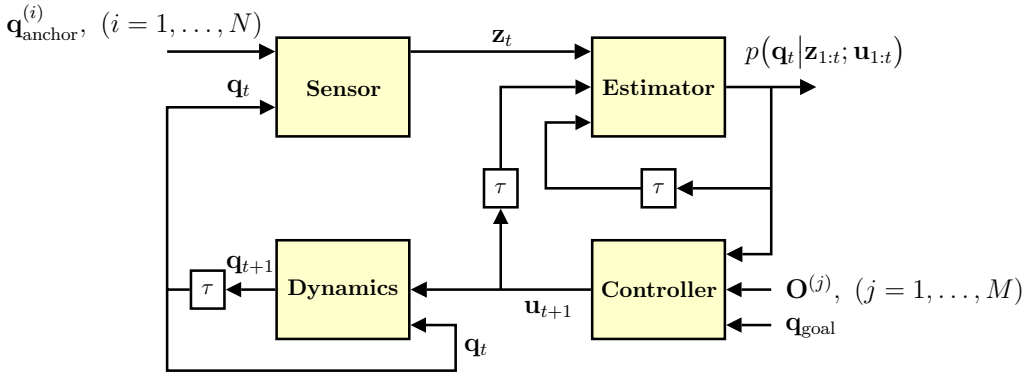


Fig. 1.2 The overall signal processing system.

the control vector to move the agent to its next position on a collision free path. The strategy for computing the control vector is further discussed in Chapter 2, and in Chapter 4, the algorithm for computing the probability of collision is presented. Finally, the dynamics block, which is discussed in Subsection 1.6.1, executes the control vector and moves the agent to its next position.

1.6.1 Motion Model

The motion model governs the evolution of the agents state from one time step to the next. With the simplifying assumption of an holonomic point agent, the process equation can be written as:

$$\mathbf{q}_t = \mathbf{q}_{t-1} + \mathbf{u}_t\Delta_t + \mathbf{v}_t. \tag{1.5}$$

The process noise $\mathbf{v}_t$ is assumed to be an independent and identically distributed (iid) zero mean Gaussian random vector with covariance matrix $\sigma_v^2\mathbf{I}_2$, where $\mathbf{I}_2$ denotes the 2×2 identity matrix. As a result, $\mathbf{v}_t$ is described by a bivariate normal distirbution $\mathscr{N}(\mathbf{0},\sigma_v^2\mathbf{I}_2)$. In addition, realizations of $\mathbf{v}_t$ at different times are also assumed to be uncorrelated. The states of the agent form a Markov sequence since the state $\mathbf{q}_t$ given $\mathbf{q}_{t-1}$ and $\mathbf{u}_t$ is independent of the previous states of the agent [31]. Physically, this means that the next position of the agent depends only on its current position and control vector. Therefore, the probability density in (1.3) simplifies to:

$$\mathbf{q}_t \sim\ p\big(\mathbf{q}_t\big|\mathbf{q}_{t-1};\mathbf{u}_t\big). \tag{1.6}$$

This equation is in agreement with the dynamics block of Figure 1.2 up to a unit time-shift, which accepts $\mathbf{q}_t$ and $\mathbf{u}_{t+1}$ to produce $\mathbf{q}_{t+1}$. For the interpretation of $\mathbf{u}_t$ within Equation (1.1), it is assumed that the agent knows the orientation of the global frame of reference.

1.6.2 Measurement Model

Localization of the mobile agent is accomplished by performing distance measurements relative to the N anchors, present in the workspace of the agent at the positions $\mathbf{q}_{\text{anchor}}^{(i)}$, $i = 1,\ldots,N$. The measurement equation can be written as:

$$z_t^{(i)} = \left\|\mathbf{q}_t - \mathbf{q}_{\text{anchor}}^{(i)}\right\| + w_t^{(i)}. \tag{1.7}$$

The measurement noise is assumed to be an iid zero mean Gaussian random vector with covariance matrix $\sigma_w^2\ \mathbf{I}_N$, that is $\mathbf{w}_t \sim \mathcal{N}(\mathbf{0}, \sigma_w^2\ \mathbf{I}_N)$. Furthermore, the measurements are assumed to be conditionally independent, i.e. the current measurement $\mathbf{z}_t$ given the current state $\mathbf{q}_t$ is conditionally independent of measurement and state histories [31]; the measurement at time t depends only on the state of the agent at time t. Therefore, the probability density in (1.4) can be written as:

$$\mathbf{z}_t \sim\ p\big(\mathbf{z}_t \big| \mathbf{q}_t\big). \tag{1.8}$$

Equation (1.8) corresponds to the sensor block of Figure 1.2 which accepts $\mathbf{q}_t$ and anchor positions $\mathbf{q}_{\text{anchor}}^{(i)}$ and outputs $\mathbf{z}_t$.

Chapter 2

Estimation and Control Strategy

Chapter 1 introduced the subtasks that an agent needs to be able to perform to achieve autonomy. The agent is able to perceive the workspace through the measurement model introduced in Section 1.6.2, and motion model introduced in Section 1.6.1 controls the movement of the agent. In this chapter, we look at the problems of localization and cognition. We start with a review of sequential Bayesian estimation in Section 2.1, which forms the basis of the cubature Kalman filter discussed in Section 2.2. In Section 2.3, our control strategy for collision free navigation is presented. In the final section, we formulate the relationship between the APFs and the control vector the agent computes.

2.1 Sequential Bayesian Estimation

Given a probabilistic state-space model, the sequential Bayesian estimator calculates the conditional posterior distribution of the unobserved state vector given the history of observed measurement vectors and control vectors. The conditional posterior distribution is denoted as

$$p\big(\mathbf{q}_t \big| \mathbf{z}_{1:t}; \mathbf{u}_{1:t}\big). \tag{2.1}$$

Taking into account the Markovian property of the motion model and the conditional independence of subsequent measurements in time (see Section 1.6.1 and Section 1.6.2), the posterior distribution can be written as:

$$p\big(\mathbf{q}_t \big| \mathbf{z}_{1:t}; \mathbf{u}_{1:t}\big) = \frac{p(\mathbf{z}_t|\mathbf{q}_t) \int p(\mathbf{q}_t|\mathbf{q}_{t-1}; \mathbf{u}_t)\; p(\mathbf{q}_{t-1}|\mathbf{z}_{1:t-1}; \mathbf{u}_{1:t-1})\; d\mathbf{q}_{t-1}}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1}; \mathbf{u}_{1:t})}. \tag{2.2}$$

A dynamic system with this property is also called a Hidden Markov Model [32]. Fig 2.1 depicts the evolution of states and measurements which are defined by Equations (1.6) and (1.8). The sequential Bayesian estimator establishes a direct relationship between the previous posterior density $p\big(\mathbf{q}_{t-1} \big| \mathbf{z}_{1:t-1}; \mathbf{u}_{1:t-1}\big)$ and the current posterior density $p\big(\mathbf{q}_t \big| \mathbf{z}_{1:t}; \mathbf{u}_{1:t}\big)$, more specifically, us-

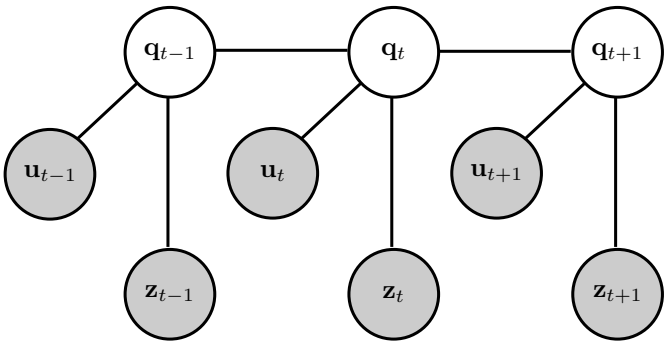


Fig. 2.1 The hidden Markov model of the dynamic system used for sequential Bayesian estimation.

ing the observed measurements $\mathbf{z}_t$ and the control vector $\mathbf{u}_t$ (see Figure 1.2), the current posterior $p\big(\mathbf{q}_t\big|\mathbf{z}_{1:t};\mathbf{u}_{1:t}\big)$ is obtained as follows:

- Prediction step:

$$p(\mathbf{q}_t|\mathbf{z}_{1:t-1};\mathbf{u}_{1:t}) = \int p(\mathbf{q}_t|\mathbf{q}_{t-1};\mathbf{u}_t)\; p(\mathbf{q}_{t-1}|\mathbf{z}_{1:t-1};\mathbf{u}_{1:t-1})\; d\mathbf{q}_{t-1}. \tag{2.3}$$

- Update step:

$$p(\mathbf{q}_t|\mathbf{z}_{1:t};\mathbf{u}_{1:t}) = \frac{p(\mathbf{z}_t|\mathbf{q}_t)\; p(\mathbf{q}_t|\mathbf{z}_{1:t-1};\mathbf{u}_{1:t})}{p(\mathbf{z}_t|\mathbf{z}_{1:t-1};\mathbf{u}_{1:t})}. \tag{2.4}$$

The position estimate for the time step t is the expectation of the posterior probability density:

$$\widetilde{\mathbf{q}}_{\mathrm{t}} = \mathbb{E}[\mathbf{q}_t|\mathbf{z}_{1:t};\mathbf{u}_{1:t}] = \int \mathbf{q}_t\; p(\mathbf{q}_t|\mathbf{z}_{1:t};\mathbf{u}_{1:t})\; d\mathbf{q}_t. \tag{2.5}$$

2.2 Cubature Kalman Filter

The celebrated Kalman filter provides an exact representation of the posterior distribution if the state-space equations are linear and the probability distributions are Gaussian. In practice, these criteria are often not met and most real-world systems have non-linear process and measurement equations. To deal with these non-linearities, several modifications have been proposed to the original Kalman filter.

The extended Kalman filter (EKF) is one of the earliest modifications of the Kalman filter that can deal with non-linear state-space equations. While the Kalman filter propagates a Gaussian random variable (GRV) through linear system dynamics, the EKF propagates the GRV through linearized non-linear system dynamics. As a result, the EKF is a sub-optimal filter that approximates the true posterior distribution with a GRV. The EKF is conceptually simple and easy to implement, it has a major disadvantage: if the state-space equations are highly-nonlinear, the EKF provides a poor

approximation of the true posterior distribution. This is due to linearizing the non-linear equations around a single point, which disregards the underlying uncertainty of the random variable.

An alternative method for dealing with the non-linearities is provided by the family of Sigma-Point Kalman filters (SPKF) [31, 33]. One of the most well known SPK filters is the unscented Kalman filter (UKF) and the cubature Kalman filter (CKF), which was derived in [34], can be viewed as a special case of the UKF. The basic idea behind the SPKFs is to deterministically choose a fixed number of sigma-points that capture the mean and covariance of the prior distribution and then propagate the sigma-point through the non-linear state-space equations. The transformed sigma-points are then used to estimate the mean and covariance of the posterior distribution.

The CKF algorithm calculates the mean $\mu_{\mathbf{q},t}$ and covariance matrix $\mathbf{C}_{\mathbf{q},t}$ corresponding to the Gaussian approximation of the true posterior density $p\left(\mathbf{q}_t \middle| \mathbf{z}_{1:t}; \mathbf{u}_{1:t}\right)$ by performing the following steps:

- Prediction:

  1. Compute the sigma-points from the prior probability distribution as:

$$\chi_{k-1}^{(m)} = \begin{cases} \mu_{\mathbf{q},t-1} + \sqrt{M}\left(\mathbf{C}_{\mathbf{q},t-1}\right)_m^{1/2}, & m = 1,\ldots,M \\ \mu_{\mathbf{q},t-1} - \sqrt{M}\left(\mathbf{C}_{\mathbf{q},t-1}\right)_{m-M}^{1/2}, & m = M+1,\ldots,2M, \end{cases} \tag{2.6}$$

     where M is the dimension of the state vector $\mathbf{q}_t$, and $\left(.\right)_m^{1/2}$ denotes m^{th} column of the square root of the matrix.

