Forward Chaining for Potential Field Based Navigation

A thesis to be submitted to the

UNIVERSITY OF ST ANDREWS

for the degree of

DOCTOR OF PHILOSOPHY

by

Graeme Bell

School of Computer Science
University of St Andrews

September 2005

Abstract

A key ability for any real world robotic agent and for many simulated agents in virtual environments is the ability to navigate successfully to goal positions within their world. It is often necessary to do this in the presence of obstacles that limit access to parts of the agent's world.

This thesis addresses the problem of agent navigation in the situation where an agent is seeking a goal in the presence of obstacle configurations. Potential fields are used to model the navigational problems, and purely potential field based heuristics are given as solutions. Existing potential field based techniques and alternative navigation approaches are shown to lack certain desirable properties that might allow them to be considered completely successful.

The general problem of potential field navigation is addressed here using subgoal chaining. A series of novel potential field based subgoal selection heuristics are developed that guide the agent to success in the presence of the stated problem conditions. Development of the heuristic begins by modelling a solution for the simplest two dimensional environments, and then extending it to solutions for problems of greater complexity.

The techniques developed here are shown to be successful on a wide variety of problems. Applications for these solutions are suggested in robotics, animation and computer games as well as in other areas of potential field research such as neural networks. Suggestions are given for ways in which competing techniques within navigation can be augmented by applying the results of this research. The remaining problems for potential field based agent navigation are highlighted for future study.

I, Graeme Bell, hereby certify that this thesis, which is approximately 71000 words in length, has been written by me, that it is the record of work carried out by me, and that it has not been submitted in any previous application for a higher degree.

date _____ *signature of candidate* _____

I was admitted as a research student in September 2001 and as a candidate for the degree of Doctor of Philosophy in September 2002; the higher study for which this is a record was carried out in the University of St Andrews between 2001 and 2005.

date _____ *signature of candidate* _____

I hereby certify that the candidate has fulfilled the conditions of the Resolution and Regulations appropriate for the degree of Doctor of Philosophy in the University of St Andrews and that the candidate is qualified to submit this thesis in application for that degree.

date _____ *signature of supervisor* _____

In submitting this thesis to the University of St Andrews I understand that I am giving permission for it to be made available for use in accordance with the regulations of the University Library for the time being in force, subject to any copyright vested in the work not being affected thereby. I also understand that the title and abstract will be published, and that a copy of the work may be made and supplied to any *bona fide* library or research worker.

date _____ *signature of candidate* _____

“If I have seen farther than others, it is because I was standing on the shoulders of giants.” - Isaac Newton

“If I have not seen as far as others, it is because giants were standing on my shoulders.” - Hal Abelson

“In computer science, we stand on each other’s feet.” - Brian Reid

Acknowledgements

I would like to thank Mike Livesey and Michael Weir for being my supervisors during this PhD, and for the invaluable advice and suggestions they made throughout my PhD.

I would like to thank the Carnegie Trust for the Universities of Scotland for their generous funding of my work, and Prof. Ronald Morrison and Alex Bain for helping to secure various pieces of financial support for my work.

I would like to thank the School secretaries, Helen and Joy, for their help during my PhD.

I would like to thank the other students in the Intelligent Computation research group - Jonathan, Adrian and Gordon - for their friendship and support.

I would like to thank Phil Rutherford of Kuju Games for taking the time to participate in a short email interview, Mitul Saha for supplying Figure 5.3 and Paul Bourke for supplying Figure 8.2.

I would like to thank Tim, Mike and Graham for their help in proofreading this dissertation.

I would also like to thank Scott, Davie, Andy, Gregor, Kath, Ruth, Tim, Iain, Brian, Yang, Olga, Svetlana, Lisa, Jenna, Youyi, Nick, James, Tad, Blerina, Gil and Ciara for their friendship and support during my PhD.

Finally, and most of all, I would like to thank my family for their support: my dad, Andrew Bell, my mum, Caroline Bell, and my brother, Cameron Bell.

Published Research

Listed here are the publications made during this research.

1. M.K. Weir and G. Bell, "The Incorporation of Intentional Action into Robots", St Andrews University School of Computer Science Technical Report. (2001) [236]

2. G. Bell, M.K. Weir, "Forward Chaining for Robot and Agent Navigation using Potential Fields" (2004). ACM International Conference Proceeding Series, Proceedings of 27th Australasian Computer Science Conference (ACSC2004), Dunedin, New Zealand. [71]

3. G. Bell, M.K. Weir, "Agent Navigation using Potential Fields and Forward Chaining" (2004). Proceedings of Postgraduate Research Conference in Electronics, Photonics, Communications and Networks and Computing Science, (PREP2004), University of Hertfordshire, UK. [70]

4. G. Bell, M. Livesey, "The Existence of Local Minima in Local-Minimum-Free Potential Surfaces" (2005). Proceedings of Towards Autonomous Robotic Systems, (TAROS 2005), Imperial College, London, UK. [69]

Apart from the Technical Report, I am the primary author of all of the publications [69, 70, 71]. I am the secondary author of the Technical Report [236].

All of the publications apart from the Technical Report underwent peer review before being accepted for publication.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Guide to Chapter 1

This chapter gives motivations for conducting research in the fields of robotic, cinematic, and computer game agent navigation. A brief overview of the problem to be tackled in this thesis and its application domains are given. A summary of the thesis contributions is provided. An overview of the thesis structure in the form of a series of short chapter abstracts can be found towards the end of this introductory chapter.

Real-world agents (robots) and virtual-world agents (such as computer generated film or game characters) play an increasing role in everyday life. While robots have yet to enter their ascendancy in the consumer's world, they are becoming increasingly important in scientific, military and industrial applications. Computer controlled virtual-world agents, often inhabiting faux-physical environments (that is, environments with simulated physics that closely resemble the physics of the real world), can be found in most computer games and in many major films - particularly those featuring large groups of people or animals, where neither human actors nor hand animated agents are a sufficiently affordable or flexible approach.

These industries are of enormous economic significance. The United Nations Economic Commission estimates that by the end of 2003, the service robotics industry had a value of around $3.8 billion, which they expect to rise to $9.6 billion (mainly through growth in personal and private use) by the start of 2007 [59]. The British Entertainment and Leisure Software Publishers Association, ELSPA, records the world market for games and edutainment software at $16.9 billion by the end of 2002 [49]. In cinema, the 'Lord Of The Rings' trilogy alone has already achieved $3 billion in box office takings [56], a success story that could not have been achieved without extensive use of computer controlled virtual-world agents in the making of the films.

In all of these fields and applications, a common skill that is needed is the ability to allow agents to automatically navigate between positions in an environment containing obstacles. For agents constrained to travelling along the ground in 3-D environments, it is usually sufficient to be able to navigate in two dimensions.

Despite various algorithmic and heuristic attempts to tackle the problem of specific cases of 2-D or 3-D navigation, it remains to some extent an unsolved problem. The best existing navigation algorithms for 2-D environments are not well suited to many navigation scenarios - often they rely on the agent's world remaining static, combined with an ability on the part of the agent to provide perfect obstacle information and localisation to the algorithm. Techniques sometimes depend on the agent possessing the ability to steer in the real world precisely as the algorithm suggests the agent should, with little tolerance of error. When attempting a heuristic rather than algorithmic approach, failure is often encountered as environments become more complex in terms of size or number of obstacles. In short, there is no single catch-all algorithm or heuristic that can reliably govern general navigation in a variety of scenarios of arbitrary complexity.

This situation is worsened by various factors: the near absence of textbooks guiding detailed implementation of agent navigation (Latombe [138] being the notable exception); and the fact that the implementations of existing approaches are seldom as straightforward as they first appear, with algorithms and heuristics often relying on further sub-algorithms being provided, or implicitly needing low levels of sensor noise and movement error - or both. It is hardly surprising that in practice, robot and virtual-world agent programmers are sometimes forced to resort to 'steering' their agents by giving them a human-designed physical or virtual track to follow.

1.1 Hypothesis

This thesis proposes the following hypothesis:

> "The Potential Fields Method has been prematurely abandoned as an approach to agent navigation, and the existing techniques have unaddressed weaknesses. Novel descent heuristics can be designed that allow successful navigation in an improved number of cases and that address the weaknesses of existing potential field techniques."

The Potential Fields Method (PFM) is an approach to agent navigation that was first introduced in 1979 [123], and despite its appearance in a variety of papers, most recently [169, 184, 221, 226, 243] and books [92, 138, 207], there have been only a few significant improvements in the capabilities of the basic heuristic beyond the very first model that was suggested over 15 years ago, and these are limited in their effectiveness due to high computational costs [65, 132]. Since then, potential fields have gradually been relegated from a serious navigational approach to a 'join-the-dots' local planning technique, essentially only providing continuous trajectories between targets selected by another navigation technique.

This research proposes that it is possible to augment the standard approach to Potential Field based navigation. By carefully designing potential field based steering heuristics, the Local Minimum Problem (LMP) [207] for potential field based navigation can be bypassed, allowing traditionally difficult obstacle configurations to be overcome, and improving navigation success rates significantly. The resulting heuristics display useful properties that

are not available in other approaches to agent navigation. These heuristics are developed through the analysis of the heuristic's behaviour, when interacting with the increasingly difficult classes of problem that are faced in navigation. The heuristics that are developed are tested in a number of navigational situations, and with each of the different classes of obstacle configuration that are identified in this thesis.

1.2 High level overview of the main problem addressed

This main part of this thesis is research leading to a significant enhancement in the performance of the Potential Field approach to navigation. The situation initially addressed in the main part of the research is as follows.

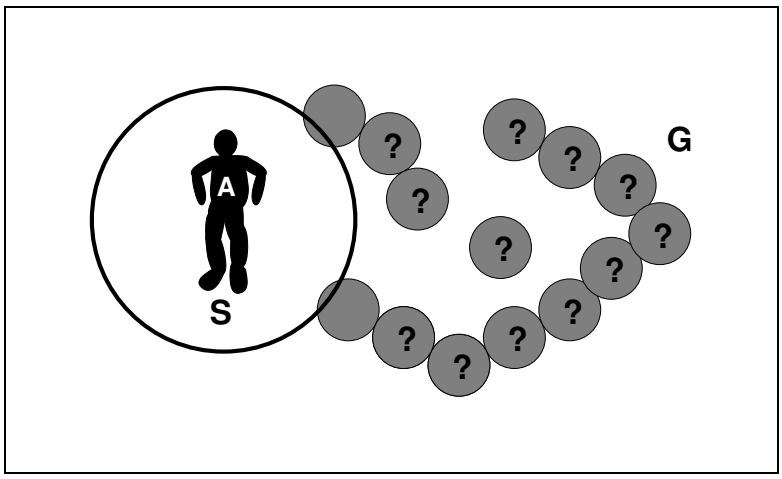


Figure 1.1: Example of a navigation environment. The thick line represents the radius of the field of vision of the agent A. The shaded circles represent obstacles. Obstacles with question marks are not currently visible/known to the agent. S represents the start position and G represents the goal position. The agent must create a path leading from S to G that does not intersect obstacles as they are encountered.

Consider a point sized agent A in a two dimensional environment. The agent's starting point S is represented as a 2-D coordinate, and the agent is seeking to reach a goal position G which is also represented as 2-D coordinate. There are some impenetrable obstacles lying between S and G that must be navigated around. This is shown in Figure 1.1.

The agent has a limited sensor range and incomplete knowledge of the environment, so some obstacle configurations may lie between S and G that are not known a priori. The

agent has an awareness of the size and position of nearby obstacles. The agent has at least some awareness of the direction of the goal position relative to the agent's current position, either by being able to sense its direction or by some global relative or absolute positioning service. The agent may be affected by sensor and effector mis-calibration or noise - i.e. there may be a small amount of error in the detection of obstacles or in the way that the agent moves. It is not guaranteed that the agent can see over or past an obstacle, further limiting the agent's field of vision.

As travel is not permitted through an obstacle, to navigate successfully the agent must generate a path as it encounters obstacles (i.e. using local information) which does not intersect (collide with) the obstacles and which ultimately reaches the goal position G. In behavioural terms, the agent must *persistently* try to reach its goal, while *plastically* reshaping its path to deviate around obstacle configurations as they are encountered to allow successful long-term navigation.

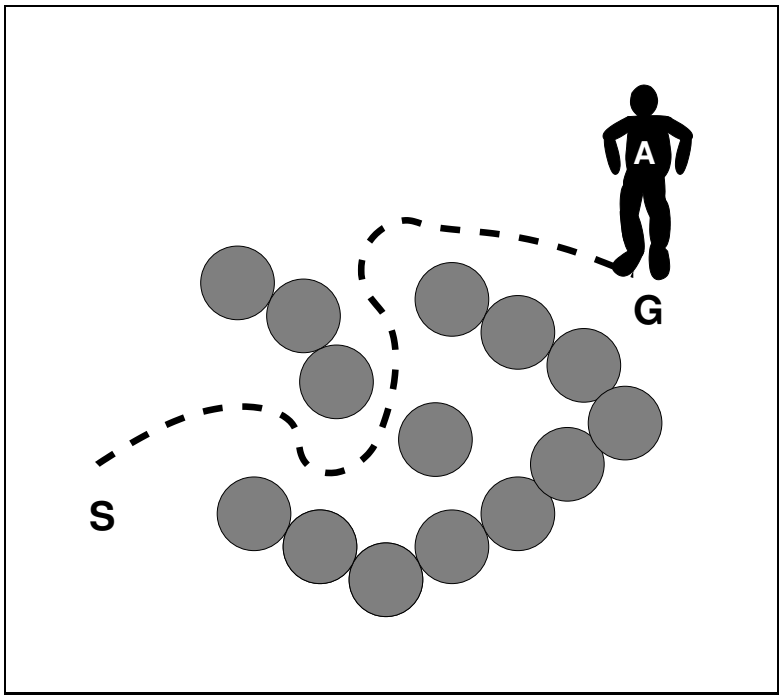


Figure 1.2: Initially unaware of the obstacle configuration OC, the path taken by the agent can be seen to travel directly towards the goal, leading into the obstacle configuration. When the agent becomes aware of all the obstacles comprising the configuration, it reshapes its path to guide it plastically around the obstacle configuration. Once it is safe to resume direct travel towards the goal, it does so. The agent reaches the goal successfully. The paths taken look like those a natural agent - such as an animal, or a human with limited awareness of the environment - might take.

It is important that the solution heuristic or algorithm must perform very well on its first attempt at navigation. This is because the heuristic may be used to control robotic agents perched on the edge of a cliff, or real-time virtual agents in games or films, and so exhaustive search, repeated attempts at navigation, or random movement strategies may not be an option. In cinematic agent control and computer game agent control, first time navigation that looks 'natural' to a human observer is a desirable skill. An example of the type of path desired is shown in Figure 1.2.

1.3 High level overview of the approach taken

This research sought to improve the *Potential Field* approach to the problem of navigation. Classically, the potential field approach fails in navigation upon encountering the (commonly-occurring) *Local Minimum Problem*. An example of this is shown in Figure 1.3. The enhancement to existing methods is achieved by identifying a novel solution to the Local Minimum Problem for potential field based navigation.

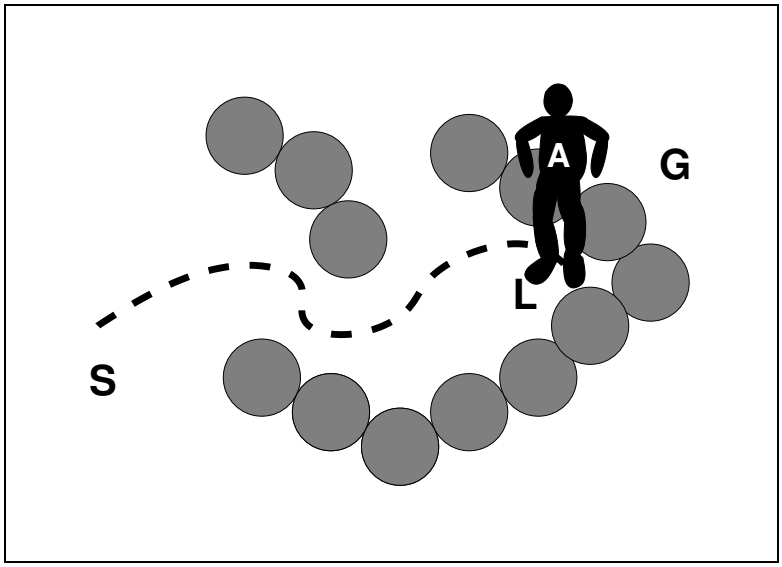


Figure 1.3: The thick-dashed line shows the agent's path into a local minimum. At point L, a local minimum, the potential field generated by the obstacle configurations and the goal G is unable to provide the agent with direction. This is because the attractive potential function of G and the repelling potential function of the obstacles balance out exactly, and any movement away from L results in the heuristic directing the agent to return to L.

The problem is that nearby this local minimum, attempting to reach the long term goal

directly will result in oscillation around the local minimum. A solution can be constructed by recognising that travel back towards a previous path is not a good way to make progress towards the goal. Placing limitations on the way descent takes place on the potential field surface can prevent this occurring, and encourage useful travel towards the goal in the long term, in the manner shown in Figure 1.4.

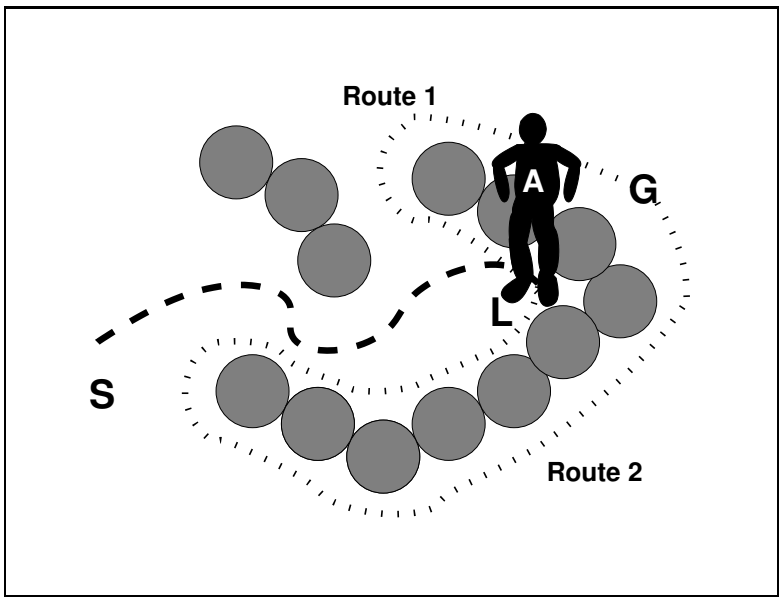


Figure 1.4: The thick-dashed path shows the agent's route into a local minimum. The thin-dashed paths show examples of the type of 'natural' paths that might be expected from a heuristic that is capable of overcoming such navigation problems.

The heuristic developed shows that such potential field based heuristics do not require steering from a non-potential fields based technique, nor access to low level sensor information - or even access to the original map of obstacles. It can be made to work using only the potential field generated from the map and the agent's internal state. This is very important, as it means that the potential field heuristic provided may be suitable for re-application within other areas of potential field research besides navigation, such as neural networks [72, 171], inverse kinematics [139], automatic system configuration [219] and other optimisation problems.

1.4 Contributions

This section details the novel contributions made by this thesis.

Major contributions

- PRIMARY CONTRIBUTION: A novel, straightforward and purely potential field based technique that allows navigation in environments containing concave obstacle configurations (Chapters 6,7).
- Introduces the novel LPCIRCLE and FWDS1 general potential field descent techniques (Chapter 6).
- Proposes techniques for allowing the approach to operate in 3-D environments (Section 8.1).
- Proposes techniques for allowing the approach to operate in n-D environments, including a novel potential field descent technique (Section 8.2) (with further improvements in the 3-D case (Section 8.2.2)).
- Proposes techniques by which two categories of existing alternative navigation approaches can be improved via augmentation with this research (Section 9.1).

Minor contributions

- Context chapters and a literature survey that show the research context of the thesis. The survey makes a contribution towards highlighting areas of mutual interest between the real-world and virtual-world navigation research communities. It also gives historical overviews of navigation for robotic agents, cinematic agents and computer game agents (Chapters 2, 3, 4).
- Identifies a family of simple, practical, and effective novel potential field based subgoal selection heuristics, and discusses the design and evolution of this family of heuristics (Chapter 6).
- Provides pseudocode and an implementation that demonstrates the novel heuristics (Chapter 6).
- Provides experimental results as evidence of the capabilities of the heuristics (Chapter 7).
- Identifies classes of pathological problems that may still exist, and suggests solutions (Appendix E).

- Discusses how the research may be re-applied into areas of research unrelated to navigation, such as neural networks and automatic system configuration (Section 9.2).

- Provides suggestions for future avenues of research into moving obstacles, multiple-agent and non-holonomic constraint scenarios (Section 10.1).

1.5 Thesis structure

Chapter 2 presents an overview of the problem of agent navigation, and of the characteristics of navigation approaches. This is followed by a summary of the main families of navigation techniques. The chapter concludes with overviews of each application domain.

Chapter 3 presents an introduction to the Potential Fields Method, a problem-solving approach within AI. The chapter discusses the underlying metaphor, the terminology, the standard techniques and the problems encountered with this approach.

Chapter 4 describes the standard mapping between navigation problems and potential field problems, along with a worked example of the technique successfully navigating in an environment. A variety of potential field surface problems are identified and interpreted in the context of navigation.

Chapter 5 is a survey of research into potential fields for navigation. The chapter shows that existing research techniques have partly succeeded in addressing some aspects of the problem of linking potential fields to navigation. Existing research can be seen to have failed to completely address all of the issues involved in navigation.

Chapter 6 presents a novel approach to potential field based navigation: Forward Chaining. A family of purely potential field based heuristics are incrementally developed to overcome the increasingly difficult problems for navigation that are identified in this chapter.

Chapter 7 provides experimental data confirming the theoretically predicted behaviour of the heuristic, and examines the performance of the heuristic in light of known problems for potential fields. The technique is assessed as a general navigation approach.

Chapter 8 looks at ways in which the Forward Chaining heuristic can be transformed for use in 3-D and n-D environments.

Chapter 9 describes ways in which this research can improve the performance of other navigation approaches, and suggests ways in which the results can be applied throughout AI more generally.

Chapter 10 presents avenues for long-term future research in Forward Chaining, summarises the research, considers the thesis hypothesis, and presents conclusions.

Appendix A is a short description of the A* graph search heuristic.

Appendix B contains extracts from a short email discussion that was conducted with an computer game AI programmer who has experience of industrial application of potential field based navigation.

Appendix C contains a chronology of the major publications that have shaped potential field based agent navigation, as well as noting some recent publications in the field.

Appendix D discusses the originality of one of the techniques described in this thesis.

Appendix E discusses pathological problems for Forward Chaining, and a range of techniques to overcome them.

Appendix F provides printouts for a number of cases of navigation in the Bigmap example environment.

Appendix G provides source code taken from an implementation of Forward Chaining.

Appendix H provides a guide to the contents of the DVD at the back of this thesis document.

Chapter 2

Context: Navigation

Guide to Chapter 2

This background chapter introduces the first of the two areas of computer science in which this thesis is set. The field of *navigation* is concerned with providing the computational facility for an agent to plan or find a sequence of movements or positions that link to form a useful path to a goal position, in an environment containing obstacles. This ability is sometimes called 'path-planning' or 'path-finding'.

The particular subfield of research within navigation that this chapter will focus on is the case where the input and output to the navigation heuristic or algorithm is provided in terms of a map and map positions, that is, without reference to raw sensor data or effector output for example.

The application domains of this form of navigation (and hence this thesis) include robotics, cinematographic agent control and computer games.

2.1 Chapter Overview

This section gives an overview of the problem of constructing navigational paths. There are three areas of discussion: looking at the problem; the typical approaches to addressing this problem; and the application domains. The chapter structure is as follows:

- The general problem of navigation - refinements and variations of the problem, and measurement of solutions.
- Standard approaches to navigation - topological, roadmap, cell decomposition, geometrical and potential fields.
- The application domains of navigation - real world and virtual world situations and their demands and constraints.

2.2 The general problem of navigation

What is navigation? Dudek and Jenkin write:

> "The basic path planning or trajectory planning problem refers to determining a path in configuration space between an initial configuration of the robot and a final configuration such that the robot does not collide with any obstacles in the environment and that the planned motion is consistent with the kinematic constraints of the vehicle. The initial configuration is known as the *start location* or pose, and the final location or pose is known as the *goal*." [93]

Though primarily concerned with the navigation of robots rather than with the navigation of agents more generally, these words serve well as a description of the problem of general agent navigation as it has been faced since Nilsson first formalised an approach to navigation in 1969 [193].

What is meant though, by a *path*, a *configuration space*, an *environment* and an *obstacle*?

Definition *Paths* can have continuous or discrete interpretations, but in both cases, a path is comprised of a sequence of consecutive neighbouring states. *A navigational path* will be one that connects from the start state to the goal state.

Definition *Configuration space*, or as it is known in robotics, *Cspace*, is the set of states which the agent has available to occupy during navigation[1].

The set of states representing the configuration space might be infinitely large in the case of an unbounded or continuous environment. A key consideration, in the case of navigation, is the dimensionality of the configuration space of the agent.

Typically, navigation is considered in the simplest case of two-dimensional[2] navigation, that is, navigation on or within a plane. This is the task that human agents typically face as they move around, with the occasional exceptions of cases where it is necessary to cope with navigation in multi-floored buildings or swimming pools, where a third dimension becomes involved in path planning. Generally, navigation is not constrained to path-planning in merely 2-D, or even 3-D, though these represent the bulk of the cases that academic researchers and industrial practitioners have an interest in.

n-D navigation consists of those cases where it is necessary to navigate, for example, a robot's arm from one pose-configuration to another. A robot's arm might have ten or more joint components that specify its exact position, in terms of how its robotic arm and fingers are laid out. Accordingly, navigating the robot's arm between holding an item at one coordinate in space, and dropping it from another point in space is a highly complex problem, and requires a path to be built in the full configuration space of the robot's arm. Navigation is non-trivial, even in 2-D [141, 199], and becomes decidedly harder as the dimensionality and complexity of the navigation problem increases.

Definition *The environment* is the combination of the agent's configuration space, unrealisable sections of the configuration space (*obstacles*) and any other constraints upon the agent's movement between states.

A key part of navigation is to avoid having a path that collides with obstacles or that tries to break any other constraints of the environment - though the consequences and thus the necessity of abiding by such constraints may vary between the application domains of navigation. Collision with an obstacle is a much less serious problem for an agent in a computer game, for example, than a Mars-based robotic rover.

[1] An extremely detailed and mathematically formal definition of *path* and *configuration space* is given in [140]. This has been omitted here, as it is not necessary to use such a detailed definition in order to understand or apply the ideas in this thesis, or compare them with existing work.

[2] Hereafter dimensionality will be referred to by the common abbreviations 2-D, 3-D ... n-D.

2.3 Difficulties for navigation

Navigation is made difficult through navigational constraints. Were there no constraints, linear interpolation would suffice to connect the start and goal state with a path, or indeed the agent might simply 'warp' from the start position to the goal position in a single step. Obstacles are the most common constraint upon navigation, but they are far from the only type, as anyone who has attempted to turn a vehicle or bicycle sharply to the side at high speed in real life will have discovered. This subsection attempts to characterise many of the variations of navigation problems that exist.

2.3.1 Obstacles

Obstacles are regions of an environment that are impassable as a result of being already occupied. Factors affecting the difficulty posed by obstacles include:

Total number of obstacles. Increasing the total quantity of obstacles will often increase the cost of representing the environment, and increases the cost of calculating navigational paths in the case of almost all algorithms and heuristics; with the exception of some discrete approaches [65, 68].

Size of obstacles. Larger obstacles can require more computation, mainly in some wavefront expansion techniques, i.e. [142, 185].

Complexity of obstacles. Complex obstacles generally require greater amounts of computation and storage to represent them accurately. Indeed, some approaches either cannot model complex obstacles, or become intractable when faced with them, particularly navigational / harmonic potential fields approaches [84, 85, 94, 130, 206], roadmap generation techniques [95, 143] and some exact cell decomposition approaches [144]. A few techniques incur little or no extra cost when complex obstacles are present in the environment - approximate cell decomposition [145], discrete navigational potential fields [146] and geometric techniques [96, 119, 177, 178]. A guide to the increasing cost complexity and modelling expressivity of various obstacle representation models can be found in [97].

Existence of obstacles. It might seem absurd to think of a navigation problem lacking obstacles, but in the case of a robot arm that is navigating between pose configura-

tions, it is entirely possible that this is the case. Here, the constraints on navigation are imposed solely by the structure of the agent or by the physical constraints of the environment, rather than by obstacles. Such problems are generally addressed in the field of 'inverse kinematics' [98, 112, 147, 190, 214].

Nature of obstacles and response to collisions. Unrealisable obstacle regions may not necessarily be physical objects. It may be that an unrealisable part of the environment is 'occupied' by a 10 meter deep hole! Accordingly, it may be that the agent fails immediately upon contact with an obstacle (for example, a Mars Rover robot that falls off the edge of a cliff), or it may be relatively inconsequential if grazing or temporary contact is made with obstacles (for example, a computer game agent among a crowd of other agents), and that navigation can still continue providing no closer contact is made with the obstacle.

2.3.2 Topological vs. metric maps

The technique that builds the map to be used for navigation may be topological [180] - that is, based around positioning relative to recognised 'landmarks', rather than metric, where geometric distance based measurements are produced and navigation is based on absolute positioning [99]. This affects the type of navigation approach that may be employed.

2.3.3 Discrete vs. continuous environments

The environment may be either continuous or discrete in nature, and the agent's representation of continuous environments may also be continuous or discrete in nature. This limits the choice of navigation technique which may be employed. Examples of discrete environments can be easily found in many computer games, and examples of continuous environments can be found in computer games, cinematic agent control and robotics.

2.3.4 Limited knowledge / Online navigation

It is possible that the agent may not be equipped with an a priori map of the environment. In this case, the agent will have to adjust its path in accordance with what it discovers as

it travels. This prohibits certain types of algorithmic and heuristic solutions to navigation, such as graph search based approaches and navigational potential functions.

Navigation techniques that are designed to exploit knowledge of a global map are known as *global planners*; those that can cope with obstacles encountered during travel are known as *local planners*. The task of navigating without complete a priori knowledge will be referred to throughout this thesis as *online navigation*.

2.3.5 Unreliable knowledge / Robust navigation

It may be that the agent's map information is unreliable, in which case, the agent is effectively carrying out online planning even if it has an a priori map[3]. This unreliability may often arise from the existence of temporary obstacles in the environment. It is likely that a map built up by an agent itself during online navigation will have some mistakes, or that errors will be made in carrying out the steering actions required by navigation. In this case, performance of the heuristic should degrade gracefully as quality of input and reliability of control drops. Some techniques naturally exhibit such behaviour through smooth control surfaces (potential fields, roadmap approaches), whereas others do not (geometrical approaches assume subsystems are completely reliable and may fail if this turns out not to be the case). Approaches that can tolerate a degree of error are *robust*. Latombe discusses navigation approaches that cope with uncertainty in considerable detail in [148].

2.3.6 Computational restrictions

Limitations can be placed on the navigation problem that require the agent to compute a navigation path within some fixed period of time. It may even be that the agent must navigate in real-time. This can be particularly problematic in cases where the computational power available to run the navigation technique is limited - such as when a robot is using a limited onboard computer, or when a computer game is trying to dedicate as much CPU time as possible to other aspects of the game such as networking and graphics. Limitations may also be imposed as a consequence of many instances of

[3] A technique designed to cope with this problem by employing two types of navigation - one using the a priori map and one for anything unexpected encountered en route - can be found in [241].

navigation being carried out in parallel (for example in multi-agent navigation scenarios). In recent years, these limitations have been become more relaxed as additional processing power has become available. Additionally, task-specialised components such as sound cards, graphics cards, and very recently, 'physics' cards [5] have reduced competition for an agent's processing resources.

Some techniques (such as high resolution or 3-D graph search, and navigational/harmonic potential fields) rapidly become intractable on all but the simplest environments, due to their computational complexity rising too quickly with large environments, or complex environments with many obstacles, or complicated obstacle shapes.

2.3.7 Dimensionality

As previously mentioned, the environment itself may be 2-D, 3-D or n-D. This imposes limitations on the types of navigation approach that may be applicable in two ways - some approaches are simply undefined in anything other than 2 dimensions, and some other approaches become rapidly intractable above 2-D.

2.3.8 Subsystems

Some techniques may rely on particular reactive subsystems being available to do particular tasks, as part of the specification of the navigation problem. A classic example of this would be geometric algorithms such as Bug [178], where it is expected that 'line following' and 'obstacle circumnavigation' routines are provided as part of the construction of the agent in the environment.

2.3.9 Resolution of representation

The resolution of the environment relates to the quality of representation of obstacles, and also to the number of available states in the configuration space. As the resolution in either regard is increased to improve the quality of the navigational path produced, the computational costs of all navigation techniques increase.

2.3.10 Physical and virtual worlds

The environment may be physical (robotics) or virtual (robotic simulations, cinematic worlds, computer games) or faux-physical in nature (some computer games, cinematic worlds, robotic simulations). Faux-physical is a term used here to describe a 3-D virtual world provided with very similar properties to the real world in terms of physics. Faux-physical environments are increasingly becoming the standard for both computer games and computer generated cinematography. It is likely that the number of environments of this type will increase further as the expectations of realism from computer games and films increase.

2.3.11 Moving obstacles and multiple agents

Most navigation approaches work on the principle that obstacles are fixed, and that the environment is *static*. This removes the problem of *foliation* which is associated with moving obstacles, where at every instant, there is a tree of environment states representing every environment configuration that may exist in the next instant. Some techniques are inherently more able to cope with moving obstacles. Typically these are *local* planning techniques using online navigation, often tackling the problem of moving obstacles mainly by being unaware of them for much of the process of navigation (when they are not local to the agent). The navigation techniques presently employed are in the main *global* techniques, which tend to deal with this problem badly (forcing recalculation of the entire navigational path when any aspect of the environment changes). This can be a particular problem for real-time applications such as computer games or robotic navigation.

It may be that the agent is itself able to modify the environment - for example, an agent in a maze might push some of the walls around if they are not heavily built, so as to provide an escape route. Such modifications of the problem of navigation are not usually considered.

Finally, the moving obstacles in the environment might themselves be navigating agents, which raises the issue of prediction of other agents' behaviour, and potentially communication to avoid collisions and allow navigation. This modification of the problem is difficult to compensate for, but is a frequently considered variant of the basic problem of navigation [100, 149, 200, 201].

2.3.12 Dynamic constraints: non-holonomic agents and environments

An agent that is able to turn on the spot and thus move in arbitrary directions is termed a holonomic agent. Such an agent will have no difficulty in reaching the entirety of the available navigation space, nor will it have any problems in shaping its path as a navigation approach requests it to. To some extent, normal human navigation is like this - if humans are moving slowly, they can stop, turn to any angle, and begin movement again.

As agents begin to move more quickly, however, dynamic constraints may be imposed. An agent may not be able to turn so quickly, or it may not be able to arbitrarily select its own speed. Alternatively, the construction of an agent may limit its ability to turn at a given speed to at most a particular tightness of turning circle.

Such constraints are called non-holonomic constraints [101, 139], and navigation under these constraints is called non-holonomic path planning. Non-holonomic navigation is recognised as a different and substantially more difficult form of navigation problem[4], and each particular unique set of non-holonomic constraints requires its own equally unique solution that is typically derived by hand. General navigation techniques assume that the agent is essentially holonomic in nature.

2.3.13 Modelling of the agent

Typically, navigation approaches assume that the agent is point sized (the 'point robot assumption'). To accommodate the fact that the agent is almost certainly not point sized, obstacle shapes are usually 'dilated' [93, 102] by the longest dimension of the agent. Since many agents are approximately circular in shape, this is generally an effective solution.

Though this reduces the freedom of the agent to navigate in those cases where the agent is far from circular in shape, and may occasionally render a solvable problem intractable, it greatly reduces the complexity involved in navigation. In specific problems though, it may be that a constraint of the navigation problem is that the agent is of a particular shape and size that must be accommodated, e.g. [68, 220].

[4]Consider, for example, the difficulty that some human agents have when trying to parallel park a car correctly, which is a relatively simple non-holonomic navigation problem.

2.3.14 Terrain types

Sometimes, different costs are associated with different parts of the environment. This could be used, for example, to represent that carpeted floor is more difficult for wheeled agents to traverse than a flat linoleum floor. Some navigation approaches attempt to allow variable cost models of the terrain in the environment. Some interesting discourse on this topic within the application domain of computer games can be found in [196].

2.3.15 Arbitrary constraints

Constraints found in navigation problems are not limited to those listed here. A particular problem may require a 'safe path', or 'a path that allows easier localisation', or a path that has some particular type of shape.

2.3.16 Summary

The apparent complexity of the problem of navigation as it is represented in this section is seldom approached. Most navigation solutions restrict themselves to tackling only a very limited subset of the issues raised in this section. It is nonetheless useful to keep in mind some awareness of the ways in which the problem of navigation is often expanded.

2.4 Characterisation of approaches to navigation

Having looked at the constraints that may apply to a navigation problem, it is now useful to discuss the ways in which solution approaches to navigation problems (and the paths they generate) may be described. This section describes some of the qualities of navigation approaches and the paths they generate.

2.4.1 Generality

A consideration when selecting a navigation technique is the extent to which a technique is tied to a particular application domain or even a particular implementation. Is the technique suitable for application across a wide or narrow range of representations of the environment, and can it be applied within several different application domains or just one? A robotic navigation approach relying on the existence of a particular sensor or motor behaviour may not necessarily be suitable as a general navigation strategy.

2.4.2 Path planning vs. trajectory planning vs. tracking

Some techniques (roadmap, cell decomposition) produce only a discrete set of points that an agent must achieve to reach the goal - a path in the loosest sense. Other techniques (geometry approaches) produce a continuous trajectory through space, and some techniques go even further (potential fields) and provide tracking of the trajectory during movement - either in that they dynamically direct the agent back to the original trajectory or by implicitly correcting the trajectory to cope with the agent's errors in following it.

2.4.3 Optimality

Many techniques attempt to minimise costs (for example, by generating a path of minimum total length) - or maximise realisability of the path (perhaps by minimising curvature of the path). Optimality is sometimes desirable, but not always - i.e. it may affect other qualities of the path such as 'naturalness' (discussed in Section 2.4.9). Techniques that produce optimal behaviour tend to be computationally expensive, as more possible paths must be considered than in a heuristic technique. Some problem constraints make optimality hard to achieve - non-holonomic constraints and moving obstacles being particularly notorious in this regard.

2.4.4 Completeness

If a path exists as a solution, is the technique guaranteed to find it [103]? An example of this might be a complex maze-like environment, where a heuristic approach is likely to fail, and where only a global algorithmic navigation approach will succeed in eventually finding the way out.

2.4.5 Soundness

If the routines produce a path, is it guaranteed to be collision free? Particularly, techniques employing random movement need to have explicit collision checks to ensure soundness [103]. Some techniques (such as visibility graph based roadmaps) generate paths that theoretically do not touch any obstacle, but in practice pass so close to an obstacle that grazing can be possible. Other techniques actually require the agent to be in contact with obstacles to succeed in navigation (i.e. Bug).

2.4.6 Computational cost

Traditional computer science measurements of big-O ($O(n)$) execution time complexity and space complexity are often used to measure the cost of navigation techniques - where $O(n)$ can include parameters such as the number of obstacles in the environment, the dimensionality of the environment, the cost of representing obstacles, and the resolution of the representation of the environment or path. Of particular interest to implementors in some application domains is the actual cost of execution for particular situations, in addition to the big-O rate of increase in the cost of execution. A particular aspect of computational cost that is relevant to all domains is whether the navigation approach is capable of generating a path in real-time, in the environment at hand. A final aspect of computational cost is that it may relate to the cost of 'setting up' the technique as well as the cost of running it for a particular case of navigation. In other words, some (offline) approaches require an expensive piece of pre-computation to be carried out on the representation of the environment in order to allow a computationally cheap approach during navigation (i.e. certain potential fields approaches).

2.4.7 Limitations upon representation of paths, agents and obstacles

Some techniques use an approximate representation of the environment or its contents, which results in the technique often being either un-sound, or incomplete. The accuracy of the approximation is sometimes referred to as the resolution of the approach, particularly in metric grid-based techniques. The idea of *resolution completeness* refers to a technique which is *complete*, in so far as the resolution of the representation of the environment allows it to be [92].

2.4.8 Suitability to the problem constraints

Some techniques work across many types of navigational constraint, and others are more specialised and cannot be used at all outside of a particular type of navigational problem. Particularly, approaches are sometimes characterised in terms of their ability to cope with:

- Online navigation - the ability to navigate with incomplete knowledge.
- Robustness - the ability to cope with probabilistic or inaccurate data.
- Multiple agents - the ability to include agent communication as an aspect of planning.
- Suitability for holonomic vs. non-holonomic planning (most techniques cope with holonomic navigation).
- Dependence upon subsystems.

Particularly though, the following four aspects of the navigation problem tend to be considered in evaluating a solution approach:

- Local vs. global planning - is the technique suitable for online navigation or only capable of path planning using a complete a priori map? Online planners are also suitable for offline navigation but tend to produce inferior results to techniques designed to exploit global knowledge.
- Suitability for complex environments and complex obstacles. Some techniques can cope with complex environments (many obstacles) and complex obstacle configurations and shape, without incurring extra costs in computation of the

navigational path. Other techniques incur a moderate cost in dealing with complexity, and yet other techniques are unable to cope with either large numbers of obstacles or high complexity of obstacles.

- Suitability to dimensionality of problems. Many complete/global path planners are useful in 2-D, sometimes feasible in 3-D, but unusable in n-D due to the number of possible paths that could exist and must be considered.
- Discrete versus continuous approaches. Irrespective of the environment, or even of the model of the environment supplied by a mapping system, a technique may apply a discrete or continuous approach to navigation. Historically, the nature of a technique in this regard is closely related to the application domain - robotic techniques feature a mix of continuous and discrete approaches and computer games tend to feature discrete approaches (though there are exceptions). Discrete techniques tend to deal quite well with high obstacle or environment complexity, and quite poorly with large environments or high dimensionality; with continuous techniques, the opposite is often the case.

More information on these aspects can be found in Section 2.3.

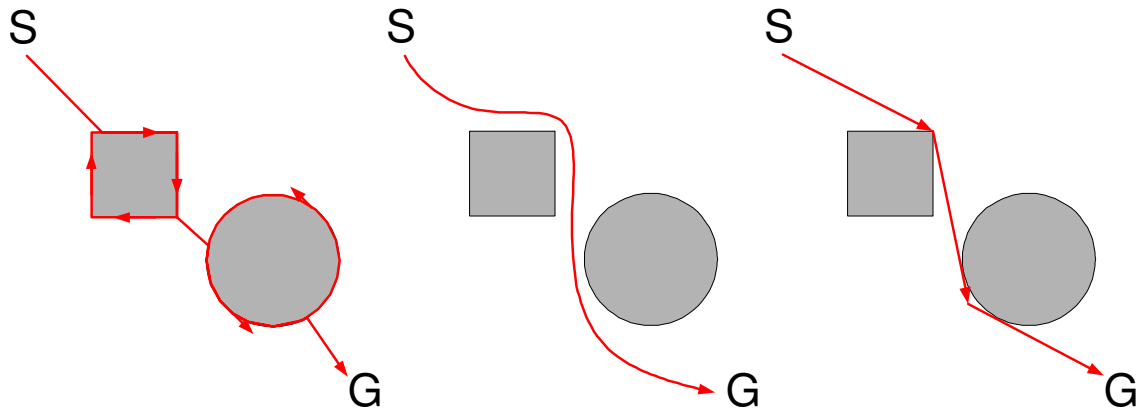
2.4.9 Non-standard ways of classifying navigation approaches

The following qualities will be mentioned occasionally in this thesis, but do not appear to have been of interest in historical evaluations of navigational strategies, and do not represent well-established ways of describing navigational approaches in academia.

Naturalness

This is difficult to quantify but easy to recognise. Does the agent exhibit behaviour that might be associated with natural behaviour? Is there any circumstance under which a natural agent might possibly follow the path generated by the technique?

This quality is is one of the essential ingredients for agents in certain application domains where human interaction is involved, where the agent is meant to represent a real-world agent in its behaviour. A lack of naturalness can detract from the ‘realism’ of the agent. Consider Figure 2.1.



(a) (b) (c)

Figure 2.1: Three examples of paths. Properties such as optimality and completeness may reduce the naturalness of movement.

It can be observed that some of the navigation behaviour is unnatural, some might be natural, and some does not fall clearly into either category. How then might natural navigation behaviour be described or quantified? Natural behaviour is usually close to optimal, but certainly not as consistently and perfectly optimal as a machine might be. Natural navigation behaviour is usually fairly predictable (by humans). Natural navigation behaviour is smooth (in the everyday sense, and to some extent also in the mathematical sense), and does not usually exhibit sharp changes of behaviour or direction unless faced with immediate and unexpected changes. In the absence of a precise definition, perhaps the only way to tell if generated movement is natural is to conduct experiments and ask people if the behaviour appears natural to them. This is a difficult issue and research coverage seems scarce, outside of work on crowd-modelling [224]. As the issue of naturalness is not central to this thesis, it will not be addressed any more thoroughly than it is in this section. Quantifying the naturalness of navigation would however be an interesting topic for further research outwith this thesis.

Ease of comprehension

It seems possible that a technique that is based on a simple theoretical model is more likely to be adopted widely than a difficult to comprehend technique. How can this be measured though? This quality is dependent on the availability and simplicity of analogies as well as the brevity and simplicity of pseudo-code. It is difficult to quantify, but likely to relate strongly to the industrial acceptance of a technique.

Ease of implementation

Related to ease of comprehension, this quality could be said to exist when a navigation technique can be implemented by a non-expert in minutes or hours rather than days or weeks. A technique which is easy to implement and works in 99% of cases could be seen by many as being more useful than a difficult technique that also works in the remaining 1% of cases.

2.5 Overview of mainstream navigation approaches

An overview of the main families of navigation approaches will now be presented with examples. There are many one-off navigation approaches in robotics and computer games that are particular to unique navigation scenarios, and that are based on neural nets or particular subsumption architectures with pre-supplied behaviours and so on, but this thesis (and therefore this section) is concerned more with more abstract navigation approaches that can be re-applied without modification across a wide range of navigation problems.

Since A* graph search is mentioned in the descriptions of several of the approaches, a short guide to this heuristic has been provided in Appendix A.

2.5.1 Topological Maps / Landmark-based navigation

This approach takes as its axiom the idea that the agent has some number of recognisable environment features (landmarks) and that the agent either has an a priori map of how these landmarks fit together, in the form of graph, or has some capability to build such a map itself through exploration. This does not in itself sound like an unreasonable basis for navigation. Humans often communicate instructions to one another in a similar form - “travel down the hall, take the second door on the left, then go to the end of the corridor” rather than via precise metric map data. A graph search technique such as A* search is typically used to find a path between nodes in the topological map.

Unfortunately the situation in which landmarks can be uniquely identified is rather limited. In robotics, typically landmarks in the form of bar codes have to be put on the ceiling or at a special height so that the agent can identify them, and also be able to see other landmarks,

to tell which direction it should travel in to reach the next landmark.

Moreover, generating landmarks without such 'friendly' environments being prebuilt by a human can be extremely difficult, since it relies on finding places that are so sufficiently different when examined by the agent's sensors, that they can be regarded as having a unique 'signature'. Identifying signatures that can be easily recognised from a range of distances and angles is not straightforward [104].

In practice, topological maps can be used to steer a physical agent around a prebuilt track with clearly labelled landmarks at an appropriate position for the agent to read, combined with an a priori map of connectivity, but it is not otherwise very practical as a form of navigation for physical agents. The technique can be considerably more useful with virtual agents who can rely on being able to identify topological locations and move between them more easily and reliably. Topological map locations can also be well suited for use with symbolic AI reasoning about the world. Further coverage of this approach can be found in [105, 180, 186, 208].

This type of non-metric approach will not be discussed further in this thesis, since it is effectively not a general purpose form of navigation, suitable for arbitrary goals and obstacle arrangements, without some a priori preparation of the environment by a human. From here on, only metric navigation approaches will be considered.

2.5.2 The Roadmap approach

This approach is called the 'roadmap' approach because a set of connected virtual 'roads' for the agent to travel along are built by the approach to facilitate navigation. The roadmap (or skeletonization) approach is similar in some ways to the topological approach, in that it relies on a graph-based model of how the environment is connected. However, it is quite different in that an a priori metric map is used as the basis of the model that is generated. The agent creates from this metric map a series of arcs that represent the connectivity of parts of the agent's free space. The agent is aware of its current position on the metric map using some form of localisation such as GPS or dead-reckoning.

There are several ways that such a connectivity map can be built up - the well-known approaches being 'voronoi diagrams' [80, 106, 150, 187, 209], 'visibility graphs' [107, 151, 193, 210], 'freeway nets' [152], 'silhouettes' [153, 211] and 'probabilistic

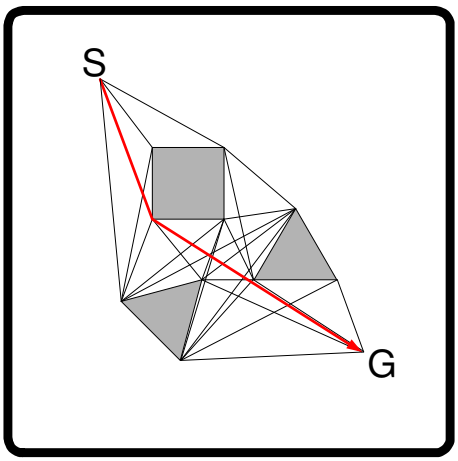


Figure 2.2: Example of a visibility graph based roadmap.

roadmaps' [121, 122]. An example of a roadmap built using a visibility graph technique in a polygonal environment is shown in Figure 2.2. The techniques to generate a roadmap vary in computational cost and completeness, but almost all share the property of being at best only slightly intuitive and are non-trivial to implement - in some cases requiring

> "involved tools from Differential Geometry ("stratifications") and Elimination Theory ("multivariate resultant")" [154]

Use of the roadmap is relatively straightforward, though, once built - the agent just moves to the nearest piece of road (if it can), performs A* graph search to find the shortest path to the piece of road closest to its destination, and then tracks the trajectory of the roads leading it towards its destination. The agent then leaves the roadmap when it has arrived at its final road segment and travels to the goal.

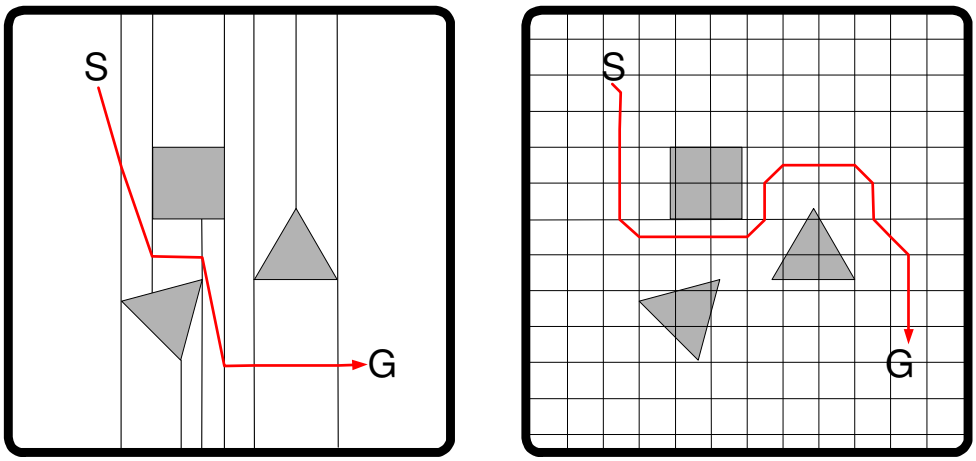
Roadmaps can be constructed so that roads keep close to obstacles (visibility graph), or remain at a safer distance (voronoi diagrams, freeway nets) - albeit at the price of longer paths. Roadmap techniques are generally unsuitable for online planning since they require a complete a priori metric map, though some attempts to use them in local planning scenarios have been made [155]. It is usually computationally expensive to compute the roadmap and so the roadmap techniques are typically limited to 2-D, or at best 3-D environments[5]. The techniques are generally only suited to polygonal representations of obstacles. A comprehensive survey of roadmap methods can be found in [143].

[5]Though the silhouette roadmap technique does allow an extremely computationally expensive approach to n-D environments.

2.5.3 Cell Decomposition approaches

Cell decomposition is a global navigation approach, and is therefore unsuitable for online navigation. There are two basic forms of cell decomposition - exact cell decomposition [144, 212] and approximate cell decomposition [145]. The idea behind both approaches is similar though, in that a metric map is broken up into a number of non-overlapping regions, and navigation takes place by the agent travelling from cell to cell. Both techniques are illustrated in Figure 2.3.

Graph search (such as A*) is used to determine which cells form a path (or 'channel', as it is known in this approach) that leads from the current cell to the cell containing the goal. The agent must then find its own way from cell to cell, until it reaches the goal.



(a) An exact cell decomposition approach based on trapezoidal cells.

(b) An approximate cell decomposition approach using square cells.

Figure 2.3: Examples of cell decomposition approaches.

There are clearly many ways in which a metric map might be divided up into cells that do not overlap - in the worst case, infinite numbers of infinitely small cells might be used. The principle behind the decomposition approaches is therefore to ensure that cells are large enough that it is computationally feasible to compute a path between any two cells in the map, but small enough that it should not be difficult for the agent to find a path leading between two adjacent cells.

The cell decomposition approach can be distinguished from the roadmap approach in that no roads are generated between cells a priori and it is left to the agent to carry out inter-cell travel - the path planning approach does not produce a trajectory for the agent directly. Consequently the approach will typically be hybridised with a continuous local planner.

In the case of exact cell decomposition, it can be computationally expensive and in some cases very unintuitive to generate the most practical types of exact cell decompositions in order to enable the agent to move effectively between cells. Obstacle representations are therefore limited to polygons in many implementations. In practice, the agent may have considerable difficulties in moving between cells, as cells may be of unusual shapes and sizes.

An alternative to this family of decomposition methods is approximate cell decomposition. The approximate cell decomposition method works by standardising all the cells to a particular simple shape, usually a rectangle or cube. Graph search for a connectivity channel from the start to the goal then takes place as before. This makes tasks such as travel between cells far simpler, but introduces other problems.

First of all, the representation is conservative and may fail to represent the entire amount of free-space that is available to the agent in the configuration space. This is because a cell that partly contains an obstacle must be entirely marked as an obstacle. This can be mitigated by using e.g. a quadtree representation, in which each cell can be further subdivided into similarly shaped (but smaller) cells, but this introduces the risk of unbounded costs in generating the decomposition as well as in using it. Some decomposition techniques generate cells that are all the same basic shape (rectangular) but are of completely different sizes.

Nonetheless, grid based decompositions are quite intuitive - indeed most human metric maps are based on this approach - and are straightforward to implement even if they are incomplete planners. Generation of the decomposition is trivial and can even be computationally cheap (providing cells are of a standard size and shape). However, searching for a path from the start to the goal is expensive even in 2-D and extremely expensive, sometimes intractable in 3-D. The cost of this approach also rises sharply as the resolution of the cell decomposition increases and as the size of the environment increases. In the case of exact cell decomposition, the cost also rises with the complexity of the obstacles, but varies according to the heuristic chosen to generate the decomposition.

Further coverage and descriptions of cell decomposition methods can be found in [156, 212].

2.5.4 Geometrical approaches

The family of geometrical approaches [96, 119, 157, 177, 178, 213] was invented by Lumelsky. Here, trajectories are generated using subsystems that (for example) carry out obstacle circumnavigation and goal-line following behaviour[6]. A common principle is that the environment can be interpreted in terms of concentric shells of obstacle configurations, which the agent must overcome. An example of a geometric approach is shown in Figure 2.4.

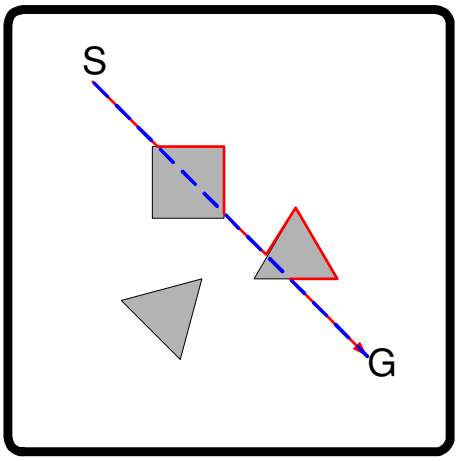


Figure 2.4: Example of a geometric approach (Bug2).

In Bug1, the simplest of the approaches, the idea is that the agent tries to travel in a straight line between the start and the goal. In the event that an obstacle is encountered, the agent employs a reactive subsystem to completely circumnavigate the obstacle. Once this is done, the agent goes back around the obstacle to the point on the obstacle that was closest to the goal. The agent then begins travelling directly towards the goal once again through free space. An example of this can be seen earlier in Figure 2.1(a).

Successive versions of the Bug approach reduced the length of the path generated by using sensor information or geometric properties to take short cuts and optimise the distance travelled.

The technique uses local information but is complete, though it does rely on its position finding and obstacle circumnavigation subsystems being faultless. The technique is limited to 2-D static environments, but some attempts have been made to translate the approach into 3-D environments [82, 225]. The paths produced are non-optimal but bounded in size.

[6]Goal-line following behaviour means to travel through free space along a line between the start and the goal.

2.5.5 The Potential Fields approach

The potential fields approach is a general problem solving approach in AI, and was introduced to the field of navigation by Khatib [123] around 1979. It is based on the metaphor that the goal should attract the agent towards it, and that the obstacles should repel the agent from them. Combining these two forces upon the agent produces a net force that (in theory) moves the agent towards the goal and away from obstacles simultaneously. An example of this is shown in Figure 2.5.

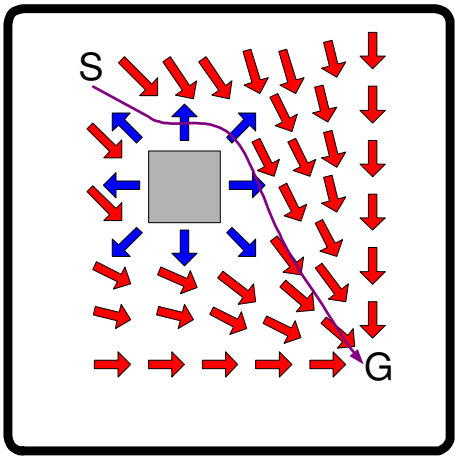


Figure 2.5: Example of a potential field approach. Red arrows represent the long range attractive effect of the goal. Blue arrows represent the short range repulsive effect of an obstacle. The purple line indicates the path taken by the agent in response to the combination of these two effects.

The technique can be used in either a continuous or discrete form; is suitable for online navigation and can be computationally cheap (both in terms of computing the function to represent the potential field, and in terms of computing the desired direction for the agent at positions on the potential field). Unfortunately, the basic technique is also incomplete and non-optimal, and in fact highly likely to fail on anything other than a trivial environment containing a few convex obstacles. Arbitrary obstacle shapes can be represented by this technique.

Whilst it is possible to alter the technique to make it a complete planner, this comes at the expense of making it extremely computationally expensive to compute the field function, so much so that the technique becomes effectively impractical on all but very simple environments [158].

There are a number of variations on this approach [66, 132]. Further details of this

technique will be given in chapters 3, 4 and 5 of this thesis, and additional information can be found in [108, 159, 188].

2.5.6 Summary of approaches

Table 2.1 at the end of this chapter summarises the properties of the metric navigation techniques in this section.

2.6 Application domains for automated navigation

This section discusses the requirements and sought-for behaviour of the various application domains of automated navigation. A background history of navigation in each application domain is given first in each case to establish a context for the requirements.

Please note that the word 'automated' has been used in this section to distinguish navigation through heuristics or algorithms, from either human-steered or random movement. Furthermore, 'navigation' is here interpreted as 'automated agent control where some form of simultaneous goal seeking behaviour *and* obstacle avoidance has been implemented'.

2.6.1 A background to automated agent navigation in robotics

This section provides some historical context to the use of automated agent navigation in robotics. Since robotics has shaped the field of navigation most significantly, it is the source of most of the terminology and ideas of the field, and contributors of some of the more important concepts are noted below.

The earliest known example of automated navigation came in 1969, when Nilsson produced a robot called "Shakey" [193]. This robot had basic symbolic path planning capabilities, based on the visibility graph roadmap approach. At this time in robotics, much of the work was centered around logical inference systems that would reason about the control of the robot, rather than numerical systems to actually drive motors and produce useful behaviour. Nillson's work marked the start of a shift of emphasis within robotics from qualitative topological navigation, to more quantitative metric navigation.

In 1976, Taylor introduced the first path-planning approach that could cope with uncertainty [227], and the idea of using a point-sized robot was also introduced around this time (1977) by Udupa [231]. Lozano-Perez introduced the idea of robotic 'configuration spaces' in 1979 and shortly afterwards also introduced the idea of the approximate cell decomposition approach [175, 176]. In 1982 Chatila applied cell decomposition to the problem of uncertain environments [81] by implementing what was technically a global planner that could cope with uncertainty by periodically updating its decomposition to include new environmental data and then re-planning navigation.

1982 also marks the year when the first Roadmap-type approach was proposed by O'Dunlain and Yap [195]. Brooks suggested a clearer example of this technique with the 'Freeway' roadmap approach [77] the following year. In 1983, the 'piano movers problem' [220] was put forward and solved by Schwartz and Sharir in a series of papers, representing the first complete path-planning approach for a non-point sized robot that was able to freely rotate and move in a 2-D environment.

The earliest recognised work in non-holonomic behaviour took place in 1986 when Laumond considered car-like mobile robots [170].

In 1986, Khatib published a paper on the potential field method for path-planning [127], which is commonly recognised as the starting point of that approach (though he had in fact produced material towards this approach at several points over the preceding seven years [123, 124, 125, 126, 128], with a flurry of papers in 1985). The approach was plagued by the local minimum problem, so two other popular variations were developed soon after, with Koditschek founding the navigational (local minimum free) potential fields approach [132] a year later, and Latombe and Barraquand producing the RPP discrete potential field based planner [66] which used random path planning to escape local minima in 1989. Another major contribution was the 'harmonic functions' approach introduced in 1990 by Connolly [84] which made mapping environments into navigational potential functions slightly more straightforward.

An early example of a robot being provided with the ability to build (and navigate using) a topological map was provided by Mataric in 1990 [180].

Geometrical approaches to navigation began in 1987 when the 'Bug1' and 'Bug2' approaches were created by Lumelsky and Stepanov in 1987 [178], which were further developed into the 'Distbug' [118], 'Tangentbug' [119] and 'Visbug' [177] approaches in later papers.

Current work in robotic navigation tends to be either practically orientated towards particular robotic platforms (i.e. AIBO [7, 27]) or competitions (particularly, Robocup [24] - a library of Robocup related research can be found in [23]). On a topical note, the problems encountered by the robotic Mars rovers "Spirit" and "Opportunity" may stimulate some new research interest in robotic navigation [18].

A more detailed history of early robotic navigation research can be found in [160].

2.6.2 Demands of robotic agent navigation

- 2-D navigation for robots navigating on a plane.
- Online navigation (e.g. local planning)
- Inverse kinematics in the case of multi-jointed robots (potentially, n-D navigation, though in practice this approach tends not to be used).
- Real-time navigation (ideally without pre-planning to allow online behaviour)
- Soundness - in cases where intersecting with an obstacle may damage or destroy the robot.
- The ability to cope with environments of moderate complexity (that is, neither trivial nor labyrinthine).

2.6.3 Typical approaches to robotic agent navigation

All of the navigation approaches that are discussed in this chapter can be found in use in the field of robotics, along with some other approaches such as 'hybrid' approaches [135, 241, 243] that feature a combination of different navigation techniques - for example, adding a continuous planner to interpolate smoothly between the states of paths generated by a discrete technique, or adding an online navigation approach to augment the ability to realise paths suggested by an offline path-planner.

2.6.4 A background to automated agent navigation in films

This section provides a historical context to the use of automated agent navigation in cinematography. The history of automated agent use in cinematography has been extremely limited, indeed almost non-existent, until very recently.

The earliest example of a form of automated navigation (where other agents represent obstacles) is perhaps Eurythmy (1985) [61] which featured the first procedural animation of flocking agents. Little information is available on the operation of this however, besides that provided by Reynolds in [205].

A far better example of automated navigation in cinematography though came in 1986 in the form of Reynolds' "Boids" [202, 205], which was a computer generated animation featuring a computer controlled flock of birds. Reynolds specialised in producing simple mathematical functions that could be combined through a weighted sum to create interesting behaviour. The original Boids heuristics contained a simple attractor equation providing goal-seeking behaviour to allow the flock to be steered around an environment, basic obstacle avoidance of circular obstacles, and flock control behaviours to keep the virtual birds at roughly the same distance from one another and thus avoiding one another. The work was presented in the form of a 40 second short film at SIGGRAPH in 1987 called "Stanley and Stella in: Breaking the Ice" [203].

This work is particularly interesting, because not only does it have a very clear example of goal and obstacle driven navigation, but also additional behaviour - group flocking. The resulting animation was unfortunately somewhat crude in appearance due to the limited graphical capabilities of computers of the time, but is a powerful example of how automated agent navigation frees animators from having to manually specify cinematic agent behaviour.

The next major exhibition of automated agent navigation again featured flocking behaviour with basic goal-seeking and limited obstacle avoidance, in Tim Burton's "Batman Returns" (1992) [36, 202], where computer simulated bats and penguin flocks were created with software based upon Boids. Following this was Disney's "The Lion King" in 1994 [38, 202]. Here, combined behaviours similar in nature to Reynolds' Boids were used to steer a herd of computer controlled wildebeests, moving on a plain in a swarm towards a goal, and avoiding convex-shaped obstacles. The following year, an episode of Star Trek 'Voyager' used similar routines to generate a swarm of alien creatures in an episode called 'Elogium' [39, 202]. Reynolds gives a brief survey of other films featuring automated agent navigation

between 1995 and 2000 in [204].

It was not until very recently that the idea of using 'digital crowds' began to become extremely popular, following the success of Weta Digital in producing large scale computer generated battles for 'The Lord of the Rings' trilogy (2001) [45, 48, 52]. Weta Digital's work was groundbreaking, because they developed a general agent control system ("MASSIVE" [43]) that could be applied to any film requiring large scale automated agent control[7]. Soon after, another company called "NaturalMotion" responded by releasing their own agent dynamics system, "Endorphin" [55], which was used in "Troy" (2004) [58] to control agents. This system featured a combination of a physical dynamics system, with genetic algorithm based techniques to allow animators to generate simple behaviours for agents to use. Lastly, "XSI 4.2/Behaviour" [25] is the third well known 'digital crowd' control package that has become available in the last few years. It was used in "The One" [46] to automatically generate navigational control for a large, 1500 agent prison crowd.

It seems that war epics (as a genre) and distributed systems (as a technology) have driven the recent use of automated agent control with basic navigation. Recent work [43] has involved up to 146000 automatically controlled agents. Still, it remains to be seen whether the use of digital crowds will become a standard cross-genre feature in the film industry. It is not obvious that (in general) Hollywood is currently seeking anything much beyond the most basic 'straight line' navigation behaviour in automatically animated agents - perhaps because the degree of realism that can currently be supplied by a human animator is so much better.

However, looking to the future (and to a possible application of the research in this thesis), perhaps more intelligent and natural automated agent navigation may be useful for providing characters walking around in the background of shots - that is, 'thin crowds' - where erroneous aspects of navigation behaviours provided by techniques used in films currently will be shown up far sooner, than when the agent is part of a far larger crowd.

The current general trend in cinematography is for more and more agents (in the 'war film' genre at least) and thus there will be a desire for less and less human-generated control for these agents due to the expense of employing human animators. The increasing number of agents also introduces a need to keep the computational cost of navigation low.

[7] MASSIVE is a proprietary system and as such, information on its operation is difficult to obtain. Anecdotal evidence is available [218] that suggests the MASSIVE system is non-trivial to set up, and produces useful results only during short simulations.

In academia, agent navigation for cinematics is associated with increasing research interest. Recent papers such as [80, 218] are beginning to examine areas of agent navigation such as general navigation, collision avoidance, multi-agent navigation and navigational psychology.

Finally, (and tangentially), several other fields are incorporating virtual environments utilising a cinematic agent approach where drop-in navigation techniques would be useful. These include military and police training exercises [233, 239], architects seeking to model the flow of people in buildings [76], and emergency evacuation and disaster planning (such as computer agents seeking exits from a building during a simulation of a building fire) [224, 228].

2.6.5 Demands of cinematic agent navigation

- 2-D navigation, since to date the most popular use of automated agent navigation has involved agent movement on a plane [38, 45, 46, 48, 52, 58].
- Low computational cost, since many thousands of agents may be in use at once.
- The ability to cope with other agents as moving obstacles, perhaps even multi-agent navigation.
- Soundness, to guarantee realism. Computer generated agents that walk in and out of walls are unconvincing.
- Naturalness, since audiences are attuned to unnatural behaviour in a film setting.
- The ability to cope with environments of moderate complexity (neither trivial nor labyrinthine).
- The ability to cope with continuous environments; agents that are neatly aligned with a grid may be noticeably unnatural, particularly in scenes with a large number of agents.
- 3-D navigation, for sea, sky and air. Currently, such situations are animated by hand (with the exception of ‘Boids’ and one episode of Voyager).
- Real-time navigation for drafting work. Not essential for ‘final cut’ animation.

2.6.6 Typical approaches to cinematic agent navigation

The author was unable to obtain any information on the current trends in cinematic agent navigation, despite attempting to contact several companies involved in making cinematic agent engines[8].

2.6.7 A background to automated agent navigation in games

This section provides a historical context to the use of automated agent navigation in computer games. The research community in computer games has been small and fragmented until very recently. The only clear examples that show navigation techniques were entering use have been drawn from online descriptions, diagrams, and histories of individual games. It is worthwhile to first of all clarify what is meant by agent navigation in computer games.

In a computer game, a computer-controlled player or a human player may give an agent in the game a goal location that is not immediately adjacent to the agent's current position. The agent must then move to that goal location through a sequence of steps, overcoming obstacles (unrealisable regions) that lie between the current position and the goal. This functionality is separate and quite distinct in nature from the AI that controls how the computer allocates goals to its agents (which may stem, in part, from topological maps, AI reasoning approaches, finite state machines and so on).

Based on this definition, the earliest example of navigation may have come in the form of a game known as "Empire" [28] by Walter Bright, which appeared in 1977 for the PDP mainframe. Somewhat similar in nature to the more well known "Civilization" [35] which was published much later, "Empire" featured[9] a form of agent navigation. A player could assign an agent a distant target. The agent would then navigate there following a series of steps. Unfortunately, there is no documentation that describes the behaviour of the navigation routines on encountering an impassable piece of terrain - so it is not possible to be certain that obstacle avoidance of any kind was employed.

The first game that certainly featured navigation (albeit in an dense, maze-like obstacle environment) was Puckman [29], better known as Pac-man [26]. The existence of

[8] Massive, Endorphin, XSI/Behaviour.
[9] Or at least, appears to feature - only screenshots and the manual were found during background research.

navigation strategies in the game can be deduced from an interview conducted with the game's creator, Toru Iwatani [60, 137]. It is worth noting that Iwatani felt that designing the ghost AI navigation was the most difficult part of producing the game. [181].

> "I wanted each ghostly enemy to have a specific character and its own particular movements, so they weren't all just chasing after Pac-Man in single file, which would have been tiresome and flat. One of them, the red one called Blinky, did chase directly after Pac-Man. The second ghost is positioned at a point a few dots in front of Pac-Man's mouth. That is his position. If Pac-Man is in the centre then Monster A and Monster B are equidistant from him, but each moves independently almost 'sandwiching' him. The other ghosts move more at random[10]." [137].

The first example of navigation in a continuous space is 'Rip-Off' [30]. This game involves obstacle avoidance (AI controlled tanks would avoid one another) combined with goal-seeking (the tanks would try to reach an individual fuel tank, wherever it was on the screen) and multi-agent co-operation. Note, however, that there were no fixed obstacles in the environment, so this game might easily (and incorrectly) be mislabelled as not being an example of navigation. Almost uniquely in this domain, the original author of the game has written an article about exactly how the AI worked in the game [222].

Throughout the 1980s, a series of war games and strategy simulations were released that demonstrated agent navigation in open spaces with occasional obstacles. Some of these were turn based (where players took it in turns to move agents over a landscape, with as much time as they liked to make each move), such as "Arnhem" (1985) [32], but some games introduced real-time constraints on agent navigation - the first 'Real Time Strategy' (RTS) games [15]. These early RTS games notably included "The Ancient Art of War" (1984) [31], "Nether Earth" (1987) [33] and "Herzog Zwei" (1989) [34]. By this stage, many games contained agents exhibiting some level of limited obstacle avoidance and goal-directed behaviour.

[10] As a side note, it is not entirely true that the ghosts have any random component, though this claim is made in various sources [20, 62, 137, 181]. This can be deduced from the existence of 'patterns' [21, 60, 136]. These are a series of movements that can be carried out by the player to guarantee they will succeed in completing each level, and indeed the game. The implied determinism of the ghost movement in every game that allows the patterns to succeed means that Pac-Man ghosts must (at best) have only pseudo-random components to their behaviour, and that this pseudo-random behaviour must be reset with the same 'seed' value at the beginning of every game. It is therefore technically incorrect to describe the original Pac-Man ghosts as having random movement.

However, the most revolutionary step, as far as navigation in computer games is concerned, was to come in 1992 with Westwood Studio's release of the groundbreaking "Dune 2" [37]. This game is regarded by many as the genre-defining RTS game. A human and several computer players send units to battle in real time, using a birds eye view of the environment, by simply clicking on the unit, then clicking on the desired destination. The unit would immediately start moving towards the assigned goal, engaging enemy forces encountered en route. On a navigational level, this extremely popular game introduced several novel gaming features simultaneously:

- A large variety of different maps featuring different shapes of impassable obstacles and terrain.
- Large scale environments. In order to allow a player time to develop a base, and to enable strategic use of landscape, the player was only given a small view of the environment at any one time.
- Moving goals - it became possible to send soldiers off to attack a target or guard a vehicle that was moving around in the environment.
- Simultaneous navigation with multiple agents and with different goals for each group of agents.
- Agents treating other moving agents as moving obstacles to be avoided (to avoid collisions between agents).
- Multiple computer opponents, each controlling different sets of agents.
- Navigation took place in real-time, even with multiple large groups of agents navigating to their destination simultaneously, and even on the standard 30Mhz, 4 MB home computer circa 1992.
- (Possibly) limited knowledge navigation. Parts of the player's map of the environment were 'shaded out' if no unit had yet explored them since the start of a battle. It is unclear whether the unit navigation routines were given an a priori complete map to control navigation, or whether agent navigation routines were as 'blind' as the player was, when distant and unexplored goals were selected.

No information has been published by Westwood on the nature of the navigation heuristics used, but certain behavioural features soon became obvious. Particularly, units would

become trapped and unable to reach their destination, as a result of encountering any kind of non-convex obstacle configurations, or as a result of being temporarily blocked by other agents and consequently abandoning navigation. This might seem like a minor irritation, but it was gameplay-destroying for the player who loses the game having sent their army across the map, only to discover that only half of it arrived, and even that, in fragmented groups.

Dune 2 was a huge success though, and was rapidly followed by a series of other well known RTS games that have been played by tens of millions of people worldwide [15], in which automatic agent navigation in obstacle filled environments was a critical part of enabling gameplay. This series of games famously included the "Warcraft" [13] and "Command and Conquer" [12] series of games - within which, new titles are still currently being produced.

Improving RTS agent navigation

How was the problematic issue of failing agent navigation resolved? It transpires that, as was the case in robotics, general, reliable, real-time agent navigation was (and is) unsolved. Instead, a number of 'cheating' mechanisms were built into games.

- Waypoints [17]. These were introduced in 1997 in the RTS game 'Total Annihilation' [40]. Waypoints affect navigation in two ways. Firstly, they enable the player to specify large and complex paths for their agents explicitly, by selecting a series of short term targets for the agents to achieve. Secondly, they overcome the problems of unreliable automatic agent navigation by shifting the burden of navigation onto the human player. The human has some idea of the situations when navigation will fail, and in these situations, they take over navigation manually and specify much of the shape of the path explicitly a priori, rather than merely a single target for navigation. As a solution to the problem of navigation though, waypoints are a failure - shifting responsibility for tedious detail onto the player instead of solving the problem for them.

- Precalculated navigational maps. Here, a roadmap (See Section 2.5.2) is calculated in advance for the environment. If an agent wants to travel anywhere, it finds the way to the nearest part of the 'motorway network' connecting different parts of the map, travels along the motorway, then leaves the motorway when it is close enough to its

destination to attempt short-term navigation.

The first consequence of this is that agents tend to stick to the roadmap no matter what happens in the game. Ambushes can then be set up along the 'motorway' and agents will blindly (and repeatedly) walk into them. Sometimes, blockades can be set up across the 'motorway' using enemy units or temporary environment features such as player bases, again, causing road-map based navigation to fail for agents that encounter the blockade.

A secondary consequence of this, is that when game designers began to allow users the option of designing their own environments, players sometimes had to be instructed to be very patient with the environment building program, while it generated a navigation 'cheat-sheet' for the environment the player had designed, before the complete finished environment could be saved.

- Grid-based approaches. Rather than have a continuous model of the environment in which the battle is taking place, a grid is overlaid upon the battlefield, and units may not occupy the same grid segment. Movement between grid positions appears continuous as a result of the animations used, and the animated units may travel at any angle in moving between grid cells, giving the illusion of a continuous environment. This creates some artifacts, such as a strong tendency among agents to line up on adjacent grid squares with exact precision when grouped together. This has the benefit though of making it feasible for the computer to conduct A* graph search between grid locations, to try and find a route leading to the goal for the agent. Unfortunately, this does not work so well for far-off goals, as conducting deep A* search is still expensive[11]. This grid approach can be combined with a roadmap approach - A* search to move quickly to the 'road', follow the road to the distant region, A* search to move to the target grid coordinate from the road.

- Simplified environments. By far the most common approach, environments were often designed to consist of simple, convex areas, linked by bridges or narrow canyon passes, with no other obstacles to navigation except small convex obstacles that could be easily navigated around.

  This approach also removes some of the artificiality associated with roadmap approaches to navigation. Users are less likely to wonder why all of their agents

[11] A consequence of this is that in some early RTS games, the game could freeze for a few seconds if several agents were sent to a distant location, consequently spoiling the gameplaying experience. Subsequently, time-slicing approaches which limit the amount of computation the AI navigation routines can carry out (per second) have reduced this problem.

are lining up and following exactly the same path, if there is only one narrow bridge exiting from their region of the environment and thus an apparently sensible reason for this behaviour.

- Giving the computer unfair knowledge. In most cases of 'limited knowledge' online navigation, a probabilistic navigation method would be too difficult or expensive (in terms of CPU time) to implement, so the computer or individual units are given access to knowledge of the entire environment, while the human player must visually coordinate behaviour with a limited view of the environment. This can result in players complaining that the computer player is cheating and that its behaviour is unrealistic.

Recent work in computer game navigation

> "Finding high-quality paths quickly in 2D terrains is of great importance in RTS games. In the past, only a small fraction of the CPU time could be devoted to AI tasks, of which finding shortest paths was the most time consuming... the presence of hundreds of moving objects and the urge for more realistic simulations in RTS games make it necessary to improve and generalise path-finding algorithms" [78]

Computer games requiring automated agent navigation are currently branching in several directions. 3-D environments have been showing up for the last few years ("Homeworld" (1999) [42], "Homeworld 2" (2003) [51], "Freelancer" (2003) [50], "Eve Online" (2003-2005) [53]). "Rome: Total War" [57] and similar war-games are moving towards supporting the behaviour of tens of thousands of units onscreen simultaneously [57]. First Person Shooter games (FPS) [14], where the player takes a first-person view of their environment, and role playing games (particularly Massively Multi-player Online Role-playing Games (MMORPGS) [19]) are pushing for a higher quality, almost cinematic experience for players[12].

[12] An excellent recent example of a game attempting to replicate a cinematic experience is the game "Battle for Middle Earth" [54]. The designers of this game advertised it by reproducing a three minute segment of the "Return of the King" [52] film entirely using the in-game engine - given the complexity of the scenes involved, this was a remarkable achievement and demonstrates the complexity already inherent in modern computer games.

Navigation is having a new impact on player experience as new games develop. There are a number of examples of players exploiting in-game navigation algorithm bugs[13]. The cost of a player being able to exploit an flaw in a computer-controlled agent's navigation errors is low in a single player game - the gaming experience is all that is damaged. In a MMORPG, the consequences can be serious and financial, as there are thriving markets trading real-world currency for in-game possessions [16]. Exploitation of in-game bugs can allow virtual cash to be rapidly acquired, which in turn allows some players to become rich in the real world, at the expense of damaging or destroying the gaming environment for all other players, sometimes indirectly via in-game economic inflation. Poor navigation, it seems, can literally carry a price.

There is also a shift towards massively multiplayer environments where thousands or even millions of mobile objects (MOBS) must be simultaneously controlled in the environment. It is interesting to note that the most recent game in the "Warcraft" series (which is already being played by more than 3.5 million people worldwide) is an example of a MMORPG rather than a RTS game. The computers that maintain the virtual environment must now simultaneously control millions of groups of agents interacting with millions of players, rather than a few groups of agents interacting with one or two players.

Last of all, computer game-type environments are becoming increasingly popular in real-time training systems. Generalised drop-in systems for automated agent navigation may come to be in increasing demand in systems ranging from simulator-based driving lessons with simulated pedestrians, to military training systems. A recent and well known example of such a training system is "America's Army" [47], a highly realistic infantry warfare game developed both for recruitment and publicity as well as military training, with a $7.5 million development budget and ongoing costs of $4.5 million annually.

2.6.8 Demands of computer game agent navigation

- In cases where a game experience approaching cinematic quality is required, the requirements for cinematic agent navigation may be relevant (see Section 2.6.5).
- A strong need for good, computationally cheap 2-D navigation, and increasingly for 3-D navigation (space-based games, flight simulators).

[13] For example, 'Eve Online' [53], an MMORPG in a 3-D space environment has suffered from players exploiting a navigation bug and trapping other players ships in an auto-pilot navigation mode, so that they can be attacked easily[22].

> "3D Path-finding cropped up as a perennial problem, just as in previous year's roundtables" [6]
>
> "Developers were keen on discussing ways to speed up path-finding in 3D space" [6]
>
> "In the next year, we will likely see more true 3D games, necessitating the use of path-finding algorithms that work in three dimensions rather than a hacked-up 2.5 dimensions (two dimensions with a small number of third-dimension planes at fixed heights)." [11].

- A strong need to be able to cope with worlds of arbitrary size, and containing some global complexity in structure (though perhaps of far lower local complexity in each region of the environment).
- Online and real-time navigation is often required.

> "Unfortunately, AI research often focuses in a direction that is less useful for games. A* is the most successful technique that AI research has come up with - and nearly the only one applied in computer games. The research community is nearly exclusively concerned with tuning its approaches for computational efficiency and does not care about features such as dynamics, real-time, and software-engineering-related properties." [192]

- Where games feature only a small number of agents, natural behaviour is likely to be desirable.
- Low computational cost is desirable, particularly where games feature large numbers of agents.
- The capability to cope with moving obstacles (and possibly other agents through communication) is desirable.
- Navigation within non-holonomic constraints would be useful in some areas of the games industry: racing simulations, and currently, FPS games with increasingly well modelled in-game physics.
- Easy to understand, easy to implement routines. The games industry is fast-paced, and speed of development is crucial. This is perhaps one reason why A* has dominated the industry so much, as it is an easy to implement and well-documented approach.

> "Genetic Algorithms and Neural Networks are very popular in academic circles but have found little use in the game industry. For the most part they're considered both too unpredictable and too difficult to understand once they are working - its tough to tune what you don't understand." [6]

- Generality - a technique should work across many types of games if it is to be well accepted. In a sense, this precludes the use of subsystems, which make a technique less suitable for 'drop in' navigation.
- Soundness is preferred; agents that collide with obstacle configurations damage the realism of the gaming experience.
- Consistent behaviour. For example, learning or random approaches to navigation are less popular, as they produce unreliable behaviour. Reliability is important to ensure a consistent game experience between players, which is vital for quality assurance and also for technical support.

  > "You'll have all kinds of problems with your QA and Tech people - try explaining to *them* that your AI will never play the game the same way each time, and that you can't necessarily recreate any problems the user might report." [6]

- The capability to cope with *moving goals* is very valuable, since computer game agent behaviours often involve navigation to another agent.

Further coverage of the navigational demands of computer games can be found in [198] and [74].

2.6.9 Typical approaches to computer game agent navigation

> "AI approaches from academia ... are hardly ever used in game development" [192]

The available literature suggests the most commonly adopted approach is cell decomposition (i.e. a grid-based approach) combined with A* graph search.

> "The so-called A* algorithm is the most common basic ingredient for computing a long-distance route for an NPC." [192]
>
> "A* is the most successful technique that AI research has come up with - and nearly the only one applied in computer games." [192]
>
> "A* is the most common solution used by game developers because it works." [6]
>
> " The A* algorithm continues to reign as the preferred path-finding algorithm, although everybody has their own variations and adaptations for their particular project. Every developer present who had needed path-finding in their game had used some form of the A* algorithm." [10]
>
> "The A* algorithm is probably one of the most, if not *the* most used pathfinding algorithm in game development today" [75]

The potential fields approach to navigation has been found infrequently in the gaming industry [64]. Often though, use of potential fields is restricted to virtual force based flocking behaviours in the manner of Craig Reynold's 'Boids' [202, 205].

In recent years, only one game seems to have had its AI navigation strategy explicitly advertised [9], a 3-D adventure game released in 2001 called "Prisoner of War" [44]. Even more curiously, it transpires that the strategy employed is partially based on potential fields. This was so surprising, (in light of the hegemony of A* based approaches), that a short email interview was arranged with the AI developer for this game, Phil Rutherford of Kuju Games [1]. The interview can be found in Appendix B. In brief, it turned out that the game still uses A* search for long distance navigation on a topological map, but uses continuous potential fields for steering. The game is therefore an A* / potential field hybrid.

Finally, a short academic survey of steering in computer games can be found in [230].

2.6.10 Convergence of gaming, cinematography and robotics

Irrespective of the number of agents that are involved, both games and films feature increasingly realistic displays of environments, and continually improving underlying physics for those environments. This trend could be interpreted as a move from merely virtual worlds, to increasingly faux-physical worlds.

Why is this important? As virtual worlds more closely resemble the real world, so should navigation research in computer games and cinema become more closely linked with navigation in robotics (and vice versa).

All three areas - cinema, games and robotics - could benefit from a computationally cheap, natural-looking automated navigation behaviour that is simple to understand and implement. It would also make sense to provide a single approach that is general enough that it can be directly applied to all three domains with minimal effort, rather than having several specialised techniques for each domain.

2.7 Chapter Summary

This chapter has given:

- An overview of the basic problem of navigation and some of the constraints that can be added to navigation problems to make them more challenging.
- An overview of characteristics and metrics of navigation solutions, along with descriptions of the main families of navigation approaches. Table 2.1 summarises the properties of the major families of navigation approach.
- A survey of the application domains of automated navigation as part of the context of this thesis.

It is clear that it would be beneficial to have an approach to navigation that is computationally cheap, and as complete, optimal and natural as possible and that is capable of coping with a range of environments in terms of obstacle complexity and total number of obstacles. It would also be useful to have an approach that is general in nature so that it could be applied to all of the application domains.

Table 2.1: General properties of families of metric navigation approaches.

	Roadmap variants	Cell Decomposition	Geometric (Bug)	Potential Fields
Online navigation?	✗	✗	✓	✓
Computationally cheap?	✗[1]	✗	✓[2]	✓
Simple model/easy to understand?	~[1]	✓	✓[3]	✓
Simple implementation?	✗	✓	~[4]	✓
Robust (vs. noise or errors)?	✓	✓	~[5]	✓
Continuous navigational state space?	✗	✗	~[6]	✓
Can generate trajectories directly?[7]	✗	✗	✗	✓
Can navigate in 2-D?	✓	✓	✓	✓
Can navigate in 3-D?	~[8]	~[8]	✗	✓
Can navigate in n-D?[9]	~[10]	✗	✗	✓
Can model non-static obstacle scenarios?	✗	✗	✗	✓
Can model non-holonomic constraints?	✗	✗	✗	✓
Suitable for non-trivial environments?	✓	✓	✓	✗[11]

[1] Although use of a Roadmap is intuitive and computationally cheap, Roadmap generation is often unintuitive and computationally expensive.
[2] Bug1,Bug2 are extremely computationally cheap, Visbug less so.
[3] Bug1 is obvious, but as the more efficient Bug algorithms are increasingly opaque and non-intuitive.
[4] Bug is straightforward to implement, particularly Bug1/Bug2, but the necessary sub-algorithms can be non-trivial particularly in virtual world scenarios.
[5] Bug algorithms may suffer if errors affect their ability to recognise the current position or circumnavigate.
[6] Technically, BUG does not represent the environment at all in its state space in order to navigate. The sub-algorithms it depends on will generate continuous trajectories, however.
[7] That is, can the algorithm itself output coordinates or effector signals for the agent's direct use, or does it require sub-algorithms to be implemented?
[8] Can navigate in 3-D albeit at much lower resolution representations of the environment, or at far greater computational cost.
[9] In other words, can it be re-applied to inverse kinematics - navigation between joint-configurations in a high-dimensional space, e.g. robotic arms?
[10] Possible (via silhouette approach) but non-trivial and computationally expensive.
[11] Unfortunately, this is by far the most important property in this table.

Chapter 3

Context: Potential Fields

Guide to Chapter 3

This background theory and context chapter introduces the second branch of computational theory in which this thesis is set. The 'Artificial Potential Fields' general heuristic approach offers a metaphor based on the physical phenomenon of potential fields. The potential fields metaphor has been employed throughout the field of artificial intelligence as a problem solving approach, enjoying particular success in neural networks.

The metaphor suggests that if a problem can be modelled by a function that assigns a value to each state configuration (position) in a continuous state space based on its usefulness, then the optimally useful state configuration can be found by minimising the value of the function.

The power of the technique derives from the typical simplicity and elegance of models and heuristics that result from it. This leads to a general approach that is easy to comprehend, computationally cheap, highly visualisable and capable of coping with both continuous and discrete multi-dimensional problems. The weaknesses of the technique come from the difficulties that can ensue in trying to find optimal states in a multi-dimensional continuous space. These difficulties are mainly caused by only a few categories of surface feature. The nature of these surface features is discussed.

3.1 Chapter Overview

This chapter presents an introduction to the Potential Field (PF) method in its general form, and gives an overview of common heuristic approaches used to address problems represented as potential fields. Basic PF terminology is defined. Several key problems for general PF methods deriving from surface properties are highlighted - the local minimum problem, the plateau problem, the ravine problem and the saddle problem.

3.2 Terminology

3.2.1 General Potential Fields

Potential Fields (PFs) can be found in the literature of many scientific disciplines, including applied mathematics, computing, physics and chemistry. They occur in the physical space of the real world (gravitational potential, electrostatic potential) as a result of natural forces. Potential fields associate coordinates within a space with scalar or vector *potential values*. In the real-world, these potential values are often energy levels (a scalar field) or physical forces (a vector field).

Artificial potential fields are analogues of real-world potential fields, used throughout the field of Artificial Intelligence (AI) to model a variety of problems. Transformations of AI problems into PF problems, by representing the desirability of each possible AI problem solution as a potential value, are intended to allow the application of *potential field heuristics*. These heuristics use optimisation techniques to travel through the *configuration space* of the potential field (which contains all possible solutions of the original AI problem) towards a useful or optimal solution.

In this thesis, the following definition of a *potential field* or *potential field function* is used:

Definition A *Potential Field Function* is a mapping from each possible coordinate in an n-D space to either a scalar or vector value. The coordinate space may be discrete or continuous. If the value that is mapped to by a continuous potential field function is a scalar, then the corresponding *Potential Field* is normally a continuous n-D surface in a $(n+1)$-D space.

By 'normally' and 'reasonably', what is meant is that most potential fields are represented by simple mathematical functions that will not generate discontinuous surfaces. Totally continuous or, better still, mathematically smooth potential fields are desirable in PF research, as they make it more practical to solve problems using the field. Throughout this thesis, it will be assumed that the potential fields being discussed are continuous scalar fields, unless a vector potential field is specified.

3.2.2 General Potential Fields: Example

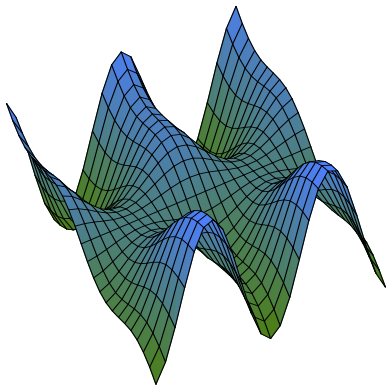


Figure 3.1: An example of a potential field.

The 3-D space in Figure 3.1 contains a 2-D surface. Many 3-D potential fields might look similar to a physical landscape when plotted. This intuition of a field resembling a physical landscape is useful, and will be used in later examples.

3.2.3 Potential Gradient Fields

Continuous scalar potential fields (usually those represented by mathematical functions) have a corresponding vector potential field implicitly defined, known as a gradient field. This gradient field is used directly by many traditional potential field based heuristics.

The gradient field represents the local variation in the scalar potential value when looking at points closely surrounding a particular coordinate, and is equivalent to a vector potential field operating over the original n-D coordinate space.

Since many potential fields are represented by continuous mathematical functions, it is often possible to derive a gradient field directly by differentiation.

Definition A Gradient Field or Potential Field Gradient Function is a mapping from each possible coordinate in an n-D space to a vector value. It is typically defined by differentiating a continuous potential field function (piecewise, if there are some minor discontinuities).

3.2.4 Potential Gradient Fields: Example

Consider the problem of finding the lowest point on a physical landscape. One practical way to attempt this would be to allow the physical gravitational potential field to solve the problem for you. You would allow a ball to roll downhill from a number of random positions on the landscape, and observe where it eventually became stuck.

The potential field in this example is simply the shape of the landscape. The height of each position on the earth's surface corresponds to a scalar value in terms of gravitational potential. The gradient at each point on the potential field can be seen by observing the speed and direction in which a ball would start rolling if it were released from a particular point. The gradient field is the collection of all of these vector values at each point in the landscape, and is constructed by summing the gradient of the earth's gravitational potential field with the reaction force provided by the landscape at each point. Figure 3.2 shows part of the gradient field corresponding to the potential field surface in Figure 3.1.

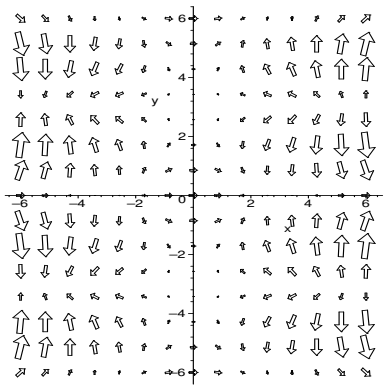
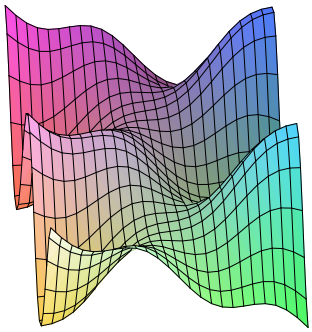


Figure 3.2: An example of a potential gradient field.

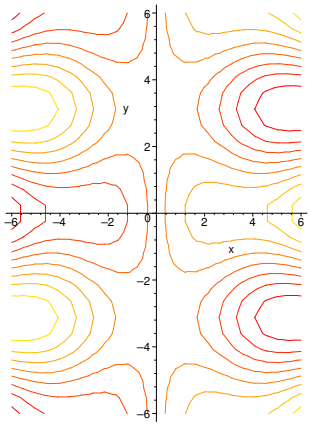
This simple 'ball rolling down a hill' analogy has a strong correspondence with many of the approaches taken to date in applying the potential fields metaphor to modelling and solving AI and navigational problems.

3.2.5 Visualisation of Potential Fields

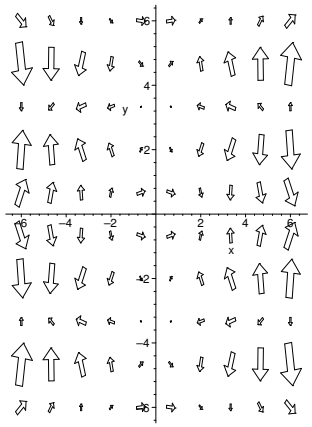
Surface rendering, *contour maps*, *gradient maps* and *flow maps* are techniques that can be used to help visualise a potential field. Examples of each of these are shown below in Figure 3.3.



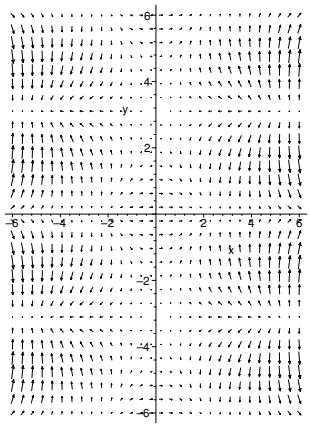
(a) 3-D surface rendering of the potential surface.



(b) Contour map, showing equipotential lines.



(c) Gradient field, showing the strength and direction of the gradient at discrete points.



(d) Approximation of a flow map, which shows continuous lines of downward potential flow.

Figure 3.3: Examples of visualisations of potential fields.

3.3 Common heuristic approaches used with PF surfaces

3.3.1 Aim of the heuristics

The potential value of a point corresponds to the desirability of that position's coordinates within the configuration space. If an optimally desirable solution is being sought, then a heuristic is needed to try and obtain a corresponding point of global extremum on the potential field surface. This extremum could be either a minimum or maximum, depending on whether the designer of the potential field has chosen for high potential value to indicate 'highly desirable' or 'highly undesirable'. From here on, unless otherwise specified, it will be assumed that a high potential value means a highly undesirable position in the configuration space, and so the lowest point on the field, a point of *global minimum potential*, will be sought for an optimal solution.

It may be useful to note at this point that the potential field representation of a problem may not always capture all of the constraints or conditions of a problem. For example, some common conditions that may exist are:

- It may be that perhaps only one attempt at solving the problem is permitted.
- It may be that certain paths or positions on the potential field do not correspond to usable or permissible solutions in the original problem.
- It may be that the path taken by the potential field heuristic represents the solution, or it may be that the final position reached in the space represents the solution.
- It may be that certain knowledge is (or is not) available to the implementor or the heuristic; for example, the heuristic may be given knowledge of the location of the global minimum, or the implementor may have knowledge of the approximate shape of the potential field that may be faced.

Constraints and conditions may need to be imposed manually upon the descent technique in order to achieve useful solutions via the potential field method. For example, a descent approach which selects unusable solutions regularly may need to be ruled out immediately, or it may be necessary to check a solution (or each stage in finding a solution) to ensure that it is valid, and perhaps take some other action if it is not.