  2. Propagate the sigma-points through the state-transition model:

$$\widetilde{\chi}_t^{(m)} = \mathbf{g}\left(\chi_{t-1}^{(m)}, \mathbf{u}_t\right) + \mathbf{v}_t, \quad m = 1,\ldots,2M. \tag{2.7}$$

  3. Compute the predicted mean $\widetilde{\mu}_{\mathbf{q},t}$ and predicted covariance $\widetilde{\mathbf{C}}_{\mathbf{q},t}$:

$$\widetilde{\mu}_{\mathbf{q},t} = \frac{1}{M}\sum_{m=1}^{M} \widetilde{\chi}_t^{(m)}, \tag{2.8}$$

$$\widetilde{\mathbf{C}}_{\mathbf{q},t} = \frac{1}{M}\sum_{m=1}^{M} \left(\widetilde{\chi}_t^{(m)} - \widetilde{\mu}_{\mathbf{q},t}\right)\left(\widetilde{\chi}_t^{(m)} - \widetilde{\mu}_{\mathbf{q},t}\right)^{\mathrm{T}} + \sigma_v^2 \mathbf{I}_2. \tag{2.9}$$

- Update:

  1. Compute sigma-points from the predictive probability distribution:

$$\zeta_t^{(m)} = \begin{cases} \widetilde{\mu}_{\mathbf{q},t} + \sqrt{M}\left(\widetilde{\mathbf{C}}_{\mathbf{q},t}\right)_m^{1/2}, & m = 1,\ldots,M \\ \widetilde{\mu}_{\mathbf{q},t} - \sqrt{M}\left(\widetilde{\mathbf{C}}_{\mathbf{q},t}\right)_{m-M}^{1/2}, & m = M+1,\ldots,2M. \end{cases} \tag{2.10}$$

2. Propagate the sigma-points through the measurement model:

$$\widetilde{\zeta}_t^{(m)} = \mathbf{h}\big(\zeta_t^{(m)}\big) + \mathbf{w}_t, \quad m = 1, \ldots, 2M. \tag{2.11}$$

3. Compute the predicted mean $\widetilde{\mu}_{\mathbf{z},t}$ and predicted covariance $\widetilde{\mathbf{C}}_{\mathbf{z},t}$, predicted cross-covariance of state and measurement $\widetilde{\mathbf{C}}_{\mathbf{q},\mathbf{z},t}$:

$$\widetilde{\mu}_{\mathbf{z},t} = \frac{1}{M} \sum_{m=1}^{M} \widetilde{\zeta}_t^{(m)}, \tag{2.12}$$

$$\widetilde{\mathbf{C}}_{\mathbf{z},t} = \frac{1}{M} \sum_{m=1}^{M} \big(\widetilde{\zeta}_t^{(m)} - \widetilde{\mu}_{\mathbf{z},t}\big)\big(\widetilde{\zeta}_t^{(m)} - \widetilde{\mu}_{\mathbf{z},t}\big)^{\mathrm{T}} + \sigma_w^2 \mathbf{I}_N, \tag{2.13}$$

$$\widetilde{\mathbf{C}}_{\mathbf{q},\mathbf{z},t} = \frac{1}{M} \sum_{m=1}^{M} \big(\widetilde{\chi}_t^{(m)} - \widetilde{\mu}_{\mathbf{q},t}\big)\big(\widetilde{\zeta}_t^{(m)} - \widetilde{\mu}_{\mathbf{z},t}\big)^{\mathrm{T}}. \tag{2.14}$$

- Estimate:

1. Compute filter gain $\mathbf{K}_t$:

$$\mathbf{K}_t = \widetilde{\mathbf{C}}_{\mathbf{q},\mathbf{z},t} \widetilde{\mathbf{C}}_{\mathbf{z},t}^{-1}. \tag{2.15}$$

2. Compute posterior mean $\mu_{\mathbf{q},t}$ and covariance $\mathbf{C}_{\mathbf{q},t}$ conditioned on measurement $\mathbf{z}_t$:

$$\mu_{\mathbf{q},t} = \widetilde{\mu}_{\mathbf{q},t} + \mathbf{K}_t\big(\mathbf{z}_t - \widetilde{\mu}_{\mathbf{z},t}\big), \tag{2.16}$$

$$\mathbf{C}_{\mathbf{q},t} = \widetilde{\mathbf{C}}_{\mathbf{q},t} - \mathbf{K}_t \widetilde{\mathbf{C}}_{\mathbf{z},t} \mathbf{K}_t^{\mathrm{T}}. \tag{2.17}$$

2.3 Control Strategy

The controller incorporates the uncertainties in measurement and localization by the following iterative procedure. At time step t, the agent performs distance measurements to the anchor nodes in the workspace $\mathscr{W}$. The measurements are stacked in a measurement vector $\mathbf{z}_t$. The estimator uses $\mathbf{z}_t$, $\mathbf{u}_t$ and $\mathbf{q}_{t-1}$ to estimate the agents current position $\mathbf{q}_t$, which is used to produce a control vector candidate $\widetilde{\mathbf{u}}_{t+1,1}$. This control vector candidate is used to predictive density of the agent at the next time step:

$$p(\mathbf{q}_{t+1}|\mathbf{z}_{1:t};\mathbf{u}_{1:t},\widetilde{\mathbf{u}}_{t+1}) = \int_{\mathbb{R}^2} p(\mathbf{q}_{t+1}|\mathbf{q}_t;\mathbf{u}_t,\widetilde{\mathbf{u}}_{t+1,1})\; p(\mathbf{q}_t|\mathbf{z}_{1:t};\mathbf{u}_{1:t})\; d\mathbf{q}_t. \tag{2.18}$$

The predicted position information $p(\mathbf{q}_{t+1}|\mathbf{z}_{1:t};\mathbf{u}_{1:t},\widetilde{\mathbf{u}}_{t+1})$ is used to calculate the probability of collision P_{coll} with obstacles, which is then compared to a threshold P_{th}. If $P_{\mathrm{coll}} \leq P_{\mathrm{th}}$, the candidate control vector is accepted, i.e.,

$$\mathbf{u}_{t+1} = \widetilde{\mathbf{u}}_{t+1,1}. \tag{2.19}$$

Otherwise, the potential field strength and its effective range are increased to produce a larger repulsive force from the obstacles, and a new control vector candidate $\widetilde{\mathbf{u}}_{t+1,2}$ is generated. The iterative procedure continues until $P_{\text{coll}} \leq P_{\text{th}}$ or the maximum number of iterations $k_{\max}$ is reached. A summary of the iterative procedure is shown in Algorithm 1.

Algorithm 1: Control algorithm

Data: $\mu_{\mathbf{q},t}$, $\mathbf{C}_{\mathbf{q},t}$, $\mathbf{q}_{\text{goal}}$, $\mathbf{O}^{(j)}$

$k = 0$

while $k < k_{max}$ **do**

Calculate $\phi(\mathbf{q}_t) = \phi_{\text{a}}(\mathbf{q}_t, \mathbf{q}_{\text{goal}}) + \phi_{\text{r}}(\mathbf{q}_t, \mathbf{q}_{\text{goal}}, \mathbf{O}^{(j)})$ (See Chapter 3)

$$\tilde{\mathbf{u}}_{t+1,k} = \begin{cases} v_{\max} \frac{-\nabla\phi(\mathbf{q}_t)}{\|\nabla\phi(\mathbf{q}_t)\|}, & \|\nabla\phi(\mathbf{q}_t)\| \neq 0, \\ 0, & \|\nabla\phi(\mathbf{q}_t)\| = 0. \end{cases}$$

$p(\mathbf{q}_{t+1}|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}) = \int p(\mathbf{q}_{t+1}|\mathbf{q}_t; \tilde{\mathbf{u}}_{t+1}) \; p(\mathbf{q}_t|\mathbf{z}_{1:t}; \mathbf{u}_{1:t}) \; d\mathbf{q}_t$

Calculate $P_{\text{coll},k}\big(p(\mathbf{q}_{t+1}|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}), \mathbf{O}^{(j)}\big)$ (See Chapter 4)

if $P_{coll,k} < P_{th}$ **then**

$\mathbf{u}_t = \tilde{\mathbf{u}}_{t,k}$

$\mathbf{q}_t = \mathbf{q}_{t-1} + \mathbf{u}_t \Delta_t + \mathbf{v}_t$

else

Adapt $\phi_{\text{r}}(\mathbf{q}_t, \mathbf{q}_{\text{goal}}, \mathbf{O}^{(j)})$ (See Chapter 3)

end

end

2.4 Calculation of Control Vector

The controller is responsible for calculating the control input that will drive the mobile agent to its next position while keeping the probability of collision under a given threshold. The kinematic model of the mobile agent is described by the following equation [35]:

$$\dot{\mathbf{q}} = \mathbf{u}. \tag{2.20}$$

Since the only kinematic constraint is in the form of a velocity constraint, the overall system is a first-order system, and the negative gradient of the potential field can be interpreted as a velocity vector field. Therefore, the control vector is proportional to the negative gradient of the potential field:

$$\mathbf{u}_t \propto -\nabla\phi(\mathbf{q}_t). \tag{2.21}$$

The negative gradient of the potential field $\phi(\mathbf{q}_t)$ might produce control vectors of unrealistically high velocities. Therefore the potential field must be modified to incorporate the maximum velocity $v_{\max}$ of the mobile agent. For this reason, in this thesis, we always move the agent with $v_{\max}$ at all time steps, and let the negative gradient of the potential field dictate the direction of motion. Therefore, the modified control vector is given by:

$$\mathbf{u}_t = \begin{cases} v_{\max} \frac{-\nabla\phi(\mathbf{q}_t)}{\|\nabla\phi(\mathbf{q}_t)\|}, & \|\nabla\phi(\mathbf{q}_t)\| \neq 0, \\ 0, & \|\nabla\phi(\mathbf{q}_t)\| = 0. \end{cases} \tag{2.22}$$

Chapter 3

Artificial Potential Fields

3.1 Background

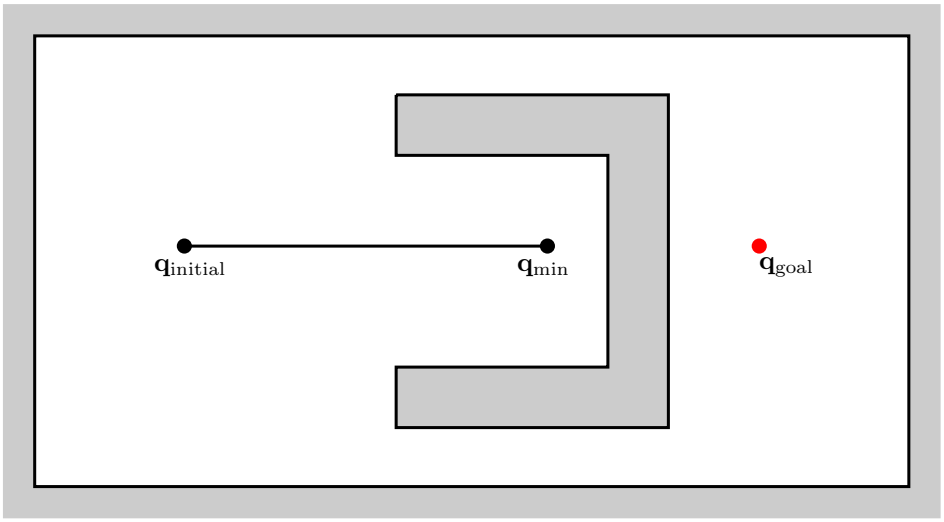
For most problems, the configuration space is a high dimensional manifold as shown in Section 1.3. Therefore, it is not possible to construct an exact representation of the free configuration space. For such problems, an alternative is to develop a search algorithm that will explore the configuration space to find a collision-free path from the agent's initial configuration to the goal configuration. One such search algorithm is the gradient descent method in an APF, first proposed in [17].

APFs were initially proposed for multi-joint robot articulators, however due to the simplicity of method, it has also found use in path planning problems for mobile agents [1]. The basic idea behind the APFs is to construct an attractive field around the goal configuration and repulsive fields around obstacles. The attractive field moves the agent to the goal configuration while the repulsive fields push the agent away from obstacle. The combination of the fields directs the robot to the goal configuration.

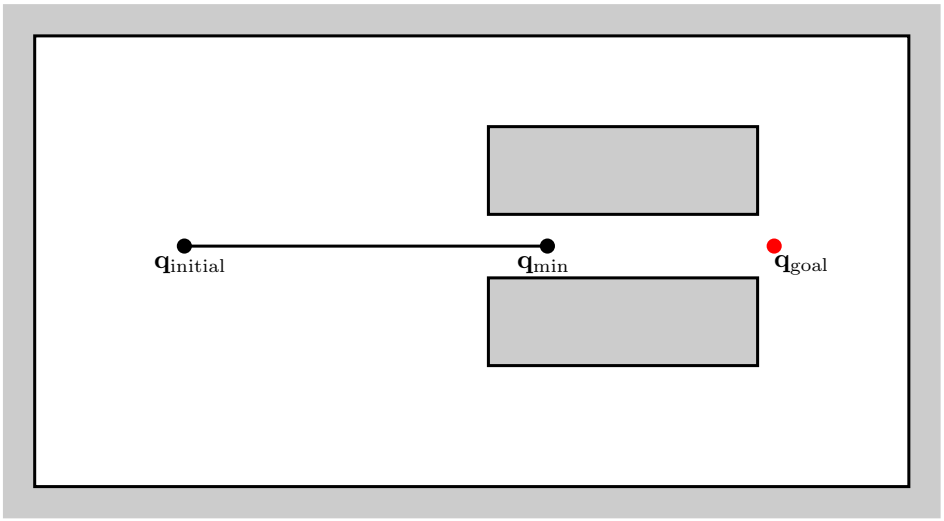
3.2 Local Minima

Unfortunately, APF methods suffer from the problem of local minima, which are points where the potential field obtains it's lowest value within a given neighborhood of the point. As a result, a mobile agent might get stuck at a local minimum despite not reaching the goal configuration. Figure 3.1 shows two scenarios which produce local minima in the APF of the configuration space $\mathscr{C}$. Over the years, many authors have attempted to solve the problem of getting stuck in local minima. In the context of APFs, two classes of approaches exist that attempt to address the problem: the first is to modify or augment the gradient descent based search method and the second is the modify the APFs to remove the local minima.