Below, descriptions are given of the two most well-known classic heuristic approaches that attempt to identify a point of optimal potential. The two heuristics can be said to represent opposite ends of a spectrum of optimising heuristics: an entirely deterministic technique from numerical analysis, and an entirely non-deterministic approach based on statistics.

3.3.2 Gradient Descent

Background

Gradient Descent (aka Steepest Gradient Descent) is a classical iterative general optimisation heuristic from numerical analysis, which is intended to minimise a function f that is dependent on a set of parameters, using the derivative of the function [215].

When global maxima are being sought rather than global minima, the corresponding *Gradient Ascent* heuristic can be used. Since the potential field functions in this research were designed to place the goal position at a global minimum, *Gradient Descent* will be referred to throughout the rest of this thesis.

Intuition

The idea behind Gradient Descent can be illustrated by the following example.

Consider a mountain climber trying to find his way back down from the side of a mountain in the fog. He can tell how the ground is sloping around him, but only very locally within his field of vision. He must pick a direction to travel in. He reasons that the best way to reach his car at ground level (the global minimum, here) is to travel downwards wherever he can, and to take the steepest descent possible, to enable him to get there as quickly as possible. Every ten seconds or so, he checks to make sure he is still travelling in the direction of steepest descent and alters his course as necessary to ensure he travels as quickly as possible downwards.

Alternatively, consider a ball allowed to roll down the side of a hill. The ball would be expected to roll preferentially along the steepest slope downwards, taking into account its momentum. To imagine what it would look like without momentum, think of a ball rolling down a sticky surface, and visualise it following the steepest path available at each point.

Definition

A function mapping a number of real parameters to a value, $f : R^n \rightarrow R$, can be locally minimised using Steepest Gradient Descent (the most common way of using Gradient Descent) by following these steps:

1. Start at an initial position p_0.
2. Determine the direction from p_0 that results in the fastest decrease in the value of f by calculating and negating the derivative of $f(p_0)$: i.e. -$\nabla f(p_0)$.
3. Move some distance along this direction, and call the new position p_1.
4. Repeat from step 1, using p_1 in place of p_0.

Detail

There are some other gradient descent techniques based upon the steepest descent approach given above, such as conjugate gradient descent and quadratic bowl descent. However, the steepest descent method described above is the one presented in most literature on potential fields for navigation [108, 127, 161] and machine learning approaches such as neural networks [72, 171]. Throughout the rest of this thesis, 'gradient descent' will refer to the steepest gradient descent method.

Commonly used variations within the steepest descent method include using either a fixed step size or a varying step size - where the step size can vary, steps are typically taken in proportion to the magnitude of the gradient in step 3 above.

Strengths of the approach

- Gradient Descent is *concise*, with only a few steps to be implemented within an iterative loop. This means it is straightforward for programmers to implement, and that it can be implemented even on robotic controllers that have very limited storage space for software.
- Gradient Descent is *computationally cheap*. Potential field functions are typically chosen so that they are easy to differentiate a priori by analysis. The result is that

when using Gradient Descent, only a single equation representing the gradient needs to be numerically evaluated each time round the loop. This is beneficial within many application domains; neural networks, where fast training is desired; robotics, where computational speed may be limited; and gaming or automated cinematic agent control, where a large number of agents may need to be controlled simultaneously, limiting computational resources per agent and making a computationally cheap solution desirable.

- Gradient Descent is *direct* - it travels as quickly as it can directly towards a minimum[1] and does not follow a meandering path in terms of movement down the field. This can be useful where the underlying problem requires a direct path to a minimal solution - for example where the path is representing some natural activity or phenomenon that should occur in a 'direct' manner.

- Gradient Descent is *easy to understand*. It would be surprising if an undergraduate student did not understand the intuition of a ball rolling down a hill, as a way of finding the lowest points on the hill. It seems reasonable to say that this improves the likelihood that implementations will be carried out correctly. From the prominence of Gradient Descent throughout all artificial potential field research, it can be reasoned that its simplicity has improved the likelihood that someone will consider using the algorithm within their application.

- Gradient Descent is *defined for n-D spaces*, and the computational cost of calculating each step rises only linearly with the dimensionality of the space. This makes it suitable for finding minima on potential surfaces of high dimensionality.

Weaknesses of the approach

Although Gradient Descent has useful properties stemming from its simplicity, it is vulnerable to several equally simple problems. The most well known of these are *local minima*, *ravines*, *plateaux*, and *saddle points*. Since the details of these problems are quite important to this thesis, they are merely summarised below and elaborated upon in Section 3.4.

Local Minima These exist whenever there is a point P on a surface where all of the local gradients surrounding that point lead steepest gradient descent back to point P, but

[1] Though not necessarily a global minimum.

point P is not the lowest position on the field (the global minimum). Local minima are the most significant problem for almost all descent heuristics, but particularly gradient descent, which suffers from complete termination of progress when a local minimum is encountered.

Plateau A plateau is an area of the potential field surface that is nearly flat, so that it is difficult to obtain a strong direction towards the global minimum. They are mainly a problem for gradient descent where the step size is linked to the strength of the gradient, significantly slowing progress for this form of gradient descent. The flatness of a plateau can also mean that the gradient can vary abruptly in direction over short distances, which can affect all forms of gradient descent.

Ravines Ravines are areas of a potential field where a slope leading downwards is surrounded to the left and right by steep slopes upwards on either side (like a ravine in real life). They are primarily a problem for techniques that use momentum. However, in all forms of gradient descent they induce oscillations in the descent path that can be problematic within many problem domains.

Saddle points Although usually unproblematic, saddle points can sometimes be a significant, progress-halting problem for gradient descent under certain conditions [69].

Other notes

- Gradient Descent often fails. The most common solution to this problem is to simply execute the heuristic many times under different initial conditions. Usually, if the problem type allows it, gradient descent will be initialised from a number of different starting locations, and left to find whichever minima it can, in order to eventually find the global minimum. If the problem does not allow this, gradient descent can be initialised with varying parameter settings to control step size, or the behaviour can be altered by simulating different amounts of 'momentum' in the travel.

3.3.3 Simulated Annealing

Background

The *Simulated Annealing* heuristic for combinatorial problems was first presented by Kirkpatrick et al. in 1983 [131] as a generalisation of the Metropolis Monte Carlo scheme [182]. The inspiration for this approach comes from the metallurgical process of annealing. Annealing is a technique used by blacksmiths and metallurgists to produce high quality steel, by controlling the way the steel is cooled. If molten steel cools too quickly, cracks and bubbles form in the metal, which disrupt the surface and the internal integrity of the final product. In order to bring the steel safely to the lowest energy state (e.g. allow it to set into shape and become cold), a 'cooling schedule' is used.

Intuition

The agent is given a 'temperature', much like an atom within the metal. This temperature allows the agent to sometimes travel upwards on the surface as well as downwards. As the temperature is cooled, the agent loses its ability to travel upwards. Given infinite time and an appropriate *cooling schedule*, the agent is guaranteed to end up at the lowest point on the landscape, no matter how many local minima are present to trap the agent as it moves over the surface.

This is much like attaching a motor to the ball in the 'ball-rolling-down-a-hill' analogy. The ball can now randomly travel up the slope if it chooses to, instead of following the steepest slope downhill constantly. Providing this freedom is exercised in a particular way (i.e. randomly and over an infinitely long period of time), the ball is statistically guaranteed to eventually settle in the deepest part of the surface.

Definition

A function mapping a number of real parameters to a value, $f : R^n -> R$, can be globally minimised using Simulated Annealing by following these steps, where T represents the current temperature at each step according to the cooling schedule being used:

1. Evaluate f at an initial position p_0.

2. Consider making a transition in a random direction from p_0 to a new position p_1.
3. Evaluate f at p_1.
4. If $f(p_1) \leq f(p_0)$, move to p_1.
5. If $f(p_1) > f(p_0)$, move to p_1 with a probability equal to the Metropolis probability $P(p_0, p_1, T) = e^{-(f(p_1)-f(p_0))/T}$.

Initially this should result in a random walk through the search space. As time goes on, it should result in steepest descent steps being taken most often. Further description of the behaviour of this heuristic can be found in [131].

Strengths of the approach

- This approach is guaranteed to reach the global minimum with an appropriate cooling schedule.
- This approach demonstrates that certain types of random movement are sufficient to overcome local minimum problems that are present, given enough time and controlled application of the random movement. Like gradient descent, the Simulated Annealing approach generalises to n-D, so that it can find a global minimum on an n-D potential surface.

Weaknesses of the approach

- In general it takes infinite time for a solution to be guaranteed, and although finite time cooling schedules may be created with knowledge of the type of surface faced, Simulated Annealing still takes a very long time to settle at the lowest point on the potential field. This is very likely to make it impractical for most problems.
- The path taken over the potential field is long and meandering. This is not a desirable property for certain problem environments where efficient or natural-agent-like path shapes are sought in addition to achieving the global minimum position.
- By allowing occasional uphill movement, simulated annealing carries the risk of breaking the rules of the problem environment being dealt with, if the modelling of the underlying problem implicitly assumes that the agent will always be using

gradient descent and travelling downhill. Therefore although simulated annealing allows travel to reach the global minimum, the resulting path generated may still be unacceptable for the problem being faced.

Other notes

Simulated Annealing has been outlined here to provide an example of a useful controlled random-search based strategy, and because (like gradient descent) it is an extremely well known general potential fields approach that is diametrically opposed to the efficient but non-robust behaviour of gradient descent.

3.3.4 Summary

Gradient Descent is an approach that has many flaws (as a consequence of surface features) both in theory and practice, but which is straightforward to implement and stands some chance of providing a useful result in application. Simulated annealing is theoretically flawless, but requires either infinite time to be sure of success, or that the problem is suited to a fast cooling schedule (which is usually hard to identify).

Most practical applications of potential fields to problems in computing rely on gradient descent based approaches (neural nets, existing navigation approaches) for reasons of practicality; in many cases it is sufficient to find a solution by running gradient descent many times. When the problem type prevents this approach from working, some techniques use the principle of occasional random behaviour to augment the behaviour of the descent heuristic. Direct use of 'full-blown' Simulated Annealing in AI is much less common than gradient descent.

It will be useful to keep in mind that Gradient Descent follows direct paths that are often too direct to allow success - i.e. it cannot travel backwards, out of a local minimum. Conversely, pure Simulated Annealing follows impractically long, meandering paths that lack sufficient directness to be of practical use for most problem types. More ideal approaches can perhaps be found between these two ends of the spectrum of potential field heuristics.

3.4 Types of problems for gradient descent heuristics

The problems described in this section are problems that afflict descent heuristics for potential fields across all problem domains. All of these problems can occur in n-dimensional spaces, but in diagrams below are demonstrated using 3-D potential field examples.

3.4.1 The Local Minimum Problem

The *Local Minimum Problem* (LMP) for gradient descent techniques is almost certainly viewed as the most serious problem faced across all areas of research utilising potential field techniques.

> "Local minima remain an important cause of inefficiency for potential field methods." [138]

The LMP occurs when gradient descent upon a potential field encounters a temporary dip in the potential field landscape, within which the gradient at all points surrounding some local minimum position P directs the descent towards P. The local minimum P can be said to be an attractor, and the positions whose gradients direct descent towards P form the attractor basin. Steepest descent finds itself being continually directed towards the local minimum at the heart of the basin. At the local minimum position itself, the gradient vector is undefined, as it has zero magnitude, and should the agent reach this position, gradient descent is left without any indication as to where to move next. Examples of local minimum problems are shown in Figures 3.4 and 3.5.

The problem is that once gradient descent has entered such a minimum, no more useful progress can be made. It is possible for many of these local minima to exist on a potential field, in addition to the global minimum. In some potential fields problems, it is not possible to determine if the current minimum is a local minimum or a global minimum (i.e. neural networks [72], automatic system configuration [219]). In other problems, the location of the global minimum (or an estimate of it) is known a priori (i.e. navigation [159], and certain forms of neural network training [171]).

A local minimum can be described as shallow if the minimum does not have a large local minimum basin surrounding it, which means that the agent is less likely to encounter the

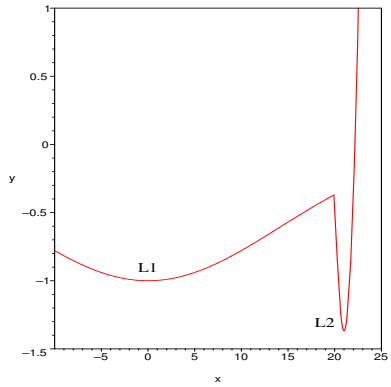
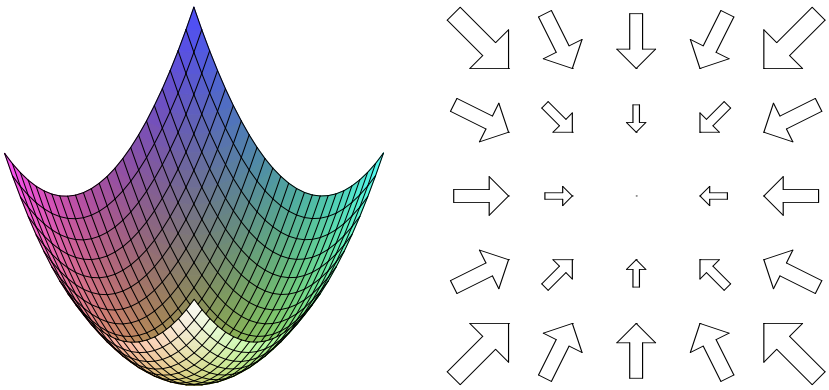


Figure 3.4: A single variable potential function. L1 is a local minimum, L2 is both a local minimum and a global minimum.



(a) Potential surface showing a local minimum.

(b) Gradient field corresponding to a local minimum.

Figure 3.5: Example of a local minimum for a 2 variable function.

minimum, and that the agent may stand a small chance of escaping the minimum with some variants of gradient descent. These variants (some are detailed below) are able to deal with shallow minima by modifying their behaviour so that descent on the field is not simply determined by the gradient at the current position on the field, but also by other factors, such as parts of the agent's path history represented in the agent's internal state. This is often enough to allow descent to continue to travel 'through' shallow minima basins, allowing an increased chance of success of reaching the global minimum.

3.4.2 General techniques used to overcome the LMP

A number of general heuristics are available that try to reduce the effect of the LMP, or even overcome it. A number of these are detailed below.

- Some techniques rely on the idea of momentum to try and keep travel moving through shallow local minima [72]. In the real world, momentum acts as a state variable that is the integral of forces acting over time upon the agent. In potential fields parlance, gradient descent with momentum is simply descent that takes into account the gradients at previous positions by summing them together into a momentum term. Often, a discount factor is applied so that less-recently visited positions have less influence upon the momentum value, a little like friction in the real world. Momentum is a passive strategy in that it is a continuously present part of the descent behaviour that does not rely on being 'activated' in the presence of a local minimum.

- If it is possible to detect the minima by noticing a lack of progress or examining the shape of the potential field, a more active alternative is to temporarily resort to particular escape strategies when the agent finds itself in a minimum. One such escape strategy might be random movement, somewhat akin to simulated annealing. Another approach, suitable for some problem domains (including navigation), might be to abandon the potential fields technique completely on encountering a local minimum, change to a different problem solving approach and then return to potential fields heuristics once the agent is away from the problematic position in configuration space.

- Other active techniques alter the potential field, attempting to 'fill' the minimum by raising the potential values at positions surrounding the minimum. This can work on discrete potential fields where positions correspond to grid locations, and reshaping the field is simpler [64].

- Another family of approaches try to avoid local minima by using special mathematical functions to build the potential field that do not introduce local minima. This approach is found particularly within the robotic navigation community and is covered in Section 5.3.

Generally though, the solution opted for within potential fields research has been to try to recognise when a local minimum has been encountered by either explicitly noticing that no further significant progress in field travel is being made, or implicitly discovering lack of progress by having some cut-off number of iterations of movement after which descent is considered to be complete if no sufficiently deep minimum has been found (sufficiently deep being decided by evaluating the potential against some threshold 'acceptable' value). Typically, if descent is considered to be suffering from a lack of progress, then travel

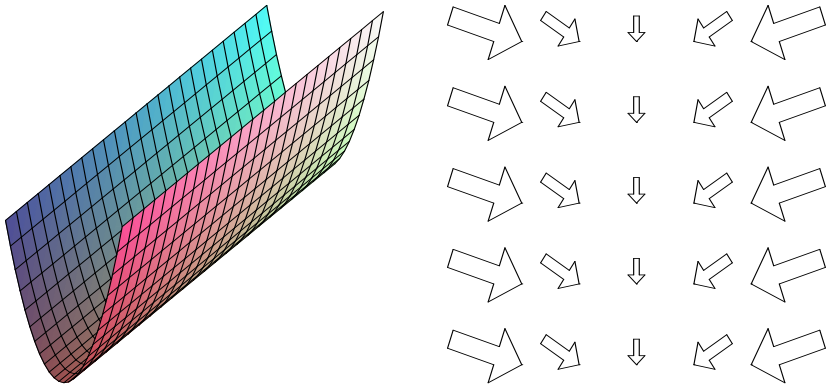
over the field is abandoned and steepest gradient descent is then restarted from another, hopefully better position[2]. The fact that gradient descent is computationally cheap (merely the iterated evaluation of the gradient function at various points on the field) makes this approach quite feasible; and in some scenarios (such as neural network training) it can be sufficient to allow the user of gradient descent heuristic to ignore rather than overcome the problems of local minima in this way.

When the global minimum is unknown, it is usual to restart a large number of times from random positions and then consider the deepest minimum found from any position to be the best candidate for the global minimum. Sometimes this strategy may not be an option, if a restart is unviable due to the nature of the problem, i.e. if the travel over the potential field is required to be successful on its first effort. This can be the case in domains where real-time response or online learning is required.

Local minima can present further unique difficulties to particular types of problem depending on the frequency with which they occur, their depth, and the limitations upon the nature of travel over the potential field that exist as part of the problem specification.

3.4.3 The Ravine Problem

The ravine problem is much less serious than the local minimum problem, in that it usually merely impedes potential field travel rather than preventing it entirely.



(a) Potential surface showing a ravine.

(b) Gradient field corresponding to a ravine.

Figure 3.6: Example of a ravine.

The problem is caused by a part of the potential field that is shaped like a ravine. While it is

[2]In some potential field problems, such 'restarts' are not feasible due to the nature of the underlying problem.

useful to travel towards the ravine floor from the sides of the ravine, some techniques may overshoot the ravine floor and travel some distance up the other side of the ravine, causing a symmetric problem on the other side (see Figure 3.6). This is inconvenient because now part of the travel that is taking place is being wasted travelling unproductively up and down the sides of the ravine, rather than straight down the middle of the ravine (which is the most effective way to travel towards the global minimum). Particularly, this problem can affect gradient descent with momentum, and gradient descent without momentum but with a sufficiently large fixed step size.

Momentum causes the agent to continue to travel up the other wall upon reaching the ravine floor during continuous travel[3]. A large step size causes a zig-zag behaviour as the agent jumps from one side of the wall to the other, never actually touching the ravine floor.

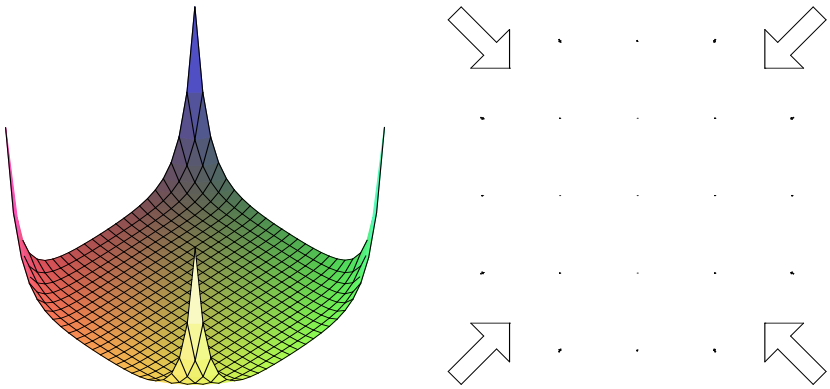
It is possible though for the ravine problem to become very troublesome in exceptional circumstances. If travel is sufficiently impeded by the oscillating behaviour induced by the ravine, then the agent may be wrongly considered to have stopped making progress by a heuristic that measures progress across the field or total number of steps taken in gradient descent - i.e. a ravine slowly leading to the global minimum may be wrongly interpreted as itself being a local minimum.

The winding, oscillatory behaviour induced by ravines is an interesting feature in itself, even if it does not prevent the global minimum from being reached. This shape of movement over the field may itself be considered highly undesirable within the particular problem domain that potential fields are being applied to, particularly if the 'wobbly' unstable path generated corresponds to some kind of 'wobbly' unstable path in real life.

3.4.4 The Plateau Problem

Plateaux also impede rather than prevent progress. They are mainly of consequence for descent heuristics based upon gradient descent with a variable step size. In certain areas of the potential field, the field can become so flat that the magnitude of the direction suggested by the field gradient is very small. If the agent's rate of movement is proportional to this gradient, then movement over the field may effectively halt. This can be seen in Figure 3.7.

[3]Momentum sometimes alleviates the ravine problem too, if the momentum term is small enough in relation to the local gradient term that the unhelpful oscillatory behaviour is damped down rather than amplified or sustained. This is unlikely to occur by chance though, and momentum more often exaggerates the problem of ravine oscillation.



(a) Potential surface showing a plateau.

(b) Gradient field corresponding to a plateau.

Figure 3.7: Example of a plateau.

A flat gradient can also mean that it is easier for the gradient to reverse direction in the space of only a few iterations of gradient descent; whereas a steep gradient can ensure some continuity in the agent's movement. This can cause meandering progress over the field.

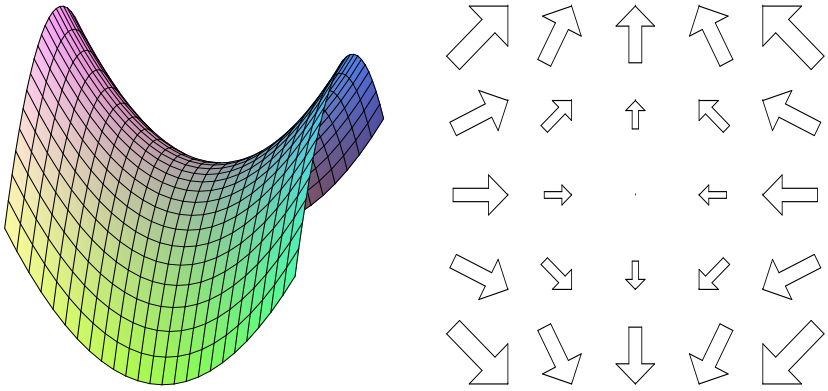
Under variable step size gradient descent regimes, a problem sometimes occurs where the gradient of the field can vary between very different magnitudes when on the plateau and when off it - essentially there are very flat and very steep parts of the field. Here, travel either moves very slowly over the plateau and ultimately never reaches the goal, or, if the rate of movement relative to the gradients magnitude is scaled up, then upon reaching a steeper non-plateau part of the field, the agent very rapidly moves off in that direction with a huge step. Neither of these behaviours is desirable.

The plateau problem can be generally mitigated by using a model of momentum to accelerate travel over flat areas of the ridge, or by utilising fixed step size descent.

3.4.5 Saddle points

Saddle points have the form of an unstable attractor - the direction of the gradient is undefined at the saddle point (as it is at a local minimum), but around this point, only some positions have gradients leading back to the saddle point position, while others lead away from the saddle point. This is shown in Figure 3.8.

Most problems involving saddle points are caused by movement being initialised at a saddle



(a) Potential surface showing a saddle point.

(b) Gradient field corresponding to a saddle point.

Figure 3.8: Example of a saddle point.

point position (the agent is thus not able to obtain a gradient for descent), or by movement being initialised in some region surrounding a saddle point where movement can be forced to oscillate around a saddle point. Accordingly, saddle points are not so much a general problem for potential field heuristics, but are merely a field feature that can be a problematic for particular cases of potential field approaches. Some examples of when these cases can occur are detailed in [69].

3.5 Chapter Summary

This chapter has presented:

- Definitions of the key terminology used in this thesis relating to potential fields, and of the standard approaches used within general potential fields.
- Descriptions of the key techniques and key problems faced within general potential field application.
- An overview of the general application of gradient descent-based and statistical methods to potential fields to find optimal configurations.
- An overview of the surface features that are problematic for gradient descent.

Chapter 4

Context: Potential Fields for Navigation

Guide to Chapter 4

Having provided some background to the problem of *Navigation*, and having introduced the problem-solving *Potential Fields* approach, the thesis now considers the amalgamation of this problem and problem-solving approach.

4.1 Chapter Overview

This chapter is in 4 parts :

- Motives for the use of potential fields for navigation.
- Mechanisms for modelling navigation problems.
- A worked example of potential field based navigation.
- A discussion of the problematic surface features associated with potential fields for navigation and their effects on navigation.

4.2 Why use potential field methods for navigation?

4.2.1 General PF benefits: Understandability, visualisability, ease of implementation

> "One of the reasons for the popularity of this approach is its *simplicity* and *elegance*. Simple PFMs can be implemented quickly and initially provide acceptable results without requiring many refinements." [134]

What properties do potential fields have that are useful for navigation? One useful property is that the intrinsic behaviour implied by the potential field model is *extremely straightforward to understand*, simply the metaphor of a ball rolling down a hill, towards the solution which lies at the lowest point on the surface. The particular transformations that can be used to model the potential field landscape for navigational tasks, (such as those given later in Section 4.3) can also be quite straightforward.

> "Path planning using artificial potential fields is based on a powerful analogy..." [109]

Specifically, there are two metaphors that can be employed throughout potential field based navigation that make the intuition of the approach easy to comprehend. The *potential*

metaphor for navigation is to suggest that each position has a certain desirability in the task of navigation (with the goal being of optimal desirability and the starting position of non-optimal desirability), and that by trying to optimise the desirability of the current position by moving to better positions, you generally achieve navigation. The *gradient metaphor* for navigation suggests that the agent is being 'pulled' by the goals or 'pushed' by obstacles in the environment, and that the resulting navigational behaviour is based on the summation of all the pushing and pulling that is occurring to the agent at any time. It is interesting to observe that both metaphors are simultaneously valid, and so a student, implementer or researcher is free to understand the technique using whichever they prefer.

Potential fields are also *highly visualisable*[1], which assists in understanding the behaviour of this approach. The range of visualisation techniques available in general potential fields methods allow this method a large number of visual metaphors to assist understanding of the behaviour of the technique and the solutions it provides.

As well as being straightforward to understand and visualise, potential field based navigation is *straightforward to implement*. Once a mathematical model of the navigational environment has been chosen, it is just a matter of converting the obstacles and goal into their corresponding mathematical potential and gradient functions. When these functions are in hand, all that is needed to carry out navigation with simple gradient descent is for the gradient to be re-computed iteratively. The entire process of field-modelling and navigating can often be encapsulated in around ten lines of implementation-level code.

These three properties - simplicity of metaphor, wide range of visualisation and ease of implementation - make the technique appropriate for teaching, where simple models with visual metaphors are useful, and straightforward implementations appropriate for practical work - and research, where simple theoretical models are always desirable and where less time spent programming allows more time thinking and modelling. Simplicity in both the model and typical implementation is an attractive property for industrial applications, where implementation time has a premium, and where programmers may be unkeen to learn or implement mathematically advanced or difficult to comprehend strategies in order to achieve navigation in even simple environments.

These useful properties are true of potential fields wherever they are applied, and are generally desirable in solution approaches in all areas of research. However, there are further properties of potential fields that are specifically useful in the case of navigation.

[1] See Section 3.2.5.

4.2.2 Continuity of the model

> "Path planning based on artificial potential fields has a number of attractive features: spatial paths are not preplanned and can be generated in real time; planning and control are merged into one function, which simplifies the overall control structure; smooth paths are generated; and planning can be coupled directly to a control algorithm." [109]

Potential fields, unlike discrete or symbolic AI navigation techniques, possess the capability to *have a continuous model of the continuous space of the problem*, and furthermore they allow a continuum of responses to be provided through the gradient (which is also defined through a continuous space). This means that a navigational response can be defined for all possible points that may be reached in a space without approximation of locations. This means there is no need for a discretisation (e.g. such as a grid or a graph) that is not present in the original environment to be implicitly or explicitly applied over the model of the environment. The continuity of response has another benefit; in some applications, potential fields are applied to directly provide effector control without the need for a lower-level system to convert desired navigation control to effector output. In other words, in real-world applications, the computed gradient can be used to directly control the motors of a robot[2]. Similarly, in virtual world scenarios, an artificial agent can be moved directly according to the gradient - there is no need for further processing of long-term navigational instructions by additional lower-level local navigation routines.

4.2.3 Computational cost

> "In comparison to other methods, potential field methods can be very efficient." [159]

Potential fields are also generally *computationally cheap* to work with in navigation, particularly when the potential function has been analytically differentiated to provide the gradient in advance. The cost of evaluating the gradient or the potential at any position is proportional to the number of terms in the relevant equation, and to the complexity of evaluating each term of the equation. In practice, the terms in the equations tend to be low

[2]In practice, this is one of the most common applications of potential fields - as a local navigation technique, to turn longer term instructions from another navigation technique into continuous patterns of short term effector output.

order polynomial functions, which are computationally cheap to compute, or exponential or square root based functions, which can be calculated immediately using lookup tables, or in the worst case using a few iterations of a numerical technique such as Newton-Raphson. This property has endeared potential fields to roboticists who have been plagued with only limited computational facilities on physically embedded computers. Low computational costs are also a boon to those working with agents in films or computer games, where large numbers of agents may need to be simultaneously controlled. In these application domains there are often hard real-time limits on available computation time and there is usually a strong need to devote CPU cycles to other aspects of the virtual world such as graphical rendering, physics modelling, and in the case of gaming, network delay compensation and 'CPU player' game strategy AI [6, 10].

4.2.4 Suitability for online and offline navigation

Potential fields can be applied to navigational problem scenarios requiring *online navigation*. An agent may begin with an incomplete awareness of their navigation environment, with obstacles only being discovered while navigation is occurring. In this scenario, the agent's navigation approach must have the flexibility to incorporate new information about discovered obstacles during execution of the heuristic or algorithm. Potential fields can permit this by allowing the field function to be modified dynamically during execution. Potential fields can also be used for offline navigation, where knowledge of an environment is available a priori - which is considered a simpler problem.

4.2.5 Suitability for problems in high dimensionalities

The potential fields method can be applied to problems of any dimensionality. Unlike geometric approaches, whose properties depend on the environment being essentially 2-D, and graph-search approaches, which tend to suffer when highly dimensional problems are encountered, the potential fields method retains the properties of understandability, ease of implementation, computational cheapness, continuous modelling and online navigation suitability as the navigation problem moves from a 2-D environment to a 3-D environment, or even to an n-D environment (in the case of inverse kinematics). This is particularly useful as it means that problems involving navigation between configurations in highly dimensional navigation environments - such as control of the joint-angles of fingers while

navigating between two possible hand positions - can be solved using the same basic approach as more traditional 2-D environment navigation problems.

4.2.6 Extensibility

Traditional potential field navigation uses the goal and obstacles to generate a field, and gradient descent to navigate on the field - but why should this be all that navigation involves? The desirability of locations may be affected by other factors of navigation. In potential field based navigation, this might include the ability for obstacles themselves to move and for the agent to anticipate this in its field-derived behaviour. It is also possible to weight certain parts of the field as being of higher or lower value to represent easy or difficult terrain, or to represent non-holonomic constraints [221].

4.2.7 Summary

Potential fields offer several simple to understand metaphors, have many forms of visualisation, and are easy to implement. Potential fields can directly model continuous spaces, are computationally cheap, and are suitable for both online and offline navigation. Particularly, extensibility and usability with high-dimensional problems are useful and interesting properties for researchers and implementers, that are generally unavailable in other standard approaches. So, given that it is argued that potential fields offer essentially a superset of the benefits of other approaches, at first glance it may seem puzzling that they are not more commonly employed. Unfortunately, unlike the other types of approach discussed in Section 2.5, basic approaches to potential fields for navigation simply do not work for most navigation problems. The reasons for this will be discussed later in this chapter.

4.3 Modelling potential fields for navigation

A traditional modelling of potential fields for navigation using the FIRAS[3] function will now be given, based on [127, 123]. This illustrates one of the simplest ways in which a

[3]FIRAS is an abbreviation of the French for 'Force Inducing an Artificial Repulsion from the Surface'.

problem of navigation can be mapped into a potential field problem. At a fundamental level, the heuristic principle behind this model is to say that *moving towards the goal is good* and that *moving towards obstacles is bad*, or alternatively that *positions closer to the goal are better* and *positions closer to obstacles are worse*.

In order to apply the potential fields metaphor to this problem, it is necessary to give a transformation of the problem description to a potential function, which associates each position with a potential value.

4.3.1 Modelling the environment

Consider a 2-D space, with an x and y axis. The potential function must map from each (x, y) coordinate position to a potential value.

The environment contains obstacles. Consider first of all the case of a single obstacle. According to the potential metaphor, the space within the obstacle itself should have a very high potential value (since 'low potential' is being used to represent desirable positions). The space that is nearby an obstacle is less desirable than the space far from an obstacle, so the potential should start high inside the obstacle, then reduce further away from the obstacle. The gradient metaphor suggests that the gradient should direct the agent away from positions near the obstacle. A function must be chosen which encapsulates these properties and which is centered on the obstacle.

Khatib suggests a potential falloff function from the edge of an obstacle [123, 127]:

$$P(x,y) = \begin{cases} \frac{1}{2}\eta(\frac{1}{\rho(x,y)} - \frac{1}{\rho_0})^2 & \text{if } \rho(x,y) \leq \rho_0, \\ 0 & \text{if } \rho(x,y) > \rho_0 \end{cases} \tag{4.1}$$

In Equation 4.1, $\rho(x, y)$ represents the Euclidian distance from the nearest part of the surface of an obstacle, and ρ_0 represents the distance at which the obstacle ceases to have an influence on the agent. η is a scaling constant representing the strength of the obstacle's repulsion of the agent. Inside the obstacle, the potential is defined to be of infinitely high value. An example is shown in Figure 4.1.

To combine the effects of many obstacles, the linear sum of the obstacle functions is taken (Equation 4.2).

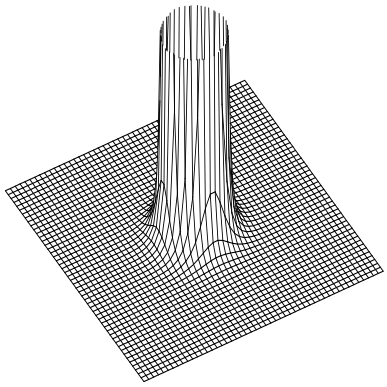


Figure 4.1: A typical potential function representing an obstacle.

$$P_{obstacles}(x, y) = P_{obstacle1}(x, y) + P_{obstacle2}(x, y) + P_{obstacle3}(x, y) + ... \quad (4.2)$$

This sum is not necessarily fixed. If obstacles are discovered to be in the environment by the agent during online navigation, an extra term corresponding to each new obstacle can be added to the obstacles function to include it in the environment's potential function. Navigation can then continue using the revised potential function.

4.3.2 Modelling the navigation problem

A navigation problem consists of start, goal, and environment. The environment function has already been described. How should the goal position and start position be modelled?

According to the potential metaphor, in the absence of any obstacles, the goal should lie at a position of optimal potential. According to the gradient metaphor, in the absence of any obstacles, the gradient at all other positions should direct an agent using gradient descent towards the goal. To achieve this, a function must be chosen which places the goal at the bottom of the basin of some attractor function. A cone or a parabolic well is appropriate for this. The function in Equation 4.3 (based on [127, 138]) is shown in Figure 4.2, but any function that has similar properties is likely to be appropriate. Latombe discusses the merits and demerits of several candidate attractor functions in [162].

$$P_{goal}(x, y) = \rho^2_{goal}(x, y) \quad (4.3)$$

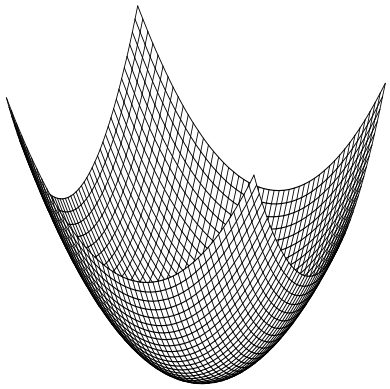


Figure 4.2: A quadratic bowl potential function representing the goal.

The start position requires no further modelling if the goal is represented in this way. If the start position coincides with the goal position, then the agent will begin at the bottom of the goal's attractor basin and will not navigate elsewhere - as is desired in this situation. If the start position does not coincide with the goal position, then the gradient of the quadratic bowl or conic function (in the absence of obstacles) will direct the agent directly towards the global minimum (the goal position) from any possible start position, without the need for a separate potential field component to represent the start position[4].

To combine the goal function with the obstacles function, and thereby create a complete navigation problem function, we take the linear sum of the two functions (Equation 4.4). The desirability of each position in the environment is then based upon its desirability in terms of its proximity to obstacles, added to its desirability in terms of proximity to the goal. In optimising the combined function, the agent will now be trying to simultaneously minimise its distance from the goal, while keeping its distance from obstacles.

$$P_{goalobstacles}(x, y) = P_{obstacles}(x, y) + P_{goal}(x, y) \tag{4.4}$$

4.4 Gradient Descent on a potential field for navigation

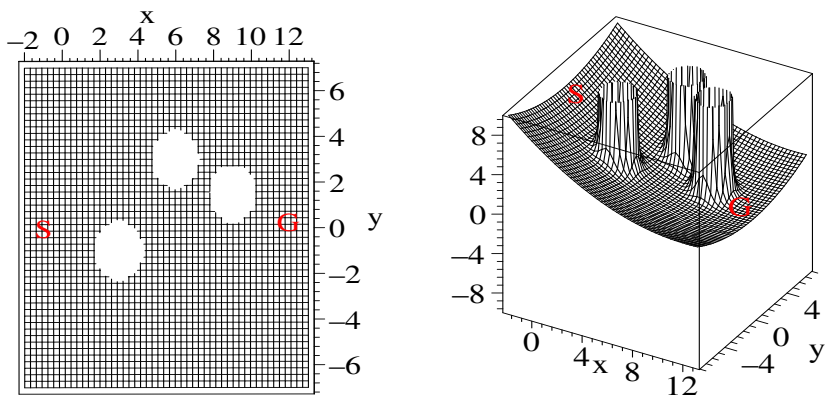
A simple and traditional technique to create a potential field surface using the goal and obstacle configuration of the navigational problem has been described. In order to create a path connecting from the start position to the goal position, it is now necessary to travel over

[4]Some potential field models require a component that partially represents the starting position, such as [130].

the resulting potential field in such a way that the agent descends from the start position towards the global minimum (the goal position).

The standard way of doing this is to use gradient descent (described earlier in Section 3.3.2). Descent is initialised at the start position, and steps are taken iteratively to move the agent according to the negated direction of the gradient. In the event that descent reaches a position suitably close to the goal position, descent is considered to be successful.

4.4.1 Worked example



(a) Overhead view of the environment. The white circles represent obstacles.

(b) Potential field corresponding to the environment in Figure 4.3(a).

Figure 4.3: An example navigation environment. S is the start position, G is the goal.

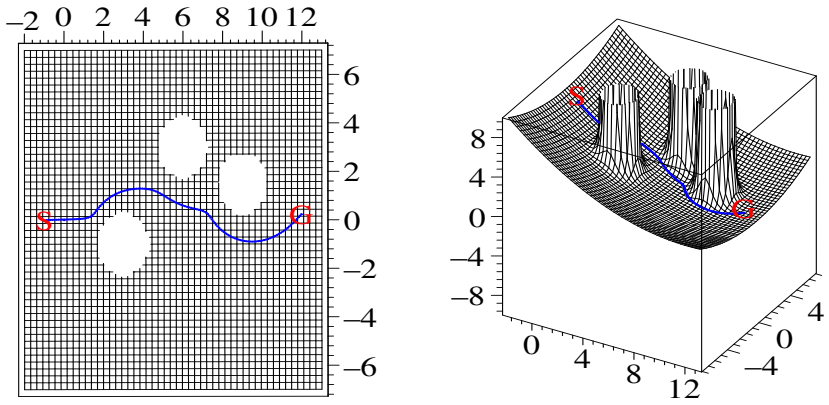
An agent is in the environment shown in Figure 4.3(a). The goal function will be a quadratic bowl attractor centered on the goal position, with a scaling constant[5] of 0.02.

The direction of steepest gradient can be calculated by evaluating and negating the partial derivative of $P_{goalobstacles}$ in x and y, or by calculating and combining the equations given in [127, 138] for the magnitude and direction of the gradient contributions of the goal and obstacle functions. The corresponding potential field is shown in Figure 4.3(b).

Gradient descent is then carried out using these gradient functions. Here, the following parameters are used: a fixed step size of 0.1 units, with no momentum. Pseudo-code for the descent routine is then as follows.

[5]The constant must be chosen so that the agent cannot 'step' into an obstacle because the attraction of a goal located behind the obstacle is too strong. This value could be chosen analytically by finding a value so that the gradient at one step's distance from the obstacle boundary points away from the obstacle, or by making a conservative estimate. The value, once set, need not be changed when the goal or obstacle layout is changed.

```
%%Initialise descent routine
x:= 0; y:= 0; Gx:= 12; Gy:= 0.25;
stepSize:= 0.1;
%%Main loop
while (dist(x,y,Gx,Gy)>stepSize) do
xStep:= -gradx(x,y);
yStep:= -grady(x,y);
magnitude:= sqrt((xStep*xStep)+(yStep*yStep));
x:= x + (xStep*stepSize/magnitude);
y:= y + (yStep*stepSize/magnitude);
print("Next move is to [",x,",",y,"]");
end while;
```



(a) The path taken by the agent in the environment.

(b) The path of gradient descent on the potential field.

Figure 4.4: The path taken in the environment. S is the start position, G is the goal.

Figure 4.4 shows the path taken by the agent in the environment, and the path taken by gradient descent on the potential field surface. This example has illustrated that the basic model of potential field based navigation works in at least simple navigation problems, and is relatively straightforward to implement - the gradient equations and a loop are all that is needed to get direct navigation instructions. The gradient equations can be automatically generated from templates once obstacle positions and the goal position are known. The big-O computational cost of generating the potential field is low - the linear sum of a series of equations based on a single template as given, hence $O(n)$ for n obstacles and a single goal. The big-O computational cost of traversing the field with gradient descent is also low: for n obstacles with obstacle function big-O complexity $f(o)$, and a goal of big-O complexity $f(g)$, the big-O computational cost for gradient descent is $O(min(n.f(o)), f(g))$.

4.5 Problems with potential fields for navigation

The potential fields approach offers compelling incentives to those considering employing it for navigational problems. Unfortunately, the mapping given in Section 4.3 results in a potential field that possesses several extremely problematic 'geographical' surface features on the mathematical landscape. Some of these features have already been discussed in the context of the general application of the potential fields approach. The material that follows is concerned with how these features affect the results of navigation - particularly in terms of navigation success, and the nature of the navigational path formed.

This section will describe how surface features confound traditional potential field approaches to navigation in all but the most trivial of cases, i.e. failure occurs in environments containing anything other than a few, simple convex obstacle shapes.

4.5.1 Local Minima

The local minimum problem for general potential fields was discussed earlier, in Section 3.4.1. Unfortunately, the local minimum problem for navigation occurs very frequently with the mapping given in Section 4.3. Any concave or even flat obstacle, or arrangement of nearby/touching obstacles in such a shape, will result in a significant progress-halting local minimum problem being present on the resulting potential field, for many goal locations.

Further, it has recently been noted [69] that even in the very simplest case of a navigation problem (a single completely convex obstacle, placed directly between the start and the goal), there can exist a form of local minimum problem. Therefore, every obstacle or obstacle configuration, whether convex, flat or concave, has the ability to introduce an associated local minimum problem depending on the position of the goal to be reached. It is well recognised that concave ('C-shape') obstacles lying between the agent and the goal represent the greatest threat in terms of the associated local minimum [109, 163]. Figure 4.5 illustrates the local minimum problem that is present with concave obstacles when combined with a goal.

Since many such flat or concave obstacle configurations will exist in a typical non-trivial environment, the local minimum problem is therefore usually present en masse. This situation worsens as navigation environments increase in terms of obstacle shape complexity, and quantity of obstacles present. If gradient descent is being used as the

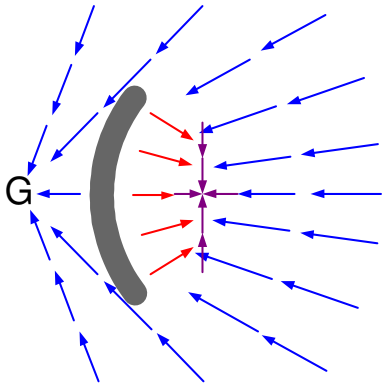


Figure 4.5: A local minimum trap situation caused by a concave obstacle. Red vectors indicate the gradient when the obstacle's contribution to the gradient is dominant; blue when the goal's contribution to the gradient is dominant; and purple around the local minimum.

descent heuristic - and it almost always is - then these local minima act to terminate further progress towards the global minimum on the field, and thus prevent successful navigation from taking place.

> "A key issue for path planning using potential fields is knowing how to deal with local minima that may occur in the field" [108]

> "...Dealing with local minima is *the* major issue that one has to face in designing a planner based on this approach." [161].

It might be suggested that since local minimum problems are present to such an extent, a better choice of descent heuristic would be one based more upon simulated annealing[6] rather than gradient descent. Unfortunately, when applied to the domain of navigation, simulated annealing suffers from three key problems, two of which have already been discussed in the context of the general potential fields approach. First of all, simulated annealing produces random, meandering paths that are far from being direct, and do not exhibit 'natural' or efficient path shapes. This is a serious problem in navigation, where an agent is likely to be expected to travel at least reasonably directly or 'naturally' towards a goal. Secondly, simulated annealing takes theoretically infinite time to reach a global minimum. This is not suitable for either real or virtual world agent navigation. Thirdly, and uniquely problematic in the case of navigation (where path shape is an essential part of

[6] See Section 3.3.3

the problem), simulated annealing allows the agent to take steps 'upward' on the potential field. This results in the agent being able to move into areas of higher potential - which in turn means that there is nothing to prevent the agent from trying to step 'into' obstacles on the potential field, and thereby breaking the rules of the original navigation problem.

The key property offered by gradient descent - that it will generally not step 'upwards' towards areas of higher potential in its direct minimum-seeking behaviour[7] is the same property that ensures the generation of *useful* paths that will never intersect obstacles in a navigation problem. In navigation, local minima exist close to obstacles. Any technique that tries to improve on gradient descent by encouraging travel upwards from a local minimum, is highly likely to simultaneously introduce the unwanted capability for the agent to step into obstacles, thus generating useless paths.

It is also usually not possible to rely on re-initialising gradient descent at a new starting location when using potential fields for online 'single attempt' navigation, as would be the case in potential fields used for heuristic AI problems such as neural networks or evolutionary algorithms. In applications such as robotics, gaming and film animation, an agent cannot simply 'warp' to a new location to attempt navigation again - if it could, it would suggest there was no need for navigation in the first place, as the agent could simply warp directly to the goal immediately.

In summary: Any non-trivial navigation problem will result in a potential field with an abundance of local minima. If gradient descent is used, this will result in total failure to navigate if the minimum basin of any one of these minima is encountered during navigation. If a different descent heuristic is employed that allows occasional ascent to take place in order to escape from minima, it may allow paths to meander or even intersect obstacles, and the resulting navigational path is likely to then be worthless as a solution to navigation. Finally, the nature of navigation problems prevents descent heuristics from being 'restarted' at random positions on the landscape, ruling out the standard re-initialisation technique used to overcome the local minimum problem in other fields of AI.

[7]Note though that it is technically possible for gradient descent to move towards areas of higher potential, if the step size is large and the gradient does not give a good estimate of the variation in local potential, so descent step size is usually set to prevent this from occurring to the extent that the agent steps into an obstacle.

4.5.2 Other landscape features

Local minima are not the only surface feature that is present. Even in those cases where the environment is very limited indeed (short-term, very local navigation with only one or two non-dangerous obstacles), other problems may exist to confound descent.

Ravine Problems

Ravines[8] present a significant problem. Consider the case where an arrangement of obstacles places two obstacles (or two sections of the same obstacle) in parallel[9]. This is shown in Figure 4.6.

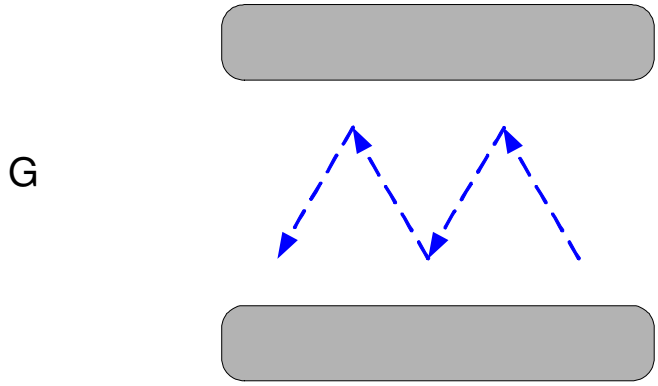


Figure 4.6: An exaggerated example of a ravine problem as found in navigation. Gradient descent oscillates across the middle of the ravine on the potential field, caused by two parallel obstacles and the goal.

The resulting potential field can be seen to contain a ravine feature in a position corresponding to the location of the parallel walls. A ravine may seem unthreatening, as all it introduces is a progress-slowing oscillation depending on the configuration of gradient descent that is being employed. However, this oscillation is a serious problem, because of the domains in which navigation is applied. In robotics, the oscillation is highly impractical, causing a robot to shuffle on the spot, making it more difficult to estimate bearings, and preventing continual progress in a particular direction. Furthermore, non-holonomic constraints on acceleration or turning may make it impossible to realise travel in the directions suggested by the gradient in a ravine.

[8] See Section 3.4.3

[9] This is the case in most human buildings (real world or virtual world) - walls are parallel, and objects within a room tend to be lined up in parallel with a wall.

In virtual world scenarios, the problem is no less serious. A character or agent that begins rapidly oscillating from pointing left to pointing right, and therefore follows a sinusoidal or zigzagged pattern of motion when a viewer would naturally expect it to move forward in a straight line, will soon lose any semblance of realism it might have had, easily destroying a gaming or cinematic experience. Oscillations in either the real world or within a virtual-world are therefore a serious problem.

Saddle points

Saddle points[10] can exist, but are generally not problematic for potential field based navigation, as almost all coordinates around the saddle point position will have gradients that push gradient descent away from the dangerous undefined gradient at the saddle point position itself. A discussion of some dangers that saddle points can pose in potential field navigation can be found in Section 7.5.1, with further detail in [69]. Compared with the more serious local minimum and ravine problems, however, saddle points are little more than a minor nuisance.

4.5.3 Summary of surface features

In summary, it has been observed that the *local minimum problem* occurs in most non-trivial environments, is associated particularly with *concave* and *flat* obstacle configurations, and is the major problem for potential field based navigation in that it terminates progress in most cases when it is encountered. The *ravine problem* causes an agent using gradient descent based techniques to exhibit highly inefficient and quite unrealistic behaviour when the ravine is encountered. Finally, the *saddle point problem* causes aberrant behaviour in a small number of navigation scenarios.

4.5.4 Research into surface features

Coverage of research efforts to address the local minimum problem for potential field based navigation can be found in [108, 206, 243], as well as in Chapter 5. However, there is only one well-known potential field navigation paper that actually attempts to identify general

[10] See Section 3.4.5.

problems for potential field based navigation besides the local minimum problem. Koren and Borenstein published a paper in 1991 [134] that claimed something quite remarkable: that they had found not one, but four serious problems for potential fields, and that these problems would be present in all potential field methods for navigation. This is quite a strong claim to make.

> "We identified the following 4 significant problems that are inherent to PFMS [Potential Field Methods] and independent of the particular implementation." [134]

1. Trap situations due to local minima (cyclic behaviour).
2. No passage between closely spaced obstacles.
3. Oscillations in the presence of obstacles.
4. Oscillations in narrow passages.

While Koren and Borenstein were correct to bring to light these important non-local-minimum problems, they may have overclaimed in saying that these problems will affect all potential field based techniques. Their claims will be examined in the experimental results section of this thesis (Chapter 7) as one of the measures of the success of the novel technique presented in this thesis.

4.6 Chapter Summary

This chapter has provided:

- Motives for the use of potential fields for navigation.
- Mechanisms for modelling navigation problems.
- A worked example of potential field based navigation.
- A discussion of the problematic surface features associated with potential fields for navigation and their effects on navigation.

Chapter 5

Survey of Potential Field Based Navigation Techniques

Guide to Chapter 5

Having built up a basic awareness of the issues involved in using potential fields for navigation, this survey chapter looks at the more advanced approaches taken to try and combine *Navigation* with *Potential Fields*, and describes the historical context of this field of research.

5.1 Chapter Overview

Research in potential field based navigation has been tightly focused on overcoming the local minimum problem, rather than the other problematic surface features that appear in potential field based navigation, since local minima represent the most serious threat preventing potential fields from being successful. This rest of this chapter is comprised of an overview of research in potential field based navigation, as follows:

- A discussion of the techniques invented to try and overcome the local minimum problem on continuous potential fields.
- An analysis of one of the most successful of these techniques - *navigational potential fields*, in which the local minimum problem is overcome by defining a special type of mathematical mapping between navigation problem and potential field that does not allow local minima to appear in the first place.
- A discussion of approaches to potential fields based on discretisation of the field.
- An overview of the historical context of the research field.
- An evaluation of the first part of the hypothesis.
- Conclusions and characterisation of an ideal potential fields approach to navigation.
- A table summarising the properties of the main techniques.

5.2 Attempts to overcome the LMP for navigation

5.2.1 Techniques from other fields

Potential fields are in use in other fields (such as neural networks) and these fields have developed techniques for addressing the local minimum problem to some extent. Unfortunately, the nature of the problem of navigation - i.e. that there are unrealisable parts of the field that the path cannot intersect - means that the main techniques used elsewhere in potential fields research cannot be re-applied. For example:

- Momentum: If the agent has a momentum term added to the negated gradient term, in order to try and keep it travelling 'uphill' on the potential field long enough to escape a minimum, it will simply crash into the obstacle configuration that is causing the minimum!
- Restarting descent from a new initial location: This is inappropriate in the case of navigation - if the agent could immediately re-start travel at a new location, it might as well jump directly to the goal! Furthermore, a key aspect of navigation is that the agent must solve the problem from a particular defined starting position on the field, not from a randomly initialised position.

This section will now discuss the myriad of alternative techniques that have been developed and deployed within navigation to address the local minimum problem.

5.2.2 Minimum avoidance

Elliptical Potentials

The idea here is that some problematic obstacles that generate local minima can be made less dangerous, by selecting a specially shaped elliptical obstacle potential function that makes them appear to be a different kind of obstacle from a distance. This new obstacle function, when combined with a symmetric goal function (such as a quadratic bowl or a cone), results in equipotentials that converge towards the ends of the obstacle as the distance to the obstacle goes towards zero. In other words, instead of travelling directly towards the obstacle, if the agent is slightly off-center, it is increasingly pushed towards the edges, so that it grazes past the obstacle [164, 234]. This is shown in Figure 5.1.

In effect, the resulting goal-obstacle potential for a flat obstacle is shaped as though there was a traditional convex obstacle there in its place. This has the effect of reducing the size of the basin of the local minimum, and thus reduces the chance that the agent's navigation will be halted by it.

The drawbacks to this technique are that:

1. It does not remove the problem of local minima - it merely reduces the chances that the agent will step into an area where it can be trapped at a minimum.

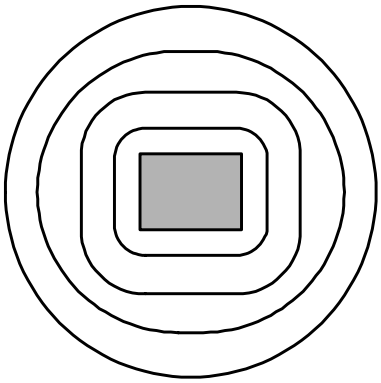


Figure 5.1: Equipotential contours of an elliptical potential function.

2. Since the obstacle now has the influence of a larger convex obstacle, a large area of free space may be made inaccessible by this approach as the agent is pushed away from the 'virtual' convex obstacle - this may be problematic or may lead to unusual paths being formed.

3. The technique can only be effectively applied to simple flat shapes such as rectangles (generally, trapezoids) in two or three dimensional spaces [129, 164].

4. The technique requires that obstacles are sufficiently separated that they cannot influence each others potential.

5. In Khosla/Volpe's technique [234], no direct mechanism is given by which potential can be directly calculated - instead, interpolation is carried out between positions with well defined potential values. This represents an added computational and implementational burden.

5.2.3 Minimum detection and escape

Given that dangerous local minima exist on a potential field - is it possible to detect when one has been encountered, and then escape from it?

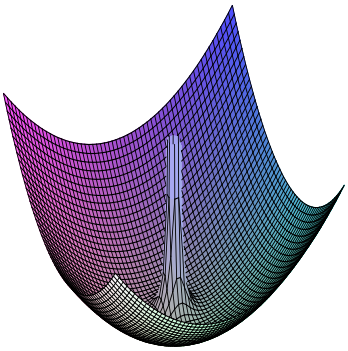
Minimum detection

> "In general... simply observing the minimum can be problematic." [94]

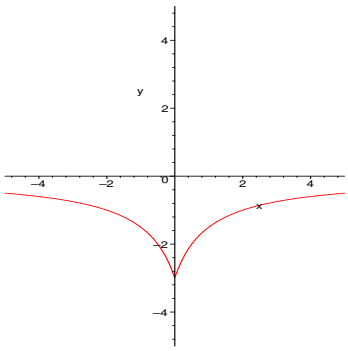
Broadly, there have been two forms of minimum detection employed on continuous surfaces - mathematical and heuristic.

1. Mathematical: Here, the agent attempts to detect when it has travelled into a local minimum basin by detecting if it is near a local minimum. This is done through mathematical analysis of the surface. For example, the Hessian can be evaluated at the current position and examined to see if the mathematical shape implies a local minimum[1] [172].

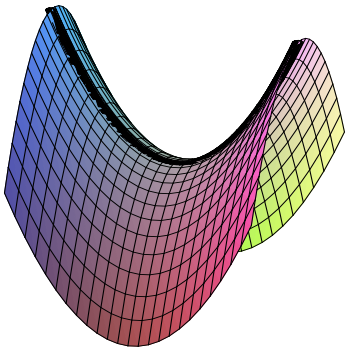
   The problem faced by this approach is that the agent may be in a local minimum basin, without being near the minimum. Even when the agent is near the minimum, it will not land exactly on the minimum position itself, but rather it will oscillate around the position. This makes mathematical observation of a local minimum difficult [110]. Finally, certain types of minimum may not appear as the classical mathematical example of a minimum. Figure 5.2 below illustrates some examples of minima that would not be recognisable via the Hessian. In particular, mathematically unrecognisable[2] minima can occur when the first order derivative of the field is discontinuous.



(a) A 'moat minimum' has the form of a circle of points of equal lowest potential.



(b) A 'seam' minimum' can be formed by a point of discontinuity in the gradient.



(c) A 'saddle minimum' is a local minimum problem in the highlighted axis of a saddle point.

Figure 5.2: Examples of minimum situations that do not correspond to the classical mathematical definition of a local minimum.

2. Heuristic: Here, the agent attempts to determine (through its own lack of progress in terms of displacement from previous positions) that it has reached a local minimum position. In potential field based navigation, this can be done in a straightforward

[1] Lewis suggests in [172] that [194] is an excellent reference on this matter.
[2] Via the Hessian, at least.

manner by checking that the agent has not successfully reached the goal (by checking the current location against the goal location) and by totalling the distance travelled by the agent over some number of iterations of gradient descent.

Alternatively, another approach taken from general potential fields is to allow a preset limit of some reasonable number of steps of gradient descent to be taken in order to reach the global minimum. If the agent has not successfully descended to the global minimum by the time the limit is reached, then it is assumed that it is trapped in a local minimum and that navigation has failed.

Problems can occur when navigation is merely difficult or long-winded in nature, and the agent is incorrectly identified as having encountered a local minimum when in fact no local minimum is present.

Minimum escape

Given that it has been identified that the agent is in a local minimum, and given that the agent is indeed in a local minimum, how may the agent then escape? [110], based on [91], suggests a number of strategies that may be employed - and it may be noted that many of them resemble strategies a natural agent might employ when trapped.

Random movement

Upon discovering that it is in a local minimum, the agent takes some pattern of random steps to try to escape the minimum. Some implementations of this technique [117] use a form of simulated annealing as the escape technique; others simply use randomly chosen paths [66]. This category of navigation techniques is distinct though from simulated annealing. In simulated annealing, no attempt is made to recognise minima or act in any different way when they are encountered. In this regard, the random movement employed here is better - it is only applied as an escape strategy when direct steepest gradient has lead to disaster, and the rest of the time the agent's behaviour is directed towards movement towards the goal and away from obstacles.

As is the case with simulated annealing though, there is a risk of stepping uphill, into an obstacle, since the repelling gradient of the obstacle is ignored by random movement. This problem can be overcome by checking each point to ensure that it lies in free space and not

inside an obstacle, before the agent moves.

The paths generated by random movement are extremely unsmooth, which causes problems for real world agents trying to follow them, and prevents virtual world agents from looking realistic in their behaviour. This can be dealt with to some extent by smoothing the path before allowing the agent to follow it.

Finally, random movements may cause the agent to re-enter the basin of a minimum that has already been escaped from. In the case of a series of nested minima, this can be a particularly serious and progress-terminating problem, as the agent works it way up and down the series of minima with random paths but fails to escape the configuration as a whole.

In short, although random movement can allow escape from a minimum, it makes movement difficult for real-world agents and unconvincing for virtual world agents. It risks failing to solve the navigation problem by undoing its own useful progress, and it poses new problems in terms of how to set random path lengths correctly.

Hybridisation

The idea here is to employ a different type of navigation or movement technique to escape minima [111]. On encountering a local minimum, gradient descent is abandoned, and some other non-potential fields technique is employed instead. Typically, since local minima are most commonly associated with obstacles, a low-level wall-following algorithm using raw sensor data or a touch sensor may be sufficient to move away from the configuration causing the minimum basin, and thus the minimum basin itself. However, since the minimum itself may occur at some distance from the obstacle, the use of such a technique is not guaranteed to succeed.

Adopting a hybrid approach, it is argued, is an inappropriate answer to the question of how potential fields can overcome the local minimum problem and provide general navigation, as hybrid approaches essentially concede defeat and do not use potential fields to overcome the local minimum problem. A hybrid approach is likely to be closely tied to a particular implementation (for example relying on the particular map representation used, or sensor data gathered) and is unlikely to be suitable for generalisation in terms of application domain, dimensionality of problem solved, or allowing extensions of the navigation problem.

Backup / Back-tracking

The agent may move back along the path that lead it into the local minimum problem, and then attempt to navigate in another direction, employing some kind of avoidance procedure to prevent the minimum from being immediately re-encountered [111]. However, there is no guarantee that the minimum will be avoided after backup, and it is even possible that the agent might return to an earlier minimum in backing up. The resulting navigation path may look a little unnatural (if being carried out by a virtual world agent) and may be inefficient. This approach cannot be adapted easily to suit non-holonomic constraints - since an agent may not actually be able to re-traverse its path backwards. There is a conflict between backing up far enough that the agent does not immediately fall back into the minimum, and not backing up so far that it retraces its steps back to an earlier problem. Since this will vary from minimum to minimum, backtracking may be very difficult.

Another important question is how an avoidance procedure may be implemented to prevent re-entering the escaped local minimum problem. It is not obvious that the gradient can be modified to prevent re-entering the minimum again, so it may be that another form of navigation is employed to direct the agent after it has backed up (see 'Hybrid techniques' above). If this is the case, the backup technique also inherits the weaknesses of the approach it is hybridised with, such as losing generality through being tied to a single physical robotic implementation for example.

Minimum filling

Here, the idea is to increase the potential value of the local minimum, so that it is no longer a minimum, and so that the area now instead repels the agent much like an obstacle. The problem here is how to raise the potential in such a way that no new local minima are formed, and so that the agent does not cut itself off from being able to reach the goal, and also so that the gradient as well as the potential is raised appropriately in filling the minimum. No general solution to this problem has been found for continuous models of potential. A model involving a discretisation of the field that works quite well will be discussed later in this chapter.

5.2.4 High-level hybrid approaches

The high-level hybrid approach to overcoming local minima is to recognise that attempts to achieve global navigation with potential fields are highly likely to encounter local minima, but that local navigation is much less likely to encounter them, and to therefore use some other technique for global navigation rather than potential fields.

In this situation, potential fields are now relegated to 'joining the dots' generated by the higher level approach. This does nothing to achieve the result of allowing potential fields to achieve global navigation. None of the benefits of potential fields navigation are conferred when another technique is employed at the top level of navigation, so the hybrid technique may lack generality in terms of usefulness in different dimensionalities, application domains and so on.

The distinction between this and the hybrid approach listed earlier, is that before, another type of technique was used to provide heuristic escape from minima situations, in order to correct the weakness of gradient descent and allow it to escape local minima if detected. Conversely, in this approach, one recognises that gradient descent will fail at navigation, and so it is gradient descent that is provided for some higher level technique to correct the weaknesses of that technique (such as inability to produce a continuous smooth path). The local minimum problem (and escaping it) is no longer an issue, since gradient descent is not 'in charge' of navigation - the fact that gradient descent is unreliable for anything other than very local movement is something for the high level navigation technique to cope with.

The earliest example of this technique can be found in [135]. Other examples of the hybrid approach in robotics can be found in [116, 169, 243] with connectivity graph based planning, geometry based planning and A* search respectively, and a publicised example can be found in computer games in [44] (see Appendix B for more information).

5.3 Navigational Potential Fields

5.3.1 Overview

This chapter has considered techniques that attempt to circumvent the problem of local minima by trying to detect them, then avoid them or escape from them. Other techniques

attempt to reduce the scale of minima in order to reduce the threat they pose to navigation.

A major branch of techniques for addressing the local minimum problem within robotic navigation takes a different approach, attempting to completely prevent local minima from ever appearing in the potential field to begin with.

Koditschek was the first to suggest such an approach, in 1987. His paper [132], written only a year after Khatib's most well-recognised paper on potential field based navigation, suggested that it might be possible to restrict the choice of mathematical functions used to model a potential field, to those functions that would allow only non-dangerous points of equilibrium such as saddle points. Since the resulting field would lack local minima, it appeared ideal for carrying out gradient descent upon, as it would not suffer from the navigation failure caused by local minima.

Koditschek's work was welcomed, and several groups of researchers published papers in this genre of potential field research. The main families of techniques based on this idea came from Connolly [84, 85], Kim and Khosla [130] and Rimon and Koditschek themselves [132, 133, 206]. They examined the properties of such a navigational field, and attempted to think of types of navigation environments best suited for its implementation, rather than attempting to find a general technique to generate navigational fields for arbitrary environments of any kind. In 1992 particularly, a number of papers were published relating to navigational potential fields [85, 130, 206], but this represented a high tide mark for this field of research, and relatively few papers have been published on the matter since. Latombe and Dudek & Jenkin provide coverage of this technique in [94, 158].

Definition A *global navigation function* is a potential function over the free space in the configuration space, with a minimum at the goal position, whose domain of attraction includes the entire subset of positions in free space that can be connected to the goal.

If such a field were constructed, then gradient descent would be guaranteed either success - the minimum reached from any position connected to the goal would be guaranteed to be the goal - or at least recognisable failure if navigation starts from a position that cannot be connected to the goal. In other words, a complete global path-planner, based on potential fields.

One of Koditschek's primary results was to show that such a global path-planning potential field cannot in fact exist [132]. However, useful approximations of such a field may exist,

'almost-global' fields, and they will contain a number of saddle points where success cannot be achieved. A global navigation function could therefore exist that is functional for almost all of the possible starting positions in the field, except for a few positions[3].

5.3.2 Generation of Navigational Potential Field surfaces

Koditschek and Rimon's 'Disc world', 'Star world'

Given that an 'almost-global' navigational field could exist in theory, in practice how might it be generated? In addition to presenting the idea of a navigational potential field, Koditschek suggested models [132] whereby a disc (representing the world) could have circular obstacles cut out from it. A corresponding navigational function could be generated directly from this that allows descent to the goal. Rimon and Koditschek suggest the use of transformations upon this 'disc world' to allow other types of navigation environment to exist, culminating in the suggestion of a 'star world' containing some symmetric, non-convex obstacles in their 1992 paper [206]. The mathematical basis for the existence of this representation, and the constructive technique they outline to build the functions corresponding to such worlds involves *decidedly* non-trivial mathematics.

Connolly's Harmonic Functions

Connolly suggested the use of *harmonic potential functions* [84, 85]. These are functions which lack local minima or maxima. Furthermore, when linearly summed (superimposed), no local minimum is introduced into the resulting field. The mathematical model originates in fluid dynamics. Consider the streamlines of an incompressible fluid, with sources (obstacle boundaries) and sinks (the goal) in the environment. The 'fluid' possesses streamlines that lead from the boundaries, to the goal, without local minima - since this would represent either an unallowed additional sink, or a case where the fluid is becoming compressed. An alternative physical metaphor is to consider the case of non-conducting solids in a conducting medium - here the starting point is a source of current and the goal is an equal and opposite sink for current.

[3]Latombe writes [158] that a function could exist for all points "except a set of points of measure zero which are saddles of the potential function". Bell and Livesey suggest in [69] that this may be an underestimation of the problem generated by saddle points for navigation.

Definition A harmonic function on a domain $\Omega \subset R^n$ is a function which satisfies Laplace's equation (Equation 5.1):

$$\nabla^2\phi = \sum_{i=1}^{n} \frac{\partial^2\phi}{\partial x_i^2} = 0 \tag{5.1}$$

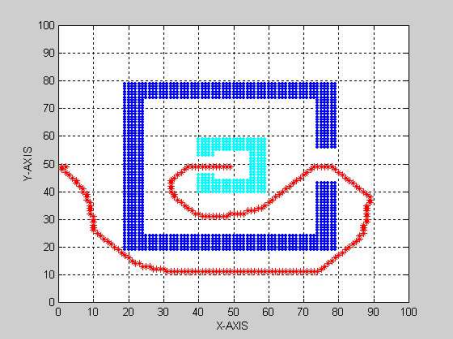
The value of ϕ is given on the closed domain Ω. Here, the closed domain Ω is the potential function space. The boundaries of Ω consist of the goal and all obstacle boundaries in the original problem.

Implementation

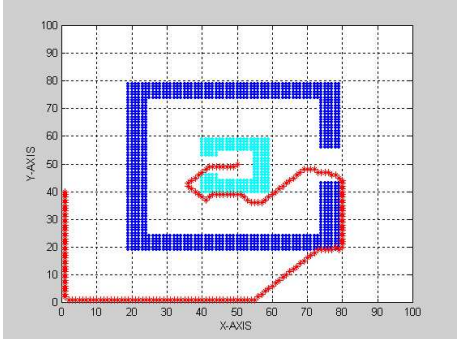
Connolly suggests deriving the harmonic potential field by using iterated Finite Difference methods (a numerical analysis technique) on a grid representation of the space. Boundary conditions are imposed on the grid according to the position of the obstacles and the goal. The idea is to try and converge on a numerical approximation that satisfies the Laplace equation under the conditions imposed by the obstacles and the start/goal position. The resulting grid of values is then used for navigation when it has been iterated upon sufficiently to guarantee that there are no local minima. The resulting navigation function is resolution complete. Connolly suggests that it is reasonable to then interpolate between positions in the grid using multi-linear functions to 'average them out' into a smooth function, as the underlying harmonic function approximated by the numerical technique is a smooth function (because it is harmonic). Some examples of navigation based on such an implementation are shown in Figure 5.3.

Kim and Khosla's panel-based approach

In 1992, Kim and Khosla [130] proposed an alternative technique. Instead of using a numerical approximation to a harmonic function, they suggested building a harmonic function using closed form equations representing different components of the problem (in a manner analogous to Khatib's approach, see Section 4.3). Their technique requires that all obstacles are represented as panels, and suffers from various problems such as the risk of discontinuity in the surface's gradient [2].



(a) Harmonic function with Direchlet boundary condition. The function is prone to shallow gradients due to plateaux.



(b) Harmonic function with Neumann boundary condition. The path is less stable and the agent risks grazing collisions.

Figure 5.3: Examples of navigation using an unsmoothed harmonic potential field. These pictures were supplied by Mitul Saha of Stanford University.

5.3.3 What are the benefits?

Complete global planning

The primary benefit offered by the navigational potential field approach is that in theory it can be used to almost guarantee convergence to the goal due to the absence of local minima. It is possible to retain the simple steepest gradient descent technique as the descent heuristic for the field. Conquering the primary problem for potential field based navigation of local minima is a significant development and should not be undervalued.

Effector control

The technique can be altered to produce torque calculations to control a robot's motors directly - in other words simultaneously addressing all of the issues of path planning, trajectory generation, trajectory tracking and effector control using the same process. Initially, techniques providing this feature had problems in that they were unable to bound the level of torque produced; and thus approaching an obstacle lead to control signals that could not possibly be realised. Rimon and Koditschek consider this problem in their 1992 paper however [206], and provide a bounded control signal within the environments their technique is able to model.

Indicates when a path does not exist

The technique is also complete, and not only guarantees that a solution will be found if one exists, but also can give an indication that a solution does not exist if this is the case.

Robustness

If the agent slips from its trajectory as a result of an external force or as a consequence of a large gradient descent step size, the nearby streamline trajectories will eventually steer the agent back towards the goal.

> "Robust control in the presence of environmental model error. Unexpected forces result in the 'sidetracking' of the effector to alternative streamlines." [86]

5.3.4 What are the weaknesses?

The technique can be very effective for a single agent in non-complex, static environments, if this is all that is required by the problem. Unfortunately, this family of techniques has a number of weaknesses that make them either impractical, or at the very least much less useful, in a number of situations.

Lack of generality

Although in theory there is no reason why a global navigational field cannot be created for an arbitrary environment, in practice, only a few forms of environment (star, panel and disc worlds) have had transformations laid out to produce corresponding fields, and these transformations are complex. The technique is therefore not as general or straightforward in practice as the heuristic transformation given in Section 4.3.

Conceptual and implementational difficulty

Some of the transformations that are given for navigational potential fields are *extremely* complex conceptually and mathematically, particularly compared with the traditional heuristic mapping. An understanding of basic mathematics will see the researcher or implementor quickly through the use of the traditional FIRAS mapping of environment to potential field, but to use the mappings of [85] requires at least an understanding of fluid dynamics and other advanced mathematical ideas. Documentation of these techniques is frequently opaque to the non-mathematician and simplified descriptions (such as in this chapter) are absent. It appears that permitting complex obstacle shapes can introduce difficult mathematical problems (such as integration along curves) that are beyond the capabilities of modern computer based mathematical tools [2].

Problems with path shape

The technique is not in any way aimed at producing natural path shapes. The paths generated by this family of techniques are associated with the behaviour of streamlines of an incompressible fluid, and not with the behaviour of a natural agent. One consequence of this is that the resulting paths lack 'directness' and can sometimes be seen to meander somewhat in reaching the goal. This can lead to unnatural path shapes that may be unsuitable for some application domains such as gaming or cinema. It may also be inefficient. Behaviour nearby obstacle boundaries may appear particularly unusual depending on the choice of boundary condition, and paths can be prone towards arcing towards a goal rather than travelling directly there. A secondary consequence is that if the field is recalculated to include new information (i.e. in the case that a new obstacle is added or moved in the middle of navigation), the path may not flexibly accommodate the new obstacle. The global navigational field might even suggest travelling back the way the agent has just come, even if the new obstacle does not lie in the original path of the agent. Finally, the choice of either of two nearby starting points (A, B) may cause completely different behaviour to be exhibited during navigation. In applications where an agent much convey a semblance of 'natural path' generation, this may be inappropriate.

Suitability for application domains

While the technique is well suited to a single agent in a simple and static environment, this does not correlate well with the problem normally faced by a robot navigating in a real world environment, by an agent in a video game, or by groups of agents in cinematic media. The navigational potential field techniques available are impractical for multiple moving agents, and this makes the technique less useful for certain application domains. Similarly, the technique is inappropriate for online navigation, where obstacles are being discovered as the agent moves around. This can rule out gaming or robotic applications, where the structure of the environment itself may have been changed by other agents, and no static global map is present.

Suitability for non-holonomic constraints

The issue of representation of non-holonomic constraints (navigation in situations where only limited control is present due to the momentum, orientation, construction or environment of the agent) presents another problem. It is possible to include this in the traditional potential fields model, for example, one might try to introduce fake 'obstacles' to bias the gradient away from locations that the agent is not actually capable of navigating towards due to non-holonomic constraints. It is not obvious how one might do this within the global navigational fields approach; one cannot alter the field easily, and in the event that a new field is generated somehow, there is no guarantee that the streamlines will direct the agent in a manner that satisfies the non-holonomic constraints. There is at least one research group working in this area, however [235].

Robustness

As well as being unsuited for representation of non-holonomic constraints, navigational potential fields can present problems in that they are not so robust as traditional potential fields in the event that the agent strays from the 'path' designated by the flow of the field due to non-holonomic constraints that are not being represented in the model, or due to the size of steps being taken by gradient descent. Consider a stick being carried along by a current in a river. If the water changes direction quickly, the stick's momentum may carry it into another current which takes it back in the direction it just came from! A stick can

be trapped in a series of currents in such a manner, and similarly can an agent with non-holonomic constraints that cannot precisely obey the dynamical instructions (in terms of velocity and acceleration) of the navigational potential field, or an agent with a gradient descent step size that carries it into another streamline of potential. In some ways this represents a limitation on robustness. Consider the quote from Connolly's paper:

> "Unexpected forces result in the 'sidetracking' of the effector to alternative streamlines." [86]

Generally, Connolly is right in that nearby streamlines are likely to lead in much the same direction as the previous streamline did, steering the agent on much the same trajectory to its goal, and hence providing robustness as a FIRAS-based field does. In some cases though (perhaps near a complex group of obstacles) streamlines may behave unusually, and when combined with non-holonomy or large gradient descent step sizes, it is possible that the agent may end up travelling on a looping or meandering course.

Dimensionality

The potential fields approach is traditionally well suited to highly dimensional problems. However, techniques such as [130] provide solutions only in 2-D. A family of functions which is harmonic in 2-D, may not necessarily be harmonic in 3-D or above. Therefore the technique may need to be redesigned with different equations to suit re-application in problems of higher dimensionality, such as 3-D navigation. The primary impact of dimensionality however comes from its effect upon the computational expense of the technique, which is significant (see 'Computational expense', Section 5.3.4).

Lack of 'drop-in' implementation

There is no simple 'drop-in' algorithm or practical guide to implementation available for these techniques. This is a significant barrier to implementation and further research, as one must try to decipher how the theory presented in e.g. [206] should be transformed into pseudo-code and hence an implementation. This is less of an issue for researchers who are likely to be prepared to take a few days, weeks or months to fully understand the complexities involved, but for a video game or cinematic animation developer, the

time involved may be an unaffordable expense. This is not a problem with the global navigational field technique per se; merely with its presentation to date.

Making the environment suit the technique

There is no standard way to perform decompositions of the environment into some of the world-types required by e.g. [130]. For example, [130] suggests using 'panels' to build a world - but in a non-trivial environment, it is likely that there are many ways in which the environment can be decomposed into sets of straight lines. Implementing a technique to carry out this additional step represents an added burden on the implementor or researcher, and makes the technique even less suitable for 'drop-in' application. The nature of such deconstructions may even introduce new problems - work carried out by Livesey [2] suggested that equivalent panel representations of an environment can create extremely different streamlines on the potential field that is generated. The resulting unpredictability of navigational behaviour is undesirable.

Computational expense

The FIRAS model of potential fields has the advantage of being extremely cheap to compute, as its cost is linear in terms of the number of obstacles. The computational expense of the harmonic potential field model depends on the complexity of the environment, and the dimensionality of the world. Practically, it is certainly too expensive to consider computing in real-time except for trivial environments in 2-D.

> "The construction of a navigation function, i.e. a potential field with no local minimum other than the goal configuration, is a difficult problem that has a known solution only when the obstacles have simple shapes and/or when the dimension of the configuration space is small (m = 2 or 3). " [165]

Even attempts to make the approach feasible in low-dimension, discretised environments meet with failure as soon as the resolution of the discretisation is increased [94]. Generally, as the cost of generating the navigational field is high, it renders the technique unusable for much beyond a 2-D, simple, static environment containing simple shapes, with a single agent.

Possibility of non-standard local minima

Some of the techniques can be vulnerable to non-standard 'local minima'. The minimum of a function can be defined as a point at which all the second derivatives are non-zero and positive, and the first derivatives are zero. This type of minimum is prohibited by the Laplace equation, and the potential fields of [85, 130, 206] lack these minima. However, it is possible for other types of 'minima' to exist; 'saddle minimum' problems which contain a form of local minima that can satisfy the Laplace equation as laid out in [69]. Similarly, 'discontinuous' minima [2] can exist if there are discontinuities in the navigational potential field. As an example, consider the seam of a pocket of a pair of trousers- clearly, it acts a local minimum for any change you put in your pocket, with coins being trapped at the bottom and unable to escape - yet it has not been recognised as a local minimum by the designers of some of the navigational potential field approaches. If any discontinuities in the first-order derivative are introduced in overcoming the traditional local minimum, there is a risk of e.g. the 'seam minimum' problem. It appears that this problem may exist in fields based upon the description of the navigational potential field method in Kim and Khosla's work [2, 130].

5.3.5 Other surface problems

Although this approach removes local minima if correctly implemented, it does not address the other types of problematic surface feature that can cause problems for navigation. In particular, the following features may still exist and prevent successful navigation:

- Plateaux are frequently seen to occur, often when large numbers of saddle points are present. These plateau represent the 'flow' of the liquid slowing down to crawl, and may cause navigation to stall indefinitely depending on the implementation of gradient descent used. Quantisation errors in the approximation to the harmonic potential field can also lead to the gradient becoming arbitrarily small at certain parts of the environment and thus somewhat impractical for use [85, 94].
- Saddle points exist, sometimes en masse - these are discussed earlier in this section.
- Ravines exist. Koditschek notices in [206] that ravines are frequently found to form along the boundaries of obstacles. In other words, an agent travelling alongside an obstacle may begin to start twisting back and forth (as per the Ravine problem). This

is not the kind of behaviour one wishes an agent to exhibit in any application domain, much less when next to an obstacle and risking collision.

5.3.6 Conclusions

In overcoming the single (albeit highly serious) problem of local minima, many new problems are introduced by this approach. An unfortunate consequence is that to a large extent, potential fields have been considered 'solved but impractical' from 1992 onwards. It is also possible that the complexity of the proofs and inaccessibility of the mathematics involved in the above research papers contributed towards the decline in popularity of the potential fields approach to navigation in the early 1990s, and to the fact that it has never been employed by modern application domains seeking virtual agent navigation such as films and computer games.

Furthermore, though potential fields generally offer a number of benefits for navigation such as computational affordability, ease of comprehension, suitability for high dimensional problems and so on - in overcoming the primary problem for potential fields (local minima), the 'global navigational field' approach removes most of the original advantages of the potential fields approach.

To some extent, it is understandable that the disadvantages introduced by the technique were never really addressed at the time - it was meant to cope with simple, static environments such as those a robot might encounter in a research environment. Even by 1992, when further research was (largely) abandoned, only one or two games had ever been written where multiple agents had to navigate in a dynamic world, and any computer animation present in films was conducted by direct hand-programmed animation rather than computer control of agents. The application of robotic navigation techniques to simulated environments that are not simple or static, and where ease of implementation and low computational cost are paramount, was far outside the scope of the original papers. Researchers possibly did not foresee the real demand for efficient online navigation, particularly for 2-D and 3-D agent navigation in large, complex and arbitrary environments, possibly alongside many other agents.

5.4 Discrete approaches

Having seen the difficulties inherent in making continuous potential field techniques overcome the local minimum problem, it is worth considering a final variant family: discretisation/grid approaches. There are discrete potential field techniques corresponding to several of the continuous heuristics that have been discussed so far.

Instead of using a continuous space, with a real number coordinate system, a grid is laid over the potential field, and potential is only used at integer grid coordinate positions. If the environment is 3-D or n-D, then a 3-D grid or n-D grid is used. A potential value is then associated with each grid cell - either precomputed using a discrete potential field generation technique, or sampled from a continuous potential field generated from the traditional heuristic mapping. The techniques that can be derived from this approach are much more suitable for implementation and practical use - generally, the ravine, plateau, and saddle point problems disappear, and later in this section it will be seen that there are several quite usable and computationally affordable techniques that can overcome the local minimum problem (at least in 2-D).

5.4.1 Problems with discrete approaches

Discrete techniques sacrifice several of the benefits of continuous potential fields - they do not directly generate smooth trajectories, and they lose some semblance of natural behaviour. In traversing the field, computational cost quickly rises above that of a simple continuous technique due to the increased number of calculations needed to look at the potential value of many points on the grid, rather than just a single gradient. Often, the grid must be stored in memory in its entirety, and this makes demands upon storage that invariably become impractical at some level of grid resolution.

The real problem however is that many discrete techniques become intractable for large environments, and all of the techniques become effectively intractable for high dimensional potential fields. In the worst case, the computational cost can rise as high as $O(X^n)$ with a memory requirement of X^n, where X is the highest resolution of any dimension (i.e. the width or height or depth of a grid), and n is the number of dimensions. Even in the best case (for a heuristic that does not precompute the entire grid), the cost of navigation is $O(2nYf(p))$ where Y is the number of steps required, and n is the dimensionality of the

field, and $f(p)$ a function calculating the potential at a position. The computational cost of movement or of precalculating a field does not substantially increase with the complexity of the environment in terms of obstacles (unlike continuous navigational potential fields) - even in the case where a discrete variant of the navigational potential field is generated.

Further problems are introduced by discretisation. By measuring potential at a distance, it is also possible that there is some intervening obstacle that the agent will collide with when it attempts to move. It is possible that the discretisation fails to capture all of the shape of an obstacle, allowing collisions with the unrepresented part. Alternatively, an overcautious discrete representation of an obstacle may cause more of the environment to be marked impassible than is actually the case, which may make navigation more difficult.

Finally, there is the question of where potential should be sampled from, within a grid cell. No matter where the potential is sampled, it is possible that the sampling point is unrepresentative of the rest of the cell's volume, and that the agent may collide with an obstacle as a result, swerve unnecessarily to avoid an obstacle, or generally take a poorer choice of path.

5.4.2 Overview of discrete approaches

Discrete Gradient Descent

One of the main characteristics of continuous gradient descent is that a vector value is produced, as the result of a single computation based on the current position on the potential field. Since the gradient vector could point anywhere, it is therefore possible for the agent to step in any direction on the field, and reasonably smooth paths may be formed as a result.

In the case of a grid approach, this vector quantity is abandoned. Instead, descent is carried out by comparing the potential values of the immediate neighbouring grid cells to find the neighbouring cell of lowest potential. Steepest descent is then carried out by moving to the neighbouring grid cell of lowest value, providing it is of lower potential value than the current position. Immediately, it may be realised that the higher the dimensionality of the grid, the more cells will neighbour the current cell. To check all neighbouring cells takes $O(2n)$ computations of the potential value of a single cell. If the number of obstacles remains constant, then the cost of carrying out discrete gradient descent on the field rises with the dimensionality of the problem.

Discrete minimum detection

Minimum detection becomes (relatively) trivial and much more accurate. To detect if the agent is at a minimum for Discrete Gradient Descent, simply check the neighbours of the current cell to see if all of their potential values are higher than the current cell. If this is the case, the agent is certainly at a 'discrete local minimum'.

Discrete minimum escape: Backtracking

Immediately, one of the problems of backtracking is eliminated - the question of how to alter the descent mechanism so that an alternate path is taken to try and avoid the local minimum. In the case of continuous gradient descent, there is only the gradient to work with. In the discrete case, previously selected grid cells can be removed from consideration, allowing a different route to be taken.

There is a question of how many steps should be taken in backtracking. It is possible for the agent to try backtracking and exhaustively searching, one step at a time, until it succeeds in finding its way out of a minimum. This will be extremely computationally expensive to compute, even in 2-D. A far better approach, that is more commonly chosen, is discrete minimum filling.

Discrete minimum filling

Here, the agent is not faced with the problem of altering the gradient such that the agent is compelled to leave the minimum and not return, without becoming trapped in the process. Instead, since only potential value is being used, it is only necessary to raise the potential values of cells when they are visited. Since there are only a finite number of cells in each minimum that have to be considered, it is possible to iteratively raise the potential in each until the 'discrete minimum basin' is completely filled, and the agent finds that a more appealing route leads out of the (ex-)minimum. This navigation technique is sometimes used in the application domain of computer games [64], as it requires no precomputation, and even when the grid must be altered (to raise the potential value at certain cells nearby a local minimum), it is possible for the agent to keep memory requirements low by using a sparse array rather than a complete array to keep track of the positions it has altered[4].

[4] At the expense of a small degree of computational effort to operate the sparse array.

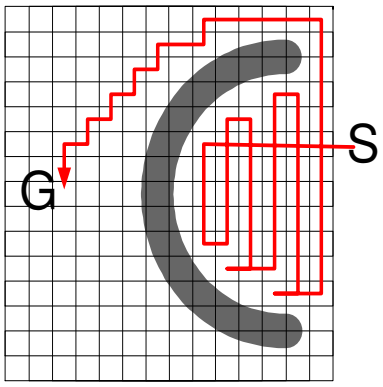


Figure 5.4: Example of minimum filling.

Unfortunately, as shown in Figure 5.4, the resulting path looks quite artificial, even in 2-D, with the agent zigzagging backwards and forwards much in the manner of a bricklayer building a wall, as it fills in the minimum basin. In higher dimensionalities, the technique may be prohibitively expensive, as it is likely to be infeasible to visit the finite volume of even a shallow discrete minimum basin.

A well known example of the minimum-filling approach can be found as part of Latombe's "Best First Planner" [166].

Discrete random movement

The most well-established technique for discrete potential-based randomised path planning is the Randomized Path Planner (RPP) [66, 67, 167].

RPP causes the agent to descend towards the goal when in-between minima, avoiding convex obstacles using discrete gradient descent, and upon becoming stuck in a minimum, it generates a path based on Brownian random motion to try and shake itself loose of the minimum before restarting descent. It is necessary however to check the positions suggested by random motion, to ensure they are in free space and are not inside obstacles, and thus avoid collisions during the random motion phases of RPP.

This approach is well suited to both low and high dimensional spaces. First of all, in high dimensional spaces, the problem of deciding that the agent is at a minimum can be made feasible by only considering a subset of the cells neighbouring the current cell, rather than every single cell, and thus estimating the likelihood that the agent is at a minimum. Secondly, traversing a random path of n steps in a highly dimensional space

is no more expensive that traversing a path in a 2-D space. Latombe describes in his book the successful application of RPP to simple eight and ten jointed robotic arm problems [167].

However, RPP faces problems when it is applied in the case where a large local minimum basin exists (e.g. a large concave obstacle arrangement). Here, because RPP is too defocused in its efforts to escape, it may not succeed - in other words, it becomes increasingly difficult to use random motion rather than directed motion to escape as minima become larger.

The paths generated by RPP are extremely unsmooth and un-natural as a result of the Brownian motion model employed. A smoothing technique is therefore necessary to make the suggested paths more useful for robotic or virtual world agents. There remains the problem of selecting a suitable random path length (though Latombe gives estimates of suitable path lengths for some configurations), and there is the risk of randomly re-entering a minimum that was previously escaped from. This is particularly problematic where there is a series of nested minima, perhaps due to a particularly jagged shaped obstacle lying between the agent and the goal.

Discrete Navigational Functions

The discrete navigational function might be described as the greatest triumph of discrete navigation approaches, in that it makes 2-D navigation and to a lesser extent 3-D motion in finite static environments highly feasible. In at least one sense, the discrete technique is more successful than its continuous equivalent - the cost of calculating a discrete navigational function does not rise with the number of obstacles, or their complexity.

A numerical navigational function can be defined iteratively as follows. The goal location is given value zero. The cells surrounding the goal are given the value 1. The cells surrounding those, are given the value 2 (if they have not already been given a value). This 'wavefront expansion' [65, 142] from the goal is repeated either until the environment has a complete navigational function defined, or until the starting location has been reached by the wavefront (depending on whether the discrete navigational function will be used for one-off navigation, or will be used many times from different starting locations).

The agent then uses discrete gradient descent from the starting location until it reaches the goal. Unfortunately, a key problem with the generated paths is that they 'cling' to the

surface of obstacles in building the shortest path backwards from goal to start. This means that an agent descending on the surface may risk grazing collisions with obstacles during its descent.

An alternative technique aims to prevent this by sacrificing the 'minimal-length' attribute that is inherent in the above algorithm, and instead generates a discrete navigational map that is something akin to the map of a voronoi-based 'Roadmap' technique [106], with paths as far from the obstacles as possible. Detail of this technique can be found in [142].

In both cases, there are a number of problems. This approach is impractical either for high-resolution grids, or for environments of more than two dimensions, due to the high computational cost of calculating the values at every grid location. The approach is also dependent on knowing the obstacles a priori, which makes it generally unsuitable for online navigation. There is an issue regarding realism of the derived behaviour too - virtual world agents that target discrete grid cell locations may have unusually 'jagged' paths that are dissimilar to those that might be chosen by a natural agent. Similarly, paths which always graze obstacle surfaces or always stay as far from obstacles as possible may not necessarily resemble the behaviour of a natural agent, and may damage the realism of virtual agents attempting to demonstrate natural behaviour - though this is a minor concern.

5.4.3 Conclusions

Although generally far more effective and suited for implementation than their continuous equivalents, the discrete forms of potential field techniques lose the benefits of being able to operate on arbitrarily large or high resolution environments, or in arbitrarily high dimensions, due to computational costs. A small element of realism is lost too, in constraining the agent to travel only between the discrete coordinates corresponding to the grid sampling points. The risk of grazing collision that is introduced by discrete techniques may also be a serious problem for real-world robots that are not suited for collisions, or for virtual world agents whose semblance of realism depends on them not overlapping obstacles in their environment.

5.5 Historical context of potential field based navigation

Potential field based navigation has been shaped by roboticists. A chronology of important and recent publications in potential field based navigation is given in Appendix C. Briefly though, the evolution of the field of research could be characterised in terms of landmark papers as follows:

In 1978, Khatib published his first paper with La Maitre [128] to suggest a need for a computationally lean yet flexible control technique for robots in a cluttered environment, and posits a potential fields approach. This is followed by his thesis in 1979 [123], and a series of related papers [124, 125, 126] on similar material, leading up to 1986. That year, Khatib published the landmark journal paper, "Real Time Obstacle Avoidance for Manipulators and Mobile Robots" [127], that is widely recognised as the start of mainstream research in this field. The approach faces several problems though, particularly the local minimum problem. A year later in 1987, Koditschek presents a landmark paper containing the minimum-free navigational potential field approach [132]. This is followed up by other papers by Koditschek et al. in 1989 and 1992 [133, 206] improving his technique.

By 1990, most of the research into Khatib's original heuristic model of potential fields has stopped, with hybrid approaches and discrete potential field approaches having been the best practical efforts at overcoming minima. Latombe published the most significant work in discrete techniques in [65, 66, 67].

Attempts were made to try and make the navigational field approach more general, by Connolly et al. in 1990-1992 [84, 85, 88] and Kim and Khosla in 1992 [130].

As early as 1991, doubts are being expressed as to whether the potential field method can ever be made to work [134], and by 1993, most of the research into variations of the navigational potential fields approach has been stopped, without managing to overcome key problems such as computational cost, inability to cope with complex environments or the limitation to offline navigation.

After 1993, there were no more major historical breakthroughs in this field, but rather there was a path of research focused on applications of potential fields in particular niches of robotics (e.g. [232] looks at adaptations for robotic walking sticks for blind people), and on making small modifications to existing work to try and incrementally improve, rather than revolutionise the field. The main areas of work were: hybridising potential

fields with some other mechanism - with potential fields being used to provide continuous trajectories [83, 169, 243]; slightly improving upon the computational costs of the potential field approach [179]; or re-applying potential fields to more complicated problems than basic online navigation, i.e. multiple agents [120, 174, 200, 201, 223, 226] or navigation under non-holonomic constraints [113, 221, 235].

Since the mid 1990s, potential fields have been found in lecture notes and books about navigation, but no revolutionary work has taken place - such as novel types of approach to the basic problem of navigation, or novel approaches to overcome the local minimum problem. Instead, small revisions are being made to existing approaches that have been around since the late 1980s and early 1990s.

5.6 Conclusions

Although some good efforts have been made at applying the potential fields approach to navigation, all have been hamstrung by a mixture of computational cost; vulnerability to potential surface features causing navigation failure or inefficiency; limitations to particular types of problem environment (disc-worlds, star-worlds, panel-worlds); problems through discretisation of the environment; problems through hybridisation with another technique to do the 'real steering'; dependence on obstacles not being nearby one another; ability to detect minima; poor quality of movement (that seems unrealistic for virtual world agents, and is hard or even impossible for real-world agents to enact); generally inefficient movement; revisiting minima that have been escaped; increased chance of collision and so on. Furthermore, the best continuous techniques are 'inelegant' in that they often require long descriptions, complicated maths and complex conceptual models to capture their behaviour - which in turn makes them impractical for implementation in some of the application domains of this research.

5.6.1 Consideration of hypothesis: Part 1

From the thesis hypothesis:

> "The Potential Fields Method has been prematurely abandoned as an approach to agent navigation, and the existing techniques have unaddressed weaknesses"

This section will consider the plausibility of the first part of the thesis hypothesis.

Has the Potential Fields Method been prematurely abandoned in navigation?

Since a handful of papers are still currently being published each year, either positing small refinements to potential fields navigation techniques, or using potential fields as part of a hybrid navigation technique, it is perhaps a little unkind to describe the field as being 'abandoned'. Unfortunately though, it seems that research has been almost entirely given up in the area of potential field based navigation, in terms of addressing key topics for basic navigation such as the local minimum problem through a pure, non-hybridised potential fields technique. The absence of 'landmark' papers with fundamentally new approaches since 1993, the gradual withdrawal of well-known researchers from publication in the area, and the general lack of industrial application suggests that the field of research has (at best) been in long-term decline over the last decade.

Nonetheless, the fact that even continuous gradient descent is still sometimes employed for local potential fields based navigation [44] despite its almost complete inability to provide reliable navigation in even slightly non-trivial environments, suggests that simple, computationally cheap approaches that generate smooth trajectories are still in demand. It seems reasonable to consider that if a successful potential fields technique similar to gradient descent could be identified - that lacked the local minimum problem yet retained the simple theoretical model, computational affordability and generation of smooth trajectories - then the potential fields approach might well attract attention once again in both research and industry. It therefore seems that the potential fields method does deserve to have further research effort applied to it, in order to find such a technique.

Unaddressed weaknesses of existing approaches

The weaknesses of the existing approaches have been highlighted in the sections corresponding to each technique in this chapter.

Are these introduced weaknesses still unaddressed? It seems that, given the lack of major publications to significantly better the situation of potential field based navigation as it stood in 1993, these weaknesses have persisted. The only technique which has seen any improvement is harmonic potential fields - and even then the improvements have been limited to a small reduction in computational cost and an improvement in the smoothness of paths generated.