One of the earliest and simplest solutions to the local minima problem was presented in [36], named the *wave-front planner*. The wave-front planner expands a wave-front created at the goal,



(a) Concave obstacle.



(b) Narrow path.

Fig. 3.1 Two scenarios that produce a local minimum $\mathbf{q}_{\text{min}}$.

and terminates when a point on the wave-front reaches the agent. More sophisticated variations of the wave-front planner, called the *fast-marching methods*, have been recently proposed [37]. While these planners solve the local minima, they have several disadvantages; they require prior knowledge of the environment which renders them unsuitable for navigation in unknown or partially known environments with dynamic obstacles, and they are only applicable to environments represented by grids.

In [38], the idea of constructing the APF with the global minimum at the goal configuration and no other local minima was introduced. These new APFs were called *navigation functions* (NF).

While NF avoid the problem of local minima, they introduce other problems:

1. NF require tuning to remove local minima, which necessitates a precomputation stage.
2. NF require knowledge of the entire workspace $\mathscr{W}$
3. NF are only applicable to so called sphere-worlds and star-worlds [12]

The first two points make NF unsuitable for mobile agents operating in real-time in unknown environments or environments with dynamic obstacles. Indeed, when a new obstacle is detected or a previously known obstacle moves to a new position, the NF has to be tuned again to remove any local minima. The last point limits the applicability of NF to a small subset of all possible workspaces. Recently, an algorithm for automatically tuning NF for sphere worlds was introduced in [39].

A probabilistic method that attempts to solve the local minima problem is the *random-path planner* (RPP) proposed in [36, 40]. This algorithm follows the gradient of the potential field and if it gets stuck in a local minimum, it initiates a random walk of finite number of steps in an attempt to get out. If the random walk moves the agent out of the local minimum, it starts descending the gradient of the potential field again; otherwise another random walk is executed. The probability of escaping certain local minima might be low, which results in a long runtime.

In [41] the concept of virtual obstacles to escape from local minima was proposed. The idea is to iteratively increase the potential at a local minimum by placing virtual obstacles with repulsive fields around them until the local minimum disappears. In the same vein, a method based on virtual water flow was proposed in [18].

Starting in the early 2000s, path planning methods combining APF approaches with evolutionary algorithms received considerable interest. Evolutionary algorithms are population-based metaheuristic optimization-algorithms based on principles that can be found in biological evolution [42, 43]. The *evolutionary APF* approach which uses a *genetic algorithm* to derive optimum potential fields for a given scenario was proposed in [44]. Furthermore, a hybrid algorithm which combines APF with *ant colony optimization* (ACO) for path planning in dynamic environments was proposed in [45]. ACO algorithms mimic the behavior of ants that randomly explore the environment, and mark paths that lead to the goal destination with pheromones. Pheromone evaporation ensures that only shorter paths that are reinforced often remain alive. The planner uses ACO for the global path planning stage based on static environment information, and the APF is used as the local planner due to its reactive nature to avoid dynamic obstacles. In addition, in [46] a *particle swarm optimization* (PSO) method based on APF approach was proposed. PSO algorithm searches the workspace for a path using swarms of particles where the particles located at a lower potential in the field are deemed more fit.

In general, pure APF methods do not find the shortest path to the goal. Recently, in [47] an APF path planner was augmented with regression search. The resulting planner provides near-optimal paths at the expense of increased computation time.

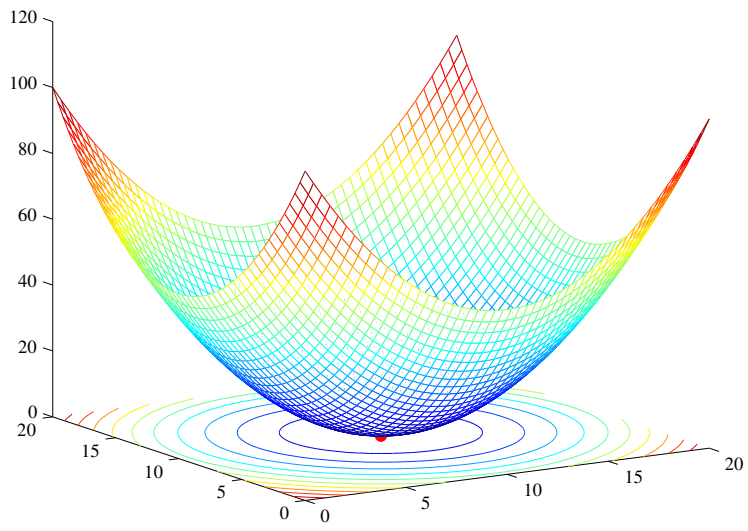


Fig. 3.2 Visualization of the attractive field.

3.3 Attractive Field

The attractive potential is a monotonically increasing function with the minimum located at the goal configuration. Furthermore the potential function is required to be continuously differentiable to avoid stability problems with the numerical implementation. The simplest potential function that fulfills these criteria is the parabolic well potential that grows quadratically with distance and is defined as:

$$\phi_{\mathrm{a}}(\mathbf{q}) = \frac{1}{2}\eta_{\mathrm{a}}\|\mathbf{q}-\mathbf{q}_{\mathrm{goal}}\|^2, \tag{3.1}$$

where $\mathbf{q}$ is the agent position, $\mathbf{q}_{\mathrm{goal}}$ is the goal position, η_{a} is a design parameter used to scale the magnitude of the field. The negative gradient of the attractive field which equals the attractive force is given by:

$$-\nabla\phi_{\mathrm{a}}(\mathbf{q}) = -\eta_{\mathrm{a}}(\mathbf{q}-\mathbf{q}_{\mathrm{goal}}). \tag{3.2}$$

Figure 3.2 shows the attractive field over a 20×20 region with $\mathbf{q}_{\mathrm{goal}} = (10,10)^{\mathrm{T}}$.

3.4 Repulsive Field

To construct the repulsive field, we borrow from electrostatics. The electrostatic field around a charged line segment of length $2L$ [48] is given by

$$\phi(x,y) = \frac{\lambda}{4\pi\varepsilon_0}\int_{-L}^{L}\frac{dx'}{\sqrt{(x'-x)^2+y^2}}. \tag{3.3}$$

Figure 3.3 depicts the underlying geometry of the integral, where (x,y) is the point at which the potential field strength is calculated, r_1 and r_2 are the distance from the endpoints of the line segment to $\mathbf{q}$. Evaluating the integral results in the following equation:

$$\phi(x,y) = \frac{\lambda}{4\pi\varepsilon_0} \ln\left(\frac{\sqrt{(L-x)^2+y^2}+L-x}{\sqrt{(L+x)^2+y^2}-L-x}\right). \tag{3.4}$$

The square root terms in the equation are distances r_1 and r_2. This motivates further simplifications to the equation [48]. With $r_1 = \left\|[-L,\ 0]^{\mathrm{T}} - [x,\ y]^{\mathrm{T}}\right\|$ and $r_2 = \left\|[L,\ 0]^{\mathrm{T}} - [x,\ y]^{\mathrm{T}}\right\|$, define

$$\xi = \frac{1}{2}(r_2 + r_1), \tag{3.5}$$

$$\nu = \frac{1}{2}(r_2 - r_1). \tag{3.6}$$

Subsuming the term before the integral into a parameter w, the equation can be written as:

$$\phi(x,y) \ = w \ln\left(\frac{\xi+v+L-x}{\xi-v-L-x}\right). \tag{3.7}$$

Then, using the identity $\xi\nu = -xL$, the numerator and denominator of the fraction inside the natural log can be factored as follows:

$$\phi(x,y) \ = w \ln\left(\frac{(\xi+L)(1+\nu/L)}{(\xi-L)(1+\nu/L)}\right). \tag{3.8}$$

Canceling out similar terms results in:

$$\phi(x,y) \ = w \ln\left(\frac{\xi+L}{\xi-L}\right). \tag{3.9}$$

Using Equation (3.9), the repulsive field at $\mathbf{q} = \begin{bmatrix} x, & y \end{bmatrix}^{\mathrm{T}}$ for a 1D obstacle is defined as:

$$\phi_{\mathrm{r}}(\mathbf{q};R,w) = \begin{cases} w\ \eta_{\mathrm{r}}(\mathbf{q}) \ln\left(\frac{\xi(\mathbf{q})+L}{\xi(\mathbf{q})-L}\right), & \text{if } \xi(\mathbf{q}) \le L+R, \\ 0, & \text{otherwise}, \end{cases} \tag{3.10}$$

where w and R are treated as tuning parameters. Increasing w results in a larger repulsive force due to the repulsive field, and increasing R makes the repulsive field effective over a larger area since the boundary of region defined by the ellipse $r_1 + r_2 = 2R + 2L$ grows as well. Figure 3.4 shows the repulsive field around a 1D obstacle with endpoints at $[5\ \ 5]^{\mathrm{T}}$ and $[5\ \ 15]^{\mathrm{T}}$, and $\mathbf{q}_{\mathrm{goal}} = [10\ \ 10]^{\mathrm{T}}$. The effects of varying w and R can be seen in Figure 3.5.

The term $\eta_r(\mathbf{q})$ is used to prevent the repulsive field from diminishing in strength in comparison

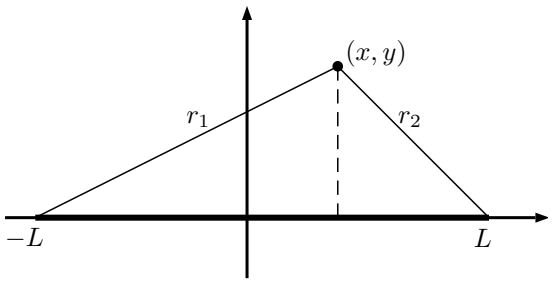


Fig. 3.3 Geometric underlying the integral in Equation (3.3).

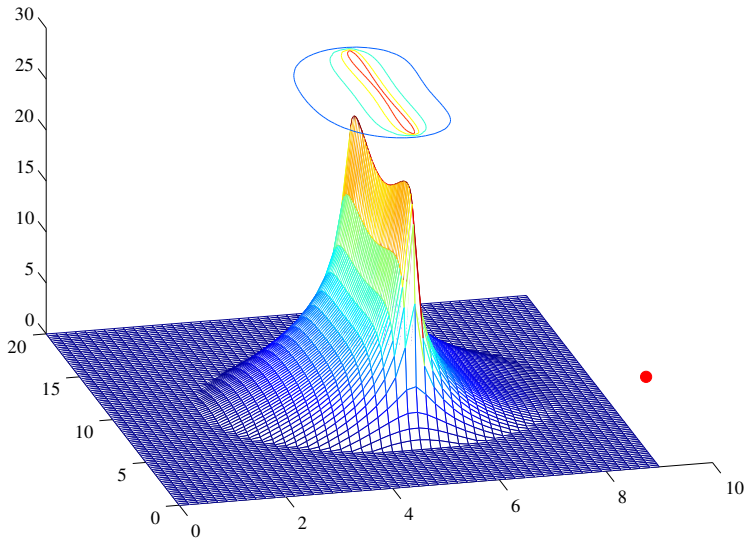


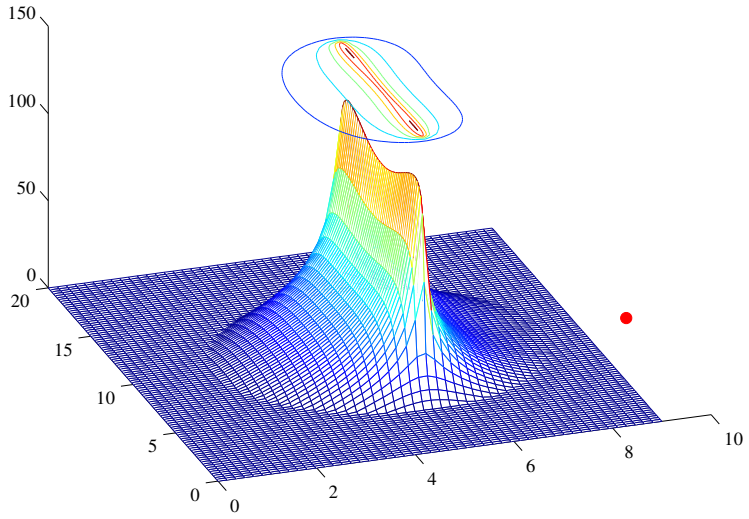
Fig. 3.4 Repulsive field around a 1D obstacle with $w = 1$ and $R = 1$.