Furthermore, not only have the identified weaknesses been left unaddressed, but attention has been shifted away from potential fields for 'basic navigation' and onto areas such as hybrid systems, multi-agent behaviour and non-holonomic constraints, so that there is now little chance that the weaknesses will ever be addressed in future, without a significant change in the focus of the researchers in the field.

Hypothesis Part 1: Conclusion

It seems that the potential fields method has been largely, if not absolutely abandoned. The reasons for this stem not from proven limitations of the technique, but rather from a longterm loss of research momentum, leading to a failure to find ways in which the technique can be made practical, and thus eventual abandonment of the idea that it can be made practical at all. Existing techniques clearly have unaddressed weaknesses that have stood uncorrected for well over a decade. It therefore seems that the first part of the thesis hypothesis is broadly correct, notwithstanding the ongoing trickle of research in the field.

5.6.2 Characterisation of an ideal potential fields approach

The historical context shows that research into the potential field approach to navigation has to some extent fallen into neglect. However, it is worth noting that even now, no non-potential field navigation technique (i.e. roadmap, cell-decomposition etc.) has emerged as a problem-free answer to real-world or virtual agent navigation. Potential field approaches should therefore still be considered an open area of research, that is worthy of further work.

It may be useful to characterise the features that a new potential fields technique should aim to provide.

- It should be 'purely' potential field based, so as to be general and not reliant upon a particular type of implementation. It should be a non-hybrid approach to avoid introducing the weaknesses of other approaches, and to ensure that the listed benefits of potential fields techniques are retained (see Section 4.2)
- It should be computationally cheap both to generate and use the potential field.
- It should be capable of online navigation - i.e. capable of adding new obstacles to the problem in real time.
- It should not rely on a particular type of environment (i.e. should not require that obstacles are represented as panels, trapezoids, or disks).
- It should be conceptually straightforward, avoiding complex mathematics or unnatural analogies where possible. This implies that it may need to use the traditional 'obstacles push, goal pulls' model of potential fields, rather than the navigational fields model of 'hydrodynamic behaviour of incompressible fluids'.
- The approach should be suitable for straightforward and rapid implementation.
- The approach should not be vulnerable to the problematic surface features identified in Chapter 4.
- The approach should ideally not be vulnerable to the weaknesses identified by Koren and Borenstein (which they believed affected all potential field methods).
- The approach should ideally produce behaviour identifiable as being similar to that of a natural agent such as a human or animal in the same circumstance.
- The approach should avoid re-visiting local minima that have been escaped from.
- It should be feasible to consider extending the approach to non-holonomic constraints, multiple robots and higher dimensions.
- The approach should be continuous in nature, rather than resorting to a discretisation of the environment - particularly since discretisation can reduce the usefulness of the approach in the case of navigation in environments of higher dimensionality.

These characteristics form the motivation for the novel research contribution that will be presented in the remainder of this thesis. Some of the criteria are qualitative and will be difficult to assess. They are discussed in the experimental results in Chapter 7.

5.7 Chapter Summary

This chapter has presented:

- A discussion of the variety of techniques invented to try and overcome the local minimum problem on continuous potential fields.
- A detailed examination of the most successful of these techniques - *navigational potential fields*, in which the local minimum problem is overcome by defining a mapping between navigation problem and potential field that does not allow local minima to appear in the first place.
- A discussion of approaches to potential fields based on discretisation of the field.
- An overview of the historical context of the research field.
- An evaluation of the first part of the hypothesis.
- Conclusions and characterisation of an ideal potential fields approach to navigation.

On the next page, Table 5.1 summarises the properties of the main potential field based techniques discussed in this chapter.

Table 5.1: General properties of significant potential field navigation approaches. Footnotes overleaf.

	Gradient Descent	LM Detection + Random Movement	Elliptical Potential	Continuous Navigational Function	Discrete Gradient Descent	Discrete Random: RPP	Discrete Minimum Filling	Discrete Navigational Function
Online navigation?	✓	✓	✓	✗	✓	✓	✓	✗
Computationally cheap to generate field?	✓	✓	✓	✗	✓	✓	✓	✗
Computationally cheap to use field?	✓	✓	✓	✓	✓[1]	✓[1,2]	~[3]	✓[1]
Simple model/easy to understand?	✓	✓	✓	✗	✓	✓	✓	~[4]
Simple implementation?	✓	✓	~[5]	✗	✓	✓	✓	~[4]
Robust (to noise or errors)?	✓	✓	✓	✓[6]	✓	✓	✓	✓
Generates smooth trajectories directly?	✓	✗	✓	✓	✗	✗	✗	✗
Can navigate in 2-D?	✓	✓	✓[7]	✓	✓	✓	✓	✓
Can navigate in 3-D?	✓	✓	✓[7]	~[8]	✓	✓	✓[3]	~[9]
Can navigate in n-D?	✓	✓	✗[7]	✗	✓	✓[2]	✗	✗
Unaffected by local minima?	✗	~[10]	✗	✓	✗	~[10]	✓	✓
Unaffected by ravine problems?	✗	✗	✗	✗	✓	✓	✓	✓
Unaffected by plateau problems?	✓	✓	✓	✗	✓	✓	✓	✓
Unaffected by saddle points?	✗	✓	✗	✗	✓	✓	✓	✓
Allows arbitrary obstacle shapes?	✓	✓	✗	✗	✓	✓	✓	✓
Doesn't re-visit visited parts of the environment?	✓	✗	✓	✓	✓	✗	✓	✓
Avoids grazing collision with obstacles?	✓	✓	✓	~[11]	✓	✓	✓	~[4]
Works in trivial environments?	✓	✓	✓	✓	✓	✓	✓	✓
Works in intermediate environments?	✗	~[10]	✗	✓[12]	✗	✓	✓	✓
Works in complex environments?	✗	✗	✗	✗	✗	✗	~[13]	✓[14]

Footnotes to Table 5.1

[1]Cheap in 2-D and 3-D, but expensive in high dimensional problems (i.e. inverse kinematics), as computational cost rises exponentially with the number of dimensions.

[2]Latombe has demonstrated RPP being effective for e.g. 11-D inverse kinematics [167].

[3]Filling a minimum can be computationally expensive even in 2-D and 3-D depending on the number of possible states lying inside the minimum basin.

[4]The minimum-length approach is easy to understand and implement, but risks grazing collisions with obstacles. The 'voronoi-roadmap'-like approach is much more difficult to understand and implement but avoids the risk of grazing collisions.

[5]Latombe comments on the practicalities of implementing this approach in [164].

[6]See Section 5.3.3.

[7]The functions are only defined in 2-D and 3-D [129, 164].

[8]Can be done but at very high computational cost.

[9]Computational cost rises with the number of possible states in the environment. In 3-D, this implies a limited size and resolution of grid, to compensate for the increased dimensionality of the state space. The technique can therefore be used in 3-D, but only with limited effectiveness.

[10]Varying degrees of success depending on technique.

[11]Depends on the choice of boundary conditions (Direchlet or Neumann). Direchlet is more meandering and inefficient, but Neumann risks grazing obstacles.

[12]Works to a limited extent - can support complex arrangements of a few obstacles, but not obstacles of complex shape, or environments with many obstacles in total.

[13]The paths generated will be *extremely* non-optimal, as it will take a very long time to fill in all the minima in a complicated environment.

[14]This will work in arbitrarily complex environments in terms of numbers of obstacles and their shapes and arrangement, but the technique is limited by the size, resolution and dimensionality of the environment.

Chapter 6

Novel Technique: Forward Chaining

Guide to Chapter 6

The preceding chapters have looked first at a type of problem, then at a type of problem solving approach, and finally at the compositions of the two to date. The demands of the application domains (robotics and virtual world agents) provide a strong incentive for an elegant and useful join to be made between *Navigation* and *Potential Fields*.

This chapter introduces the primary novel aspect of this thesis - the development of the 'Forward Chaining' potential field based navigation technique.

The technique derives its name from the combination of the ideas of 'Subgoal Chaining' and 'Forward Motion', and is unrelated to the 'Forward Chaining' technique used in the field of logical inference.

6.1 Chapter Overview

This chapter is split into two sections.

- A brief introduction to the research that inspired this work.
- An overview of the development of a novel potential field based navigation heuristic.

6.2 Relevant local potential fields research: Subgoals

6.2.1 Overview

This chapter begins by considering a recent successful piece of potential field research carried out at the University of St Andrews [171, 173, 238] within the field of neural network training (an area of research that is not closely related to navigation, except in that potential field techniques are used in both areas).

6.2.2 Problem addressed

The problem addressed by Lewis is that of training neural networks [171]. Lewis addressed several problems including the benchmark neural network problem, '2-spirals', but the main part of his thesis concentrated on training on potential field surfaces that he constructed, which were guaranteed to contain local minima. The problem was to successfully reach a global minimum (and thus a trained neural net) given the presence of problematic local minima on the potential field between the start and the goal.

The presence of these local minima means that gradient descent techniques fail almost 100% of the time [173], as any random starting configuration chosen for training will usually tend to be within the basin of a local minima rather than a global minimum. Lewis' research focused on finding a means to train the neural network reliably using gradient descent on such problematic surfaces.

6.2.3 Technique

Firstly, Lewis promoted the idea of *subgoals*, previously developed in neural networks in [114, 115, 173, 237, 238]. The idea here is that rather than immediately targeting the global minimum of a potential field, the heuristic instead temporarily targets a more achievable position nearby the agent's current location, by turning it into a temporary global minimum. This is done through temporarily modifying the original problem's potential field. The idea is to ensure that although only a small amount of local travel will be possible, the travel will not be held up by local minima on the original potential field. It is highly unlikely that there will be a local minimum on the temporary field between the current position and the nearby temporary goal.

Though this technique was only adopted in training neural nets in 1993 [114, 237], it has been quite commonly found in early navigation research, particularly since 1986 when Krogh's paper used potential fields as an interpolating mechanism between pre-selected subgoals as part of a hybrid technique [135]. 'Waypoints' [17], essentially human-generated subgoals to guide any navigation technique, have been a feature of navigation approaches in computer games since 1997[1]. A significant difference between Lewis' work and existing work is that Lewis' technique generated subgoals dynamically using the agent's current state information at each point, rather than through a pre-computed analysis of the surface or of the underlying problem it represents.

Secondly, Lewis promoted the idea of using such subgoals in a sequential manner (known as *Subgoal Chains*), to allow steering of an agent using gradient descent. The idea here was that the agent could travel in a safe manner by being carried along in a series of temporary 'local' global minima - in other words a changing global minimum that is kept close enough to the agent to be continually achievable. This idea of sequential subgoals was also published in [115] as part of an unsuccessful linear-chain technique and is implicit in hybrid techniques for navigation. The up-front observation that subgoals must be kept close enough to the current state that they can be sequentially achievable is an important contribution from this work. It is interesting to notice that some other waypointing mechanisms that exist (in games, or hybrid potential field techniques) do not make this requirement explicit, which may lead to navigation failure if a subgoal is sufficiently far enough away to be unachievable.

Thirdly, he developed the idea of a '*cheat chain*' - stating that if a researcher knew a priori

[1] See Section 2.6.7.

where the global minimum was, and if the researcher knew where local minima were (or were likely to be), then it should be possible to plot 'cheat' subgoal chains that lead the agent safely along a path to the global minimum, even when passing near local minima on the *original* potential field surface.

Finally, he developed a technique which used cubic splines to extrapolate the good, realisable positions that had been traversed already, in order to try and predict future good, realisable positions to use as subgoals - in other words, he attempted to generate a cheat chain automatically and dynamically through a heuristic which exploited knowledge it gathered during travel.

6.2.4 Applicability to this research

Weir, Lewis and Milligan [173, 238] recognised that non-linear subgoal chains are an inherently more powerful form of potential field travel than simple gradient descent. Importantly, Lewis recognised that if an a priori cheat chain of subgoals will allow an agent to succeed, then a system that is able to produce cheat chains (or an approximation) dynamically should have an improved chance of success over gradient descent alone.

Lewis' neural networks research leads to a motivation for novel navigation research. If a suitable heuristic technique for producing 'cheat chain'-like subgoal chains can be developed, then the resilience of gradient descent based navigation to local minima might be improved.

Unfortunately it is not possible to apply Lewis' work directly to the problem of potential field based navigation, for many reasons - significantly, his technique depended upon the agent having the ability to move anywhere in its configuration space at will. Navigating agents do not have such freedom, or they would be able to move straight through obstacles to reach a goal - indeed they could even jump immediately to the goal in a single step, defeating the purpose of navigation. However, it is still useful to keep in mind the idea of trying to dynamically generate heuristic approximations to 'cheat chain' subgoal chains, using the local knowledge the agent has available.

6.3 Subgoal Chaining

The abstract idea behind subgoal chaining is essentially that of divide and conquer. Navigation to a goal using potential fields can be a difficult or intractable problem, and by breaking the task into smaller subproblems, it may become more tractable. This implies that instead of carrying out descent on a fixed potential field based upon the goal, descent will be carried out on a dynamically varying potential field according to the subproblem currently being addressed.

It is therefore necessary to provide a technique for breaking down the large navigation problem into tractable subproblems, as well as a technique that can solve the more feasible subproblems. In the case of potential field based navigation, a subgoal chaining template heuristic of the following form can be constructed:

Definition Template Subgoal Chaining Technique for Potential Field Descent

1. Pick a subgoal position SG that, when navigated towards using a potential field based interpolative technique, is likely to produce progress towards the goal position.
2. Generate a potential field for navigation, PF_{SG}, that corresponds to the problem of navigating to the subgoal SG in the original obstacle environment.
3. Carry out a potential field based interpolative technique to generate a path between the current position and the subgoal SG using PF_{SG}. In the event that navigation to SG fails (i.e. has not been achieved after some period of time attempting to navigate to it), return to step 1.
4. If the final goal has been reached, then exit. Otherwise, return to 1.

The single problem of potential field navigation is here transformed to four different, but hopefully simpler problems. These will be considered in order of increasing difficulty.

6.3.1 Generating a potential field for subgoal navigation

Step 2 can be addressed directly. Since the subproblem is itself still a navigational problem, the subgoal's temporary potential field can be generated in exactly the same manner as the

potential field of the original navigation problem was generated, with the subgoal being substituted in place of the goal in the equations.

6.3.2 Ending navigation

Step 4 is also straightforward to address. If the agent's current location is within some threshold distance of the goal, then navigation can be considered as successful. The threshold can be set as close to the goal as is necessary to suit the specific nature of a given problem.

6.3.3 Potential field based interpolation between subgoals

Step 3 is less trivial. How can an interpolative trajectory be produced in the case of navigation to a subgoal, when the problem of navigation is such a difficult task in the first place?

It is important to realise that the navigation that is now taking place is towards a target that is much closer than the original goal. Indeed, if the subgoal is sufficiently close to the current position, it is probable that there should be few obstacles near the subgoal, and therefore few - *if any* - local minimum basins or other problematic surface features lying between the current position and the subgoal position to prevent successful navigation to the subgoal. The definition of 'close' will be refined further later in this chapter.

Given that this is the case, and looking back to Table 5.1, it seems that continuous steepest gradient descent might be a good option to consider as an interpolative technique. It is already well known and accepted as an option for local navigation. It is suitable for use in navigation in spaces of any dimensionality, is computationally cheap, is suitable for navigation in continuous spaces, and can produce arbitrarily smooth paths. It has a simple intuition behind it, and it should be already familiar to most researchers and implementers.

A problem may still remain, however - what if the subgoal chosen is unachievable for some reason by gradient descent, perhaps because it is placed too close to an obstacle, or because there is still a local minimum on the surface?

To overcome this, the subgoal selection mechanism must either avoid producing this

problem in the first place, or it must be possible to generate subgoals under the condition that the previous subgoal may not have been achieved. If one of these conditions can be met, then continuous gradient descent should provide a suitably useful potential field based interpolative technique as the solution to Step 3.

6.3.4 Subgoal selection

The only remaining problem is the generation of subgoals. This is by far the most difficult task to overcome in implementing a subgoal chaining technique, and it may not be immediately obvious that this step is even possible to achieve. It may even seem that this task is itself a repetition of the problem of navigation.

This is not the case however. Here, the task is not to generate a trajectory to the goal, but to select optimal positions for a series of subgoals that may allow gradient descent the best possible chance to generate a useful trajectory. By removing the concern of actual trajectory generation, it is possible to concentrate on selecting subgoals that shape the behaviour as desired. The task is therefore to try to find a mechanism that is sufficiently capable in the task of selecting subgoals that the sequence of subgoals selected approximates one of the 'cheat chains' discussed earlier.

In implementing the task of 'cheating' however, it is important to utilise only information that is contained in the potential field, information about the agent's current position and the goal position on the field, and any internal state information gathered en route. Breaking from this, and developing an approach that is computed using (for example) the shape or nature of the obstacle configurations as they are sensed directly by the agent or represented in an abstract metric map, will reduce the generality of the technique particularly in terms of re-applicability across domains. Furthermore, it would disqualify the algorithm or heuristic from being considered as the successful 'pure' potential fields approach that this thesis moots the existence of.

The next sections describe the evolution of a subgoal selection heuristic. To keep the problem of navigation as tractable as possible, and to ensure that it is straightforward to visualise the ideas in the development of the subgoal selection heuristic, the problem of navigation in only 2-D environments will be considered from here onwards, throughout the rest of this chapter. Later sections of this thesis (Chapter 8) will demonstrate techniques to extend the approach to allow 3-D, and n-D navigation.

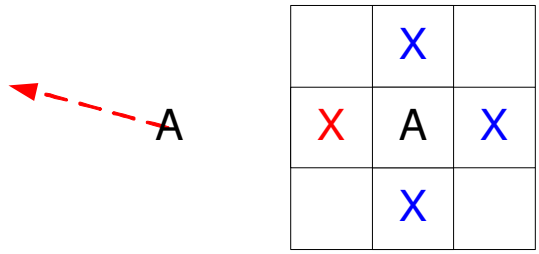


Figure 6.1: Behaviour of continuous and discrete gradient descent.

6.4 Subgoal Selection: Polynomial Splines

The first technique considered for subgoal selection was the polynomial spline. This technique was considered as the initial approach in this research partly because it is the classical approach in mathematics for generating a parameterised path, and partly since it had already been applied successfully in Lewis' research [171] as a steering mechanism for subgoal chaining in neural networks.

Unfortunately, polynomial splines lack an awareness of obstacles or potential functions, and so the subgoals they place repeatedly direct the agent straight towards the goal despite the presence of an intervening obstacle that prevents success. Polynomial splines were therefore disqualified as an approach to subgoal generation.

6.5 Reflections upon gradient descent

After initial experimentation with polynomial splines failed, it seemed that replacing the failure-prone interpolation technique (continuous gradient descent) with something a little more robust might not place such high demands upon the subgoal selection mechanism.

Consider that the existing interpretations of steepest gradient descent are trying to do approximately the same thing in two different ways. Recall that there are two basic forms of steepest gradient descent: continuous gradient descent, which chooses a direction for descent using the local gradient at the agent's current position, and grid based discrete 'gradient' descent, which actually selects a neighbouring grid square of lowest potential value, rather than using the mathematical gradient. Figure 6.1 illustrates the differences between the two.

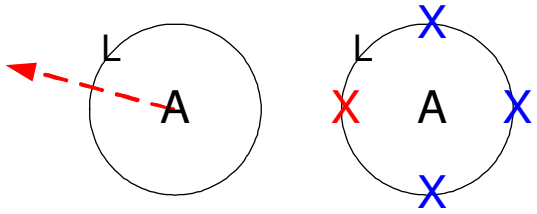


Figure 6.2: Behaviour of continuous and discrete gradient descent. 'L' signifies the position of the point of lowest potential lying at a fixed distance from the agent.

Essentially, both techniques are trying to move rapidly downhill on the potential field. In order to do so, they both perform a computationally cheap 'trick' to estimate where the lowest point on the potential field will be, in the area around them. This principle will no doubt be familiar to anyone who has ever considered the idea behind the design of the gradient descent heuristic. However, now consider Figure 6.2.

In a sense, one might say that both techniques are trying to obtain a cheap estimate as to which point on the circles shown on the potential field is the best place to go, with the aim of moving downwards quickly.

Consider though, the situation of a researcher given access to an unlimited amount of computational power. The thought might occur to them that there is actually a much better way to obtain such an estimate of the lowest point on the circle. They could directly evaluate the potential value not just at the four points of grid-based discrete gradient descent, but at *every single point on the circle*. This would give a technique that is completely continuous in nature (in that a point in any direction could be chosen, like continuous gradient descent), and that gives a completely accurate measure of the position of lowest potential value at a particular distance, rather than an estimate. The technique would scale to higher dimensionalities by considering every point on the surface of a sphere in 3-D, or hypersphere in n-D.

A more feasible approximation to this technique is to instead consider examining every point on a very good approximation of the circle - perhaps by taking one sample at, say, each of the 360 degrees spaced around the circle. Let us call such a technique, the *LPCIRCLE descent technique*, as it selects the 'lowest point on a circle' around the current position.

6.5.1 Definition of the LPCIRCLE field descent heuristic

1. Start at an initial position p_0.
2. Compare the value of $f(p_0)$ at all the points (or an approximation of all the points) that lie at positions at a fixed distance D from p_0, and make a note of the point of lowest value, p_1.
3. Move to p_1.
4. Repeat from step 1, using p_1 in the place of p_0.

6.5.2 Could LPCIRCLE be a useful alternative to gradient descent?

Might it be possible to replace continuous gradient descent with LPCIRCLE for the task of subgoal interpolation, in order to gain improved performance and thus facilitate an easier problem for the subgoal selection heuristic to solve?

Computational cost

First of all, computational cost is considerably worsened, rather than improved by LPCIRCLE. Sampling so many points around the current position is far more expensive than either continuous gradient descent (which requires only a single sampling of the gradient per step, irrespective of the dimensionality of the environment) - or grid based gradient descent (requiring $2n$ samples for a n-D environment per step, which is a quite reasonable 4 samples in 2-D and 6 samples in 3-D). In comparison, the LPCIRCLE technique would need 360 samples in 2-D (one at every 1-degree interval, say), and $2.(180^{n-1})$ samples in n-D. Even in 3-D, carrying out a single scan of the sphere around the agent at regular 1 degree spacings would take 64,800 evaluations of the potential field function.

To repeat Latombe's experiments [167] with planning in 11-dimensional space would require $2.(180^{10})$ computations[2] of the potential field function just to take a single step down the potential field - and there might be tens of thousands of such steps down the field.

[2] $2.(180^{10}) = 7.1 \times 10^{22}$.

It would be possible to imagine cutting down the number of samples, so that they are spaced at every 10 degrees around the circle, and therefore allow a ten-fold better execution speed in 2-D, and a 10^{n-1}-fold better execution speed in n-D. Nonetheless, the amount of computation involved still vastly exceeds either continuous or discrete gradient descent, with $2.(18^{n-1})$ samples necessary to take a single step along the field in an n-dimensional space. Even a *single step* of descent may be impossible in practice in environments of dimensionality above 5-D or 6-D at this resolution (on some platforms).

Perhaps fewer steps can be taken though, to mitigate the high cost of a single step? Unfortunately, there is also now the additional risk that if part of an obstacle lies between the agent and the arc of the sampling circle, LPCIRCLE will register the low value of the potential on the far side of the obstacle. If one of these low-valued points is the lowest point on the circle, then the agent might consequently try to 'step through' the obstacle to try and reach the low potential position, in a similar manner to the way in which continuous gradient descent may be misled by the gradient into stepping 'into' an obstacle. This can be prevented from happening (given that LPCIRCLE will generally keep away from the boundary of an obstacle anyway, due to the fact that positions there have extremely high potential value) by keeping the descent step size small. Effectively, the LPCIRCLE descent step size must be of similar size to other forms of gradient descent - so the increased computational cost per step of this descent technique is not mitigated by allowing fewer steps in descent.

Overall, LPCIRCLE offers only a tremendous worsening of computational cost compared with the standard techniques of continuous gradient descent or grid based discrete gradient descent, and no possibility for computational cost reduction at all. LPCIRCLE descent is effectively intractable in problems of much higher dimensionality than 2-D or 3-D environments.

Success rate in descent

Given that LPCIRCLE has such considerable computational costs associated with it, one might hope that it provides considerable improvements in success rates in return.

Unfortunately, this is not the case. The technique does provide a more accurate estimate of the lowest point at a distance from the current position, but there are two problems with this:

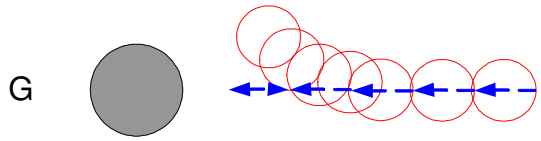


Figure 6.3: LPC (red) vs. GD (blue), saddle point.

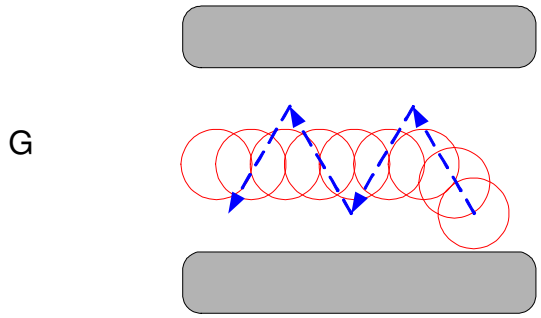


Figure 6.4: LPC (red) vs. GD (blue), ravine problem.

- In cases where the agent is near a local minimum, LPCIRCLE is much *more* likely to fall into the minimum problem, as it samples potential directly and exhaustively as it travels (unlike the other forms of gradient descent, which can 'miss' a local minimum by chance, depending on the exact points at which gradient or potential are sampled).
- The ability to make an improved estimate of the point of lowest potential only ensures that the agent will stay absolutely trapped in almost all local minimum problems.

Several small benefits over continuous gradient descent are provided however when facing surface features other than the progress-halting local minimum problem.

- In the case where descent is initiated on a saddle point (or in the problematic axis of a saddle point [69]), LPCIRCLE will be able to descend successfully unlike continuous gradient descent, as positions will be sampled on the sides of the saddle that are of lower potential than the saddle point position (Figure 6.3).
- When a plateau is faced, the fixed step size ensures that progress should continue at a steady pace.
- When a ravine is faced, LPCIRCLE will descend towards the center of the ravine, and will immediately 'lock on' to the center of the ravine as soon as the scanning circle overlaps the bottom of the ravine, so that subsequent steps of descent take place directly down the center of the ravine, without oscillation (Figure 6.4). This

happens because there is no chance that the heuristic will select a point lying up the side of the ravine, as such points will be of higher potential than the point leading most directly down the center of the ravine. This is *highly* desirable behaviour in the case of a ravine surface feature.

Important differences

It is important to note the difference in the nature of the descent and dependencies of the techniques. Gradient descent uses the *gradient* function to try and estimate the most optimal position to move to from the current position. LPCIRCLE uses the *potential* function to find out for certain which is the most optimal position to move to at a certain distance. It is unnecessary to have access to a gradient function in order to carry out LPCIRCLE descent.

Summary of LPCIRCLE properties

In terms of path shape and success rates, LPCIRCLE combines the best elements of continuous and discrete gradient descent but does not exceed them. Paths can be arbitrarily smooth (depending on the resolution and radius of the scanning circle), and LPCIRCLE is as successful as discrete gradient descent would be on a grid. Although the technique is (in rare circumstances) able to escape very small local minimum problems, the technique is generally still vulnerable to local minima and can even be captured by local minima that other descent techniques might miss. The cost of the technique, relative to the other approaches, varies from a couple of orders of magnitude worse (in 2-D), to many orders of magnitude worse (in 3-D), and to such high costs in n-D that the heuristic is rendered effectively intractable.

6.5.3 Why introduce LPCIRCLE?

Given that the LPCIRCLE descent technique introduces remarkable computational penalties in exchange for no real improvement in success rate when facing the main problem of local minima, why introduce it as part of this research?

Though the LPCIRCLE heuristic may carry a high price, as a form of potential field

descent, it provides two things that the other techniques do not.

1. It provides the best performance in terms of smooth path shape and response to problematic surface features (excluding local minima).
2. It gives the researcher *the circle*[3]. What is meant by this, is that LPCIRCLE provides the researcher with a straightforward mechanism by which they might easily express ideas such as wishing to select points only from particular parts of the circle, or wishing to de-select points from parts of the circle. This will later turn out to be a very useful property.

Unfortunately, the high computational expense of the technique and inability to overcome significantly more local minimum problems than continuous gradient descent, force LPCIRCLE to be ruled out as an improved descent heuristic, suitable for replacing continuous gradient descent in the task of subgoal interpolation.

However, LPCIRCLE represents an excellent candidate technique for a *subgoal selection technique*! It could help continuous gradient descent overcome non-minima surface features, and via the *circle* metaphor described above, it has the potential to be further refined in order to overcome local minimum problems. If it can be used in this way, then it will not be necessary to replace continuous gradient descent with an alternative potential field based subgoal interpolation technique after all.

Particularly, if LPCIRCLE is only used to select subgoals, it is not necessary to carry out LPCIRCLE so often in order to move over the potential field. Instead, LPCIRCLE - or indeed, its variants - can be used only occasionally, in order to select subgoals that steer the computationally cheaper continuous gradient descent heuristic.

The LPCIRCLE technique is believed to be original and novel (though it is not the primary contribution of this thesis). A short discussion of the novelty of the approach is provided in Appendix D.

This chapter now presents an initial subgoal selection technique based on the LPCIRCLE potential selection method, which will use continuous gradient descent to interpolate between the selected subgoals.

[3]Or hypersphere.

6.6 Subgoal Selection: LPCIRCLE

This section will now describe how LPCIRCLE could be used as a subgoal selection heuristic, and will examine the qualitative performance of the heuristic.

6.6.1 Fitting LPCIRCLE into the Subgoal Chaining template

LPCIRCLE will now be used to try to implement the first step of the template subgoal chaining technique discussed earlier. To recap:

1. Pick a subgoal position S that, when navigated towards using an interpolative technique, is likely to produce progress towards the goal position.

It is hoped that LPCIRCLE can be made to select subgoals that produce the desired progress towards the goal, when interpolated between using gradient descent. To do this, first it is necessary to define the way in which subgoals are chosen by LPCIRCLE, which can then be sequentially targeted by interpolating gradient descent as discussed earlier in Section 6.3 to achieve long-term navigation.

LPCIRCLE selects the point of lowest potential value on a circle surrounding the current position. In order to choose a point to use directly as a subgoal, it is necessary to define the radius of the circle that LPCIRCLE will use, and the number of points around the circle to sample.

6.6.2 Radius of the sampling circle

- If the radius of the circle is too small, or in other words, if the 'subgoal step size' is of similar size to the continuous gradient descent method being used to interpolate, subgoal selection will be carried out far too often, and we may as well simply use LPCIRCLE directly as the descent mechanism rather then pretending that gradient descent is really being used at all - as the subgoals will not be so much an aid to navigation, but rather the entire navigation itself.
- If the radius of the circle is too large - for example, if it is set to be of similar magnitude to the distance between the current position and the goal, then the subgoal

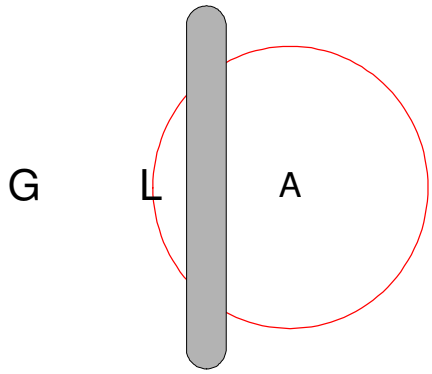


Figure 6.5: The agent (A) will select an unachievable subgoal at position L in trying to reach the goal (G).

chosen will immediately be the goal - and subgoal chaining will not have improved the tractability of navigation in any way, as the original navigation problem will not have been reduced to new subproblems.

- Intuitively, one would expect that the radius of the sampling circle will be of an order of magnitude that lies around midway between these two extremes.

Assuming that subgoals must usually be achievable in order for them to guide gradient descent based navigation, a question can therefore be put forward: "how far apart can subgoals be set, before they start to become repeatedly unachievable?". Since generating extra subgoals incurs extra computational cost, it is ideal that subgoals are generated as infrequently as the navigation problem permits, but often enough that each subgoal is achievable and able to steer gradient descent effectively to success.

6.6.3 Interaction of LPCIRCLE with obstacles

Consider what will happen in the situation in Figure 6.5. Here, an agent is scanning for a subgoal, at all positions on the circle shown. The circle overlaps the obstacle, and part of the circle to be sampled lies beyond the obstacle, and points here are likely to be of lower potential than any other points sampled around the circle. The agent will therefore select a subgoal that essentially requires the interpolating gradient descent technique to travel through or around the obstacle - neither of which gradient descent is capable of.

As was the case with splines, when gradient descent is set a subgoal that lies in the same direction as the goal, local minima will exist for the subgoal at the same positions as they

exist for the goal. The agent will move forward under gradient descent, until it becomes trapped in the local minimum basin caused by the subgoal and the obstacle.

Eventually, gradient descent will report failure to reach the subgoal. At this point the subgoal selection routine will attempt to generate a new subgoal using LPCIRCLE. This subgoal will be set at approximately the same point as before, and gradient descent will now be starting from a position that is at the bottom of a local minimum on the potential surface created by the combination of subgoal and obstacles (let us call this potential field the *subgoal-obstacle field*). Therefore, allowing subgoals to be set beyond obstacles will result in repeated failure to achieve subgoals[4]. LPCIRCLE based techniques must therefore have a subgoal step size that is no larger than the smallest dimension of any obstacle - i.e. perhaps half the width of the smallest obstacle, for example.

The other factor affecting the subgoal step size is the scale of detail in the environment. It is possible to imagine a situation such as a cliff-face with a small cave within which it is the agent's task to navigate. Here, the obstacle thickness is very high as the cliff is potentially kilometres thick. Subgoal step size should therefore be set appropriately so that LPCIRCLE is allowed a reasonable number of steps in the free space of the environment in which it is trying to navigate - a step size of 1 kilometre does not help the agent navigate in a 10 metre long cave. On the other hand, there is no point in setting the subgoal step size so small that it will take thousands of steps to move even a tiny distance within the navigation problem. In essence, there is a certain degree of common sense involved in setting the subgoal step size according to the scale of the two environmental factors of obstacle thickness and the typical size of areas of free space as a result of obstacles.

6.6.4 Number of points around the circle

It is necessary to choose a number of points to be evaluated around the circle. It is not obvious that the spacing of the points will greatly affect the behaviour of the heuristic, providing there are enough points for the technique's sampling to approximately represent a circle, and providing that the gap between each sample is not so large that there could be a large variation in the potential values between samples, making the lowest point selected on an approximated circle perhaps unrepresentative of the lowest point on a whole circle.

[4]It is worth observing that it is of course impossible that a subgoal might be set inside an obstacle, since the LPCIRCLE technique picks out points of lowest potential, and the area (or volume) inside an obstacle is always of the highest potential on the field - indeed, ideally it will be of infinite potential - and thus will not be selected by the heuristic.

Since variation in the field is largely caused by the non-linearity of the obstacles function, it is therefore best to select a sample spacing that would not allow an obstacle to lie between consecutive sampled points. This depends on the radius of the circle, and the size of obstacles on the field.

If the radius of the circle is (in the worst case) the width of an obstacle as defined above, this implies that the number of samples taken around the circle should be greater than 2π, implying a worst-case sample spacing of around 57 degrees. It is however a good idea to use a far higher number of samples than this, in order to allow the agent to choose from a wider range of possible directions and thus more closely approximate a continuous technique.

6.6.5 Configuring gradient descent for subgoal interpolation

The step size of gradient descent will normally be set according to the capabilities of the agent to move reliably in the physical or virtual world in which it resides. The minimum resolution of a step will however typically be very much smaller than the width of an obstacle and will also be set so that the agent cannot step 'into' an obstacle in following the gradient. Typically, 10-100 gradient descent steps might be used between subgoals to generate a smooth-looking path.

6.6.6 Definition of the LPCIRCLE subgoal selection heuristic

1. If the distance between the current position and the long term goal is less than distance D, set the subgoal to be the goal. Otherwise:

2. Scan the circle at a distance D surrounding the current point, for the point of lowest potential and use it as the subgoal. To do this:

   (a) Calculate the value of the goal-obstacle $U(x, y)$ potential field at a distance D from the current position for a selection of angles in the range 0-360 degrees around the current position.

   (b) Pick the angle at a distance D which resulted in the lowest potential, and use the corresponding position as the subgoal for gradient descent.

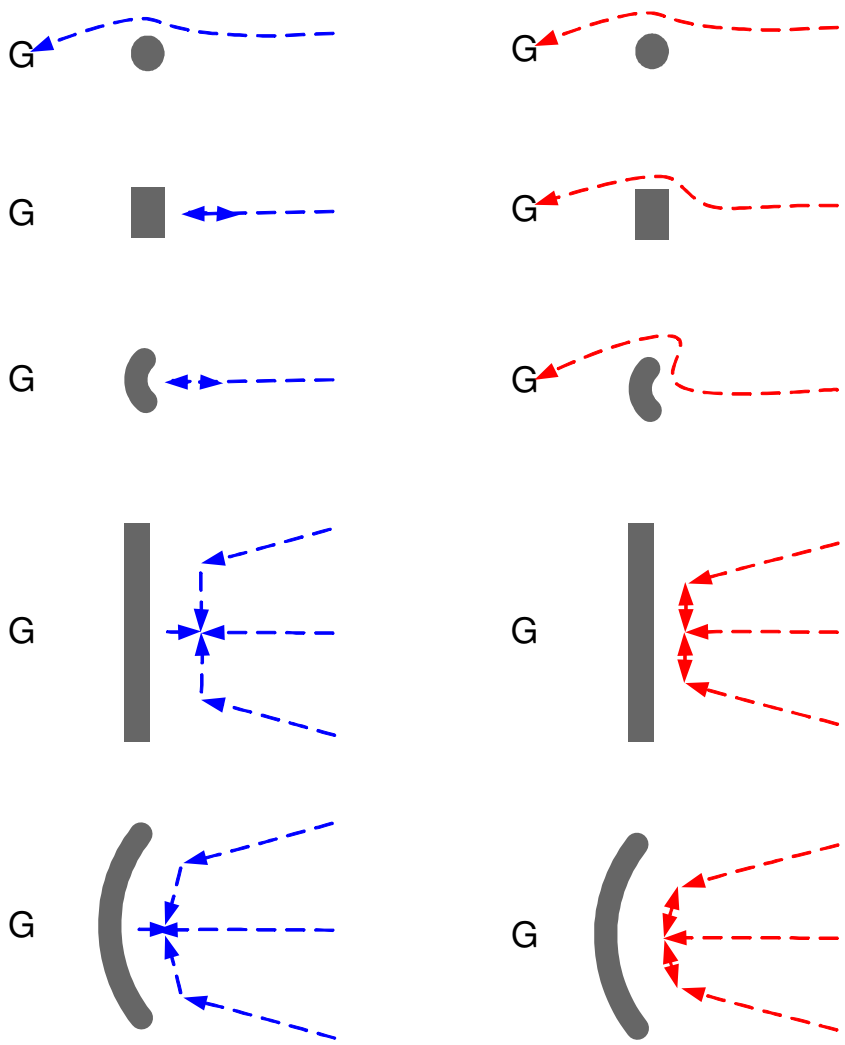


Figure 6.6: Behaviour of Gradient Descent (left) and LPCIRCLE subgoal chaining (right) facing a number of obstacle types.

6.6.7 Performance of LPCIRCLE

Figure 6.6 shows examples of the behaviour of LPCIRCLE subgoal chaining contrasted with that of gradient descent when faced with: a convex obstacle; a small flat obstacle; a small concave obstacle; a flat obstacle; a concave obstacle. In each case, the obstacles are set between the agent and the goal (G).

The following observations can be made.

- LPCIRCLE subgoal chaining generates a path of similar smoothness to traditional gradient descent.

- Gradient descent is only able to navigate past convex obstacles placed between the agent and the goal.
- LPCIRCLE subgoal chaining is able to navigate in environments with convex obstacles and small flat or concave obstacles lying between the agent and the goal.
- LPCIRCLE behaves differently to gradient descent when faced with larger flat or concave obstacles, but like gradient descent, is not able to overcome such obstacles.

6.6.8 Behaviour of LPCIRCLE vs. gradient descent around flat or concave obstacles.

Several qualitative observations can be made concerning the differences between the behaviour of the two potential field based navigation approaches.

Firstly, standard gradient descent travels a short distance in its oscillation around the local minimum (a single descent step backwards and forwards), whereas gradient descent with LPCIRCLE chaining travels a significant distance (a single subgoal step, and therefore many descent steps) from the minimum before returning. Secondly, standard gradient descent travels backwards and forwards - that is, towards the obstacle and then away again. Gradient descent with LPCIRCLE chaining instead moves from side to side, parallel to the obstacle. Why is the behaviour of what seems like two similar approaches so very different?

Distance

In the case of gradient descent, the local gradient is able to vary over a short distance in the area closely surrounding the local minimum. Indeed, there is only a single point separating the change of direction of gradient - the point being the local minimum position itself.

In comparison, LPCIRCLE chaining is selecting a point at or around the local minimum as its target based on *potential value*, not gradient. Upon arriving there, it is unable to select its current position again, even though that it is the lowest point in the area, since it must select the lowest point that *lies at a distance*, on the circumference of the circle. In other words, *selecting the subgoal at a distance from the current point is forcing gradient descent*

to move away from its current position, even if that position is at a minimum on the original goal-obstacle potential surface.

Using LPCIRCLE as a subgoal selection technique for steering gradient descent is already forcing gradient descent to move well away from local minima, even if only temporarily.

Forwards-and-backwards vs. side-to-side

Gradient descent follows the local gradient, which alternately varies between being more strongly attracted by the goal than it is repulsed by the obstacles (the result of which, is a gradient directing the goal forward), then vice versa, with the obstacles pushing the agent away more than the goal attracts (the result being a gradient directing the agent backwards). In either case, the force is directed forwards or backwards along an axis containing the agent's current position and the goal. The obstacle boundary is perpendicular to this axis.

In contrast, LPCIRCLE travel takes place from side to side, parallel to the obstacle boundary. Near the obstacle boundary (Figure 6.6), the lowest points on the potential field at a distance from the minimum lie to either side of the minimum. Positions of low potential cannot lie 'forwards' (which is of high potential value, lying very near or even inside the obstacle) or 'backwards' (which is of high potential value, lying far from the goal).

Once the agent has travelled to a chosen LPCIRCLE subgoal, which lies away from the local minimum position, the agent is then free to select a new subgoal. The consequence of this is that the area close to the local minimum will now be selected, as it now lies at the requisite distance from the current position, and is almost certainly going to be at the lowest potential value of any point on the circle. It can be said that the standard local minimum problem has now been transformed into a two-step subgoal oscillation problem.

6.6.9 Overcoming small obstacles

Since the oscillation is not very long in nature, what happens when the flat or concave obstacles faced are of similar size to the radius of the scanning circle? Figure 6.6 demonstrates the behaviour of LPCIRCLE guided subgoal chaining in this instance, contrasted with gradient descent.

It can be observed that LPCIRCLE subgoal chaining already represents a small improve-

ment over gradient descent. By selecting subgoals using the potential value at a distance, rather than just following the gradient (which becomes useless when near a local minimum), the technique overcomes problems associated with very shallow local minimum basins associated with small non-convex obstacles.

6.6.10 Summary: Changing the local minimum problem to a subgoal oscillation problem

In summary, in cases where the agent reaches a local minimum, and must then select a subgoal at some distance, it is forced to set a subgoal to direct gradient descent to a point away from the local minimum, because of the fixed step size condition implicit in LPCIRCLE. Having travelled to this position, in the rare case of small non-convex obstacles, LPCIRCLE is now able to select subgoals that move past the obstacle. However, in most cases of non-convex obstacles, LPCIRCLE unfortunately now re-selects the local minimum as the next subgoal, as it is now the point of lowest potential, at the required distance from the agent's current position. This is a two-step subgoal oscillation problem.

How can this two-step subgoal oscillation problem be overcome? It can be overcome by exploiting the fact that, unlike gradient descent approaches, *the circle* is now available to work with and not just a single point. The exact manner in which this can be done will now be described, in the form of the first Forward Chaining heuristic - the FWDS1 subgoal selection heuristic.

6.7 Subgoal Selection: FWDS1

The previous section described how the LPCIRCLE subgoal selection heuristic allows a local minimum problem to be transformed into a new type of path oscillation problem.

Under LPCIRCLE-based subgoal chaining, the agent is able to begin to move away from the local minimum, due to the condition that subgoals must be selected at a distance from the agent's current position. Unfortunately, LPCIRCLE then immediately sets a subgoal back towards the position it has just come from. Returning along the path just travelled does not generate progress towards the goal. To prevent this two-step oscillation problem

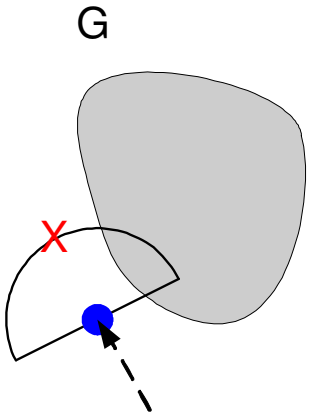


Figure 6.7: Selection of a point of lowest potential (X) relative to the goal (G) within the FWDS1 semicircle. The agent is marked as a blue dot, and the forwards direction is indicated with a dashed arrow.

and improve the quality of heuristic subgoal selection, path information can be exploited.

Since travel immediately backwards causes the oscillation problem, it seems sensible to exclude from selection those parts of the circle that lie *backwards* from the direction the agent has just travelled in. This simple refinement will be termed the FWDS1 heuristic.

6.7.1 What is meant by Forwards and Backwards?

Definition Forwards

Forwards will be defined as the direction going from the second last subgoal, to the last subgoal that has just been attempted or achieved.

The *Forwards semi-circle* will refer to those parts of the sampling circle that have a positive or zero component along the forwards direction. In 2-D, these parts of the circle all lie up to 90 degrees clockwise or counter-clockwise from the forwards direction. This is shown in Figure 6.7.

Directions that are *backwards* are those that have a negative component along the axis of the forwards direction. In 2-D, this means all parts of the circle that lie more than 90 degrees clockwise or counter-clockwise from the forwards direction.

6.7.2 Definition of the FWDS1 subgoal selection heuristic

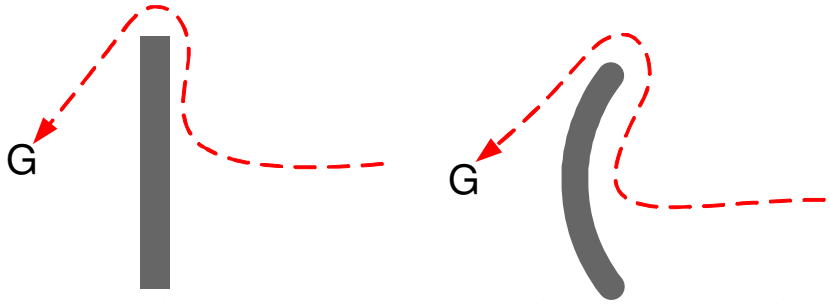
1. If the distance between the current position and the long term goal is less than distance D, set the subgoal to be the goal. Otherwise:
2. The vector from the second last attempted subgoal to the last attempted subgoal indicates the *forward direction*. If there are no previous subgoals, then initialise the *forward* direction to point directly at the goal.
3. Scan the *forward* semicircle at a distance D for the point of lowest potential and use it as the subgoal. To do this:
   (a) Calculate the value of the goal-obstacle $U(x, y)$ potential field, at a distance D from the current position. Do this for a selection of angles at most up to +/- 90 degrees from the *forward direction* (see Figure 6.7).
   (b) Pick the angle with a point at distance D which resulted in the lowest potential, and use the corresponding position as the subgoal for gradient descent.

6.7.3 FWDS1: Convex or small non-convex obstacles

The heuristic behaves exactly like LPCIRCLE when approaching a convex obstacle, since LPCIRCLE would not attempt to set a subgoal backwards towards the local minimum (and hence back along its previous path) in this circumstance, and therefore the new type of behaviour created by FWDS1 is not exhibited. The agent behaves as LPCIRCLE does in Figure 6.6.

6.7.4 FWDS1: Flat or shallow concave obstacles obstacles

Initially, the heuristic behaves exactly like LPCIRCLE, up until the moment at which LPCIRCLE would select a subgoal leading directly back to a local minimum on the goal-obstacle surface. The heuristic now rules out setting a subgoal back in the opposite direction to that in which the current subgoal was set and therefore rules out positions at or around the local minimum position. Instead, the position of lowest potential in the forwards semi-circle must be picked. This acts to drive progress forwards, out of the local minimum basin, allowing progress to resume towards the goal immediately. This is shown in Figure 6.8.



(a) Flat obstacle (b) Shallow concave obstacle

Figure 6.8: Interaction of FWDS1 subgoal chaining with large obstacle configurations.

6.7.5 Isn't this just wall-following?

It may be useful at this point to differentiate FWDS1 from a hybrid approach that identifies local minima or their associated obstacles and engages a wall-following subsystem to overcome them, also producing a form of sidewards travel. FWDS1 does not 'know' about walls, or obstacles, or their shapes. It only 'knows' about potential. Hence, the heuristic may be generalised across different application domains, obstacle shapes and map representations. Furthermore, the following observations can be made:

- Rather than having logically distinct phases of descent, minimum-recognition and wall-following, what is being proposed is an improved form of potential field descent that is not so vulnerable to local minima (as found in navigation) to begin with.
- There is no need to detect that a wall is present, or that a local minimum is present that might need to be dealt with, as there is in a hybrid technique.
- Unlike robotic navigation techniques that rely on touch sensors in order to implement their wall-following behaviour, FWDS1 achieves minimum escape without the need to actually touch the wall and thus does not require grazing collisions.
- The technique will later be generalised into 3-D and n-D in a way that wall-following, an essentially 2-D mechanism, cannot.