to the attractive field when the agent is far from the goal. It is defined as:

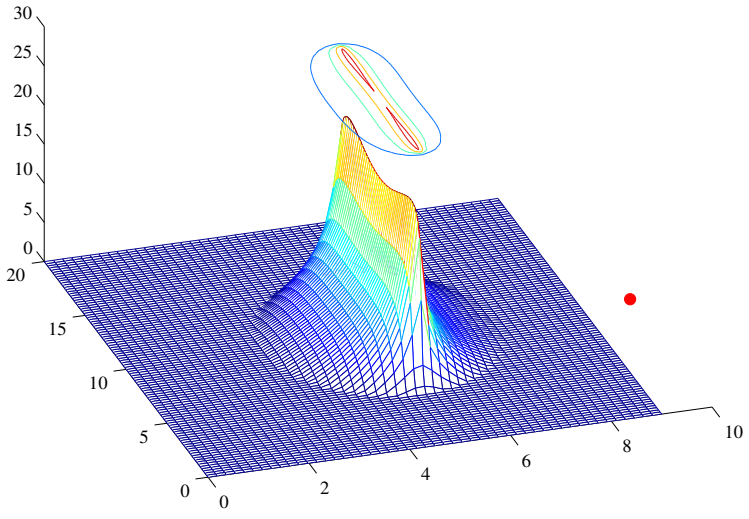
$$\eta_{\mathrm{r}}(\mathbf{q}) = \eta_{\mathrm{a}} \|\mathbf{q} - \mathbf{q}_{\mathrm{goal}}\|.$$

The repulsive force is obtained as the negative gradient of the repulsive field:

$$\begin{aligned}
\mathbf{f}_{\mathrm{r}}(\mathbf{q}; R, w) &= -\nabla \phi_{\mathrm{r}}(\mathbf{q}; R, w) \\
&= -\nabla w \eta_{\mathrm{r}}(\mathbf{q}) \ln\left(\frac{\xi(\mathbf{q}) + L}{\xi(\mathbf{q}) - L}\right) \qquad (3.11) \\
&= -w\eta_{\mathrm{a}} \left[\ln\left(\frac{\xi(\mathbf{q}) + L}{\xi(\mathbf{q}) - L}\right) + \frac{1}{2}\|\mathbf{q} - \mathbf{q}_{\mathrm{goal}}\| \left(\frac{1}{\xi(\mathbf{q}) + L} - \frac{1}{\xi(\mathbf{q}) - L}\right) \left(\frac{\mathbf{q} - \mathbf{q}_{-L}}{\|\mathbf{q} - \mathbf{q}_{-L}\|} + \frac{\mathbf{q} - \mathbf{q}_{L}}{\|\mathbf{q} - \mathbf{q}_{L}\|}\right)\right].
\end{aligned}$$



(a) Repulsive field with $w = 5$.



(b) Repulsive field with $R = 0.5$.

Fig. 3.5 Effects of varying w and R on the repulsive field $\phi_r(\mathbf{q}; R, w)$.

3.5 Obstacles and Boundaries

In this thesis, obstacles in the configuration space $\mathscr{C}$ and boundaries of $\mathscr{C}$ are represented by orientable surfaces. For simplicity, only polygons are considered, but no further restrictions, such as convexity, are enforced. The obstacles are represented by a $2 \times P$ matrix

$$\mathbf{O}^{(j)} = \left[\mathbf{q}_1^{(j)}\ \mathbf{q}_2^{(j)} \cdots\ \mathbf{q}_P^{(j)}\right], \tag{3.12}$$

where j is the index of the obstacle, and $\mathbf{q}_p^{(j)}$ is the position vector of the p^{th} vertex of the obstacle. Following the *right-handedness* convention, an obstacle is filled "inside" if its boundary is oriented *counter-clockwise*, and filled "outside" if the boundary is oriented *clockwise*. Figure 3.6 shows a configuration space with obstacles represented by orientable polygons.

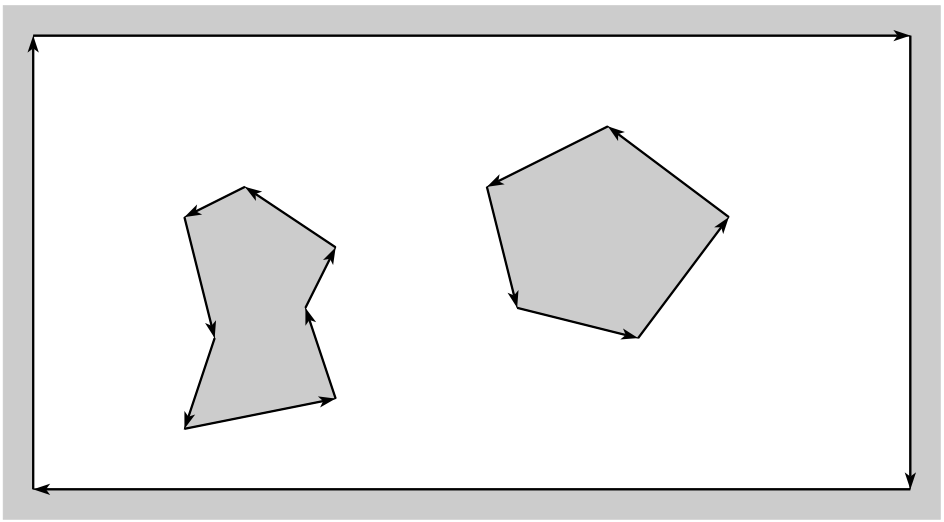


Fig. 3.6 A configuration space with orientable surfaces.

Chapter 4

Probability of Collision

The probability of collision $P_{\text{collision}}$ with an obstacle is defined as the probability that the mobile agent is located within the region defined by the boundary of an obstacle. Let $\mathscr{R}^{(j)}$ denote the region corresponding to the interior of an obstacle $\mathbf{O}^{(j)}$ with vertices $\mathbf{q}_1^{(j)}\ \mathbf{q}_2^{(j)}\ldots\ \mathbf{q}_P^{(j)}$. The probability of collision with the obstacle $\mathbf{O}^{(j)}$ at time step $t+1$ is obtained by integrating the predictive posterior density calculated at time step t over the region $\mathscr{R}^{(j)}$:

$$P_{\text{collision}}^{(j)} = P\big(\mathbf{q}_{t+1} \in \mathscr{R}^{(j)} \big| \mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}\big) = \int\limits_{\mathscr{R}^{(j)}} p\big(\mathbf{q}_{t+1} \big| \mathbf{z}_{1:t}; \mathbf{u}_{1:t}, \tilde{\mathbf{u}}_{t+1,k}\big)\, d\mathbf{q}_{t+1}. \tag{4.1}$$

The above integral has no general closed form solution and often numerical methods are used to compute the integral [49, 50]. To avoid numerical integration over non-trivial regions, a simple approximation is proposed to calculate the collision of probability. The approximation is obtained in three steps:

1. A decorrelation transformation is performed using the predicted covariance matrix $\widetilde{\mathbf{C}}_{\mathbf{q},t+1}$.
2. An upper bound on the probability of collision with each individual edge of polygonal obstacle is calculated.
3. The maximum of the resulting probabilities is taken.

The first step results in a transformed position vector $\tilde{\mathbf{q}}'_{t+1,k}$ whose components are independent. In the second step, rectangular regions behind obstacle edges are identified, and the predictive density is integrated over these regions.

4.1 Probability of Collision

Given obstacle j with a matrix $\mathbf{O}^{(j)}$ containing column vectors of vertex positions, first step is to define the rectangular regions behind the edges.

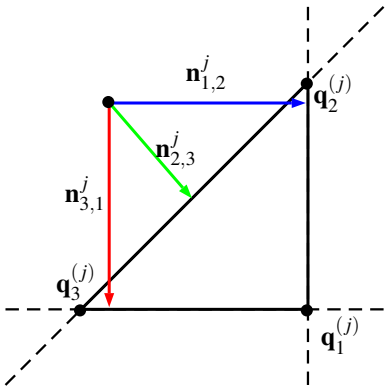


Fig. 4.1 Shortest distances to obstacle edges.

Let $\mathbf{O}^{(j)}_{l,l+1} = \begin{bmatrix}\mathbf{q}^{(j)}_l & \mathbf{q}^{(j)}_{l+1}\end{bmatrix}$ be the l^{th} edge of the obstacle and $\widetilde{\mathbf{q}}_{t+1}$ is the predicted position of the agent at time step t; define vectors $\mathbf{p}^{(j)}_{l,l+1}$ and $\mathbf{n}^{(j)}_{l,l+1}$ parallel and orthogonal to the obstacle as follows:

$$\mathbf{p}^{(j)}_{l,l+1} = \mathbf{q}^{(j)}_{l+1} - \mathbf{q}^{(j)}_l, \tag{4.2}$$

$$\mathbf{n}^{(j)}_{l,l+1} = (\mathbf{q}^{(j)}_l - \widetilde{\mathbf{q}}_{t+1,k}) - \left((\mathbf{q}^{(j)}_l - \widetilde{\mathbf{q}}_{t+1,k})^{\mathrm{T}} \hat{\mathbf{p}}^j_{l,l+1}\right) \hat{\mathbf{p}}^j_{l,l+1}, \tag{4.3}$$

where $\hat{\mathbf{p}}^j_{l,l+1} = \frac{\mathbf{p}^j_{l,l+1}}{\|\mathbf{p}^j_{l,l+1}\|}$ is the unit vector along the l^{th} edge of the obstacle. Figure 4.1 shows the geometric interpretation of the vectors $\mathbf{n}^j_{l,l+1}$ that correspond to the 3 edges of a triangular obstacle. The region $\mathscr{R}^{(j)}_{l,l+1}$ behind the l^{th} edge of the obstacle is given by:

$$\mathscr{R}^{(j)}_{l,l+1} = \left\{\mathbf{q}^{(j)}_l + u\,\mathbf{p}^{(j)}_{l,l+1} + v\,\mathbf{n}^{(j)}_{l,l+1} \;:\; u \in [0,1],\; v \in [0,\infty)\right\}. \tag{4.4}$$

Figure 4.2 depicts the region of integration for calculating the probability of collision with the edge $\mathbf{O}^{(j)}_{1,2}$ of obstacle j. The probability of collision with the l^{th} edge of the obstacle j is approximated as:

$$P^{(j)}_l = P\left(\mathbf{q}_{t+1} \in \mathscr{R}^{(j)}_{l,l+1} \middle| \mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}\right) = \int\limits_{\mathscr{R}^{(j)}_{l,l+1}} p\left(\mathbf{q}_{t+1} \middle| \mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}\right) d\mathbf{q}_{t+1}, \tag{4.5}$$

where the predictive density incorporating the candidate control vector $\tilde{\mathbf{u}}_{t,k}$ is used in the integration.

4.2 Decorrelation Transform

Due to the cross-correlation of the x_{t+1} and y_{t+1} components of the position vector $\mathbf{q}_{t+1}$, the integral is still non-trivial. Therefore, a decorrelation transformation is performed to simplify the calculation

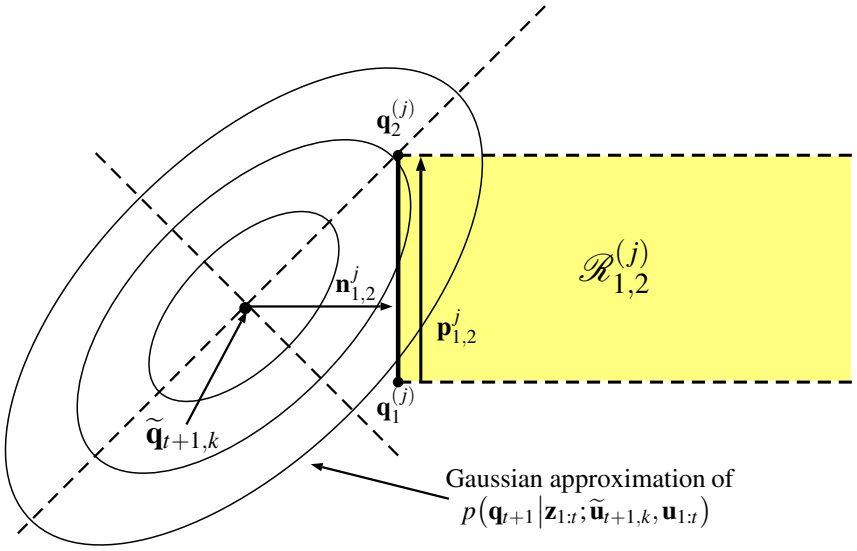


Fig. 4.2 Probability of collision with an edge.

of the integral. First, an eigenvalue decomposition is performed on the predicted covariance matrix $\widetilde{\mathbf{C}}_{\mathbf{q},t+1}$:

$$\widetilde{\mathbf{C}}_{\mathbf{q},t+1} = \mathbf{V}\Lambda\mathbf{V}^T = \begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}\begin{bmatrix}\lambda_1 & 0\\ 0 & \lambda_2\end{bmatrix}\begin{bmatrix}\mathbf{v}_1 & \mathbf{v}_2\end{bmatrix}^T, \tag{4.6}$$

with $\lambda_1 > \lambda_2$. Subsequently, a decorrelation transformation is performed on the predicted position random vector $\mathbf{q}_{t+1}$ that results from the candidate control vector $\tilde{\mathbf{u}}_{t+1,k}$:

$$\mathbf{q}'_{t+1} = \mathbf{V}^T\mathbf{q}_{t+1} = \begin{bmatrix}x'_{t+1} & y'_{t+1}\end{bmatrix}^T. \tag{4.7}$$

After the transformation, the direction of largest variance is along x'_{t+1} axis and smallest variance along y'_{t+1} axis. Figure 4.3 shows the effect of the transformation on the workspace of the agent. Since the transformed variables x'_{t+1} and y'_{t+1} are independent, the probability of collision in x'_{t+1} and y'_{t+1} directions can be computed separately.

4.3 Approximate Probability of Collision

The decorrelation transformation simplifies the integration considerably, however, integrating over non-rectangular remains a non-trivial problem due to the co-dependence of x'_{t+1} and y'_{t+1} in the definition of the region. This co-dependence manifests itself in the limits of the double integral. By replacing $\mathscr{R}_{1,2}^{(j)\prime}$ with $\mathscr{R}_{1,2}^{(j)}$ as shown in Figure 4.4, an approximation of the probability of collision

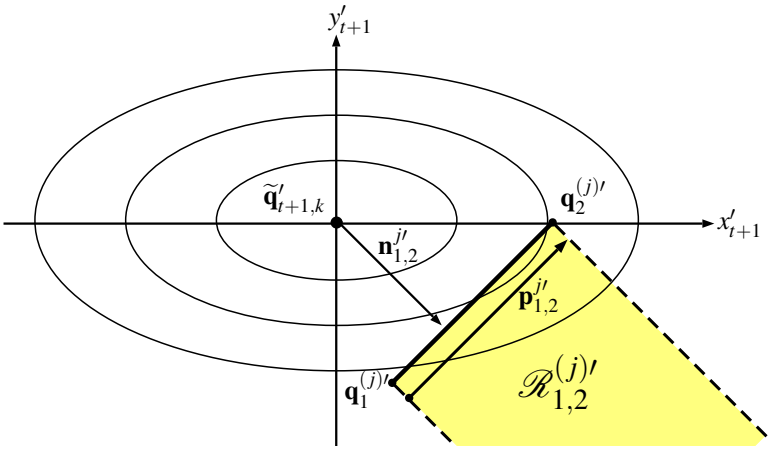


Fig. 4.3 Effect of decorrelation transformation.

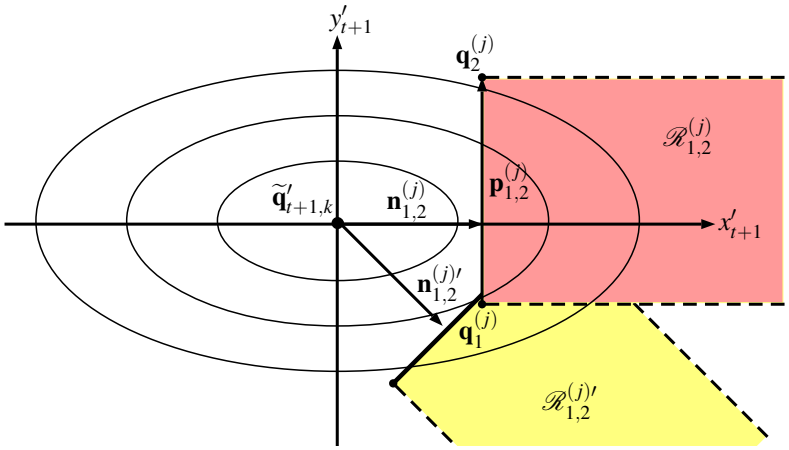


Fig. 4.4 Replacing the transformed region with the original region.

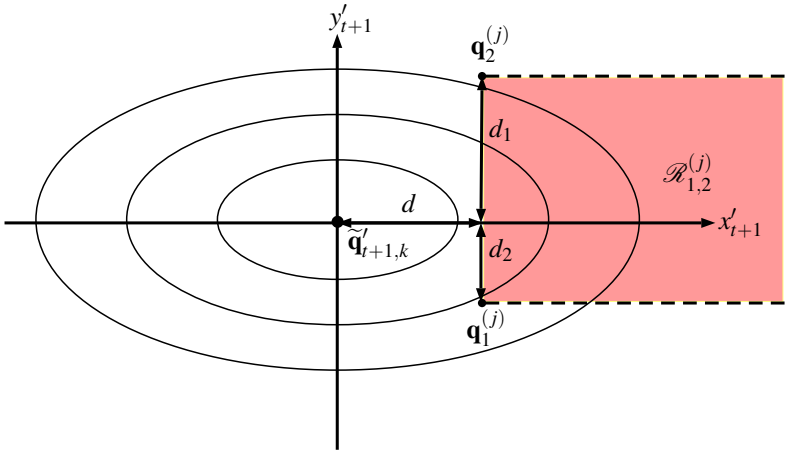


Fig. 4.5 Shortest distances to polygon edges.

$P_1^{(j)}$ with obstacle edge $\mathbf{O}_{1,2}^{(j)}$ is obtained as:

$$P_1^{(j)} = P(\mathbf{q}_{t+1} \in \mathscr{R}_{1,2}^{(j)} \,|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}) = P(\mathbf{q}'_{t+1} \in \mathscr{R}_{1,2}^{(j)\prime} \,|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}) \tag{4.8}$$

$$\leq P(\mathbf{q}'_{t+1} \in \mathscr{R}_{1,2}^{(j)} \,|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}) := \tilde{P}_1^{(j)}. \tag{4.9}$$

The limits of the double integral are now scalar values, and since x'_{t+1} and y'_{t+1} are independent, the integral can be split into a product of two integrals:

$$\tilde{P}_1^{(j)} = P(\mathbf{q}'_{t+1} \in \mathscr{R}_{1,2}^{(j)} \mid \mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}) \tag{4.10}$$

$$= P(x'_{t+1} > d \mid \mathbf{z}_{1:t-1}; \tilde{\mathbf{u}}_{t,k}, \mathbf{u}_{1:t-1})\, P(-d_2 < y'_{t+1} < d_1 \mid \mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t}), \tag{4.11}$$

where $d = \|\mathbf{n}_{1,2}^{(j)}\|$, $d_1 = \|\mathbf{q}_2^{(j)} - (\tilde{\mathbf{q}}'_{t+1,k} + \mathbf{n}_{1,2}^{(j)})\|$ and $d_2 = \|\mathbf{q}_1^{(j)} - (\tilde{\mathbf{q}}'_{t+1,k} + \mathbf{n}_{1,2}^{(j)})\|$. The posterior probability $p(\mathbf{q}_{t+1}|\mathbf{z}_{1:t}; \tilde{\mathbf{u}}_{t+1,k}, \mathbf{u}_{1:t})$ is approximated by a bivariate Gaussian distribution, therefore the integrals can be expressed in terms of Q-functions as follows:

$$P(x'_{t+1} > d) \leq Q\Big(\frac{d}{\sqrt{\lambda_1}}\Big) \tag{4.12}$$

and

$$P(-d_2 < y'_{t+1} < d_1) \leq 1 - Q\Big(\frac{d_1}{\sqrt{\lambda_2}}\Big) - Q\Big(\frac{d_2}{\sqrt{\lambda_2}}\Big). \tag{4.13}$$

Therefore, the upper bound on the probability of collision with obstacle edge $\mathbf{O}_{1,2}^{(j)}$ is given by

$$\tilde{P}_1^{(j)} = Q\Big(\frac{d}{\sqrt{\lambda_1}}\Big)\left[1 - Q\Big(\frac{d_1}{\sqrt{\lambda_2}}\Big) - Q\Big(\frac{d_2}{\sqrt{\lambda_2}}\Big)\right] \tag{4.14}$$

and the approximation of the probability of collision with obstacle j is given by

$$\tilde{P}_{\text{collision}}^{(j)} := \max_l \tilde{P}_l^{(j)}. \tag{4.15}$$

If there are several obstacles j, the control vector $\mathbf{u_{t+1}}$ is calculated with regard to the largest collision probability

$$\tilde{P}_{\text{collision}} := \max_j \tilde{P}_{\text{collision}}^{(j)}. \tag{4.16}$$

$\tilde{P}_{\text{collision}}$ corresponds to the obstacle that poses the most danger to the agent. Indeed, obstacles that do not pose any danger to the mobile agent should not influence the process of generating a control vector.

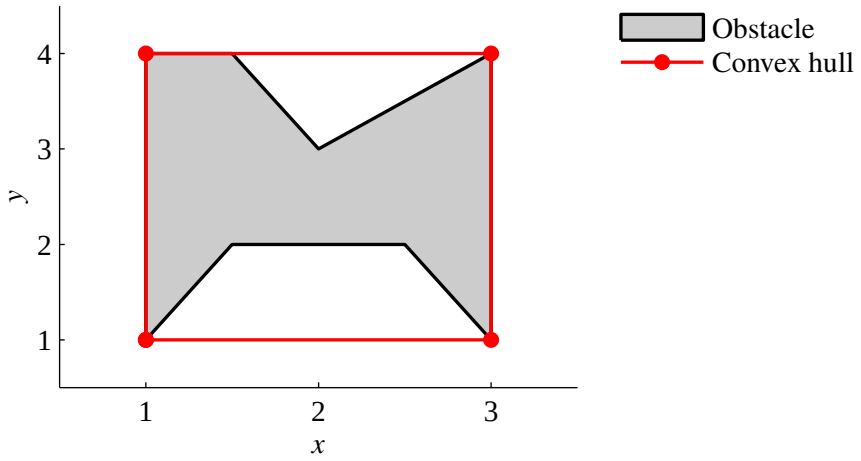


Fig. 4.6 Convex hull of a non-convex obstacle

4.4 Non-Convex Obstacles

For approximating the probability of collision with non-convex obstacles, a necessary first step is to compute the convex hull of the obstacle, and consider the edges of the convex shape. Non-convex obstacles might produce local minima in $\mathscr{C}$; calculating the probability of collision with respect to the convex hull of the obstacles assigns a high probability of collision to these regions that might possibly contain local minima without denying access to such regions. Figure 4.6 shows the convex hull of a non-convex obstacle.

4.5 Numerical Validation of the Approximation

In this section, proposed approximation to the probability of collision is validated with a simulation. The mobile agent is moved on a predetermined path as seen in Figure 4.7 and the approximate and exact probabilities of collision are calculated. To simplify the computation of the exact probability of collision, a rectangular obstacle is used. The exact probability of collision is obtained by numerically integrating the probability distribution of the agent position over the rectangular region corresponding to the obstacle.

The path of the agent is given by the curve $\mathbf{c}(t)$, $t \in [0,1]$ where $\mathbf{c}(0) = [1 \quad 5]^{\mathrm{T}}$ is the initial position and $\mathbf{c}(1) = [8 \quad 24]^{\mathrm{T}}$ is the final position of the agent. The position random vector of the agent is assumed to be Gaussian with $\mathbf{q}_t = \mathscr{N}\left(\mathbf{c}(t), \mathbf{C}_{\mathbf{q},t}\right)$. The covariance matrix is defined as:

$$\mathbf{C}_{\mathbf{q},t} = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}, \quad \rho \in [0,1). \tag{4.17}$$

The parameter ρ determines the correlation between x_t and y_t.