6.7.6 Problem: What happens if there is *no* suitable point in the forwards direction?

As the curvature of the obstacle increases, it may be that the agent is unable to select a forwards direction because the potential is infinite at all positions that are scanned by the heuristic. There are at several ways this can be addressed, for example:

1. The radius D of the circle could be temporarily reduced in size until a forwards direction can be selected.
2. A subgoal could be placed at random in the forwards semicircle to generate some movement to help escape the problematic position.
3. The forwards scanning range could be expanded from 180 degrees to a larger value, until a subgoal position is found.

In practice, this problem should be rare[5], because positions that are so close to an obstacle surface that no subgoal position can be selected will be of high potential due to their proximity to the obstacle. It is therefore unlikely that FWDS1 would direct the agent to such positions.

6.7.7 Problem: Moderately concave obstacles

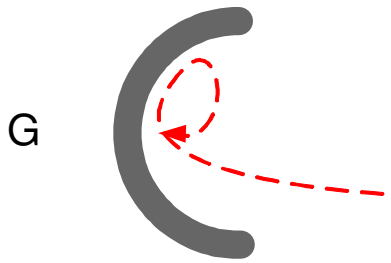


Figure 6.9: Interaction of FWDS1 subgoal chaining with an obstacle configuration of moderate total curvature.

In this case (shown in Figure 6.9), the curvature of the obstacle configuration gradually forces the agent's path to turn round over the space of several iterations of the subgoal selection heuristic, until the forwards direction is such that positions nearby the local

[5]Over 1300 experiments were carried out for Chapter 7, and no occurrences of this problem were found.

minimum position are again included in the forwards semi-circle. The agent will then select such points of lowest potential that lead back towards the local minimum position, and the agent will therefore be trapped in a multi-step oscillation problem and hence will fail to overcome the obstacle configuration.

In the event that the obstacle configuration is symmetric, the agent's forwards direction as it approaches the minimum may cause it to repeat a similar loop on the opposite side of the obstacle configuration. Alternately, the agent may simply retread its previous steps, exhibiting a looping behaviour with dynamical properties reminiscent of a limit cycle.

6.7.8 Overcoming moderate concavity through varying the forwards range

In the definition above, all the positions at a distance D within +/- 90 degrees of the forwards vector may be scanned as candidate points by the heuristic. It may seem possible to reduce the chance of turning towards the local minimum by reducing this range so that the agent does not turn back towards the minimum so quickly. Unfortunately, this technique does not work very well. The agent makes slow progress in escaping obstacle configurations; paths generated become meandering and thus less efficient and natural; some increased level of curvature will still be able to trap the agent; and the agent may later be unable to navigate towards the goal, due to the limited choice of direction available.

6.7.9 Summary: Changing the 2-step oscillation problem to a multi-step oscillation problem

LPCIRCLE fails to overcome local minima associated with non-convex obstacles because it allows the agent to move *backwards*.

By means of a simple variation on LPCIRCLE, the novel FWDS1 subgoal selection heuristic overcomes the local minimum problem for potential field navigation, in cases that cannot be overcome by either standard continuous or discrete gradient descent, by forcing subgoal selection to always take place in a forwards direction. This alone represents a significant improvement over existing potential fields techniques, permitting effective,

efficient and natural-looking directed[6] escape from standard local minimum situations involving flat obstacle configurations or those of mild concave curvature.

Problems involving concave obstacle configurations of moderate total curvature cannot be easily overcome by this technique, as the forwards direction becomes gradually rotated back by the attraction of the local minimum, resulting in the forwards direction pointing back at the local minimum, as the agent places successive subgoals. It may be possible to overcome some of these problems by reducing the 'forward range' of the agent, but this results in behaviour that is unsatisfactory in several ways.

Further refinement of the heuristic will be required to overcome the oscillation problems associated with concave obstacle configurations of moderate curvature successfully and elegantly. These refinements will now be presented, in the form of the FWDS2 heuristic.

6.8 Subgoal Selection: FWDS2

The previous section described a technique which fundamentally improves upon gradient descent techniques and allows the local minimum problem to be overcome in navigation for a number of common problems. However, it was shown that concave obstacle configurations of moderate curvature and their associated local minimum problems on the original goal-obstacle surface present a subgoal selection oscillation problem that the agent is unable to escape.

It can be observed that the reason FWDS1 is failing is because, despite de-selecting all backwards positions in order to prevent the agent from immediately travelling back upon its recent path, by selecting points of lowest potential the agent is being constantly drawn towards the local minimum position that exists on the goal-obstacle potential field.

It is therefore necessary to alter the goal-obstacle field in some way, so that this local minimum does not have this effect. Consider the possibility of replacing the goal with an alternative temporary goal. This temporary goal position would be selected so as to promote continued movement in the forward direction, but also so that its direction rapidly tends towards the position of the actual goal of navigation. A candidate mechanism will therefore be to place a temporary goal initially in the current forward direction, and bias it towards the goal by rotating it round towards the goal up to some limit. The agent then

[6]That is, non-random.

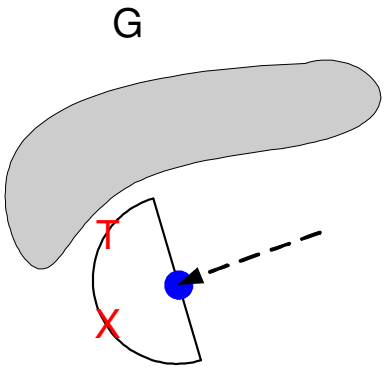


Figure 6.10: Selection of a point of lowest potential (X) using FWDS2 with the temporary goal at position (T).

uses the resulting position of the temporary goal to generate a 'temporary goal + obstacle' potential field for FWDS1 subgoal selection to take place on, as before.

The position of such a temporary goal will be called a goal-biased forward position. The field that the agent samples potential from will be based on the contribution of only the temporary goal and the obstacle function, with no contribution being made by the original goal to the potential function.

6.8.1 Definition of the FWDS2 subgoal selection heuristic

1. If the distance between the current position and the long term goal is less than distance D, set the subgoal to be the goal. Otherwise:

2. The vector from the second last attempted subgoal to the last attempted subgoal indicates the *forward direction*. If there are no previous subgoals, then initialise the *forward* direction to point directly at the goal.

3. The goal direction is defined as being the vector that points towards the long term goal from the current position.

4. A temporary goal G_1 is placed initially at a distance D in the current *forward* direction. This temporary goal is then biased towards the original goal by being swung round towards the original goal up to a limit of B degrees either clockwise or anti-clockwise. The direction in which the temporary goal is swung should be the direction which moves it towards the original goal direction most quickly. The range B limits the extent to which the original goal can bias movement away from

the *forward* direction[7]. If the original goal direction lies within the bias range of the forwards direction, the direction chosen should be that of the original goal itself.

5. Carry out FWDS1 to select a subgoal position, using a potential field constructed from the temporary goal and the obstacles, rather than the original goal-obstacle field. This is shown in Figure 6.10.

6.8.2 FWDS2: Convex, flat or shallow concave obstacles

The technique behaves like FWDS1 in these scenarios, successfully overcoming the local minima involved in the flat and shallow concave obstacle configuration scenarios.

6.8.3 FWDS2: Moderately concave obstacles

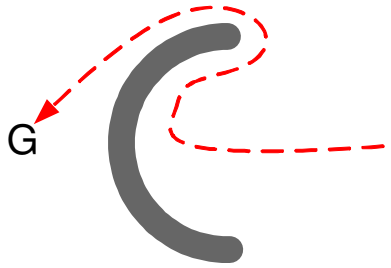


Figure 6.11: Interaction of FWDS2 subgoal chaining with an obstacle configuration of moderate total curvature.

Here (Figure 6.11), the technique succeeds in overcoming the obstacle configuration without becoming deflected into an oscillatory multi-step cycle in free-space. The technique also successfully resumes navigation towards the goal upon overcoming the configuration.

6.8.4 Problem: Highly concave obstacles

Consider the case (Figure 6.12) where an agent is faced with what is for gradient descent based approaches a certainly insurmountable obstacle configuration. Here, even a random

[7] A value of 75 degrees is derived for B in Chapter 7.

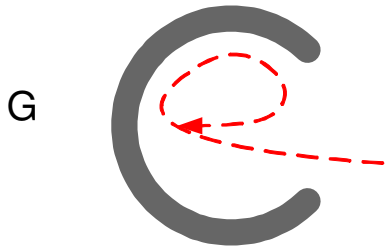


Figure 6.12: Interaction of FWDS2 subgoal chaining with an obstacle configuration of high total curvature.

path [66] would have difficulty escaping the configuration - directed travel is what is required to avoid re-entering the local minimum area repeatedly.

When FWDS2 subgoal chaining is applied to this problem, the result is unsatisfactory, as the agent becomes trapped. Analysis of this behaviour and a solution to this problem follows in the next section, in the form of the FORWARDS subgoal selection heuristic.

6.9 Subgoal Selection: FORWARDS

The FWDS2 heuristic is unsuccessful when faced with obstacles of high total curvature, as it eventually turns back towards its earlier path, with the consequence that travel enters a long multi-step oscillatory cycle of subgoal selection that fails to make any long-term progress. How can this problem be overcome?

By analysing the conditions at the point when the agent starts returning to its old path, it can be observed that the goal-bias procedure itself now leads the agent towards the dangerous area of free-space in the center of the concave obstacle configuration.

This occurs when the angle between the goal and the forwards direction exceeds +/-180 degrees. At this point, the effect of goal-bias pulls the agent back on its earlier path, and into a long multi-step oscillatory cycle, instead of allowing the agent to escape the minimum.

This is because there is no way to tell from the goal and forwards directions whether the agent has twisted through 185 degrees clockwise (and is therefore risking entering a multi-step cycle by continuing FWDS2 behaviour), or whether the agent has turned through 175 degrees anti-clockwise, which would allow FWDS2 behaviour to continue to safely operate. It is therefore necessary to introduce a state variable to keep track of the longer term behaviour of the agent and distinguish which of these situations has arisen.

It can be observed that when the agent navigates in the presence of an obstacle configuration of high total curvature, the agent's path is increasingly turned back on itself, until goal-bias eventually begins to fail. The agent can track this cumulative angle through which its path is turned from the initial goal-direction vector, by incrementally totalling the angle between the last two positions at the time of subgoal selection. This accumulated angle is referred to here as *path-twist*, as it represents the amount by which the obstacle has twisted the agent's path away from its initial direction.

There is a further complication. As the agent navigates, the obstacle configuration may be complex and may allow some circumnavigation of the goal. Tracking the extent to which the path has twisted from its original direction is not ideal. Instead, the path twist should be calculated relative to the extent to which the agent has circumnavigated the goal. Consider an agent accumulating a path twist of 180 degrees in the process of circumnavigating the goal by 180 degrees - it is now facing the correct direction to move towards the goal. It is therefore necessary to calculate the extent to which the agent has circumnavigated the goal in a separate 'goal twist' variable, to be subtracted from the path twist to create a 'relative path twist'. This can then be used to decide whether the agent's path is in danger of twisting back on itself and thus causing goal-bias to fail.

In the event that the relative path twist exceeds 180 degrees, an alternative temporary goal selection mechanism can be engaged to correct the path so that FWDS2 will be able to guide the agent successfully. Since the path is twisting back on itself, the correct strategy is to try and untwist the path by setting an appropriate temporary goal in the opposite direction. Here, this is done by placing the temporary goals at the edge of the forwards range, in the opposite direction to that in which the agent has become twisted. To untwist a path that is over-twisted in the clockwise direction, the agent should be untwisted anti-clockwise - or vice versa.

The result of this additional mechanism is that the agent now untwists its path before re-engaging FWDS2 - taking into account circumnavigation of the goal - and thereby allows the agent to successfully navigate towards the goal even in the presence of obstacle configurations of high total curvature.

6.9.1 Definition of the FORWARDS subgoal selection heuristic

1. If the distance between the current position and the long term goal is less than distance D, set the subgoal to be the goal. Otherwise:

2. The vector from the second last attempted subgoal to the last attempted subgoal indicates the *forward direction*. If there are no previous subgoals, then initialise the *forward* direction to point directly at the goal.

3. The goal direction is defined as being the vector that points towards the long term goal from the current position.

4. Calculate the angle between the vector from the second last position to the last position, and the vector from the last position to the current position. Add this angle to the cumulative *path twist* variable.

5. Calculate the angle of rotation about the goal by finding the angle between the vectors (goal $\rightarrow$ last position) and (goal $\rightarrow$ current position). Add this angle to the cumulative *goal twist* angle rotated about the goal.

6. The cumulative *path twist* minus the cumulative *goal twist* is calculated and called the *relative path twist*.

7. If the *relative path twist* exceeds 180 degrees, then a temporary goal should be set at a distance D at the edge of the forwards range in the clockwise direction, to untwist. If the *relative path twist* is less than minus 180 degrees, the temporary goal should be placed at a distance D on the edge of the forwards range in the anti-clockwise direction, to untwist. FWDS1 subgoal selection should then be used, to select a subgoal position using a potential field constructed from this temporary goal and the obstacles rather than the original goal-obstacle field.

8. If the *relative path twist* lies in the range minus 180 to 180 degrees, the agent behaves as in FWDS2.

6.9.2 FORWARDS: Convex, flat, and shallow or moderately concave obstacles

FORWARDS behaves exactly as FWDS2 on each of these structures.

6.9.3 FORWARDS: Highly concave obstacles

Here (Figure 6.13), FORWARDS successfully navigates around the obstacle.

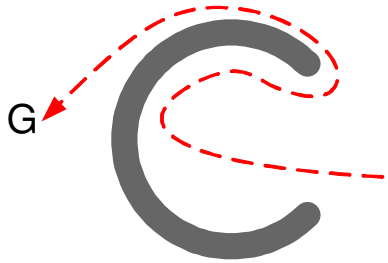


Figure 6.13: Interaction of FORWARDS subgoal chaining with an obstacle configuration of high total curvature.

6.9.4 FORWARDS: Goal circumnavigation

The idea of relative path twist allows FORWARDS subgoal chaining to continue to be able to overcome problems where goal circumnavigation is involved (Figure 6.14).

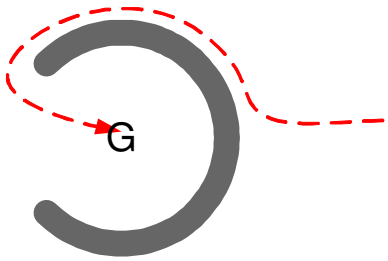


Figure 6.14: Interaction of FORWARDS subgoal chaining with a problem involving goal circumnavigation.

FORWARDS is the final version of the 'Forward Chaining' subgoal chaining heuristic to be developed in 2-D. Source code listings from an implementation of FORWARDS are provided in Appendix G for the purpose of assisting future implementations of Forward Chaining.

6.10 Chapter Summary

This chapter has charted the development of the *Forward Chaining* family of heuristic potential field methods for navigation, which are based on the ideas of *Subgoal Chaining* and *Forward Motion*.

Forward Chaining represents an entirely new branch of purely potential field based navigation, in which the descent technique uses historical path information, and knowledge of the characteristics of the potential field, in order to overcome the local minimum problem for navigation without recourse to hybrid navigation approaches or random behaviour. The subgoal selection heuristic itself is based upon what is believed to be a new form of gradient descent, and is incrementally augmented to allow more challenging local minima conditions in navigation to be overcome.

Some questions have been left deliberately unanswered in this section. These include:

- How do the Forward Chaining heuristics measure up against the criteria in Section 5.6.2?
- Can the Forward Chaining heuristics overcome the other problematic surface features for gradient descent besides local minimum problems?
- Do the Forward Chaining heuristics suffer the weaknesses pointed out by Koren and Borenstein in their 1991 paper [134]?
- Are there any potential field or obstacle configuration problems that the final FORWARDS heuristic is unable to overcome? How might they be overcome?
- Can Forward Chaining be made to operate in 3-D navigation environments, or even n-D?
- How might the technique be made to work with moving obstacles, or non-holonomic constraints?
- Can the technique be re-applied as a general potential fields method outside of the field of agent navigation?

These issues will now be addressed in the following three chapters of this thesis.

Chapter 7

Experimentation and Results

Guide to Chapter 7

Having laid out the novel Forward Chaining approach to abstract agent navigation, this chapter now analyses the performance of the heuristic under a number of experimental conditions.

These experimental results are provided here only as 'proof of concept' evidence of the behaviour of Forward Chaining. The results show the technique operating in a limited range of experimental conditions.

This chapter is not intended to be a rigorous, extensive or exhaustive empirical study of the properties of the technique.

7.1 Chapter Overview

This chapter is split into nine sections.

- Experimental setup. This describes the experimental setup used to gather results throughout this chapter.
- Discussion of parameters. This section explains how the parameters to the techniques were chosen and explains their effects on the quality of path generated.
- Basic navigation problems. This demonstrates the behaviour of the developed technique on archetypal navigation problems under a variety of starting conditions. The issue of goal circumnavigation is also considered.
- Other surface features. This section looks at the response of the developed techniques to problematic potential surface features other than local minima.
- Reflection upon classic criticisms. This section looks at the problems for potential field methods highlighted by Koren and Borenstein and the consequences for Forward Chaining.
- Pathological problems. This section suggests pathological cases of navigation that have been deliberately constructed to cause the heuristic to fail.
- Typical navigation. This section demonstrates the behaviour of the algorithm in a non-trivial environment (128 obstacles).
- Computational expense and optimisation. This section discusses the factors affecting the asymptotic cost of Forward Chaining, and concludes with notes on the expected benefits of various forms of optimisation.
- Characterisation of the technique. This is a description of the technique using the terminology given in Chapter 2.

Experimental output corresponding to each of the experiments in this chapter can be found on the DVD placed at the back of this document.

7.2 Experimental setup

7.2.1 Platform

The computer used throughout experimentation was equipped with :

- A single Intel Pentium 4 2.66Ghz processor
- 512MB of DDR266 memory.

The software platform used for the implementation was:

- Microsoft Windows XP (kernel 5.1 build 2600)
- Maplesoft Maple 8.0 (build ID 110847)

Where computational cost experiments were carried out in Linux, the software configuration consisted of:

- Redhat Linux 9.0 (Shrike)
- Linux kernel 2.4.20-19.9,
- gcc compiler, version 3.2.2 build (2003/0222)

7.2.2 Notes on the platform

Maple is a high level, interpreted language, and this has *very* serious consequences for the real world absolute computational costs of this implementation. This is discussed in Section 7.9.4. Maple was used in spite of the costs involved as it facilitated visualisation of the potential surfaces involved (which assisted development of the approach), and reduced the implementation effort necessary to allow animations and diagrams to be generated to demonstrate results.

The experimental platform used is representative of a typical mid-range consumer system as of September 2005.

7.2.3 Experimental technique

The details of the technique used in each experiment can be found in the report for each of the sets of results. Generally, where the results being generated are based upon the measurement of time, five runs were carried out to ensure that results were consistent. Where 'proof of concept' paths were being generated, 25 combinations of start and goal positions were used to ensure a variety of angles of incidence to the problematic obstacle configurations.

7.2.4 Potential field configuration

The implementation has been tested with two basic types of potential field model. The first obstacle-to-function mapping was designed by the author and can be found in [71], and the second mapping used is the standard FIRAS function detailed in Khatib's work [123, 127]. Khatib's functions were used throughout these experiments to allow comparisons to be made with other research.

The current implementation provides potential function mappings for two types of basic obstacle shapes at present - rectangular obstacles and circular obstacles. More complex obstacle configurations can be built from these, or alternatively, potential field functions corresponding to other shapes of obstacle could be implemented as necessary.

The implementation is therefore general in that it does not depend on a particular type of mapping between the abstract map and the potential field, nor does it rely on any particular underlying obstacle type in the environment or map upon which the potential function is based.

A conical goal function [138] was used so that the effect of the goal was based only on the direction of the goal and not on its distance (i.e. to bound the magnitude). The obstacle's repulsion constant η was set to a value of 3.

The circular obstacles used had a diameter of 6 units. All of the dimensions of the rectangular obstacles were 6 units or greater. The choice of obstacle size has no special significance. Obstacles were configured to have a falloff distance of 5 units - in other words, at a distance of 5 units from the obstacle, the obstacle ceased to have any effect on the potential or gradient functions of the field.

7.2.5 Heuristic configuration

Unless otherwise specified, the scanning circle was sampled every 10 degrees (the reasoning behind this choice is detailed in Sections 7.3.1 and 7.9.1).

The subgoal step size chosen was 3 units - i.e. half the thickness of the smallest obstacles[1].

The gradient descent step size was set to 0.2 units, as this setting produced a continuous trajectory onscreen when plotted in Maple and is a little over an order of magnitude smaller than the subgoal size (see Section 6.6.5).

7.3 Parameters affecting success

7.3.1 Experiment 1: Resolution of the sampling circle

Purpose

The purpose of this experiment is to give an indication of the effect of the resolution of the sampling circle upon the quality of paths generated.

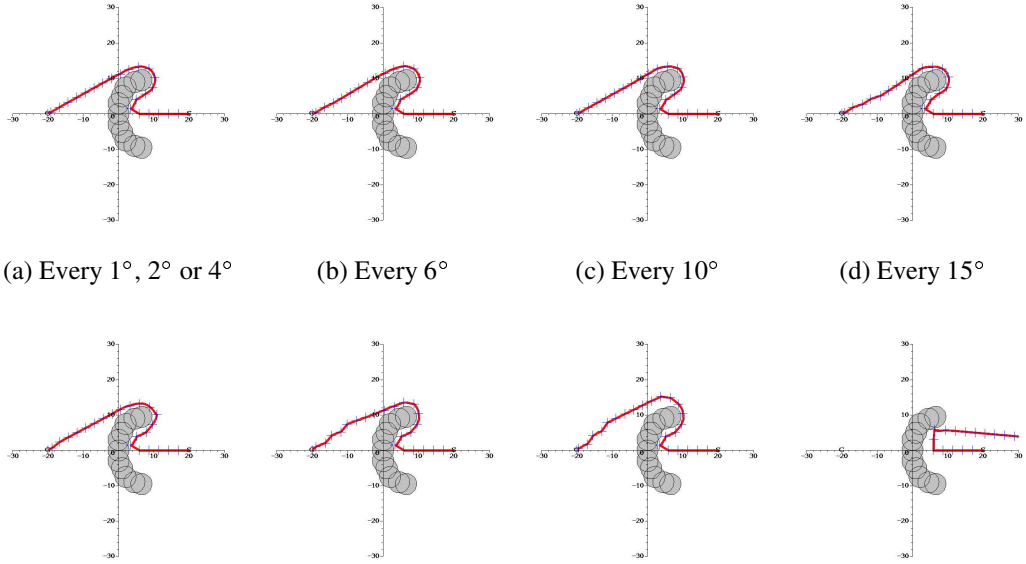
Technique

Forward Chaining with the FORWARDS subgoal selection heuristic is used to navigate around concave obstacles of varying curvature, with a sampling circle set to resolutions of 1, 2, 4, 6, 10, 15, 20, 30, 45, 60 and 90 degrees. The paths are compared qualitatively.

Prediction

It is expected that similarly smooth and successful paths will result from using a sampling circle of very high resolution, and that as the number of samples taken is reduced, the paths will become increasingly unsmooth and unnatural in nature.

[1] See Section 6.7.6.



(a) Every 1°, 2° or 4° (b) Every 6° (c) Every 10° (d) Every 15°

(e) Every 20° (f) Every 30° (g) Every 60° (h) Every 90°

Figure 7.1: Quality of path varying with resolution of scanning circle.

Results

Figure 7.1 shows the behaviour of the technique on an obstacle of moderate curvature at different scanning circle resolutions. The DVD at the back of the thesis contains examples of the behaviour on other obstacle shapes that are consistent with the results shown in Figure 7.1.

Conclusion

Qualitatively, it can be observed that the path is not noticeably improved by use of sample spacing at resolutions higher than one sample every 10 degrees. Sample spacings greater than one sample every 10 degrees begin to result in gradually more jagged and unsmooth paths, and the heuristic fails to operate at all with a sample spacing of 90 degrees. A sample spacing of 10 degrees is therefore an ideal choice from a qualitative perspective. This parameter setting will be used throughout the rest of the experiments in this chapter.

7.3.2 Experiment 2: The FWDS2 goal-bias parameter

Purpose

To give an indication of the extent to which the forwards direction should be biased towards the goal direction to allow correct operation of the FWDS2 heuristic (see Section 6.8).

Technique

Obstacles of low and moderate total curvature were designed, and FWDS2 was run with goal bias settings of 0, 15, 30, 45, 60, 75, 90, 105, 120 and 135 degrees to determine the effect on the rate of success and quality of path.

Prediction

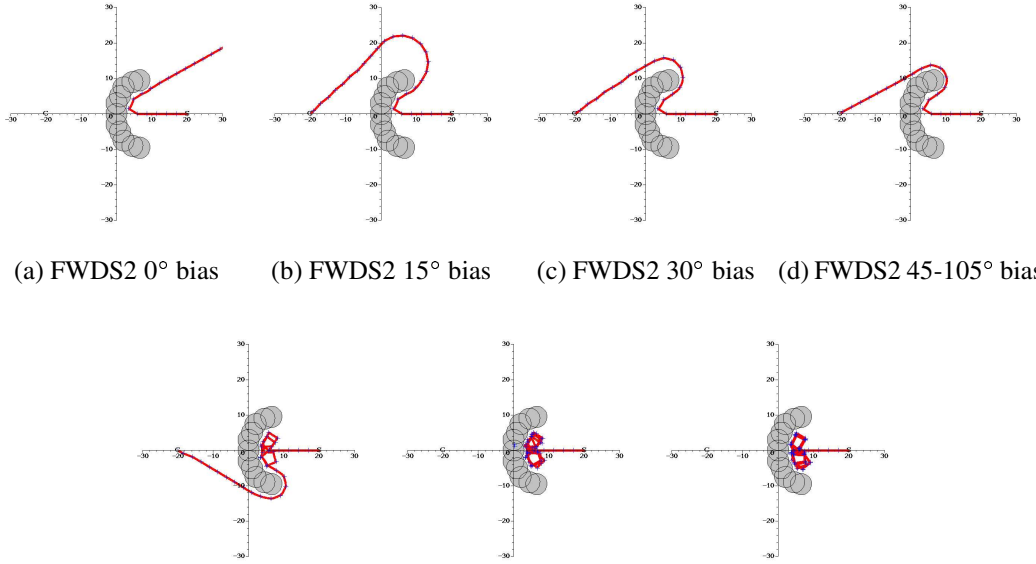
If the bias towards the goal is extremely low, the technique places the FWDS2 temporary goal effectively in the Forward direction repeatedly. The result would be that the agent would travel forwards, but with only minimal consideration for the actual goal of navigation. The agent will therefore 'bounce' from the obstacle surface, and only turn back towards the goal eventually after a large number of steps.

If the bias towards the goal is extremely high, then the agent will place the FWDS2 temporary goal in the same direction as the original goal of navigation with no regards for the Forward direction. The result of this will be that the FWDS2 heuristic behaves exactly like the FWDS1 heuristic.

At intermediate values of goal-bias, the agent should be able to travel out of the obstacle configuration successfully.

Results

Figure 7.2 shows the behaviour of the agent is as predicted. Zero goal-bias results in the agent 'bouncing' from the obstacle, small (15,30) values of goal-bias result in broad arcing paths, and high (120,135) values result in paths that are equivalent to FWDS1. For a wide



(a) FWDS2 0° bias (b) FWDS2 15° bias (c) FWDS2 30° bias (d) FWDS2 45-105° bias

(e) FWDS2 120° bias (f) FWDS2 135° bias (g) FWDS1

Figure 7.2: The effects of the goal-bias range parameter on FWDS2.

range of values between 45 and 105 degrees, the technique works correctly.

Conclusion

Values between 45 and 105 degrees are appropriate for use. In this research, a goal-bias value in the middle of the range where FWDS2 operates correctly was used (75 degrees).

7.4 Basic navigation problems

7.4.1 Experiment 3: Archetypal obstacle configurations

Purpose

This set of experiments is designed to indicate that the heuristics developed operate as has been suggested in Chapter 6, when used with convex obstacles, small obstacles, flat obstacles and concave obstacles of varying degrees of total curvature.

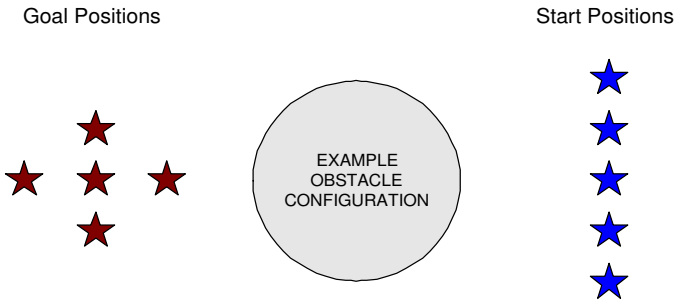
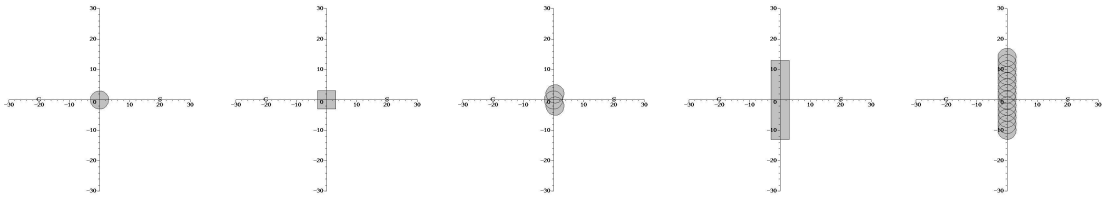
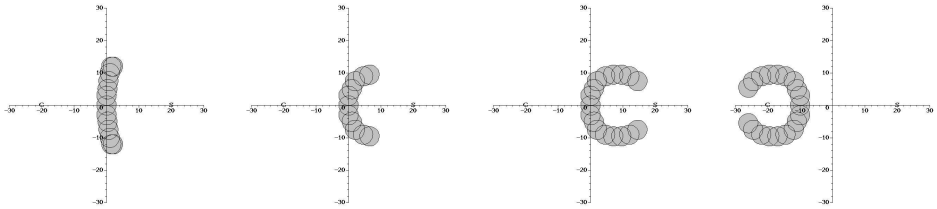


Figure 7.3: Layout of experimental configurations for Experiment 3.



(a) Convex (b) Small/Flat (c) Small/Concave (d) Flat (e) Flat(2)



(f) Shallow Concave (g) Moderate Concave (h) Highly Concave (i) Circumnavigation

Figure 7.4: Diagrams of each of the archetypal obstacle configurations used.

Technique

Forward Chaining was tested using each of the subgoal selection heuristics (LPCIRCLE, FWDS1, FWDS2, FORWARDS) on a variety of navigation problems containing a single archetypal obstacle configuration lying between the start and the goal. Five starting positions were chosen to ensure a range of entry angles intersecting the obstacle, and five goal positions were selected to demonstrate navigation to a goal position above, below or in the middle of the obstacle configuration, as well as navigation to a goal that is either more distant or much closer to the obstacle. This produced a set of 25 unique cases of navigation involving each archetypal class of obstacle. The start and goal positions are shown in Figure 7.3. The archetypal obstacle configurations are shown in Figure 7.4.

Obstacle Type	Grad. Desc.	LPCIRCLE	FWDS1	FWDS2	FORWARDS
Convex	88%	100%	100%	100%	100%
Small flat	0%	100%	100%	100%	100%
Small concave	0%	100%	100%	100%	100%
Flat (rectangle)	0%	0%	100%	100%	100%
Flat (circular)	0%	0%	100%	100%	100%
Shallow concave	0%	0%	100%	100%	100%
Moderate concave	0%	0%	0%	100%	100%
Highly concave	0%	0%	0%	0%	100%
Circumnavigation	0%	0%	100%	100%	100%

Table 7.1: Success rates for each heuristic in Experiment 3.

Additionally, it is worth noting that two different types of representation of a 'large flat' obstacle are tested - an approximation built from a number of circular obstacles, and a truly flat rectangular obstacle. This is intended to reproduce both the approximate representation of flat obstacles that might be found in real-world use of the heuristic, and the accurate representation that might be found in virtual world use.

Standard gradient descent was also tested under the same conditions to allow comparisons to be made against it.

Prediction

Gradient descent should only succeed on the convex obstacle. LPCIRCLE should succeed on the convex obstacle and the small obstacles. FWDS1 should succeed in all cases except the obstacles of moderate and high total curvature. FWDS2 should succeed in all cases except the obstacle of high total curvature. FORWARDS should succeed on all archetypes, in all cases.

Results

Table 7.1 shows the percentage of cases in which navigation succeeded using each heuristic on each archetypal obstacle configuration. The complete set of experiments and graphical output corresponding to the agents trajectory can be found on the DVD at the back of this thesis.

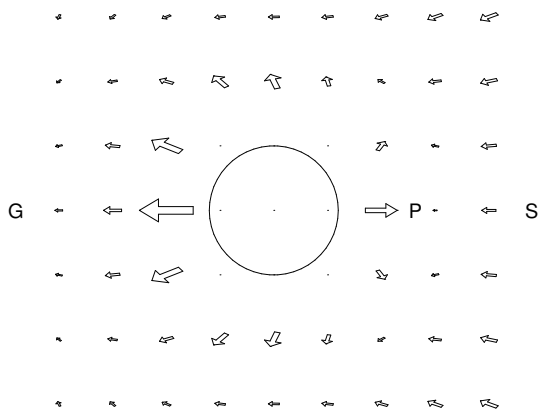


Figure 7.5: Gradient field showing a saddle point P between the start S and the goal G.

Conclusion

The techniques operate as expected (see Chapter 6), with the exception of gradient descent, which fails to navigate in 3 cases with convex obstacles. Specifically, failure occurs when the start position, the goal position and the center of the obstacle all lie on a straight line. These cases are discussed later in Section 7.5.1, in the context of saddle point problems.

7.5 Other surface features

7.5.1 Saddle Points / 'Saddle Point Minima'

Generally, saddle points are not considered to be problematic for potential field based navigation, except in the case when navigation is initialised with a starting position that corresponds to a saddle point - with the consequence that at the initial position there is no gradient value available to set gradient descent on its way.

In Experiment 3, gradient descent can be seen to fail when navigating in the presence of a circular and hence entirely convex obstacle. Gradient descent is not classically expected to fail on a simple convex obstacle problem, so this phenomenon requires explanation. Figure 7.5 shows the gradient across the potential field.

A symmetrical convex obstacle will have a saddle point directly in front of it, if a goal is placed behind the obstacle on the line of symmetry. If navigation starts from a position lying in this line of symmetry, the gradient on either side of the saddle point will always

direct the agent towards the saddle point. Effectively, a 2-D local minimum problem is set up in the 3-D space. This is classified by Bell and Livesey as a *saddle minimum problem* in [69].

Gradient descent can therefore be said to be vulnerable to saddle points in two ways.

1. When descent is initialised at a saddle point and there is no gradient available to direct descent away from the starting position.
2. When descent is initialised in a position within the basin of a saddle minimum problem. The gradient cannot direct the agent out of the subspace effectively containing a local minimum problem.

Forward Chaining's subgoal selection heuristics are not vulnerable to either of these problems. Both problem situations are avoided because Forward Chaining does not rely on the gradient to steer navigation, but rather, samples potential at positions at a fixed distance in all directions from the current position in selecting subgoals. Forward Chaining approaches therefore succeed 100% of the time in situations containing saddle points (as shown in Table 7.1). Gradient descent fails in all cases where the start position lies along a line of symmetry containing the goal and the obstacle.

7.5.2 Plateaux

Plateaux do not exist in the modelling of potential fields used here, since the goal acts as an attractor over the whole surface. Furthermore, as a fixed step size technique, Forward Chaining will be unaffected by plateaux if it is applied to potential field problems that feature plateaux.

7.5.3 Experiment 4: Ravines

Purpose

To determine if Forward Chaining is affected by the Ravine Problem for potential fields.

Technique

In the experimentation so far, no behaviour attributable to ravine problems has been found. To verify that the effect is present, an extreme scenario was constructed.

The agent was made to navigate in an environment consisting of two parallel walls as shown in Figure 7.6. The strength of the obstacle's repulsion of the agent was increased by a factor of 1000 to increase the severity of the ravine problem. The gradient descent step size was increased tenfold to make any oscillation by the agent more obvious to visual inspection.

The agent was initialised at a position displaced from the center of the ravine problem (20,1) and had to navigate to (-20,0) which lay directly in the middle of the ravine. Both gradient descent and Forward Chaining were tested for susceptibility to this problem.

Prediction

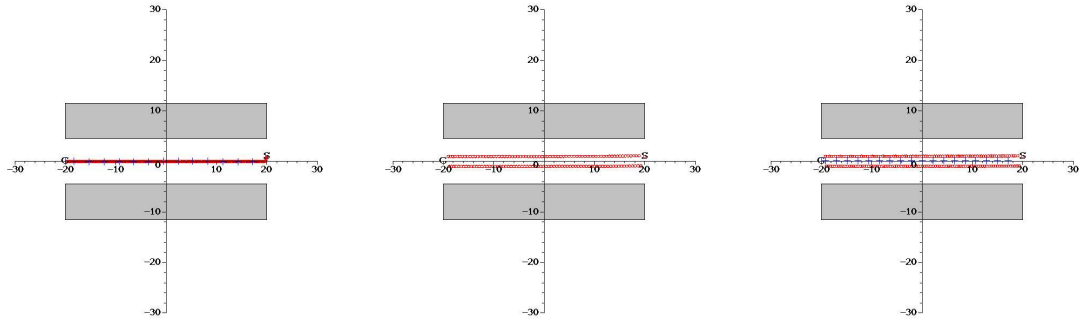
It was expected that standard gradient descent would oscillate around the floor of the ravine while making some progress towards the goal, but that Forward Chaining will rapidly 'lock on' to the center of the ravine during selection of positions of lowest potential and then travel directly to the goal along the ravine floor.

Results

Figure 7.6(a) shows the behaviour of Forward Chaining with a normal gradient descent configuration. Figure 7.6(b) shows the behaviour of gradient descent with a large step size. Figure 7.6(c) shows the behaviour of Forward Chaining, where the interpolating gradient descent is set to have a large step size.

Conclusion

- In normal practice, the faint oscillatory effect caused by the ravine problem was so small that it went unnoticed.
- When standard gradient descent was used, the agent oscillated noticeably as expected.



(a) FWDS, normal GD step size (b) GD, large step size (c) FWDS, large GD step size

Figure 7.6: The Ravine Problem.

- When Forward Chaining was used, subgoals were selected in the manner predicted, but the interpolating gradient descent *still oscillated*. In hindsight, this should have been obvious, since if gradient descent oscillates when a goal is set in the middle of the ravine, it will of course also oscillate with a subgoal set in the middle of the ravine.

- LPCIRCLE and FWDS based subgoal selection can be seen to avoid oscillation, from the series of subgoals that are set during navigation. The oscillatory effect exhibited during gradient descent based subgoal interpolation can therefore be overcome by using LPCIRCLE or FWDS1 as the interpolating descent heuristic in place of gradient descent[2]. This way, both the subgoal selection technique and the interpolating descent mechanism will track the valley floor of the goal-obstacle and subgoal-obstacle field respectively, resulting in movement directly towards the goal without oscillation. This is only necessary though in navigational scenarios where the presence of any oscillation whatsoever will cause the agent to behave in a way that is unacceptable.

7.5.4 Summary

As well as being unaffected by the local minimum problem found in potential field based navigation, Forward Chaining is generally unaffected by any of the other classic surface problems that exist on potential fields.

[2]This will incur an increased computational cost relative to gradient descent, but is certainly affordable.

Unfortunately, when used with gradient descent as the interpolating mechanism, the technique is still affected by the ravine problem if a large step size is used in cases with strong obstacle repulsion. This can be overcome through the use of LPCIRCLE or FWDS1 as the subgoal interpolation mechanism.

7.6 Reflection upon classic criticism

Koren and Borenstein's 1991 paper "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation" [134] came at a time when research interest in potential field methods began to wane, and it is an important paper to re-consider in light of this research for two reasons, even though the research in this thesis is intended for agent navigation more generally, rather than just robotics.

Firstly, no significant new non-hybrid potential field techniques have been published subsequent to their paper, so there has been little cause to revisit its claims. Secondly, it is the only paper that claims there are particular problems inherent to *all* potential field based methods for navigation.

7.6.1 Koren and Borenstein's model

The model used by Koren and Borenstein in their paper is not valid for Forward Chaining. They assume that an agent using a potential field method will navigate according to the gradient, and that the agent's target during gradient descent will lie on the far side of a long, flat obstacle.

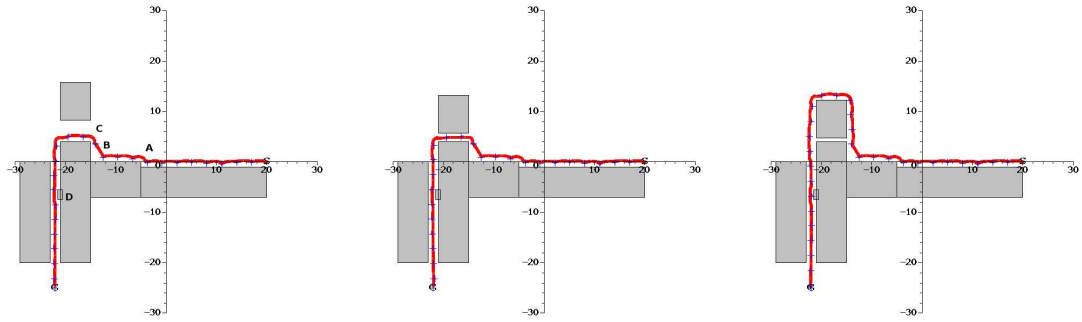
Neither of these conditions holds true for Forward Chaining. Navigation is primarily steered by the potential function, and when subgoals are placed to steer interpolation, the restriction on subgoal step size ensures that the agent's target during gradient descent is never on the far side of an obstacle. The model on which their reasoning is based is therefore not appropriate to describe Forward Chaining.

7.6.2 Experiment 5: Koren and Borenstein's problems

Koren and Borenstein claim that amongst other problems with their own technique,

> "we identified problems that are inherent to PFMs in general." [134]

The four problems the paper identifies will now be discussed in turn. Each problem scenario (A-D) is marked on Figure 7.7(a) for consideration.



(a) (b) (c)

Figure 7.7: Example environment containing Koren and Borenstein's 4 scenarios. Each scenario is marked (A-D) on subfigure (a).

A: 'Trap situations due to local minima'

"Perhaps the best-known and most often-cited problem with PFMS is the problem of local minima or trap-situations.... however, trap-situations can be resolved by heuristic or global recovery" [134]

The phrasing here seems to neglect the possibility that a qualitatively better form of potential field descent might exist that is simply less vulnerable to the local minimum problem by virtue of its means of field descent. The issue of local minima (and the way in which Forward Chaining is constructed so as to be unaffected by them) has been dealt with by the previous experiments on archetypal problematic obstacle configurations, and needs no further treatment here.

B: 'No passage between closely spaced obstacles'

The description of this problem indicates that when two obstacles are spaced closely together, with a gap between, it is possible for the summed repulsive force of the obstacles on either side of the gap to prevent the agent from being able to navigate into the gap. The example they give shows a robot having problems navigating through a wide gap, as well as a distant goal placed out of line with the gap.

Is Forward Chaining affected by this problem? Consider Figure 7.7. The area of influence of the obstacles in this example extends across the gap, and yet Forward Chaining is able to navigate through the gap in all cases except when the gap becomes extremely narrow indeed.

Why is Forward Chaining able to get through the gaps without turning off to the side as Koren and Borenstein suggest?

Primarily, this is because Forward Chaining selects subgoals lying in the gap region - so that when gradient descent is employed, the full attractive effect of the goal is directed towards making the agent move through the gap region, unlike the example in [134]. Furthermore, the agent is using a fixed step size during gradient descent, so a strongly attractive goal function can be used, which reduces the effect of the obstacles in preventing entry to the gap region.

Why does Forward Chaining have problems on the extremely narrow gap?

Firstly, it can be observed that a subgoal is not placed in the gap. This is because the agent is not navigating at the 'scale' of the detail in the environment. The narrowness of the gap is far smaller than the size of the subgoal step. Setting up the step size parameter to subgoal chaining so that it suits the scale of the environment, or establishing an adaptive regime to do this automatically, is part of this technique. Nonetheless, navigation still succeeds in Figure 7.7(b), despite the gap being narrower than the subgoal step size.

Secondly, there is a problem in that the repulsiveness of the obstacles is designed to prevent gradient descent from getting closer than a particular distance to the obstacles. It is therefore not unsurprising that gradient descent cannot navigate through a gap that would

require it to move closer to the obstacles than the potential functions have been designed to allow.

Finally, once the step size has been set so that subgoals stand a chance of being placed within the gap, it is possible to consider replacing gradient descent with (for example) the FWDS1 heuristic as the interpolating descent mechanism, which would force the agent to continue moving forwards and into the gap despite the raised potential encountered en route. Put simply, if the agent is forced to move forward while selecting points of low potential, then it will select points along the middle of the gap as these are the best choice it has under the constraint of moving forward.

C: 'Oscillations in the presence of obstacles' and D: 'Oscillations in narrow passages'

Koren and Borenstein suggest two cases where a small protrusion on an obstacle surface can perturb the agent in such a way that a serious oscillation is established in the behaviour of the agent. This oscillation persists and in some cases worsens as navigation continues. The perturbation has essentially set up an oscillation within a standard ravine problem. In the first case (C), one side of the ravine is formed by the attraction of the goal, and the other by the repulsion of the obstacle. In the second case (D), both sides of the ravine are formed by repulsion from obstacles (see Experiment 4 earlier in this chapter).

They observe that the oscillation is minor if navigation is taking place at low speeds, but that it becomes increasingly serious as the agent travels faster. The situation of the agent travelling faster is analogous to using a larger step size for gradient descent. This agrees with the findings in the Ravine problem experiments in this thesis - that gradient descent will oscillate around the center of the ravine if descent takes place too quickly.

As discussed earlier in this chapter, if the oscillation due to a ravine on the potential surface risks being a problem, it can be remedied by use of LPCIRCLE or FWDS1 as the interpolating descent mechanism in place of gradient descent. The experiments earlier clearly showed that LPCIRCLE based descent on the surface will rapidly move towards and track the center of the ravine - therefore rapidly dampening down oscillation and allowing smooth progress at high speed.

7.6.3 Oscillation in Forward Chaining?

Note that a very subtle oscillation can still be found in the paths generated by Forward Chaining when gradient descent is used (see Figure 7.7). This behaviour is unrelated to the problem identified by Koren and Borenstein, and instead relates to the fact that the subgoal used by gradient descent is now in quite close proximity to the agent.

In Figure 7.7, as the agent travels towards the subgoal, the agent's angle with respect to the subgoal may change rapidly as the agent gets close to the subgoal (because subgoals, unlike goals, are in close proximity to the agent). The summed force of subgoal and obstacle will change direction gradually as the agent gets closer to the subgoal. If the subgoal has been set very close to an obstacle wall, this may lead to a very shallow arc forming over the course of the agents movement between subgoals. In fact, a similar arcing effect should be faintly present in most potential fields methods - albeit even more shallow, since the goal is far away, and thus the arc of the curve it causes very shallow.

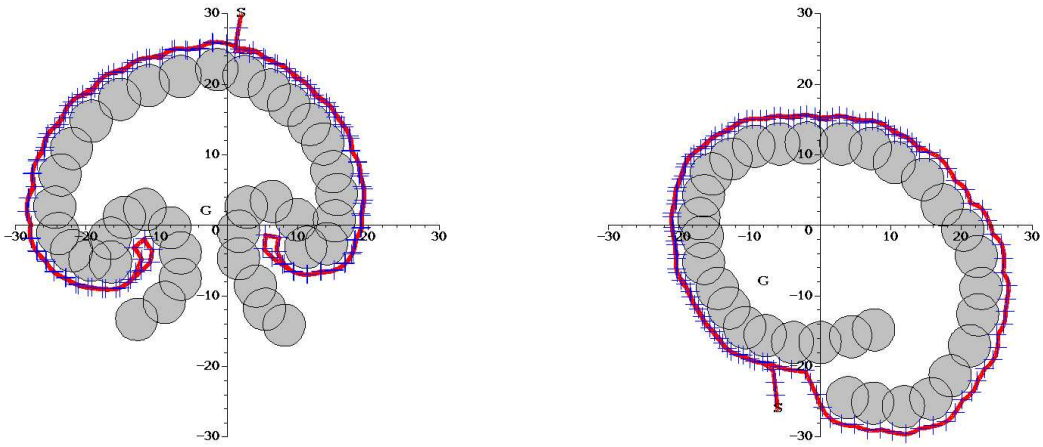
This minor problem may be remedied by use of a subgoal function that takes into account the agent's proximity to the subgoal, so as to minimise the gentle arcing effect. Alternatively, the problem can be alleviated by setting subgoals slightly 'out of reach' according to the number of steps permitted by the agent's interpolating gradient descent mechanism, so that the agent experiences less strong rates of change of angle with respect to the subgoals it is attempting to reach.

Conclusion

From a theoretical perspective, it is not obvious that the problems in [134] should affect the Forward Chaining approach. Figure 7.7 and the results shown earlier in Figure 7.6 indicate that from an experimental viewpoint they do not occur in this scenario. Koren and Borenstein's analysis and claim that their problems will affect all potential field methods is based upon the assumption that all potential field methods will use the gradient as the primary basis for directing navigation. As this does not entirely apply to Forward Chaining - which is nonetheless still a purely potential field based technique - and since Forward Chaining can be made to operate *entirely* without reference to a gradient function (via LPCIRCLE-based or FWDS1-based interpolatory descent), it seems that the claim that all potential field based approaches will be necessarily and inherently affected by the problems outlined is unwarranted.

7.7 Pathological problems

It has been possible to construct navigation problems where Forward Chaining is very likely to fail. These problems hinge on the deliberate introduction of large scale symmetry in situations where it is necessary for the agent to significantly or completely circumnavigate the goal. The situations - known as the 'Mushroom' and 'Spiral' problems - are contrived, and although diagrams are given below in Figure 7.8, a fuller discussion of these cases and possible heuristic adaptations of Forward Chaining that would overcome them has been left to Appendix E.



(a) The Mushroom Problem (b) The Spiral Problem

Figure 7.8: Pathological problems for Forward Chaining.

It does not seem worthwhile to complicate Forward Chaining in order to cope with these pathological cases and risk the introduction of further more complicated pathological cases in doing so. Forward Chaining is most certainly a *heuristic* technique. When using any heuristic technique, it will be possible to construct situations where the heuristic will fail, or will perform sub-optimally. What is important in heuristic design, is to ensure that in most cases encountered in normal use, the heuristic approach will behave competently.

7.8 Navigating in a larger environment

7.8.1 Experiment 6: 'Bigmap'

Purpose

To test the behaviour of the heuristic in a much larger environment involving many obstacles and moderately complex arrangements of obstacles, and a variety of starting positions and goals.

Technique

A total of 128 obstacles are placed on the map. A initial set of obstacles were placed randomly, and then additional obstacles were placed to ensure that some complicated obstacle structures existed in the map. Some rectangular obstacles were placed to demonstrate the ability of the technique to accommodate arbitrary types of basic obstacle shape simultaneously. No attempt was made to have obstacle layout conform to the archetypes described earlier in this chapter. The resulting environment might be interpreted as resembling a complex situation that might occur in a computer game - perhaps houses in a forest, or exhibits surrounded by people in a museum.

The obstacles used are half the size of those used earlier, so that more can fit within the same graphical display, and the Forward Chaining step size was scaled correspondingly to maintain the same ratio to the obstacle's thickness (50%) as in earlier experiments.

The agent was required to navigate from each of the edges and corners of the map, to positions lying opposite, so that navigation through the environment was required. For example, in Figure 7.9, the agent was required to navigate from position A to positions E,F,H and I; from position B to positions E,G,H and I, and so on. Navigation was carried out using Forward Chaining and also using gradient descent for comparison.

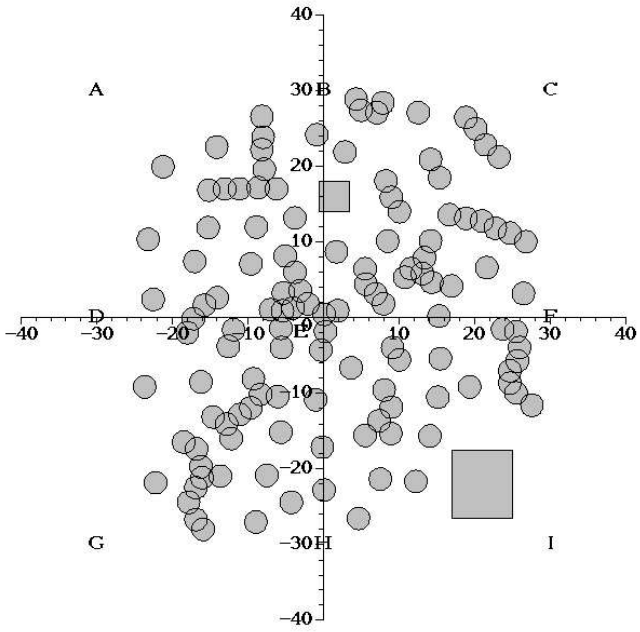


Figure 7.9: Overview of the 'Bigmap' navigation problem.

Prediction

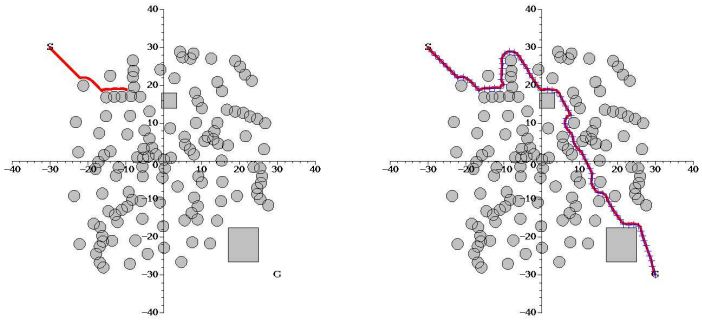
It was expected that gradient descent would fail on most cases and perhaps even in all cases, and that Forward Chaining would succeed on all cases.

Results

Gradient descent succeeded in 7 out of 32 cases.

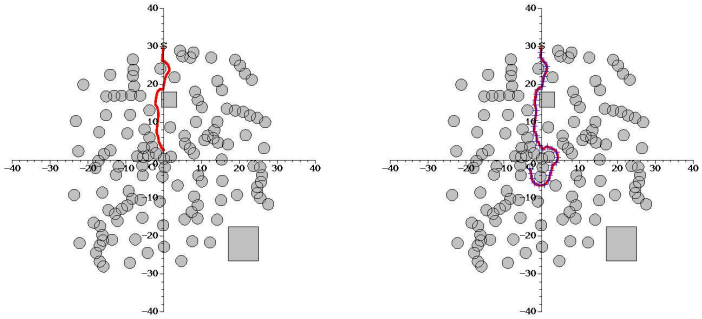
Forward Chaining succeeded in 32 out of 32 cases.

Graphical output corresponding to all of the test runs, and animations for some of the test runs can be found on the DVD at the back of the thesis. Printouts of some example runs have been provided at the end of the thesis in Appendix F. Examples are also given below in Figures 7.10 and 7.11.



(a) Gradient Descent (b) Forward Chaining

Figure 7.10: Example of navigation on Bigmap.



(a) Gradient Descent (b) Forward Chaining

Figure 7.11: Example of navigation on Bigmap.

Conclusion

Forward Chaining is considerably more successful than basic gradient descent, and as a heuristic approach it is very capable of coping with navigation scenarios of non-trivial complexity. This straightforward, online, continuous path-finder that succeeds in 100% of cases of moderate complexity is considerably more useful for practical application than a similar technique that succeeds in less than 25% of cases.

7.9 Computational expense and optimisation

A set of three experiments were conducted to determine how computational costs of the Forward Chaining heuristic (with the final FORWARDS subgoal selection heuristic) rose as aspects of the navigational problem were varied. The experiments have the same basic setup - a series of obstacles arranged in a straight line above and below the agent, so that the agent interacts with the obstacle's potential functions, but has a predictable and linear path shape. This allows measurements to be made with a known path length. Figure 7.12 shows the experimental arrangement.

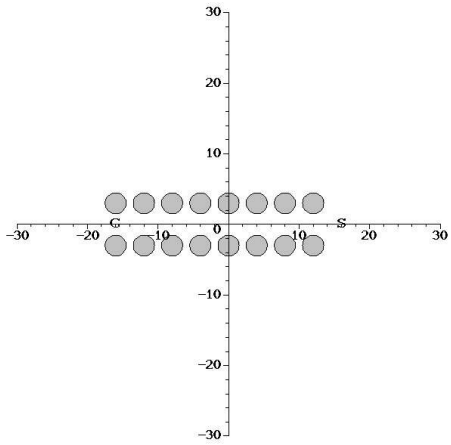


Figure 7.12: Experimental setup for timing experiments.

7.9.1 Experiment 7: Cost vs. resolution of scanning circle

Purpose

This experiment shows how the computational cost of the technique varies with the resolution of the sampling circle.

Technique

The path length is held constant at 32 units. A pair of obstacles are placed every 4 units apart along the length of the path. A selection of resolutions of sampling circle are used with samples being taken at 1, 2, 4, 6, 10, 15, 20, 30, 60 and 90 degree intervals.

Prediction

The asymptotic cost of Forward Chaining will rise linearly with the number of samples taken on the sampling circle. The total cost of Forward Chaining will therefore rise proportionally to the resolution of sampling circle.

Results

See Table 7.2 and Figure 7.13.

Run	1°	2°	4°	6°	10°	15°	20°	30°	60°	90°
1	10.172	5.953	4.016	3.188	2.688	2.469	2.359	2.172	1.985	1.969
2	10.062	5.968	3.922	3.203	2.672	2.468	2.265	2.141	2.031	1.953
3	10.172	6.000	3.938	3.188	2.704	2.344	2.360	2.172	2.015	1.969
4	10.171	5.968	3.985	3.234	2.657	2.453	2.297	2.203	2.016	1.968
5	10.125	5.969	3.938	3.234	2.687	2.391	2.297	2.125	2.016	1.953
Av.	10.14	5.97	3.96	3.21	2.68	2.43	2.32	2.16	2.01	1.96

Table 7.2: Results: CPU time (seconds) vs. sample spacing (degrees).

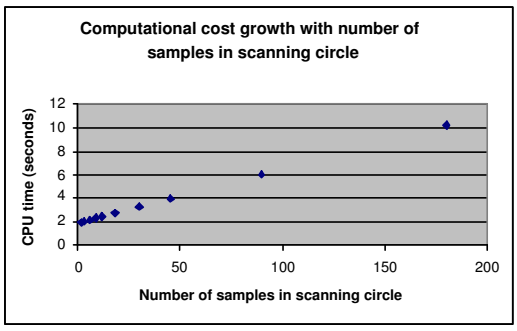


Figure 7.13: Graph of computational cost against quality of scanning circle.

Conclusion

The prediction is correct. Also, it can be seen from Table 7.2 and from Experiment 1 that a 10 degree sample spacing represents a good choice in terms of maintaining path quality while not incurring unnecessary computational expense.

7.9.2 Experiment 8: Cost vs. number of obstacles encountered

Purpose

This experiment shows how the computational cost of the technique varies according to the number of obstacles encountered along the path (and implicitly, the obstacle density of the map).

Technique

The path length is held constant at 32 units. A sampling circle with a 10 degree sample spacing is used (see Experiments 2,6). In successive runs, obstacles are encountered at a rate of 1 obstacle every 32, 16, 8, 4, 2, and 1 units of path length.

Prediction

The asymptotic cost of Forward Chaining will rise proportionally with the number of obstacles encountered over a given path length.

Results

See Table 7.3 and Figure 7.14.

Run	1/32	2/32	4/32	8/32	16/32	32/32
1	.375	.563	.937	1.656	3.188	6.859
2	.375	.547	.907	1.656	3.344	6.922
3	.375	.547	.921	1.672	3.234	7.422
4	.375	.563	.953	1.672	3.281	7.657
5	.359	.562	.938	1.703	3.390	7.719
Av.	0.37	0.56	.93	1.67	3.29	7.32

Table 7.3: Results: CPU time (seconds) vs. obstacle density along path (obstacles per unit length).

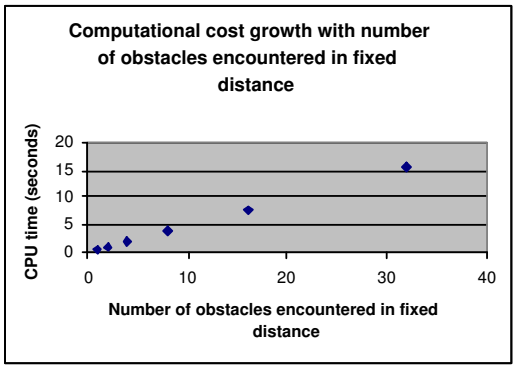


Figure 7.14: Graph of computational cost against obstacle density of map.

Conclusion

The prediction is correct. The asymptotic cost of Forward Chaining rises approximately proportionally with obstacle density.

7.9.3 Experiment 9: Cost vs. path length

Purpose

This experiment shows how the computational cost of the technique varies according to the length of the path needed, under the assumption that the density of obstacles does not vary across the map.

Technique

A sampling circle with a 10 degree sample spacing is used. A pair of obstacles are placed every 4 units along the length of the path. Path lengths of 2, 4, 8, 16, and 32 units are used.

Prediction

The asymptotic cost of Forward Chaining will rise proportionally with the length of path being generated.

Results

See Table 7.4 and Figure 7.15.

Run	PL=2	PL=4	PL=8	PL=16	PL=32
1	0.094	.156	.282	.562	1.016
2	0.093	.172	.297	.562	1.000
3	0.094	.172	.266	.579	1.032
4	0.094	.172	.281	.563	1.015
5	0.094	.172	.281	.563	1.000
Av.	0.09	0.17	0.28	0.57	1.00

Table 7.4: Results: CPU time (seconds) vs. length of path (units).

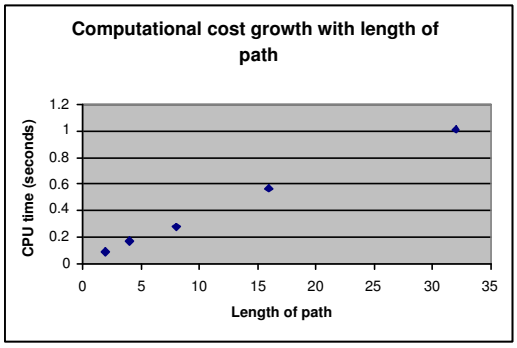


Figure 7.15: Graph of computational costs against length of path.

Conclusion

The prediction is correct. The asymptotic cost of Forward Chaining rises approximately proportionally with the length of path being generated. This is very useful since the cost of competing techniques such as graph search rise approximately exponentially with the length of path being generated. This is unsurprising, since graph search approaches are usually algorithmic global planning techniques whereas Forward Chaining is a heuristic local planning technique.

7.9.4 Optimisation of Forward Chaining

The runtime required to generate paths in this chapter is very high - for example, in the 'bigmap' environment, path generation took on average 56.8 seconds, with exact times depending on length of path. Although this level of computational cost is acceptable in some scenarios within robotics, where often only a single agent is used, such a high level of computational expense is completely inappropriate in virtual world scenarios with many agents, such as computer games or films. How can the computational expense of path generation be reduced?

Use of a low level language

The most significant improvement can be made by implementing the technique using a low level language. There was insufficient time for a complete re-implementation of Forward Chaining in C or machine code, but it was possible to generate timing information for one of the most computationally expensive parts of the program, iterated evaluation of potential value, in both Maple and C.

The task of iteratively evaluating potential is quite representative of the Forward Chaining heuristic as a whole, involving mathematical calculation, evaluation of a conditional statement, and a few function calls (with associated context switch costs) during calculation.

The structure of a code fragment for calculating potential values was replicated in C (with corresponding data types and functional decomposition). Timing information was then gathered over tens of thousands of iterations to estimate the expense incurred by using a high level rather than low level implementation. It was expected that a high level interpreted

language might be a factor of 100 times less efficient than using C.

Run	Maple (seconds)	C (seconds)
1	113.42	0.03
2	115.89	0.04
3	113.75	0.03
4	114.55	0.03
5	114.19	0.04
Av.	114.36	0.034

Table 7.5: Cost of computing potential values (seconds).

However, the results (Table 7.5) show that the high level implementation was approximately 3400 times less efficient than a naive C implementation. It does not seem unreasonable to suggest that a carefully hand-optimised machine code implementation might be a further factor of 3-10 times more efficient. This reduces the time involved in generating the entire path substantially. Further gains might be possible by selection of compiler, level of aggressiveness of compiler optimisation, reduction of code size so that it fits into CPU cache memory, unrolling the potential scanning and gradient descent loops, and so on.

Summary: Implementation in a low level language makes cost reduction by a initial factor of 3000 to 30000 feasible.

Incremental generation of the path

Forward Chaining is a local path-planning technique, so it is not necessary to generate the entire path at once, even if an a priori map is available that would allow this. Global techniques such as A* graph search must often be time-sliced so that the cost of path generation is not incurred all at once, but is spread to some extent over the period of time taken by the agent to follow the entire path. Forward Chaining is naturally suited to incremental path generation, so there is no need to implement a time-slicing approach.

If an agent takes 60 seconds to follow a path generated in its environment, then the number of agents whose navigation can be supported in parallel and in real-time is 60 times higher than if each agent takes only 1 second to realise the entire path that is generated.

Improved computational resources

If a faster processor, more cache memory or faster system memory is made available, the technique can generate paths more quickly. This is less useful in the case of computer games, where the consumer cannot be usually be expected to upgrade their gaming machine to suit a particular game, but it is very useful in automated computer generated cinematography as a means of supporting an increased number of agents simultaneously navigating in real-time on a single machine.

Selection of Forward Chaining parameters

Small gains in efficiency can be made by choosing larger gradient step sizes or lower resolutions of scanning circle, but these come at the cost of reduced quality in the generated path.

Reducing the mathematical costs

It might be possible to replace the mathematical function with an alternative but essentially equivalent function that can be calculated more quickly. It might even be possible to discretise obstacles at a particular resolution, and use a lookup table to evaluate obstacle potential and gradient functions more affordably. This could significantly reduce the cost of the technique. It may also be possible to reduce the computational cost throughout the technique by introducing a fixed point real number representation in place of floating point representations.

Better environment/map representation

In the implemented simulation, the simulated environment checked to see if an obstacle would affect the agent by carrying out a bounds test for every obstacle in the environment against the agent's current position. A more efficient implementation would group obstacles together so that bounds testing would take less computation. This might significantly improve the cost of the technique in cases where large a priori maps are involved.

Summary

By simultaneously applying all of the suggested techniques for reducing the cost of this approach, a gain in efficiency of 10^5 times over the current highly inefficient simulation is not unrealistic and a gain in efficiency of 10^4 times is certainly quite feasible. However, at present only one of the suggested optimisations has been tested, providing a gain in efficiency of 3.4 x 10^3 times.

The current implementation generates a path across an environment with 128 obstacles in approximately 60 seconds. If the agent takes 60 seconds of real time to traverse that path, then navigation for 1 agent can be supported in realtime.

If a more efficient implementation is used as detailed above, then realtime navigation of up to a hundred thousand agents might reasonably be simultaneously supported within a 128 obstacle environment. Given that the current generation of computer games using A* are limited to (at best) thousands of simultaneous agents, and that games developers have a reputation for producing highly efficient machine code implementations of core algorithms in their games, it is likely that Forward Chaining will have considerable appeal within this particular application domain.

7.9.5 Space complexity

Each agent requires several state variables. These are: the current position, the previous position, the goal position, the current subgoal position, the previous subgoal position, the accumulated path twist value, the accumulated goal twist value, the position of the temporary goal used in FWDS2/FORWARDS, and the current position being scanned during sampling of potential. If each of these state variables is represented within 4 bytes of memory, then each agent requires 36 bytes of memory at a minimum during operation.

One hundred thousand agents navigating simultaneously on a single system would therefore require approximately 3.6 megabytes of memory. This is well within the limits of current computing hardware.

7.10 Characterisation of the technique

This section uses the ideas presented in Section 2.4 to characterise Forward Chaining.

7.10.1 Generality

Forward Chaining is suited for a range of different underlying environments and obstacle representations within the map, as well as for different forms of obstacle and goal function, as the technique does not make assumptions about the nature of the environment that underlies the potential field. It can therefore be said to be general in nature.

7.10.2 Path planning vs. trajectory planning vs. tracking

Forward Chaining can be used as a path planning and path finding technique, and can also be used for path tracking if the agent's movements are unreliable and do not follow the path accurately - since the agent's path is built incrementally and is based upon the positions that the agent is actually achieving. The underlying continuous control surface of the goal-obstacle potential field ensures that when the agent is perturbed from its path by errors, compensatory corrections to the path will be continual and gradual rather than discrete and discontinuous. However, Forward Chaining does not associate positions along the path with a particular time, and in that sense, does not generate a continuous trajectory through space-time.

7.10.3 Optimality

The paths generated are not optimal. They are however usually quite efficient and direct.

7.10.4 Completeness

Forward Chaining is not guaranteed to find a path if one exists. This is because it is a local heuristic technique rather than a global algorithmic planner.

7.10.5 Soundness

The paths generated by Forward Chaining are guaranteed to be collision free, providing that the potential functions are set up correctly so that the interpolating gradient descent function cannot step onto an obstacle.

7.10.6 Computational cost

The computational cost of setting up the potential field at each step of navigation is proportional to the number of obstacles that are 'within sight' of the agent at that point.

The computational cost of navigating using the technique rises linearly with the length of path, the number of obstacles encountered on the path, and linearly with the number of subgoals and interpolatory steps used to build each part of the path.

On a more practical level, the computational costs of the technique when implemented in C or machine code should be sufficiently low to support tens of thousands, and perhaps hundreds of thousands of simultaneously navigating agents on a single midrange consumer system. This is discussed earlier in Section 7.9.4.

7.10.7 Limitations on representations of paths, agents, or obstacles

The technique currently assumes that the agent is point-sized and that obstacles have been dilated as necessary. There are no limitations upon the representations of obstacles except that it is must be possible to represent them using a potential function (which implies that it must be possible to calculate the agent's distance from the surface of the obstacle shape).

7.10.8 Suitability to problem constraints

- The technique is a local planner and is therefore suited to generating paths incrementally, and can either path-find using information discovered during online navigation, or can generate paths offline using an a priori map.
- It should possible for the technique to cope with multiple agents, moving obstacles,

and non-holonomic constraints, but currently it does not have any features implemented to suit these situations.

- The technique's dependence on subsystems are as follows:
  - The agent must be able to detect local obstacles and represent them as a potential function to Forward Chaining.
  - The agent must be aware of its position relative to the goal.
  - In the current implementation, the agent must be able to move quite reliably for small distances during subgoal interpolation to enact the transitions made by fixed step size gradient descent.

7.10.9 Applicability to environments

- It is suitable for large and complex environments and obstacle configurations but it is still heuristic in nature and cannot be guaranteed to succeed in every case. Pathological navigation problems have been identified. The technique works with complex obstacle shapes providing they can be turned into a potential function in some way.
- The technique provided so far works in 2-D. Mappings to allow 3-D and n-D navigation will be considered in the next chapter.
- The technique is continuous in nature and does not rely upon a discretisation of the environment.

7.10.10 Non-standard ways of classifying approaches

Naturalness

It is difficult to assess the technique on a qualitative basis, but it might be reasonable to suggest that in general, the paths generated are not dissimilar from those that might be followed by a natural agent operating using local information - for example, an animal with a limited field of vision, or a blind human with a guide stick who is therefore only capable of detecting the arrangement of obstacles very nearby.

The technique is not artificially optimal, nor overly inefficient, except in those cases where it navigates into closed obstacle configurations that an intelligent sighted agent might analyse from afar as being incapable of leading to the goal.

Ease of comprehension

The technique can be represented in a few lines of pseudocode, and does not require any understanding of advanced mathematics such as fluid dynamics. It should be therefore be appropriate for implementation by an undergraduate student who has basic programming proficiency.

7.11 Chapter Summary

This chapter has considered parameters to Forward Chaining and their qualitative and quantitative effects. The experimental results have demonstrated the successful operation of Forward Chaining in navigation. General surface problems for potential field approaches and specific problems for potential field methods for navigation have been discussed. Established criticisms of the potential field approach to navigation have been addressed. Pathological cases where the heuristic fails have been identified but are not considered sufficiently significant to diminish the claim that Forward Chaining is capable of navigation in scenarios of moderate to high complexity. Finally, Forward Chaining has been characterised as a navigation approach in a way that allows comparisons to be drawn with other types of navigation approach.

Chapter 8

Forward Chaining in 3-D and n-D

Guide to Chapter 8

This chapter discusses techniques by which the Forward Chaining approach may be made suitable for directing navigation in 3-D and n-D environments.

8.1 Forward Chaining in 3-D

8.1.1 Introduction and motivation

So far, this research has been concerned with the implementation of an improved potential field based navigation technique in 2-D. However, most potential field techniques have the ability to be applied to 3-D spaces (and many to n-D spaces) and so it is important to examine whether or not Forward Chaining, previously defined in 2-D, can be made to have this ability.

The operation of a navigation approach in three dimensions is relevant to all of the application domains. With space, air and underwater robotics, and space-based games and cinematic productions, it is likely that there would be some interest in heuristics that can overcome the navigational problem in a 3-D environment in a straightforward and computationally cheap manner.

Moreover, if it can be shown that the technique can be used with potential field problems of higher dimensionality, then it is more likely that the technique is suitable for re-application as a potential field based approach to other types of problem besides navigation.

Gradient descent is already known to be defined in 3-D and n-D - so the subgoal interpolation technique is already defined for movement in 3-D between subgoals. In order to translate Forward Chaining into 3-D, it is necessary for subgoal selection to also be transferred into 3-D. Each aspect of the subgoal selection heuristics will therefore have to be considered. For example, what does it mean to 'scan' points in 3-D, or to become 'twisted' in 3-D? The concepts were fairly intuitive in 2-D but become far less so, when an added dimension must be considered. These concepts will be considered in the order that they were introduced to the subgoal selection heuristics of Forward Chaining in 2-D.

8.1.2 Re-modelling LPCIRCLE in 3-D

LPCIRCLE relies on scanning equidistant points around a circle to select a point of lowest potential. What does it mean to do this in 3-D? Trivially, the answer is to scan points on a sphere instead of a circle. How can this be done in a manner keeping with the original approach?

If points on a circle are being scanned around a central position in 2-D space, it is necessary to generate the set of points to be evaluated in an automatic manner. In the case of the Forward Chaining implementation in this research, the task of generating positions was carried out by using the polar form of a circle, i.e. Equation 8.1.

$$x = Rcos(\theta), y = Rsin(\theta) \tag{8.1}$$

Here, θ is the angle taken from the y axis and R is the radius of the circle to be scanned. By taking a regular angular separation (that is, varying θ by the same amount each time), a series of points are generated that are equally distant from their neighbours in terms of metric distance, and are spaced around a circle. In 2-D, angular spacing and metric spacing are intrinsically proportional. In 3-D, this is not the case.

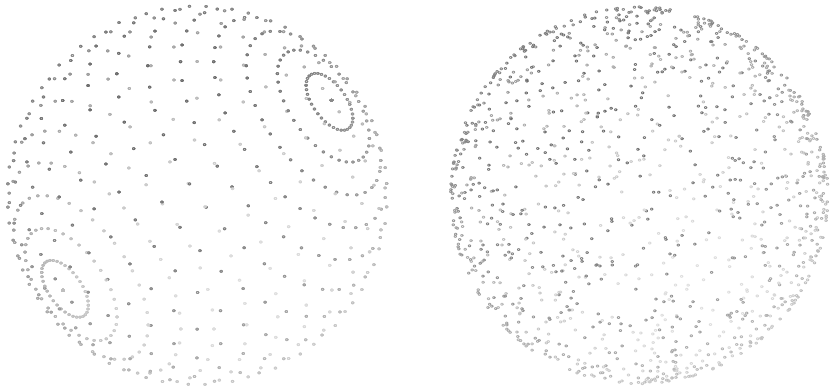
Polar coordinates

To maintain regular angular spacing, the polar coordinate form of a sphere could be used to generate a set of points on the surface of a sphere centered on the current position - i.e. Equation 8.2.

$$x = Rsin(\theta), y = Rcos(\theta)cos(\phi), z = Rcos(\theta)sin(\phi) \tag{8.2}$$

Unfortunately, this approach has a drawback. If equal angular spacing is used (that is, varying θ and ϕ by the same amount each time), then the points that are generated are not spaced equally in terms of metric distance to their neighbours. This is shown in Figure 8.1(a). Particularly, the points at the 'top' and 'bottom' of the sphere are clustered. A bias towards certain parts of the sphere around the agent's current position is therefore present when sampling in this way.

It is possible to correct the clustering problem by adjusting the size of the angular resolution of one variable (ϕ) while keeping the other (θ) fixed. There are alternative mechanisms though which may be preferable for implementation, which are detailed below.



(a) Spherical co-ordinates, regular spacing of θ and ϕ

(b) Normal random distribution

Figure 8.1: Generating a set of points to sample on a 3-D sphere.

Random sampling

A set of points to be considered on the sphere may be generated by selecting random positions. To minimise the 'clustering' problem present in the polar coordinates approach, a uniform distribution should be used. A simple way of producing such a distribution is to generate a number of 3-D vectors with x, y, z values that are selected randomly in the range [-1,1], discarding any combination where $x^2 + y^2 + z^2 > 1$, and then normalise the vectors. Unfortunately, such a distribution may still form clusters, however these will tend to be smaller and at random positions. An example of a typical sample distribution using this technique in shown in Figure 8.1(b).

Polyhedral approximation

A better alternative may be to approximate a sphere using a polyhedron of as high order as computational resources will permit. To do this, a facet splitting[1] approach may be used (shown in Figure 8.2). The 'sphere' is initialised to be a cube or tetrahedron. The shape is then refined by forming several new facets in place of each facet on the current model - typically by bisecting the edges and normalising the resulting vertices to lie on the surface of the sphere. This is repeated iteratively until a sufficiently detailed polyhedral approximation to a sphere is formed. This approach has several useful properties - it produces sets of points that are almost equal distances apart, avoids clustering, and the computational cost can be controlled by limiting the number of iterations of facet splitting.

[1]This technique is also called 'surface refinement', particularly in computer graphics.

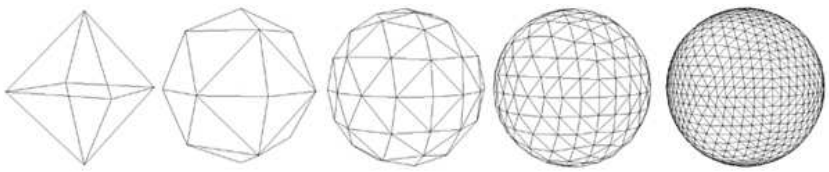


Figure 8.2: Polyhedral approximation technique for generating approximately equidistant samples on the surface of a sphere. This picture was supplied by Paul Bourke of Swinburne University of Technology, Australia.

Conclusion

It is straightforward to convert LPCIRCLE to operate in 3-D rather than 2-D. There are several approaches that can be used to generate a set of points to be considered on a sphere surrounding the agent's current position. The definition of LPCIRCLE in 3-D remains consistent with the original definition in 2-D. LPCIRCLE in 3-D should overcome convex obstacles and small flat or concave obstacles.

8.1.3 FWDS1 in 3-D

The difference between LPCIRCLE and FWDS1, is that FWDS1 only considers the points in the 'forwards semicircle'. In 3-D, this means that FWDS1 will consider points on the 'forwards hemisphere'. How can this be defined?

Re-modelling forwards

The forwards direction is defined as before in Section 6.7, but for mathematical precision, it is considered to be a vector rather than just a direction. This vector is defined as being from the position of the second last subgoal, to the position of the last subgoal.

Re-modelling the forwards hemisphere

An algorithmically simple and computationally cheap way to generate a hemisphere of points around the forwards vector is to consider placing a plane through the middle of the sphere, i.e. 'plane clipping'. The clipping plane equation is defined using the agent's current position, and the forward vector as above.

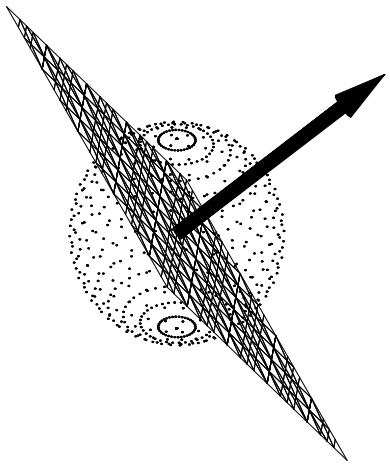


Figure 8.3: The plane-clipping technique for selecting points in the forwards hemisphere.

Each point in the sphere is checked by substituting its x, y, z coordinates into the plane equation. If the point generates a negative value, then it lies 'behind' the plane and should not be considered by FWDS1. If the point generates a positive or zero value, then it lies either 'forwards' or on the plane, and should be scanned by FWDS1. This is shown in Figure 8.3.

The resulting reduced set of 'plane-clipped' points can be then be evaluated to select a point of lowest potential in the forwards hemisphere.

Conclusion

FWDS1 can be defined in 3-D, and the definition remains consistent with the original 2-D definition. FWDS1 in 3-D should overcome flat surfaces or shallow concave bowl shapes in 3-D.

8.1.4 FWDS2 in 3-D

The difference between FWDS1 and FWDS2 is that FWDS2 introduces 'goal-bias'[2], which must be translated into a 3-D equivalent. In FWDS2 in 2-D, 'goal-bias' was implemented by placing a temporary goal at an angle up to B degrees[3] from the forwards direction,

[2]See Section 6.8.

[3]In the experiments, a value of 75 degrees was found to be suitable.

towards the goal. First, it is necessary to define the goal direction as a 3-D vector. Secondly, it is necessary to describe what it means to choose a goal-biased direction using the forwards vector and the goal vector, in 3-D.

Re-modelling the goal vector

The goal vector is defined to be the vector difference between the agent's current position, and the position of the goal.

Re-modelling goal-bias

To turn optimally (and to maintain consistency with the definition of FWDS2 in 2-D), the agent should bias the vector in the axis containing both the goal vector and the forward vector.

There is a elegant way to carry out the goal-bias operation using vector calculus. Observe that if the forward vector and goal vector are less than B degrees apart, then the 'goal biased' vector is the goal vector itself. In the case where the scalar product suggests the vectors are more than B degrees apart, the vector must be biased towards the goal by the full B degrees in the 'goal-forward' axis. Therefore, if the initial angle between the goal vector and the forward vector is d degrees, the angle between the goal vector and the resulting temporary goal vector will be $d - B$ degrees, and the angle between the forward vector and the temporary goal vector will be B degrees. This relationship allows two simultaneous equations to be solved to generate the new vector in terms of f and g, the forward and goal vectors.

Conclusion

It is possible to map FWDS2 into 3-D using the technique described above.

8.1.5 FORWARDS in 3-D

Unfortunately, no solution has been developed so far to map FORWARDS into 3-D.

8.1.6 Global search problems presented by 3-D

In 2-D, surface search is trivial. In 3-D, this is no longer the case. A new class of global problems might demand an entire surface search to stand any chance of being successful. Two examples are given below.

The Bowling Ball Problem

Here, the agent is faced with the task of navigating to a goal that is set inside a bowling ball. The only way that the agent can possibly reach the goal is if it searches the entire surface and finds the 'finger holes' leading in to the center of the ball.

The Balloon Problem

Here, the agent begins inside an 'inflated balloon' and must try to find a way out as part of the process of reaching the long term goal. Unfortunately, the only way it can do this is if it finds the 'neck' of the balloon which leads to the outside world - again, demanding an entire surface search of the inside of the balloon.

Summary

It has been suggested [3] that perhaps some kind of analogue to human sight - being able to sweep across the surface from a great distance - might lead to a solution. However, at the current time these global problems cannot be satisfactorily overcome by Forward Chaining in 3-D.

Fortunately, it is highly unlikely that an agent would in practice be required to navigate in such a pathological environment unassisted. It is much more likely that the problems to be overcome in the 3-D environment are a number of convex and concave obstacles that stand between the agent, and the goal. A simulated spaceship or fish or bird is unlikely to have to navigate its way into a bowling ball very often in practice, and is much more likely to navigate its way past shapes corresponding to planets, boats or trees, or other agents. Forward Chaining should certainly be suited for such navigation in 3-D.

8.1.7 Conclusions

Techniques have been provided that allow Forward Chaining to operate in 3-D in the cases of the LPCIRCLE, FWDS1 and FWDS2 subgoal selection heuristics. Experimentation with Forward Chaining in 3-D remains an open research topic.

8.2 Forward Chaining in n-D

In the previous section, the possibility of extending Forward Chaining to allow 3-D navigation was considered. There are two motivations for going further and considering n-D navigation problems. Firstly, this would allow navigation between positions in the configuration space of multi-joint effectors belonging to robots and possibly complicated virtual world agents - i.e. inverse kinematics. Secondly, this would allow possible re-application of Forward Chaining in other domains of AI, such as neural networks, where n-D spaces containing local minima are commonplace.

8.2.1 Re-modelling LPCIRCLE in n-D

As the task of LPCIRCLE scanning for a point of lowest potential changed from scanning a 2-D circle to scanning the surface of a 3-D sphere, the cost of scanning rose dramatically due to the increased number of samples to be taken in the higher dimensioned space.

As navigation takes place in increasingly higher dimensionalities, the cost of scanning for even a single subgoal grows exponentially with each extra dimension added. Clearly, a better mechanism must be presented in order for LPCIRCLE scanning (and consequently the entire Forward Chaining approach) to be considered feasible in n-D.

The idea of scanning is to find a good estimate of the point of lowest potential on a (hyper)spherical surface in the potential field space by measuring actual potential values. The techniques described earlier, such as random sampling of points on the surface of the sphere, could just as easily be used to find a candidate point on a hypersphere. However, could the estimate of the lowest point on the hypersphere be generated more cheaply?

8.2.2 Re-introducing Gradient Descent

It is necessary to find a point of lowest potential out of all the points on the surface of the hypersphere. What if gradient descent was used along the potential surface of the hypersphere to provide such an estimate?

It would be necessary to derive equations for the variation of the potential function in terms of angular movement around the hypersphere, in other words, gradient functions for the 'ϕ', 'θ' (and so on) angles as they vary around the hypersphere. If feasible, the result would be a cheap and perhaps effective search for the lowest point on the hypersphere surface.

Gradient descent could be carried out multiple times from many positions on the hypersphere for relatively low computational cost, and the best position found could be considered as a good estimate of the lowest point on the hypersphere surface.

The main problem with this approach would be that the gradient functions would probably be difficult to differentiate around the hypersphere surface by analysis. A secondary problem is that the position chosen to initialise the surface search may lie inside an obstacle boundary and thus have undefined gradient. This might be overcome by setting the gradient function inside an obstacle to lead the descent 'out' towards the boundary of the obstacle during scanning.

Re-application to the case of 3-D navigation

It can be observed that this new 'hypersphere gradient descent' LPCIRCLE technique carries out a gradient descent-based search for a point of low potential on a small, finite hypersphere surface within a much larger volume of space - i.e. a search problem in a space of lower dimensionality than the original environment.

In the case of a 3-D navigation environment, the surface of the sphere is 2-D. It is known that LPCIRCLE is expensive in high dimensionality, but that it is affordable in 2-D. So why not also apply the LPCIRCLE heuristic to the task of finding a low point on the surface of the search sphere?

It may therefore be unnecessary to scan a large number of points in 3-D. LPCIRCLE, or

even FWDS1[4] can be carried out on the surface of the sphere to rapidly identify a deep minimum suitable as the next subgoal.

This has a benefit in that it is now unnecessary to analytically derive a gradient function that varies according to the angle around the sphere. On the other hand, several runs of this heuristic may be required from different starting coordinates, to be sure of finding the best minimum on the sphere surface.

Summary

In the case of 3-D navigation, a subgoal can be generated by carrying out the LPCIRCLE or FWDS1 heuristic on the 'sampling sphere' surface to find a point of low potential. In general, if the hypersphere sampling problem is treated as a minimisation problem in $(n-1)$-D, any descent regime (gradient descent, LPCIRCLE, FWDS1) will be appropriate. If LPCIRCLE or FWDS1 is used, the need to solve a minimisation problem in $(n-2)$-D arises - potentially allowing a recursive application of LPCIRCLE or FWDS1.

Conclusion

It is possible to scan for a subgoal during n-D navigation by either carrying out 'gradient descent'-type search on the hypersphere surface. This should produce monotonically improving solution quality with increased computation time, in the manner of the *anytime* algorithms of [242].

8.2.3 Re-modelling the FWDS heuristics in n-D

Fortunately, the vector-based solutions presented in Section 8.1 are already sufficiently general that they can be applied directly to the n-D case to allow Forward Chaining to overcome n-D concave bowl shapes.

- FWDS1. A hemi-hypersphere is now used, rather than a hemisphere. The clipping

[4]FWDS2 would require a goal vector to be defined, which is not present in the case of blind search on the sphere surface. The goal that is available in navigation is only a goal for the problem of navigation, not for the problem of finding the lowest point on a particular surface within the potential field.

hyperplane is defined by the n-d Forward vector.

- FWDS2. Goal bias is carried out using the same vector calculus as in the 3-D case.
- FORWARDS. No proposed solution at present.

8.2.4 Conclusions

Brute-force scanning is clearly inappropriate for descent on high dimensionality potential field surfaces. A novel modification to the LPCIRCLE descent technique has been presented that may make Forward Chaining very practical for high dimensionality problems, even with limited computational resources (Section 8.2.2). A second modification was also presented that may make the heuristic more effective on 3-D problems (Section 8.2.2).

The variety of problematic surfaces that might occur in n-D are endless, but it is expected that Forward Chaining - particularly LPCIRCLE and FWDS1 - will produce superior results to traditional gradient descent or random walk approaches.

The mappings of Forward Chaining into n-D presented in this section demonstrate that the approach is likely to be suitable for other areas of AI research, where n-D potential surfaces are the norm. It is hoped that future research will clarify the suitability of Forward Chaining for such purposes.

Chapter 9

Applications of Forward Chaining

Guide to Chapter 9

This chapter suggests that the research in this thesis can be applied to improve the performance of other navigation approaches. Possible applications of this research to general heuristic AI are also highlighted.

9.1 Applying this research within navigation

Some possibilities are presented below for ways in which existing techniques within navigation research might benefit from some of the ideas in Forward Chaining.

9.1.1 Augmenting Navigational Potential Fields

This section will consider the possibility of exploiting aspects of this research to improve a major competing technique, the 'navigational potential fields' approach.

Key properties of LPCIRCLE descent

- It is practical in 2-D and 3-D situations, but much less so in n-D.
- It is not vulnerable to saddle points, plateaux or ravines, but it is vulnerable to local minima.
- It accurately tracks the potential at a distance and does not 'step uphill' in the way that continuous gradient descent can.
- It is more computationally expensive than continuous gradient descent.
- Travels very directly to points of low potential.

Key properties of Navigational Potential Fields

- They are suitable only for 2-D, or at best 3-D global navigation.
- The fields generated contain no local minima, but they do contain plateaux, saddle points and ravines.
- When used with continuous gradient descent, there is a small risk of the agent stepping 'uphill' and moving to a different flow of potential.
- It can be computationally expensive to generate the potential field and so it is only suited to cases where a good amount of computational power is available.
- They are only suited to simple environments with non-complex obstacles.

- The gradient of the field can direct the agent on meandering, arcing paths in order to reach points of lower potential.

LPCIRCLE suits navigational potential fields

If navigational potential fields are already being used in a particular application, the computational cost of LPCIRCLE is likely to be quite acceptable. If LPCIRCLE is used for descent over navigational potential fields, it overcomes several of the remaining weaknesses of the approach, such as saddle points, ravines, and plateau. It also reduces the danger from accidental movement between flows caused by the local gradient failing to capture the way the potential is changing on the surface at a distance, or by random perturbation from a flow.

Furthermore, LPCIRCLE will move downhill very efficiently, even if the local gradient suggests a more meandering route. This means that LPCIRCLE may help alleviate the meandering, arcing paths sometimes found in navigational potential fields. This is because although LPCIRCLE can 'jump' trajectories in much the same way gradient descent can, LPCIRCLE does it in a safe way by knowing what the potential value will be in the trajectory it jumps into so that it does not risk accidentally heading 'uphill' upon reaching the new trajectory[1].

LPCIRCLE cannot overcome all the problems of navigational potential fields, such as their restriction to quite simple environments with non-complex shapes and high computational cost during potential field generation, but it can noticeably improve the behaviour of the approach when the navigational field is being used.

Clearly, there is some interesting research to be carried out in this field by examining what happens when LPCIRCLE is applied to navigational potential fields. It is even possible that a Subgoal Chaining approach may be used to overcome remaining issues such as arcing behaviour, or non-classical local minimum situations that satisfy the Laplace equation.

[1] An analogy might be to consider a stick floating down a river and occasionally being caught in eddies as a result of its momentum carrying it from the current (Gradient Descent) as opposed to a strong swimmer who can see the strongest current ahead in the river, and can head directly for it to accelerate their progress downstream (LPCIRCLE).

9.1.2 Augmenting hybrid approaches

This section will look at the possibility of improving another class of approaches, the hybrid approaches, where discrete or continuous potential fields are used to augment the capabilities of another type of navigational approach.

'Continuous Gradient Descent' Hybrids

Consider hybrid approaches such as Krogh's method [135] or the hybrid approach taken in the computer game 'Prisoner of War' [44, 1]. Here, continuous gradient descent is being used to link up travel between cells of e.g. an A* cell decomposition generated path.

These techniques (such as A* cell decomposition) are usually resolution complete, and therefore not vulnerable to the kinds of pathological problems that prevent FORWARDS from being fully successful. So why not consider replacing gradient descent with Forward Chaining - and in the process, overcome pathological problems for Forward Chaining and simultaneously improve the situation faced by the other technique within the hybrid?

Advantages

It is anticipated that Forward Chaining would offer the following advantages:

- Continuity. Forward Chaining produces path shapes that are of similar continuity to those of continuous gradient descent.
- Reduced failure rate. If every other aspect of the hybridisation is left the same, then Forward Chaining will simply be more reliable. In cases where a local minimum causes local navigation using continuous gradient descent to fail - disrupting the overall process of navigation - Forward Chaining will instead succeed in navigating.
- Reduced cost in the hybridised technique. The hybrid can be changed to exploit the fact that Subgoal Chaining is a more powerful approach and can be used over larger 'local' areas than continuous gradient descent. For example: In the case of 2-D A* navigation (which can be extremely computationally expensive with high resolution grids), the grid cell size can be increased, because Forward Chaining will be able to navigate extremely reliably within much larger grid cells than continuous gradient.

It is worth highlighting the fact that keeping computational costs low is *extremely important* in cinematic and computer game agent navigation, and a technique which can be 'dropped in' to replace gradient descent, and which significantly reduces A* navigation costs, is likely to be warmly welcomed.

'Discrete Gradient Descent' Hybrids

Consider cases such as Zelinsky's hybrid model [241], where the contribution made by potential fields is not to provide continuity, but rather to provide the ability to navigate online. Here, a global planner is used to navigate between positions on a static map, and dynamic obstacles (such as humans, litter and so on) can be overcome using discrete gradient descent techniques such as minimum filling or wavefront expansion, within a local (and thus computationally affordable) area.

What advantages does Forward Chaining offer?

- Continuity. The approach produces more natural looking, smooth paths than the discrete gradient descent approach.
- Better paths. The minimum filling approach often used with gradient descent can produce very unusual and un-natural looking paths if the agent actually moves around to each position as it is 'filled in'.
- Better results in 3-D. Minimum-filling a concave bowl shape can be very expensive in 3-D, where the volume of a minimum basin may take some time to fill. Rather than spend time filling in the minimum, Forward Chaining should simply travel directly back out of the concave bowl shape.

9.1.3 Conclusions

Forward Chaining and LPCIRCLE offer a number of possibilities for improving many types of alternative navigation approaches.

9.2 Applying this research to general heuristic AI

Forward Chaining is loosely based upon Subgoal Chaining, a spline-based approach that was developed in neural networks and which was discussed at the start of Chapter 6. Is it possible that some of the ideas involved in this research can be employed in general AI research - in areas such as neural networks [171], automatic systems configuration optimisation [219] and so on?

First of all, the subgoal chaining technique itself (though not invented in this research) might be valuable for general AI, now that it has been proven in at least two different fields (navigation and neural networks).

Secondly, the idea of using the potential field itself combined with the agent's internal state to generate subgoals, rather than a non-PF technique, appears to be relatively new (it is certainly not the approach presently taken in navigation, where non-PF navigation techniques are used to generate subgoals).

Thirdly, LPCIRCLE as an alternative to standard continuous or discrete steepest gradient descent may be useful - particularly if the proposed *novel hypersphere surface search* technique was used, which can be applied generally and cheaply in n-D spaces.

Fourthly, FWDS1 may be useful as an alternative technique in place of, for example, momentum, as a mechanism for escaping local minima on general potential fields. FWDS1 does not rely on information about a 'goal', so it is suitable as an escape technique in fields where the global minimum is not known a priori (such as neural network training, and automatic system configuration optimisation). FWDS1 has already shown itself to be effective in escaping some large local minimum basins (Chapters 6, 7). It is computationally cheap to implement if *hyperplane clipping* is used, as it halves the number of points to be considered by LPCIRCLE.

Finally, FWDS2 or FORWARDS might be useful in cases where there is a known goal combined with unrealisable parts of a potential field. This may even turn out to be the case in e.g. neural networks - [171] suggests a technique for obtaining an estimate of the direction of the global minimum during neural network training.

In conclusion, there are many aspects of this research that may be suitable for re-use, either as frameworks for development of new potential field methods, or even for immediate and direct re-application in other fields as improved descent techniques.

Chapter 10

Further Work and Conclusions

Guide to Chapter 10

This chapter starts by outlining some directions for long term future work based on this research. The chapter then describes the current condition of the technique developed and discusses the validity of the hypothesis of the research. This work is concluded with a short summary of the novel contributions, avenues for future research, and overall consequences of this piece of research.

10.1 Long term avenues of research

This section contains a brief selection of partially developed ideas for extending and re-applying Forward Chaining.

10.1.1 Moving obstacles

Not all environments are static. Though Forward Chaining is capable of coping with the problem of moving obstacles to some extent[1], the potential field representations and techniques discussed so far have no ability to model moving obstacles. How might moving obstacles be represented, and how might Forward Chaining be made to cope with them? Below are some ideas that might present opportunities for further development.

Representation of obstacles

1. It might be possible to pretend that an obstacle is in *all* of the positions it will occupy in the near future. This approach could be so overcautious however, that failed navigation would result when trying to navigate around the multitude of 'obstacles' that would then exist.

2. Another representation of a moving obstacle might involve raising the potential slightly ahead of the obstacle, in the direction it is travelling, and lessening it behind the obstacle. Instead of being a fixed shape irrespective of speed, the obstacle might be 'smudged' in the direction of travel such that certain positions will gradually become less and less likely to be chosen by the agent, in the time preceding the obstacle's actual arrival at those positions.