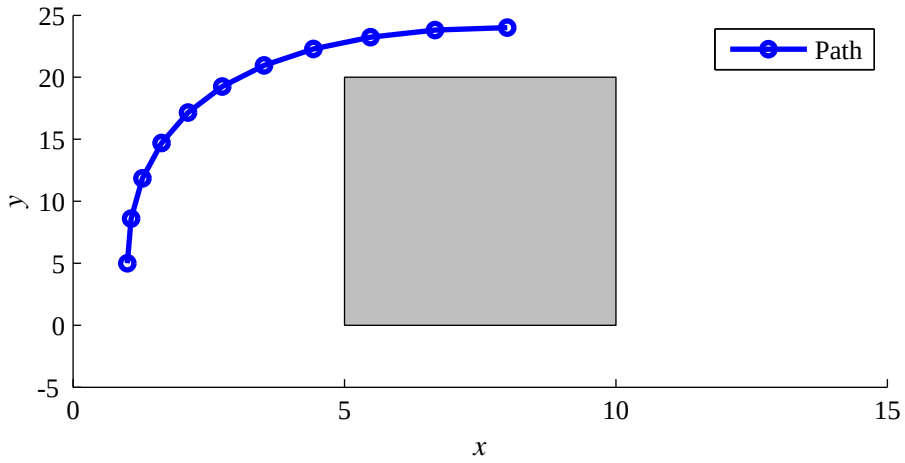


Fig. 4.7 Mobile agent moves on a predetermined path around a rectangular obstacle.

In the scenario, given in Figure 4.7, the obstacle is defined by the matrix:

$$\mathbf{O} = \begin{bmatrix} 5 & 10 & 10 & 5 \\ 0 & 0 & 10 & 10 \end{bmatrix}, \tag{4.18}$$

where the vertex at $(5,0)$ is considered the starting position. For agent moving along curve $\mathbf{c}(t)$, the edges that pose the greatest risk are:

$$\mathbf{O}_{4,1} = \begin{bmatrix} 5 & 5 \\ 10 & 0 \end{bmatrix} \tag{4.19}$$

and

$$\mathbf{O}_{3,4} = \begin{bmatrix} 10 & 5 \\ 10 & 10 \end{bmatrix}, \tag{4.20}$$

since the edges $\mathbf{O}_{1,2}$ and $\mathbf{O}_{2,3}$ are too far away from the agent position to contribute significantly to the probability of collision. Figure 4.8 shows that the approximate is almost exact when x_t and y_t are independent, i.e. $\rho = 0$. This is expected since the eigenvalues of $\mathbf{C}_{\mathbf{q},t}$ are equal, and in this scenario, the approximation is equivalent to integrating the probability distribution over the region with the exception that the regions extend infinitely behind the edges. It is important to note that this provides a higher probability of collision.

In Figure 4.9 and Figure 4.10, the approximation provides a visibly larger probability of collision than the exact value. The transition point between $t = 0.6$ and $t = 0.7$ corresponds to the agent rounding the corner of the obstacle. At this stage, the edge $\mathbf{O}_{3,4}$ poses a greater risk to agent than the edge $\mathbf{O}_{4,1}$. The high values provided by the approximation are acceptable since over-estimating the probability of collision is safe while under-estimating it might cause a collision.

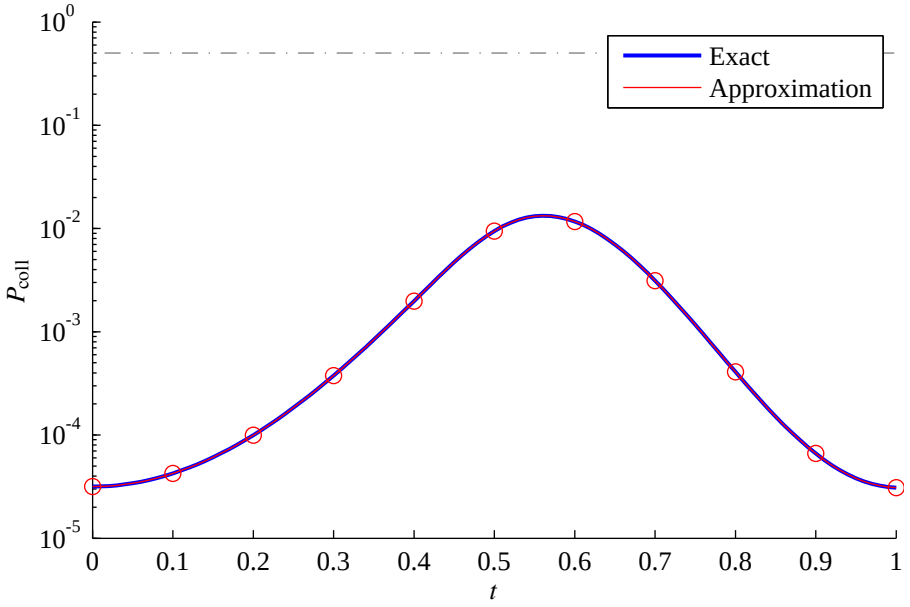


Fig. 4.8 Exact and approximate probabilities of collision with $\rho = 0$.

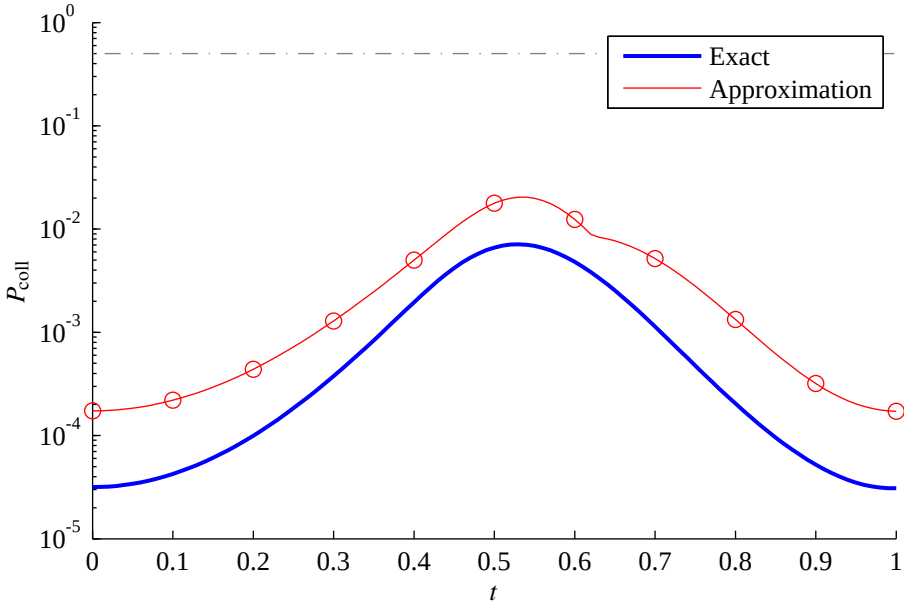


Fig. 4.9 Exact and approximate probabilities of collision with $\rho = 0.25$.

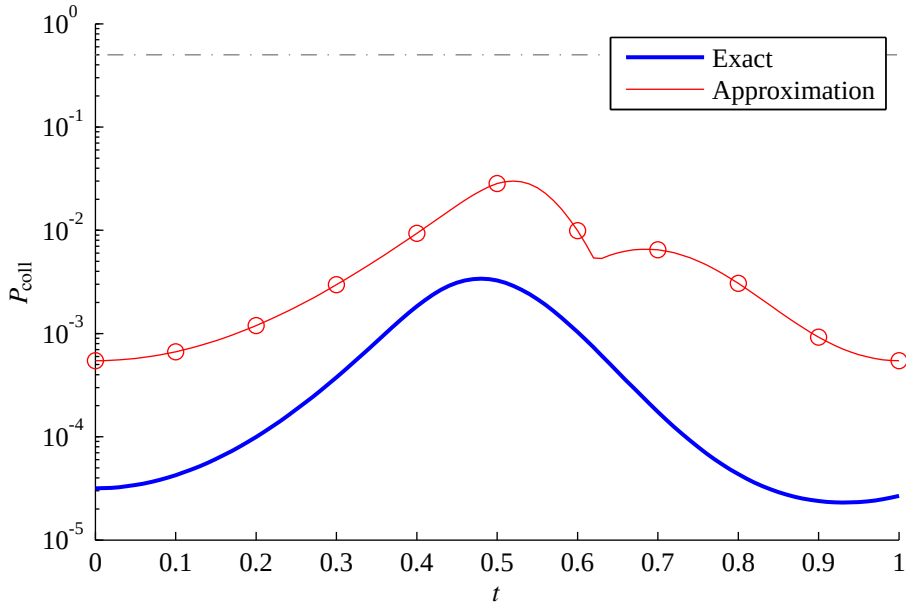


Fig. 4.10 Exact and approximate probabilities of collision with $\rho = 0.5$.

Chapter 5

Numerical Simulations

In this section, the validity of the control algorithm is demonstrated in simulations. Three scenarios are considered to measure the performance of the algorithm. The first scenario has only a single rectangular obstacle. This is the simplest scenario and is necessary to show that the control strategy is able to cope with trivial navigation scenarios. In the second scenario, a workspace with a concave obstacle which results in a local minima is used to test the ability of the controller to successfully navigate the agent to the goal. In the last scenario, the agent attempts to navigate a workspace with multiple obstacles.

The agent moves according to the motion model described in Equation (1.5). Range measurements relative to the anchor nodes are performed according to the measurement model described in the Equation 1.7. The CKF, introduced in Section 2.2, is used to estimate the position of the agent, which is then used by Algorithm 1 to produce the next control vector. The simulation terminates when the agent reaches goal position or if maximum simulation time is reached.

5.1 Effects of Simulation Parameters

In this section, the effects of varying simulation parameters on the performance of the controller are assessed. A simple L-shaped workspace without obstacles is considered to better observe the behavior of the agent when simulation parameters are changed. Table 5.1 lists the simulation parameters that remain the same under different scenarios.

$\mathbf{q}_{\text{initial}}$	$\mathbf{q}_{\text{goal}}$	σ_v^2	$k_{\max}$	w	R	Δw	ΔR
$[3.5\ 1]^T$	$[22\ 22]^T$	0.001	100	1	1	0.25	0.1

Table 5.1 Simulation parameters.

In the following simulations, measurement noise variance σ_w^2 and probability of collision thresh-

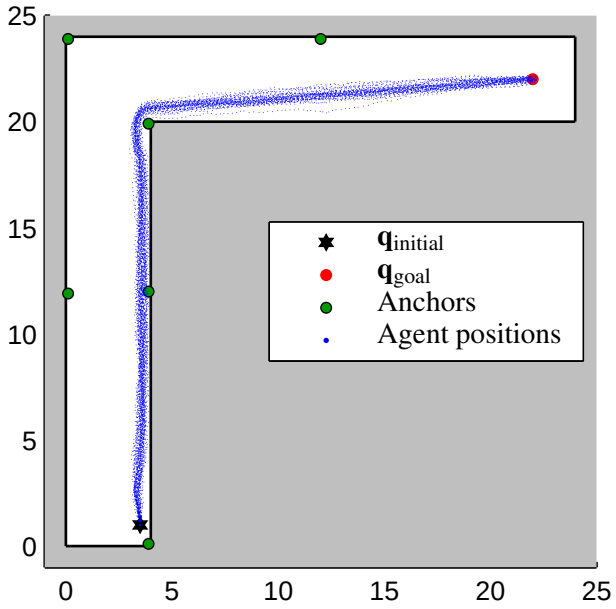


Fig. 5.1 Simulation with $\sigma_w^2 = 0.1$ and $P_{\text{th}} = 10^{-2}$.

old P_{th} are varied. Four scenarios are considered and in each case, the simulation is run for 100 times. The resulting agent positions are represented as a scatter plot in the workspace. Comparing Figure 5.1 with Figure 5.2 and Figure 5.3 with Figure 5.4, it is seen that increasing the measurement noise variance results in agent positions becoming more spread out. This is expected since increasing measurement noise variance results in decreased accuracy of position estimates, as a result, the agent deviates from its intended path.

On the other hand, comparing Figure 5.1 with Figure 5.3, and Figure 5.2 with Figure 5.4, shows that decreasing P_{th} causes the agent to keep a larger distance between itself and the obstacles to remain safe. This result is desirable, since different real world scenarios might have different safety constraints, and having a design parameter that determines how cautiously the agent navigates is useful.

This probabilistic characterization of the agents cautiousness is the main contribution of this thesis. For navigation scenarios where the number of reference nodes of the localization infrastructure, such as GPS, might change over time, our method would allow the agent to continue to operate as long as the uncertainty in localization remain relatively low. In the event that the agent can't guarantee a safe move, it can choose to remain stationary until localization improves or it can make a risky move.

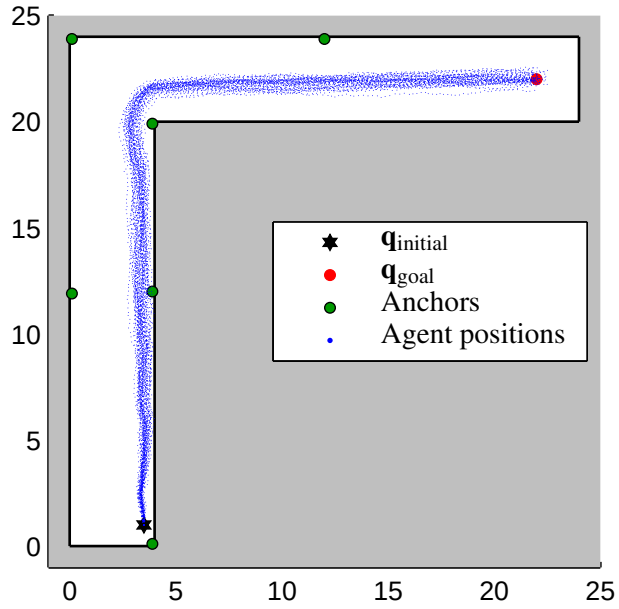


Fig. 5.2 Simulation with $\sigma_w^2 = 4$ and $P_{\text{th}} = 10^{-2}$.

5.2 Simulation Scenarios

The indoor environment is a 25×25 square region. There are six anchor nodes present in the workspace $\mathscr{W}$ of the agent, represented by green circles. The black star represents the initial position of the agent, and the red circle is the goal position. Shaded regions correspond to obstacles and boundaries as explained in Section 3.5, and thus are inaccessible by the agent.

5.2.1 Indoor Scenario with a Rectangular Obstacle

In the first scenario, only a single rectangular obstacle is present in the workspace of the agent. Table 5.2 lists the parameters used for the simulation. As mentioned in the previous chapters, $\mathbf{q}_{\text{initial}}$ is the initial position of the agent, $\mathbf{q}_{\text{goal}}$ is the goal position, σ_w^2 is the measurement noise variance, σ_v^2 is the process noise variance and $k_{\max}$ is the maximum number of iterations the agent can perform before deciding on a control vector. The tuning parameter w is used to change the strength of the force due to the APF, and R determines the distance at which the fields become active. Parameters Δw and ΔR control the rate of change of the tuning parameters. Finally, the parameter P_{th} corresponds to the value of the probability of collision which will trigger the tuning of APFs to lower it again. Figure 5.5 shows 10 runs of the simulation and the resulting path plotted in the workspace.

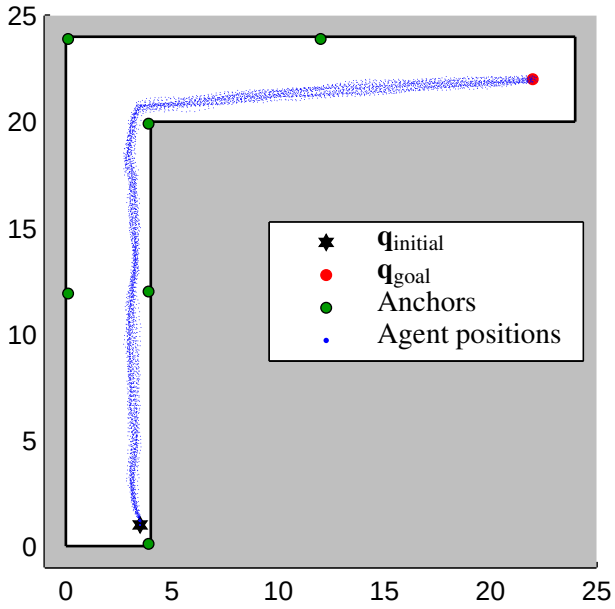


Fig. 5.3 Simulation with $\sigma_w^2 = 0.1$ and $P_{\text{th}} = 10^{-10}$.

$\mathbf{q}_{\text{initial}}$	$\mathbf{q}_{\text{goal}}$	σ_w^2	σ_v^2	$k_{\max}$	w	R	Δw	ΔR	P_{th}
$[3.5\ 10]^T$	$[22\ 17]^T$	0.1	0.001	10	1	1	0.25	0.1	10^{-10}

Table 5.2 Simulation parameters for single obstacle scenario.

5.2.2 Indoor Scenario with a Local Minimum

In the following scenario, the workspace of the agent contains a horse-shoe shaped obstacle. The shape of the obstacle produces a local minimum in the configuration space $\mathscr{C}$ of the agent, as discussed in Section 3.2. The probability of collision is computed with respect to the convex hull of the obstacle, as the agent approaches the local minimum, the probability of collision will increase. The control strategy will attempt to lower it by increasing the range and strength of the repulsive field around the obstacle and pushing the agent away. Tables 5.3 and 5.4 show the relevant simulation parameters for the navigation scenario. The two scenarios only differ in the initial position of the agent and the goal position. Simulation results show that the controller is very sensitive to the position of the obstacle in the workspace $\mathscr{W}$ as well as the initial position of the agent and goal position. In the scenario, depicted in Figure 5.6, the agent is able to escape the local minimum. In contrast, Figure 5.7, depicts a scenario where the agent was unable to escape. This is not surprising, since the edges at the two ends of the horse-shoe shaped obstacle form a new local minimum as the repulsive field around the vertical line segments grow larger and stronger. Further increasing R and w only push the local minimum away from the obstacle. As a result, the agent moves away from the

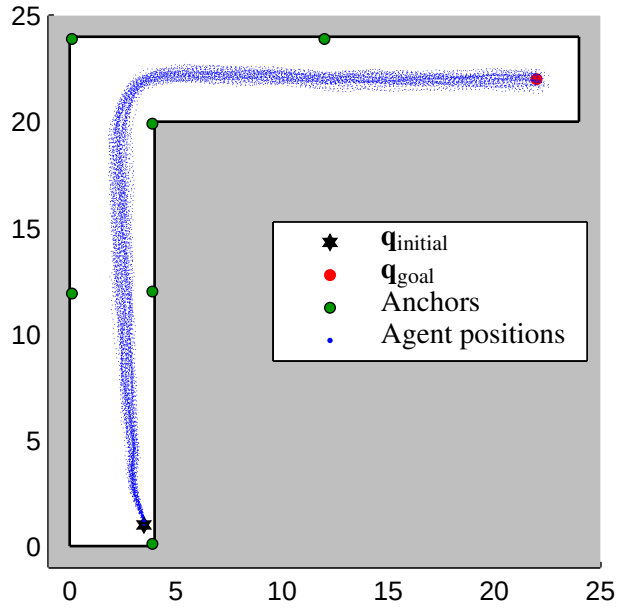


Fig. 5.4 Simulation with $\sigma_w^2 = 4$ and $P_{\text{th}} = 10^{-10}$.

$\mathbf{q}_{\text{initial}}$	$\mathbf{q}_{\text{goal}}$	σ_w^2	σ_v^2	$k_{\max}$	w	R	Δw	ΔR	P_{th}
$[3.5\ 5]^T$	$[22\ 20]^T$	0.1	0.001	10	1	1	1	0.4	10^{-10}

Table 5.3 Simulation parameters for local minimum scenario 1.

obstacle. Eventually moves towards it again as the repulsive fields slowly weaken over time. This process repeats for the duration of the simulation and the agent is never able to reach the goal. A drastic change in the initial and goal positions is necessary, such as depicted in Figure 5.6, for the agent to be able to avoid the local minimum.

5.2.3 Indoor Scenario with Multiple Obstacles

In this scenario, the workspace contains multiple obstacles. While the individual obstacles are convex, their combinations can produce non-convex structures, as can be seen in Figure 5.8. The simulation is again run 10 times and the resulting paths are shown in Figure 5.8. In all cases, the agent was able to navigate around the obstacles to reach the goal destination, despite the challenging arrangement of the obstacles in $\mathscr{W}$. Figure 5.9 shows the number of iterations performed at each time step for one run of the simulation. It is seen that at any time step no more than three iterations were required to produce a valid control vector. However, the number of iterations necessary to adapt the APF depends on the parameters Δw and ΔR that govern the rate of adaptation. Lowering their values would increase the number of iterations.

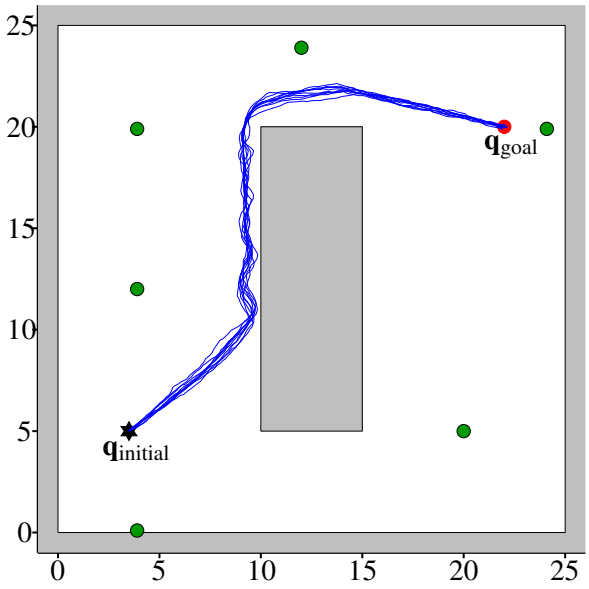


Fig. 5.5 Simple scenario with one rectangular obstacle.

$\mathbf{q}_{\text{initial}}$	$\mathbf{q}_{\text{goal}}$	σ_w^2	σ_v^2	$k_{\max}$	w	R	Δw	ΔR	P_{th}
$[3.5\ 10]^T$	$[22\ 10]^T$	0.1	0.001	10	1	1	1	0.4	10^{-10}

Table 5.4 Simulation parameters for local minimum scenario 2.

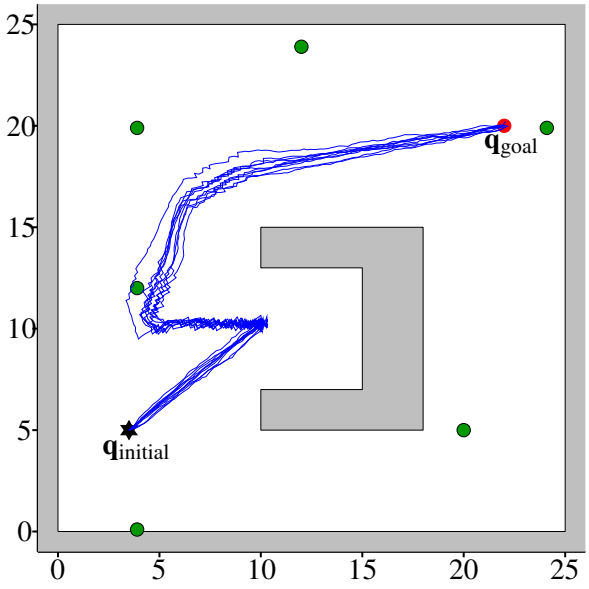


Fig. 5.6 Local minimum scenario 1.

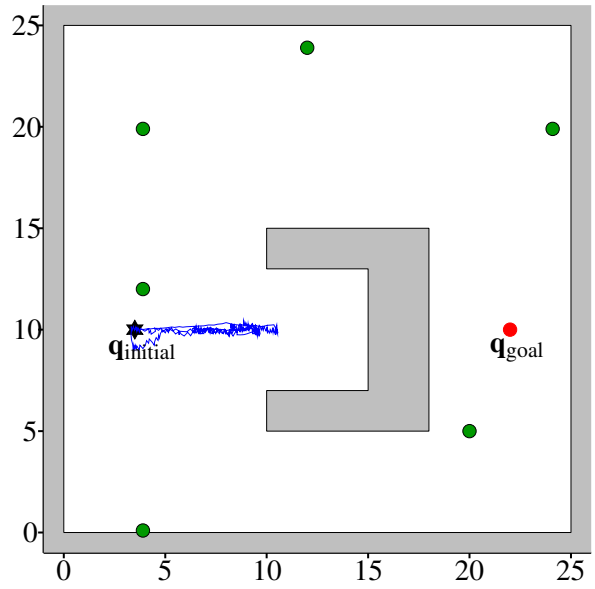


Fig. 5.7 Local minimum scenario 2.

$\mathbf{q}_{\text{initial}}$	$\mathbf{q}_{\text{goal}}$	σ_w^2	σ_v^2	$k_{\max}$	w	R	Δw	ΔR	P_{th}
$[3.5\ 10]^T$	$[22\ 17]^T$	0.1	0.001	10	1	1	0.25	0.1	10^{-10}

Table 5.5 Simulation parameters for navigation scenario with multiple obstacles.

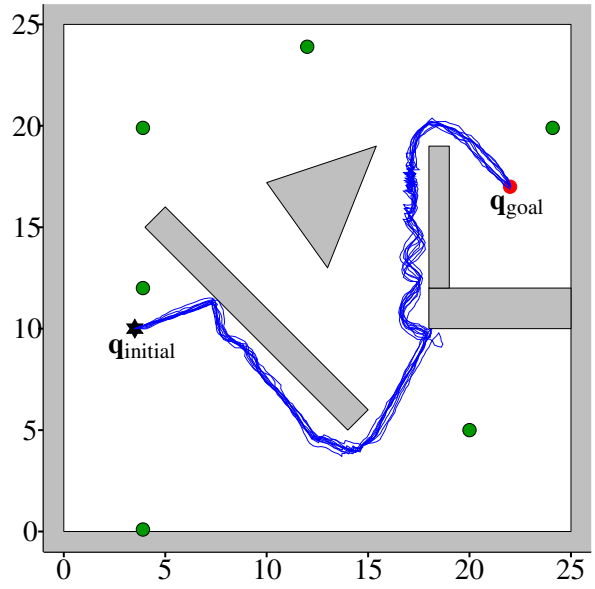


Fig. 5.8 A scenario with multiple obstacles.

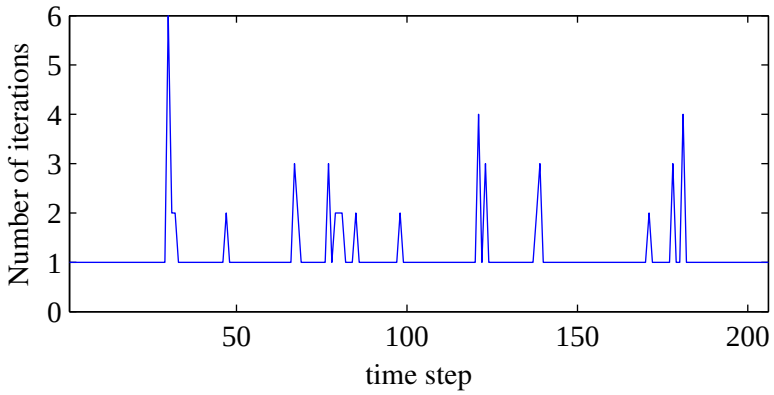


Fig. 5.9 Number of iterations per time step for the many-obstacle scenario.

Chapter 6

Conclusion

In this Master's thesis, a control algorithm was developed to navigate a mobile agent indoors. Agent localization was done in two steps: range measurements were performed relative to the anchor nodes with known positions; the cubature Kalman filter (CKF) was used to compute position estimates. The probability of collision with obstacles was defined, and used for adapting the artificial potential field (APF) to produce control vectors that directed the agent to the goal configuration while attempting to keep the approximate probability of collision under a given threshold.

In comparison to the traditional APF methods, the proposed algorithm is probabilistic and unlike hybrid APF methods, it does not rely on computationally intensive genetic algorithms to escape local minima. Simulations with different threshold values showed that lowering the threshold caused the agent to keep a larger distance between itself and the obstacles in the environment. Therefore, the threshold value can be interpreted as a measure of agents cautiousness. This makes the algorithm suitable for navigation scenarios where accurate localization is difficult due to the scarcity or temporary absence of reference systems.

Furthermore, it was shown that the resulting algorithm is able to move the agent to the goal configuration in the presence of a single obstacle, as well as multiple arbitrary obstacles. The algorithm also achieved limited success in escaping from local minima. However, the ability of the algorithm to escape local minima was found to be sensitive to the configurations of the agent, the goal and the obstacles. The parameters that adapt the potential fields required different values under different scenarios. These shortcomings limit the applicability of the algorithm to fully-known environments. Further research into deriving an optimum strategy for adapting the potential fields is a prerequisite for enabling the algorithm to function in partially known or unknown environments.

APF methods have always suffered from the problem of local minima, and while navigation functions can produce APFs without local minima in known environments, similar solutions do not exist for partially known or unknown environments. An avenue for future research is the development of a hybrid algorithm that uses the control algorithm proposed in this thesis for global planning and obstacle avoidance, and a graph search method for escaping from local minima.

References

[1] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to Autonomous Mobile Robots*. MIT press, 2011.

[2] J. Grotzinger, J. Crisp, A. Vasavada, R. Anderson, C. Baker, R. Barry, D. Blake, P. Conrad, K. Edgett, B. Ferdowski, R. Gellert, J. Gilbert, M. Golombek, J. Gómez-Elvira, D. Hassler, L. Jandura, M. Litvak, P. Mahaffy, J. Maki, M. Meyer, M. Malin, I. Mitrofanov, J. Simmonds, D. Vaniman, R. Welch, and R. Wiens, "Mars science laboratory mission and science investigation," *Space Sci. Rev.*, vol. 170, no. 1-4, pp. 5–56, 2012.

[3] C. C. Eriksen, T. J. Osse, R. D. Light, T. Wen, T. W. Lehman, P. L. Sabin, J. W. Ballard, and A. M. Chiodi, "Seaglider: A long-range autonomous underwater vehicle for oceanographic research," *IEEE J. Ocean. Eng.*, vol. 26, no. 4, pp. 424–436, 2001.

[4] A. Sinha, T. Kirubarajan, and Y. Bar-Shalom, "Autonomous surveillance by multiple cooperative UAVs," in *Proc. SPIE '05*, vol. 5913, Sep. 2005.

[5] E. Falcone, R. Gockley, E. Porter, and I. Nourbakhsh, "The personal rover project: The comprehensive design of a domestic personal robot," *J. Robot. and Auton. Syst.*, vol. 42, no. 3, pp. 245–258, 2003.

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

[7] J. H. Gallier, *Geometric Methods and Applications: For Computer Science and Engineering*. Texts in applied mathematics, New York: Springer, 2001.

[8] J. Canny and J. Reif, "New lower bound techniques for robot motion planning problems," in *Proc. IEEE FOCS '87*, pp. 49–60, Oct. 1987.

[9] J. Canny, *The Complexity of Robot Motion Planning*. MIT press, 1988.

[10] J.-C. Latombe, *Robot Motion Planning*. Kluwer Academic Publishers, 1991.

[11] Y. Li, *Real-time motion planning of multiple agents and formations in virtual environments*. PhD Thesis, Simon Fraser University, 2008, Burnaby, Canada.

[12] H. Choset, K. M. Lynch, S. Hutchinson, G. A. Kantor, W. Burgard, L. E. Kavraki, and S. Thrun, *Principles of Robot Motion: Theory, Algorithms, and Implementations*. Cambridge, MA: MIT Press, June 2005.

[13] N. J. Nilsson, "A mobile automaton: An application of artificial intelligence techniques," Tech. Rep., DTIC Document, pp. 509–520, 1969.

[14] C. Ó'Dúnlaing and C. K. Yap, "A "retraction" method for planning the motion of a disc," *J. of Algo.*, vol. 6, no. 1, pp. 104–111, 1985.

[15] O. Souissi, R. Benatitallah, D. Duvivier, A. Artiba, N. Belanger, and P. Feyzeau, "Path planning: A 2013 survey," in *Proc. IEEE IESM '13*, pp. 1–8, Oct 2013.

[16] R. Finkel and J. Bentley, "Quad trees a data structure for retrieval on composite keys," *Acta Informatica*, vol. 4, no. 1, pp. 1–9, 1974.

[17] O. Khatib, "Real-time obstacle avoidance for manipulators and mobile robots," *Int. J. of Robot. Res.*, vol. 5, pp. 90–98, Mar. 1986.

[18] L. Xie, H. Chen, and G. Xie, "Artificial potential field based path planning for mobile robots using virtual water-flow method," in *Proc. ICIC '07*, pp. 588–595, Springer, 2007.

[19] J. Sfeir, M. Saad, and H. Saliah-Hassane, "An improved artificial potential field approach to real-time mobile robot path planning in an unknown environment," in *Proc. IEEE ROSE '11*, pp. 208–213, Sep. 2011.

[20] E. W. Dijkstra, "A note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[21] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Trans. Syst., Man, Cybern., Syst*, vol. 4, no. 2, pp. 100–107, 1968.

[22] A. Stentz, "Optimal and efficient path planning for unknown and dynamic environments," Tech. Rep., DTIC Document, 1993.

[23] S. Koenig and M. Likhachev, "D* lite.," in *Proc. AAAI '02*, pp. 476–483, Jul. 2002.

[24] D. Ferguson and A. Stentz, "Field D*: An interpolation-based path planner and replanner," in *Robotics Research*, Springer Tracts in Advanced Robotics, pp. 239–253, Springer, 2007.

[25] D. Harabor and A. Botea, "Hierarchical path planning for multi-size agents in heterogeneous environments," in *Proc. IEEE CIG '08*, pp. 258–265, Dec. 2008.

[26] L. Kavraki, P. Svestka, J.-C. Latombe, and M. Overmars, "Probabilistic roadmaps for path planning in high-dimensional configuration spaces," *IEEE Trans. Robot. Autom.*, vol. 12, pp. 566–580, Aug. 1996.

[27] S. M. Lavalle and J. J. Kuffner Jr, "Rapidly-exploring random trees: Progress and prospects," 2000.

[28] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. of Robot. Res.*, vol. 30, pp. 846–894, Jun. 2011.

[29] J. Nasir, F. Islam, U. Malik, Y. Ayaz, O. Hasan, M. Khan, and M. S. Muhammad, "RRT*-smart: A rapid convergence implementation of RRT*," *Int. J. of Adv. Robot. Syst.*, vol. 10, Jun. 2013.

[30] P. Raja and S. Pugazhenthi, "Optimal path planning of mobile robots: A review," *Int. J. of Phys. Sci.*, vol. 7, pp. 1314–1320, Feb. 2012.

[31] S. Särkkä, *Bayesian Filtering and Smoothing*. Institute of Mathematical Statistics Textbooks, Cambridge University Press, 2013.

[32] S. Thrun, W. Burgard, and D. Fox, *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. MIT Press, 2005.

[33] R. Van Der Merwe, *Sigma-point Kalman filters for probabilistic inference in dynamic state-space models*. PhD Thesis, Oregon Health & Science University, 2004, Portland, OR.

[34] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1254–1269, 2009.

[35] J. Chen, D. M Dawson, M. Salah, and T. Burg, "Cooperative control of multiple vehicles with limited sensing," *Int. J. of Adapt. Control Signal Process.*, vol. 21, no. 2-3, pp. 115–131, 2007.

[36] J. Barraquand, B. Langlois, and J.-C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 22, no. 2, pp. 224–241, 1992.

[37] A. Valero-Gomez, J. Gomez, S. Garrido, and L. Moreno, "The path to efficiency: Fast marching method for safer, more efficient mobile robot trajectories," *IEEE Robot. Autom. Mag.*, vol. 20, pp. 111–120, Dec. 2013.

[38] E. Rimon and D. E. Koditschek, "Exact robot navigation using artificial potential functions," *IEEE Trans. Robot. Autom.*, vol. 8, no. 5, pp. 501–518, 1992.

[39] I. Filippidis and K. Kyriakopoulos, "Adjustable navigation functions for unknown sphere worlds," in *Proc. IEEE CDC-ECC '11*, pp. 4276–4281, Dec. 2011.

[40] J. Barraquand, L. Kavraki, R. Motwani, J.-C. Latombe, T.-Y. Li, and P. Raghavan, "A random sampling scheme for path planning," in *Robotics Research*, pp. 249–264, Springer, 1996.

[41] M. C. Lee and M. G. Park, "Artificial potential field based path planning for mobile robots using a virtual obstacle concept," in *Proc. IEEE/ASME AIM '03*, vol. 2, pp. 735–740, Jul. 2003.

[42] A. E. Eiben and G. Rudolph, "Theory of evolutionary algorithms: A bird's eye view," *Theoretical Computer Sci.*, vol. 229, no. 1, pp. 3–9, 1999.

[43] G. Jones, "Genetic and evolutionary algorithms," in *Encyclopedia of Computational Chemistry. John Wiley and Sons*, 2002.

[44] P. Vadakkepat, K. C. Tan, and W. Ming-Liang, "Evolutionary artificial potential fields and their application in real time robot path planning," in *Proc. IEEE CEC '00*, pp. 256–263, 2000.

[45] H. Mei, Y. Tian, and L. Zu, "A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment," *Int. J. of Inf. Technol.*, vol. 12, no. 3, pp. 78–88, 2006.

[46] H. Park and J.-H. Kim, "Potential and dynamics-based particle swarm optimization," in *Proc. IEEE CEC '08*, pp. 2354–2359, Jun. 2008.

[47] G. Li, A. Yamashita, H. Asama, and Y. Tamura, "An efficient improved artificial potential field based regression search method for robot path planning," in *Proc. ICMA '12*, pp. 1227–1232, Aug. 2012.

[48] A. Zangwill, *Modern Electrodynamics*. Cambridge University Press, 2012.

[49] J. V. Terza and U. Welland, "A comparison of bivariate normal algorithms," *J. of Statistical Comput. and Simul.*, vol. 39, no. 1-2, pp. 115–127, 1991.

[50] D. R. Cox and N. Wermuth, "A simple approximation for bivariate and trivariate normal integrals," *Int. Statistical Rev.*, vol. 59, pp. 263–269, Aug. 1991.