3. Another possibility might be to increase the dimensionality of the environment in which navigation is taking place, so that 'time' becomes a dimension, and 2-D moving obstacles are now fixed position 3-D obstacles that must be navigated around. Positions implying backward or static movement in time could be deselected from the sampling process automatically.

[1] For example, if an obstacle moves while outside of the 'range of vision' of the Forward Chaining heuristic, then it certainly will not present any problems. This is a property shared by all local planning techniques.

Adapting Forward Chaining to cope with moving obstacles

1. In the case where the dimensionality of the navigation environment is increased to include time, it is important to remember that Forward Chaining needs to be able to travel backwards in a controlled manner as part of its behaviour. With time represented as a dimension, it is not obvious how backwards travel can be achieved in this particular dimension.

2. In the case of the second representation, Forward Chaining might need to be able to stop, or slow down, in order to navigate successfully. One way of doing this might be to allow the FWDS heuristics to scan the potential value of points lying in concentric circles around the agent's current position, and also scan the potential of the current position, taking into account the potential contribution of moving obstacles. If a nearby circle contains the point of optimal potential, then this corresponds to the agent slowing down while travelling in that direction - conversely, if a far-away circle contains the optimal point, this corresponds to the agent speeding up to navigate best around the obstacle. Finally, if the the current position already represents the most optimal potential, then this corresponds to the agent stopping till positions of more optimal potential become available again and navigation can continue.

   The cost of scanning concentric circles is non-trivial of course, the cost of scanning rising broadly in proportion to the number of circles to be scanned.

These ideas for coping with moving obstacles are somewhat naive and are mainly taken from existing research and re-framed in light of Forward Chaining, but the second suggestion for adapting Forward Chaining is actually a novel contribution that was not available to previous potential based approaches based on discrete grids or direct use of the gradient function.

A final concern in the case of any subgoal chaining based navigation approach, is the estimation of time between subgoals - since actual interpolative travel is being carried out by gradient descent which has no inherent facility to deal with moving obstacles.

Good coverage of the moving obstacle problem for potential field based navigation can be found in [113, 168].

10.1.2 Multiple agent scenarios

How might Forward Chaining be enhanced to cope with multiple agent navigation? Clearly, the points raised under 'moving obstacles' can be directly re-applied here, particularly in regard to allowing the agent to control its speed in response to moving obstacles.

In the case of communicating agents, recall the 'circle' that agents have available in the LPCIRCLE and FWDS heuristics. The possibility now emerges that agents can deliberately de-select parts of their circle immediately if they know that another agent may use it. This might be done by the agent communicating its path shape by sending its current position, its next subgoal, as well as the size and shape of the agent. These could be broadcast to other nearby agents, who could immediately remove from consideration points lying under the area that will be used by their fellow agents.

Moreover, agents could negotiate for positions on their circles which are particularly valuable to them, and 'reserve' or even 'buy' and 'sell' them with other agents. Like people choosing to wait politely to get through a crowded door, so that they are given an opportunity to exit themselves in turn, agents could stop movement and 'sell' parts or all of their circle to the group, then use the accumulated credit to 'buy' a chance at clean navigation. The issue of *trust* may arise during this interaction.

This however does not represent a benefit of Forward Chaining itself, but rather extension or re-application of the 'circle' metaphor employed by the Forward Chaining heuristics.

Standard multiple agent techniques such as map fusion (where map information sharing takes place among agents) could also be employed. Work in potential field based multiple agent coordination can be found in [120, 174, 200, 201, 223, 226].

10.1.3 Non-holonomic constraints

In the case of non-holonomic constraints, where the agent suffers some inability to navigate as a consequence of its environment or construction, it is possible to consider running a non-holonomic constraint modelling algorithm or heuristic before FWDS potential sampling takes place, in order to de-select automatically impossible points from the 'circle' before FWDS has a chance to consider them. This way, it can be ensured that FWDS' selected subgoals will not require behaviour from gradient descent does not conform to the agent's

non-holonomic constraints. Alternatively, constraints may be modelled by associating them with a contribution to the potential value of positions to discourage selection of impossible or dangerous positions by Forward Chaining [221]. Recent examples of work that might be integrated with Forward Chaining can be found in [113, 221].

10.2 Current condition of the technique

The currently implemented Forward Chaining technique is a pure potential fields method that is able to generate navigational paths in 2-D continuous spaces of moderate complexity and that does not require an a priori map or complex a priori computation. It can cope with several forms of concave obstacle configurations that are associated with intractable local minima for many existing online potential field based navigation techniques.

The technique is not yet fully capable of navigation on entirely arbitrary 2-D environments, as pathological problems preventing navigation still exist, though it seems highly plausible that navigation within such environments is achievable through future improvement.

10.3 Suitability for application domains

In robotics, Forward Chaining can be substituted directly as the navigation mechanism in cases where existing less powerful potential field techniques (such as gradient descent in the manner of Chapter 4) are currently used. Elsewhere in robotics, supporting techniques to generate maps, give estimates of the goal direction, and control of a robot's effectors must also be provided, which may limit applications.

In computer games, the technique is particularly suitable for navigation. As reliable obstacle position and goal direction information are easily available in this domain, pathological problems can be avoided and the computational costs of C or machine code implementations of the technique should be appropriate for controlling large numbers of agents. Similarly, the technique should be suitable for use in controlling cinematic agents except where multiple-agent interaction is a key issue. Further research could substantially improve the heuristic in this regard.

Table 10.1: General properties of significant potential field navigation approaches. Numbered footnotes: see page 121 for explanation. Asterisked items: See Chapter 8. Double-asterisked item: See Section 7.5.3. Triple-asterisked items: See Appendix E.

	Gradient Descent	LM Detection + Random Movement	Elliptical Potential	Continuous Navigational Function	Discrete Gradient Descent	Discrete Random: RPP	Discrete Minimum Filling	Discrete Navigational Function	Forward Chaining
Online navigation?	✓	✓	✓	✗	✓	✓	✓	✗	✓
Computationally cheap to generate field?	✓	✓	✓	✗	✓	✓	✓	✗	✓
Computationally cheap to use field?	✓	✓	✓	✓	✓[1]	✓[1,2]	~[3]	✓[1]	✓
Simple model/easy to understand?	✓	✓	✓	✗	✓	✓	✓	~[4]	✓
Simple implementation?	✓	✓	~[5]	✗	✓	✓	✓	~[4]	✓
Robust (to noise or errors)?	✓	✓	✓	✓[6]	✓	✓	✓	✓	✓
Generates smooth trajectories directly?	✓	✗	✓	✓	✗	✗	✗	✗	✓
Can navigate in 2-D?	✓	✓	✓[7]	✓	✓	✓	✓	✓	✓
Can navigate in 3-D?	✓	✓	✓[7]	~[8]	✓	✓	✓[3]	~[9]	✓*
Can navigate in n-D?	✓	✓	✗[7]	✗	✓	✓[2]	✗	✗	~*
Unaffected by local minima?	✗	~[10]	✗	✓	✗	~[10]	✓	✓	✓
Unaffected by ravine problems?	✗	✗	✗	✗	✓	✓	✓	✓	~**
Unaffected by plateau problems?	✓	✓	✓	✗	✓	✓	✓	✓	✓
Unaffected by saddle points?	✗	✓	✗	✗	✓	✓	✓	✓	✓
Allows arbitrary obstacle shapes?	✓	✓	✗	✗	✓	✓	✓	✓	✓
Doesn't re-visit visited parts of the map?	✓	✗	✓	✓	✓	✗	✓	✓	~***
Avoids grazing collision with obstacles?	✓	✓	✓	~[11]	✓	✓	✓	~[4]	✓
Works in trivial environments?	✓	✓	✓	✓	✓	✓	✓	✓	✓
Works in intermediate environments?	✗	~[10]	✗	✓[12]	✗	✓	✓	✓	✓
Works in complex environments?	✗	✗	✗	✗	✗	✗	~[13]	✓[14]	~***

10.4 Consideration of hypothesis: Part 2

> "The Potential Fields Method has been prematurely abandoned as an approach to agent navigation, and the existing techniques have unaddressed weaknesses. Novel descent heuristics can be designed that allow successful navigation in an improved number of cases and that address the weaknesses of existing potential field techniques."

The first part of the thesis hypothesis was discussed earlier at the end of Chapter 5. The second part of the hypothesis will now be discussed here.

The preceding three chapters have shown that novel potential field descent heuristics *can* be designed - but consider Table 10.1. Is it reasonable to say that Forward Chaining allows success in an improved range of scenarios and that it addresses the weaknesses of other techniques?

10.4.1 Forward Chaining vs. Navigational Function Techniques

Compared with techniques that generate potential fields lacking local minima, the primary advantage of Forward Chaining is that it allows online navigation, and neither discrete nor continuous navigational fields are suited for this task due to the amount of computation required. With further work, Forward Chaining should also be able to provide significantly computationally cheaper navigation in 3-D.

Discrete navigational field approaches are limited by environment size and resolution but not by obstacle complexity; whereas continuous navigational field approaches are limited in terms of obstacle complexity and total number of obstacles. Forward Chaining suffers neither set of drawbacks.

Unfortunately, there will certainly exist some static 2-D environments of intermediate complexity that navigational field techniques will be able to overcome, but which Forward Chaining will fail upon - for example, any environments containing pathological problems. Forward Chaining therefore cannot be said to be superior to navigational field approaches in all regards - it is still a heuristic approach and not a complete planner.

10.4.2 Forward Chaining vs. Continuous Gradient Descent Techniques

Compared with continuous gradient descent (whether by itself, or combined either with minimum avoidance, or minimum detection and escape techniques), Forward Chaining offers the same benefits of online navigation, an easy-to-understand theoretical model and straightforward implementation, as well as robust, relatively smooth trajectories.

On top of this, however, Forward Chaining also offers the capability for success in fields containing any of the problematic surface features that can exist for potential fields. Clearly, in this regard Forward Chaining is a significant improvement over the other techniques in terms of the range of navigational environments it is capable of addressing.

This comes at the price of computational expense. Forward Chaining is certainly affordable in 2-D and 3-D, but in n-D, even with the cost-reduction techniques outlined in Section 8.2.1, the technique is certainly not as practical as standard gradient descent, though if implemented with the FWDS2 heuristic, it should prove considerably more effective at n-D navigation than gradient descent approaches.

Using the gradient as a rough estimate of how the potential field is going to change locally will always be computationally cheaper than actually finding out for certain how the field changes. It is perhaps possible that there will be some way to approximate the choice of position that will be selected by LPCIRCLE, using relatively few samples of the gradient or potential in some manner, or by using mathematical analysis to characterise the local behaviour of the surface. Were this to be the case, the remaining weakness of Forward Chaining, relative to continuous gradient descent techniques, could be addressed and the cost in n-D mitigated.

In the meantime, Forward Chaining vastly improves the range of 2-D and 3-D problems that can be solved.

10.4.3 Forward Chaining vs. Discrete Gradient Descent Techniques

Forward Chaining outperforms these techniques in terms of the smoothness of paths generated and in terms of computational cost by merit of the fact that it uses gradient descent to make most of the movement over the field, and in that it escapes from

minima in an effective, directed manner. In cases where large concave shapes of high total curvature exist in the environment, or where obstacle circumnavigation is necessary, Forward Chaining should prove much more successful than either RPP or minimum filling techniques.

Forward Chaining is certainly not so successful in 2-D environments as minimum-filling approaches, which can be complete, given sufficient time. Instead, the improvement is qualitative - the paths generated by Forward Chaining are more direct and efficient in reaching the goal. Forward Chaining should also prove more effective in 3-D (and with further work) in n-D, where minimum filling rapidly becomes prohibitively expensive.

10.4.4 Conclusion

As Table 10.1 shows, Forward Chaining overcomes some weaknesses relative to all of the competing techniques, but equally, in almost every case (except discrete gradient descent and elliptical potentials) there remain some navigational situations in which an existing technique is able to achieve navigation where Forward Chaining cannot - albeit, either at huge computational expense or by allowing an extremely high chance of failure. The situations can either be addressed through further work, or accepted, with the recognition that Forward Chaining will work well in most environments.

Addressing the pathological navigation problems for Forward Chaining, and developing a more affordable form of n-D navigation will go a long way towards ensuring that Forward Chaining addresses every weakness of all of the competing approaches. Nonetheless, it can be said that at present Forward Chaining overcomes many of the weaknesses of competitors, and in all cases should be able to navigate effectively in situations where competitors cannot.

10.5 Summary of novel research contributions

Major contributions

- PRIMARY CONTRIBUTION: A novel, straightforward and purely potential field based technique that allows navigation in environments containing concave obstacle

configurations (Chapters 6,7).

- Introduces the novel LPCIRCLE and FWDS1 general potential field descent techniques (Chapter 6).
- Proposes techniques for allowing the approach to operate in 3-D environments (Section 8.1).
- Proposes techniques for allowing the approach to operate in n-D environments, including a novel potential field descent technique (Section 8.2) (with further improvements in the 3-D case (Section 8.2.2)).
- Proposes techniques by which two categories of existing alternative navigation approaches can be improved via augmentation with this research (Section 9.1).

Minor contributions

- Context chapters and a literature survey that show the research context of the thesis. The survey makes a contribution towards highlighting areas of mutual interest between the real-world and virtual-world navigation research communities. It also gives historical overviews of navigation for robotic agents, cinematic agents and computer game agents (Chapters 2, 3, 4).
- Identifies a family of simple, practical, and effective novel potential field based subgoal selection heuristics, and discusses the design and evolution of this family of heuristics (Chapter 6).
- Provides pseudocode and an implementation that demonstrates the novel heuristics (Chapter 6).
- Provides experimental results as evidence of the capabilities of the heuristics (Chapter 7).
- Identifies classes of pathological problems that may still exist, and suggests solutions (Appendix E).
- Discusses how the research may be re-applied into areas of research unrelated to navigation, such as neural networks and automatic system configuration (Section 9.2).

- Provides suggestions for future avenues of research into moving obstacles, multiple-agent and non-holonomic constraint scenarios (Section 10.1).

10.6 Possibilities for future implementations

- It is hoped that some of the techniques discussed herein for 3-D and n-D Forward Chaining based navigation will be implemented and tested.
- It is hoped that LPCIRCLE and FWDS1 will be implemented as general potential fields methods in some fields of research.
- It is hoped that symmetry-breaking heuristic techniques discussed herein will be useful as the basis for overcoming pathological problems for navigation.
- It would be good to see actual implementations of Forward Chaining navigation in computer games, robotics and cinematic agents.
- It would be interesting to see the performance of implementations of Forward Chaining and LPCIRCLE when re-applied to navigational potential fields and hybrid navigation as suggested in Section 9.1.

10.7 Avenues for future research

- Developing the 3-D and n-D techniques, particularly to cope with more complex obstacle configuration than bowl-shapes.
- Research into the enhancement of Forward Chaining to cope with non-holonomic constraints, moving obstacles, and multi-agent scenarios.

10.8 Finally

Forward Chaining is a novel technique for potential field based navigation. While not perfect, the tested forms of the technique represent a practical and novel improvement over existing potential field techniques.

The technique demonstrates that it is possible to provide a navigation technique that can overcome a number of surface problems including the local minimum problem. The basic technique, in combination with the novel research ideas provided towards the end of this thesis, represents the first few steps into several new avenues of enquiry in potential field based navigation.

This research has confirmed the thesis hypothesis, demonstrating that the potential fields method was prematurely abandoned by showing that the potential fields technique can be a computationally cheap yet effective and flexible approach to navigation. It has been demonstrated that it is possible to improve upon existing techniques and allow navigation in a far greater number of different cases.

Though Forward Chaining is not a complete navigational solution, this heuristic has improved the state of the art by offering an entirely new type of potential field approach to navigation that is one of the most interesting in terms of being: easy to understand, easy to implement, cheap to run, easy to extend, appropriate for problems in 2-D (and hopefully 3-D and n-D), real-time, online, suitable for application across a wide range of domains, and possessing a good success rate in environments of moderate to high complexity and unlimited size due to reliably overcoming the most common types of obstacle configurations.

As virtual world environments in computer games, simulations and cinematic productions improve their physical realism and more closely approximate the real world, the interplay between robotic agent navigation and virtual world navigation techniques can only increase. It is hoped that future research will build on the work in this thesis to provide further improved potential field techniques for agent navigation in virtual and physical environments.

Bibliography

[1] Personal communication with Phil Rutherford of Kuju Games, July 2005. See Appendix B.

[2] Personal communication: meeting with Dr. M Livesey, April 2005, St Andrews.

[3] Personal communication: meeting with Dr. M Weir, Nov 2003, St. Andrews.

[4] Personal communication: meeting with Dr. M Weir and Dr. J Lewis, Dec 2003, St. Andrews.

[5] AGEAI White paper: Physics, Gameplay, and the Physics Processing Unit (2005). http://www.ageia.com/pdf/wp_2005_3_physics_gameplay.pdf.

[6] AI Roundtable session of the 2001 Game Developer's Conference (2001). http://www.gameai.com/cgdc01notes.html.

[7] AIBO Global Link. http://www.sony.net/Products/aibo.

[8] Collection of sources on implementing A* based navigation in games. http://www-cs-students.stanford.edu/~amitp/gameprog.html http://www.gamasutra.com/features/19970801/pathfinding.htm http://www.policyalmanac.org/games/aStarTutorial.htm http://www.geocities.com/jheyesjones/astar.html http://en.wikipedia.org/wiki/A-star_search_algorithm.

[9] Examples of adverts and reviews for Prisoner of War. http://darkstation.com/previews/article131.html http://www.gaming-age.com/news/2001/7/6-35.

[10] Game AI: The state of the industry, part 1, by Steve Woodcock (2000).
http://www.gamasutra.com/features/20001101/woodcock_01.htm.

[11] Game AI: The state of the industry, part 2, by Dave Pottinger (2000).
http://www.gamasutra.com/features/20001108/laird_02.htm.

[12] General information: Command and Conquer game family.
http://en.wikipedia.org/wiki/Command_and_conquer
http://en.wikipedia.org/wiki/Red_Alert_%28computer_game%29.

[13] General information: Warcraft game family.
http://en.wikipedia.org/wiki/Warcraft
http://en.wikipedia.org/wiki/Warcraft_Universe#Strategy_Games.

[14] Guide to First Person Shooters.
http://en.wikipedia.org/wiki/First-person_shooter.

[15] Guide to the Real Time Strategy game genre.
http://en.wikipedia.org/wiki/Real-time_strategy.

[16] IGE: Online Game Currency Trade. A currency broker for transactions between real world currencies and online game virtual currencies.
http://www.ige.com/.

[17] Information on the historical origin of waypoints in computer games.
http://www.everything2.com/index.pl?node_id=17414
http://archive.gamespy.com/articles/july01/top505/.

[18] Mars Exploration Rover Mission, Jet Propulsion Laboratory, California Institute of Technology, 2005. http://marsrovers.jpl.nasa.gov/home/.

[19] MMORPGs - general information.
http://en.wikipedia.org/wiki/MMORPG
http://en.wikipedia.org/wiki/Massively_multiplayer_online_game.

[20] Pac-Man ghost movement.
http://www.everything2.com/index.pl?node_id=499157.

[21] Pac-Man patterns.
http://www.hanaho.com/pacman/
http://www.mameworld.net/pacman/patterns.html.

[22] Player exploitation of navigation problems within the Eve Online MMORPG.
http://www.eve-i.com/forum/showflat.php?Cat=&Board=bug1&Number=204071&p%age=0&view=collapsed&sb=5&o=&fpart=1&vc=1&PHPSESSID=.

[23] RoboCup Bibliography. Around 100 research papers relating to Robocup.
http://tralvex.com/pub/robobib/.

[24] Robocup Official Site.
http://www.robocup.org.

[25] SoftImage Behaviour for XSI.
http://www.softimage.com/products/behavior/v2/.

[26] The history of Pac-Man.
http://www.mameworld.net/pacman/history/p1_01.htm
http://en.wikipedia.org/wiki/Pac-Man.

[27] Universities and AIBO, 2005.
http://www.aibo-europe.com/5_1_casestudies.asp.

[28] Empire (game). Publisher: Walter Bright, 1977.
http://en.wikipedia.org/wiki/Empire_%28computer_game%29
http://www.classicempire.com/.

[29] Puckman (game). Publisher: Namco, 1979.
http://en.wikipedia.org/wiki/Puckman
http://www.klov.com/game_detail.php?letter=P&game_id=9149
http://www.puckman.net/main.html.

[30] Rip-Off (game). Publisher: Cinematronics, 1980.
http://www.klov.com/R/Rip_Off.html
http://coinop.org/g.aspx/100148/Rip_Off.html.

[31] The Ancient Art of War (game). Publisher: Broderbund, 1984.
http://en.wikipedia.org/wiki/Ancient_Art_of_War
http://j.lahmy.tripod.com/aaow/subp01.htm.

[32] Arnhem (game). Publisher: CCS Games, 1985.
http://www.the-underdogs.org/game.php?id=70.

[33] Nether Earth (game). Publisher: Argus, 1987.
http://www.braingames.getput.com/nether/
http://www.goodolddays.net/zx_spectrum/n/netherearth/.

[34] Herzog Zwei (game). Publisher: Technosoft, 1989.
http://en.wikipedia.org/wiki/Herzog_Zwei.

[35] Civilization (game). Publisher: Microprose, 1991.
http://en.wikipedia.org/wiki/Civilization_%28computer_game%29
Also, interview with Sid Meier:
http://www.mindjack.com/interviews/sidmeier.html.

[36] Batman Returns (film), 1992.
http://www.imdb.com/title/tt0103776/.

[37] Dune 2 (game). Publisher: Westwood Studios, 1992.
http://en.wikipedia.org/wiki/Dune_2.

[38] The Lion King (film), 1994.
http://www.imdb.com/title/tt0110357/.

[39] Star Trek Voyager: Elogium (TV), 1995.
http://www.imdb.com/title/tt0112178/.

[40] Total Annihilation (game). Publisher: Cavedog Entertainment, 1997.
http://en.wikipedia.org/wiki/Total_Annihilation.

[41] Antz (film), 1998.
http://www.imdb.com/title/tt0120587/.

[42] Homeworld (game). Publisher: Sierra Entertainment, 1999.
http://en.wikipedia.org/wiki/Homeworld.

[43] Massive Software, 2001.
http://www.massivesoftware.com/
http://www.massivesoftware.com/what_massive.html.

[44] Prisoner of War (game). Publisher: Codemasters, 2001.
http://www.codemasters.co.uk/pow/front.htm.

[45] The Lord of the Rings: The Fellowship of the Ring (film), 2001.
http://www.imdb.com/title/tt0120737/.

[46] The One (film), 2001.
http://www.imdb.com/title/tt0267804/
Details of computer controlled agents at:
http://www.softimage.com/products/behavior/v2/Case_Studies/case02/.

[47] America's Army (game). Publisher: U.S. Army, 2002.
http://en.wikipedia.org/wiki/America's_Army
http://www.americasarmy.com/.

[48] The Lord of the Rings: The Two Towers (film), 2002.
http://www.imdb.com/title/tt0167261/.

[49] ELSPA White Paper: Computer and Video Games: A British Phenomenon around the world, August 2003. Also available as:
http://www.elspa.com/about/pr/elspawhitepaper1.pdf.

[50] Freelancer (game). Publisher: Microsoft Games, 2003.
http://en.wikipedia.org/wiki/Freelancer.

[51] Homeworld 2 (game). Publisher: Sierra Entertainment, 2003.
http://en.wikipedia.org/wiki/Homeworld_2.

[52] The Lord of the Rings: The Return of the King (film), 2003.
http://www.imdb.com/title/tt0167260/.

[53] Eve Online (game). Publisher: Simon and Schuster Interactive, 2003-2005.
http://www.eve-online.com/
http://en.wikipedia.org/wiki/Eve_online.

[54] Battle for Middle Earth (game). Publisher: EA Games, 2004.
http://www.eagames.com/official/lordoftherings/thebattleformiddleearth/%us/home.jsp.

[55] Endorphin 2.0 Dynamic Motion Synthesis, 2004.
http://www.naturalmotion.com/files/endorphin_datasheet_v2_0_Feb05.pdf.

[56] IMDB Box Office Records for the Lord of the Rings, 2004.
http://http://www.imdb.com/title/tt0120737/business,

http://www.imdb.com/title/tt0167261/business,
http://www.imdb.com/title/tt0167260/business.

[57] Rome Total War (game). Publisher: Activision, 2004.
http://www.totalwar.com/community/rome.htm.

[58] Troy (film), 2004.
http://www.imdb.com/title/tt0332452/.

[59] UNECE 2004 World Robotics Survey, October 2004.
Summary available at
http://www.unece.org/press/pr2004/04stat_p01e.pdf.

[60] K. Ainsworth. Retrogamer: The magazine of classic videogames. 9:9–14.

[61] S. Amkraut, M. Girard, and G. Karl. Motion studies for a work in progress entitled eurythmy. In *SIGGRAPH Video Review. Issue 21 (second item, time code 3:58 to 7:35).*, 1985.

[62] E.F. Anderson. Playing smart - artificial intelligence in computer games. In *Proceedings of ZFXCON03, Conference on Game Development*, 2003.

[63] J.R. Andrews and N. Hogan. Impedance control as a framework for implementing obstacle avoidance in a manipulator. In *Proceedings of AMSE Winter Meeting 1983, Boston, MA*, pages 243–251, 1983.

[64] Stephen Baert. Motion planning using potential fields. 2000.
http://www.gamedev.net/reference/articles/article1125.asp.

[65] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. Technical Report Report No. STAN-CS-89-1285, Department of Computer Science, Stanford University, 1989.

[66] J. Barraquand, B. Langlois, and J.C. Latombe. Robot motion planning with many degrees of freedom and dynamic constraints. In *Preprints of the Fifth International Symposium on Robotics Research, Tokyo*, pages 74–83, 1989.

[67] J. Barraquand, B. Langlois, and J.C. Latombe. A monte-carlo algorithm for path-planning with many degrees of freedom. In *Proceedings of the IEEE International Conference of Robotics and Automation*, pages 77–103, 1990.

[68] J. Barraquand, B. Langlois, and J.C. Latombe. Numerical potential field techniques for robot path planning. *IEEE Transactions on Systems Man and Cybernetics 22*, 2:224–241, 1992.

[69] G. Bell and M. Livesey. The existence of local minima in local-minimum-free potential surfaces. In *Proceedings of TAROS 2005 (Towards Autonomous Robotic Systems), Imperial College, London, UK*, pages 15–20, 2005.

[70] G. Bell and M. Weir. Agent navigation using potential fields and forward chaining. In *EPSRC PREP 2004 Conference Proceedings, University of Hertfordshire*, pages 74–75, 2004.

[71] G. Bell and M. Weir. Forward chaining for robot and agent navigation using potential fields. In *ACM International Conference Proceeding Series, Proceedings of the Twenty-Seventh Australasian Computer Science Conference (ACSC2004)*, pages 265–274, 2004.

[72] C.M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995. ISBN 0198538642.

[73] J. Borenstein and Y. Koren. Real-time obstacle avoidance for fast mobile robots. *IEEE Transactions on Robotics and Automation*, 19(5):1179–1187, 1989.

[74] D. Bourg and G. Seemann. *AI for Game Developers*. O'Reilly, 2004. ISBN 0596005555.

[75] D. Bourg and G. Seemann. *AI for Game Developers*, pages 126–148. In [74].

[76] E. Bouvier and P. Guilloteau. Crowd simulation in immersive space management. In *Proceedings of the Eurographics Workshop on Virtual Environments and Scientific Visualisation '96*, pages 104–110. Springer-Verlag, 1996.

[77] R.A. Brooks. Solving the find-path problem by good representation of free space. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-13(3):190–197, 1983.

[78] M. Buro and T. Furtak. Rts games as test-bed for real-time research (invited paper). In *Workshop on Game AI, JCIS 2003*, pages 481–484.

[79] S. Caselli, M. Reggiani, and R. Sbravati. Parallel path planning with multiple evasion strategies. In *Proceedings of the 2002 IEEE International Conference on Robotics and Automation ICRA 2002*, pages 260–266, 2002.

[80] J. Champagne and W. Tang. Real-time simulation of crowds using voronoi diagrams. In *EG UK Theory and Practice of Computer Graphics (2005)*, pages 195–201. The Eurographics Association, 2005.

[81] R. Chatila. Path planning and environment learning in a mobile robot system. In *Proceedings of the 11th Annual ACM Symposium on Theory of Computing*, pages 38–48, 1982.

[82] E. Cheung and V. Lumelsky. Motion planning for a whole-sensitive robot arm manipulator. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 344–349, 1990.

[83] D.C. Conner, A. Rizzi, and H. Choset. Composition of local potential functions for global robot control and navigation. In *Proceedings of 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, volume 4, pages 3546–3551, 2003.

[84] C. Connolly, J. Burns, and R. Weiss. Path planning using laplace's equation. In *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, pages 2102–2106, 1990.

[85] C. Connolly and R. Grupen. Applications of harmonic functions to robotics. Technical Report UM-CS-1992-012, University of Massachusetts at Amherst, 1992.

[86] C. Connolly and R. Grupen. On the applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10(7):931–946, 1993.

[87] C.I. Connolly. Harmonic functions and collision probabilities. *International Journal of Robotics Research*, 16(4):497–507, 1997.

[88] C.I. Connolly and R.A. Grupen. Harmonic control. In *Proceedings of the 1992 International Symposium on Intelligent Control*, 1992.

[89] N.J. Cowan, J.D. Weingarten, and D.E. Koditschek. Visual servoing via navigation functions. *IEEE Transactions on Robotics and Automation*, 18(4):521–533, 2002.

[90] B. Damas, P. Lima, and L. Custdio. A modified potential fields method for robot navigation applied to dribbling in robotic soccer. In *RoboCup 2002: Robot Soccer World Cup VI, Spring Lecture Notes in Computer Science*, volume 2752, pages 65–77, 2002.

[91] B.R. Donald. A search algorithm for motion planning with six degrees of freedom. *Artificial Intelligence*, 31(3):295–353, 1987.

[92] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*. Cambridge University Press, 2000. ISBN 0521568765.

[93] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 132. In [92].

[94] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 140, 141. In [92].

[95] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 135–138. In [92].

[96] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 142–143, 219. In [92].

[97] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 125. In [92].

[98] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 13, 21. In [92].

[99] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 212, 213. In [92].

[100] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 147, 148. In [92].

[101] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 18, 21, 130–131. In [92].

[102] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 88–89. In [92].

[103] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 133. In [92].

[104] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 198–203. In [92].

[105] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 225. In [92].

[106] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 137–138. In [92].

[107] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 136–137. In [92].

[108] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 138–141. In [92].

[109] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 139. In [92].

[110] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, page 140. In [92].

[111] G. Dudek and M. Jenkin. *Computational Principles of Mobile Robotics*, pages 97, 202. In [92].

[112] R. Featherstone. *Robot Dynamics Algorithms*. Springer, 1987. ISBN 0898382300.

[113] S.S. Ge and Y.J. Cui. Dynamic motion planning for mobile robots using potential field method. *Autonomous Robots*, 13:207–222, 2002.

[114] D. Gorse, A. Shepherd, and J.G. Taylor. The new ERA in Supervised Learning. *Neural Network World*, 5:503–510, 1993.

[115] D. Gorse, A.J. Shepherd, and J.G. Taylor. The New ERA in Supervised Learning. *Neural Networks Volume 10, No. 2*, pages 343–352, 1995.

[116] Y.K. Hwang and H. Ahuja. A potential field approach to path planning. In *IEEE Transactions on Robotics and Automation*, volume 8(1), pages 23–32, 1992.

[117] F. Janabi-Sharifi and D. Vinke. Robot path planning by integration of artificial potential field approach with simulated annealing. In *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Le Touquet, France*, 1993.

[118] I. Kamon and E. Rivlin. Sensory-based motion planning with global proofs. *IEEE Transactions on Robotics and Automation*, 13 (6):815–822, 1997.

[119] I. Kamon, E. Rivlin, and E. Rimon. A new range-sensor based globally convergent navigation algorithm for mobile robots. In *Proceedings of the IEEE Conference on Robotics and Automation*, pages 429–435, 1996.

[120] A. Kamphuis and M.H. Overmars. Finding paths for coherent groups using clearance. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3815–3822, 2004.

[121] L. Kavraki, M. Kolountzakis, and J. Latombe. Analysis of probabilistic roadmaps for path planning. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 3020–3025, 1996.

[122] L. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration space. *IEEE Transactions on Robotics and Automation*, 12(4):566–580, 1996.

[123] O. Khatib. *Commande dynamique dans l'espace operationnel des robots manipulateurs en presence d'obstacles*. PhD thesis, L'Ecole Nationale Superieure de l'Aeronautique et de l'Espace, 1979.

[124] O. Khatib. Dynamic control of manipulators in operational space. In *Proceedings of the Sixth IFToMM Congress on the Theory of Machines and Mechanisms, New Delhi, India*, pages 1128–1131, 1983.

[125] O. Khatib. Real-time control of manipulators in operational space. In *Proceedings of the 28th Annual Stanford Conference of the American Society for Quality Control, Palo Alto, California*, pages 9/1–9/7, 1984.

[126] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robotics. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 500–505, 1985.

[127] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robotics. *International Journal of Robotics Research*, 5(1):90–98, 1986.

[128] O. Khatib and J. Le Maitre. Dynamic control of manipulators operating in a complex environment. In *Proceedings of the RoManSy'78, 3rd CISM-IFToMM Symposium on the Theory and Practice of Robots and Manipulators, Udine, Italy*, pages 267–282. Elsevier, 1978-1979.

[129] P. Khosla and R. Volpe. Superquadric artificial potentials for obstacle avoidance and approach. In *Proceedings of the IEEE International Conference of Robotics and Automation, Philadelphia, PA*, pages 1178–1784.

[130] J.O. Kim and P.K. Khosla. Real-time obstacle avoidance using harmonic potential functions. *IEEE Transactions on Robotics and Automation*, 8(33):338–349, June 1992.

[131] S. Kirkpatrick, C.D. Gelatt, and M.P. Vecchi. Optimization by simulated annealing. *Science*, 220(4598):671–680, 1983.

[132] D. Koditschek. Exact robot navigation by means of potential functions: Some topological considerations. In *Proceedings of the IEEE International Conference on Robotics and Automation*, pages 1–6, March 1987.

[133] D.E. Koditschek. Robot planning and control via potential. *The Robotics Review*, 1:349–367, 1989.

[134] Y. Koren and J. Borenstein. Potential field methods and their inherent limitations for mobile robot navigation. In *Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, California*, pages 1398–1404, 1991.

[135] B. Krogh and C. Thorpe. Integrated path planning and dynamic steering control for autonomous vehicles. In *Proceedings of 1986 IEEE International Conference on Robotics and Automation (ICRA '86)*, pages 1664–1669, 1986.

[136] C. Kubey. *The Winners' Book of Video Games*. Warner Books, 1982. ISBN 0446371157.

[137] S. Lammers. *Programmers at work: interviews with 19 programmers who shaped the computer industry*. Tempus Books of Microsoft Press, Redmond, WA, 1989. ISBN 1556152116.

[138] J.C. Latombe. *Robot Motion Planning*. Kluwer Academic Publishers, Boston, MA., 1991. ISBN 0792391292.

[139] J.C. Latombe. *Robot Motion Planning*, pages 403–451. In [138].

[140] J.C. Latombe. *Robot Motion Planning*, pages 8–10. In [138].

[141] J.C. Latombe. *Robot Motion Planning*, pages 2, 34. In [138].

[142] J.C. Latombe. *Robot Motion Planning*, pages 323–331. In [138].

[143] J.C. Latombe. *Robot Motion Planning*, pages 12–14, 153–199. In [138].

[144] J.C. Latombe. *Robot Motion Planning*, pages 200–247. In [138].

[145] J.C. Latombe. *Robot Motion Planning*, pages 14–18, 248–294. In [138].

[146] J.C. Latombe. *Robot Motion Planning*, pages 319–334. In [138].

[147] J.C. Latombe. *Robot Motion Planning*, pages 384–402. In [138].

[148] J.C. Latombe. *Robot Motion Planning*, pages 452–532. In [138].

[149] J.C. Latombe. *Robot Motion Planning*, pages 21–25, 356–402, 533–586. In [138].

[150] J.C. Latombe. *Robot Motion Planning*, pages 12–13, 171–174, 197–198. In [138].

[151] J.C. Latombe. *Robot Motion Planning*, pages 155–169. In [138].

[152] J.C. Latombe. *Robot Motion Planning*, pages 176–191. In [138].

[153] J.C. Latombe. *Robot Motion Planning*, pages 191–197. In [138].

[154] J.C. Latombe. *Robot Motion Planning*, page 192. In [138].

[155] J.C. Latombe. *Robot Motion Planning*, page 21. In [138].

[156] J.C. Latombe. *Robot Motion Planning*, pages 14–18, 200–294. In [138].

[157] J.C. Latombe. *Robot Motion Planning*, pages 46–47. In [138].

[158] J.C. Latombe. *Robot Motion Planning*, pages 318, 319. In [138].

[159] J.C. Latombe. *Robot Motion Planning*, pages 19–21, 295–355. In [138].

[160] J.C. Latombe. *Robot Motion Planning*, pages 50–54. In [138].

[161] J.C. Latombe. *Robot Motion Planning*, pages 19, 295, 296. In [138].

[162] J.C. Latombe. *Robot Motion Planning*, pages 298–303. In [138].

[163] J.C. Latombe. *Robot Motion Planning*, page 313. In [138].

[164] J.C. Latombe. *Robot Motion Planning*, pages 335–340. In [138].

[165] J.C. Latombe. *Robot Motion Planning*, page 340. In [138].

[166] J.C. Latombe. *Robot Motion Planning*, pages 313–316, 342–343. In [138].

[167] J.C. Latombe. *Robot Motion Planning*, pages 342–350, 398–399. In [138].

[168] J.C. Latombe. *Robot Motion Planning*, pages 377, 390–405. In [138].

[169] T. Laue and T. Rofer. A behavior architecture for autonomous mobile robots based on potential fields. In *8th International workshop on robocup (Robot World Cup Soccer Games and Conferences), Lecture Notes in Artificial Intelligence*. Springer, 2004.

[170] J.P. Laumond. Feasible trajectories for mobile robots with kinematic and environment constraints. In *Preprints of the International Conference on Intelligent Autonomous Systems*, pages 346–354, 1986.

[171] J. Lewis. *Using Subgoal Chaining to Address the Local Minimum Problem*. PhD thesis, School of Computer Science, University of St. Andrews, Sep 2001.

[172] J. Lewis. *Using Subgoal Chaining to Address the Local Minimum Problem*, pages 99–102. In [171].

[173] J. P. Lewis and M. K. Weir. Using subgoal chaining to address the local minimum problem. In *Proceedings of Second International ICSC Symposium on Neural Computation*, 2000.

[174] S. Loizou and K. Kyriakopoulos. Closed loop navigation for multiple holonomic vehicles. In *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2861–2866, 2002.

[175] T. Lozano-Perez. An algorithm for planning collision-free paths among polyhedral obstacles. *Communications of the ACM*, 22(10):560–570, 1979.

[176] T. Lozano-Perez. Automatic planning of manipulator transfer movements. *IEEE Transactions on Systems, Man and Cybernetics*, SMC-11(10):681–698, 1981.

[177] V. Lumelsky, S. Mukhopadhyay, and K. Sun. Dynamic path planning in sensor-based terrain acquisition. *IEEE Transactions on Robotics and Automation*, 6(4):462–472, 1990.

[178] V. Lumelsky and A. Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica, 2(4):403-440*, 1987.

[179] I. Mantegh, M.R.M. Jenkin, and A.A. Goldenberg. Reformulating the potential field method for goal-attaining, real-time path planning. In *Proceedings of the 3rd ECPD International Conference on Advanced Robotics, Intelligent Automation and Active Systems, Bremen, Germany*, pages 132–136, 1997.

[180] M. Mataric. Environmental learning using a distributed representation. In *IEEE International Conference on Robotics and Automation*, volume 1, pages 402–406, 1990.

[181] M. Mateas. Expressive AI: Games and artificial intelligence. In *Level Up Conference Proceedings*, pages CD–ROM, Vancouver, January 2005. University of Vancouver.

[182] H. Metropolis, A.W. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[183] F. Miyazaki and S. Arimoto. Sensory feedback based on the artificial potential for robots. In *Proceedings 9th IFAC, Budapest, Hungary*, 1984.

[184] M. Mohan, D. Busquets, R. L'opez de M'antaras, and C. Sierra. Integrating a potential field based pilot into a multi-agent navigation architecture for autonomous robots. In *Proceedings of the 1st International Conference on Informatics in Control, Automation and Robotics*, pages 287–290, 2004.

[185] R.R. Murphy. *An Introduction to AI Robotics*, pages 365–367. In [189].

[186] R.R. Murphy. *An Introduction to AI Robotics*, page 326. In [189].

[187] R.R. Murphy. *An Introduction to AI Robotics*, pages 357–358. In [189].

[188] R.R. Murphy. *An Introduction to AI Robotics*, pages 122–148. In [189].

[189] R.R. Murphy. *An Introduction to AI Robotics*. MIT Press, 2000. ISBN 0262133830.

[190] R.M. Murray. *Mathematical Introduction to Robotic Manipulation*, pages 97–114. In [191].

[191] R.M. Murray. *Mathematical Introduction to Robotic Manipulation*. CRC Press, 1994. ISBN 0849379814.

[192] A. Nareyek. Artificial intelligence in computer games: State of the art and future directions. *ACM Queue*, 1(10):58–65, 2004.

[193] N. Nilsson. A mobile automaton: An application of artificial intelligence techniques. In *Proceedings of the 1st International Joint Conference of Artificial Intelligence*, pages 509–520, 1969.

[194] B. Noble and J. Daniel. *Applied Linear Algebra*. Prentice Hall, 1988. ISBN 0130412600.

[195] C. O'Dunlaing and C.K. Yap. Deadlock-free and collision-free coordination of two robot manipulators. *Journal of Algorithms*, 6:104–111, 1982.

[196] A. Patel. Movement costs for pathfinders, 2003. `http://theory.stanford.edu/~amitp/GameProgramming/MovementCosts.html`.

[197] V.V. Pavlov and A.N. Voronin. The method of potential functions for coding constraints of the external space in an intelligent mobile robot. *Soviet Automatic Control*, 17(6):45–51, 1984.

[198] B. Reese and B. Stout. Finding a pathfinder. In *Proceedings of the AAAI 99 Spring Symposium on Artificial Intelligence and Computer Games*, pages 69–72, 1999.

[199] J. Reif. Complexity of the mover's problem and generalisations. In *Proceedings of the 20th IEEE Symposium on Foundations of Computer Science*, pages 421–427, 1979.

[200] J. Reif and H. Wang. Social potential fields: A distributed behavioral control for autonomous robots. In *International Workshop on Algorithmic Foundations of Robotics (WAFR)*, pages 431–459, 1995.

[201] J. Reif and H. Wang. Social potential fields; a distributed behaviour control for autonomous robots. *Robotics and Autonomous Systems*, 27(3):171–194, 1999.

[202] C. Reynolds. Boids (Flocks, Herds and Schools: a Distributed Behaviour Model). Excellent resource on boids and agent flocking heuristics. `http://www.red3d.com/cwr/boids/`.

[203] C. Reynolds. Stanley and Stella in: Breaking the ice. Shown at SIGGRAPH 1987. 1987.

[204] C. Reynolds. Reactive autonomous characters, 2000. Presented in Distinguished Lecture Series on Computer Games, University of Michigan's Department of

Electrical Engineering and Computer Science.
http://www.research.scea.com/research/pdfs/CWR2000_10_20_UMich.pdf.

[205] C.W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. In *Computer Graphics 21(4), (SIGGRAPH '87 Conference Proceedings)*, pages 25–34, 1987.

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

[207] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, first edition, 1995. ISBN 0131038052.

[208] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 805–807. In [207].

[209] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, page 801. In [207].

[210] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 799–800. In [207].

[211] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 800–802. In [207].

[212] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 796–799. In [207].

[213] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 806–808. In [207].

[214] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 917–918. In [217].

[215] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 110–114. In [217].

[216] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*, pages 114–115. In [217].

[217] S. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Prentice Hall, second edition, 2003. ISBN 0130803022.

[218] S.J. Rymill and N.A. Dodgson. A psychologically-based simulation of human behaviour. In *EG UK Theory and Practice of Computer Graphics (2005)*, pages 35–42. The Eurographics Association, 2005.

[219] A.I. Sage. *Observation-Driven Configuration of Complex Software Systems*. PhD thesis, University of St Andrews, 2003.

[220] J. Schwartz, M. Scharir, and J. Hopcroft. *Planning, Geometry and the Complexity of Robot Motion*. Ablex Publishing Corporation, Norwood, NJ, 1986. ISBN 0893913618.

[221] S. Shimoda, Y. Kuroda, and K. Iagnemma. Potential field navigation of high speed unmanned ground vehicles on uneven terrain. In *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, pages 2839–2844, 2005.

[222] Tim Skelly. Rip-Off: A description of Programmed Behaviour in a Video Game. 2001. Note: This is a rather unique recent discussion of game AI by the author of one of the first games to feature AI navigation.
`http://www.red3d.com/cwr/boids/RipOff_Flocking.html`.

[223] K. Souccar and R. Grupen. Distributed motion control for multiple robotic manipulators. In *Proceedings of the 1996 IEEE Conference on Robotics and Automation*, pages 3434–3439, 1996.

[224] G.K. Still. *Crowd Dynamics*. PhD thesis, Warwick University, 2002.

[225] K. Sun and V. Lumelsky. Motion planning with uncertainty for a 3d cartesian robot arm. In *Preprints of the Fifth International Symposium on Robotics Research, Tokyo*, pages 57–64, 1989.

[226] H. Tanner and A. Kumar. Towards decentralization of multi-robot navigation functions. In *Proceedings of 2005 IEEE International Conference on Robotics and Automation, Barcelona, Spain*, pages 4143–4148, 2005.

[227] R.H. Taylor. *Synthesis of Manipulator Control Programs from Task-Level Specifications*. PhD thesis, Department of Computer Science, Stanford University, 1976.

[228] P.A. Thompson and E.W. Marchant. A computer-model for the evacuation of large building population. *Fire Safety Journal*, 24(2):138–148, 1995.

[229] C. Thorpe. Path relaxation: Path planning for a mobile robot. Technical report, Robotics Institute, Carnegie Mellon University, April 1984.

[230] S.L. Tomlinson. The long and short of steering in computer games. *International Journal of Simulation*, 1-2(5):33–46, 2004.

[231] S. Udupa. *Collision Detection and Avoidance in Computer Controlled Manipulators*. PhD thesis, Department of Electrical Engineering, California Institute of Technology, 1977.

[232] I. Ulrich and J. Borenstein. Vfh+: Reliable obstacle avoidance for fast mobile robots. In *Proceedings of the 1998 IEEE International Conference on Robotics and Automation. Leuven, Belgium*, pages 1572–1577, 1998.

[233] D. et al. Varner. UMSC Small unit leader non-lethal trainer. In *Proceedings of the International Training and Education Conference ITEC '98*, 1998.

[234] R. Volpe and P. Khosla. Artificial potentials with elliptical isopotential contours for obstacle avoidance. In *Proceedings of the 26th IEEE Conference on Decision and Control, Los Angeles*, pages 180–185, 1987.

[235] S. Waydo and R.M. Murray. Vehicle motion planning using stream functions. In *Proceedings of the 2003 IEEE International Conference on Robotics and Automation*, pages 2484–2491, 2003.

[236] M. Weir and G. Bell. The incorporation of intentional action into robots. Technical report, School of Computer Science, University of St Andrews, 2001.

[237] M. Weir and A. Fernandes. Tangent hyperplanes and subgoals as a means of controlling direction in goal finding. In *Proceedings of the World Conference on Neural Networks*, volume 3, pages 438–443, 1994.

[238] M.K. Weir, J.P. Lewis, and G. Milligan. Using tangent hyperplanes to direct neural training. In *Proceedings of the 2nd International ICSC Symposium on Neural Computation*, 2000.

[239] J.R. Williams. *A Simulation Environment to Support Training for Large Scale Command and Control Tasks*. PhD thesis, University of Leeds, 1995.

[240] J.S. Zelek. Dynamic path planning. In *IEEE International Conference on Systems, Man and Cybernetics, 'Intelligent Systems for the 21st Century'*, volume 2, pages 1285–1290, 1995.

[241] A. Zelinsky and S. Yuta. A unified approach to planning, sensing and navigation for mobile robots. In *Proceedings of the Third International Symposium on Experimental Robotics, Kyoto, Japan*, 1993.

[242] S. Zilberstein. Resource-bounded sensing and planning in autonomous systems. *Autonomous Robots*, 3(4):31–48, 1996.

[243] X.Y. Zou and J. Zhu. Virtual local target method for avoiding local minimum in potential fields navigation. *Journal of Zhejiang University SCIENCE*, 4(3):264–269, 2003.

Appendix A

Quick guide to the A* algorithm

This appendix gives a very brief guide to the A* algorithm, which is often used to generate navigational paths over graph representations of the navigational environment. Additional information on A* search may be found in [8, 75, 192, 217, 230].

A.1 What is it?

A* search is a graph search algorithm from AI, that is used in navigation to find paths between nodes on graphs such as those produced by some metric navigation approaches (e.g. roadmap, cell decomposition). The algorithm operates by iteratively 'expanding' nodes and checking their neighbouring nodes, starting from the 'start' node, and trying to reach a goal node. To check each node, a value is calculated according to a cost function as given in equation A.1.

$$f(n) = c(n) + h(n) \tag{A.1}$$

$f(n)$ is the cost of a node. It is made up of $c(n)$, the lowest cost to reach node n from the start node by following the edges between visited nodes, and $h(n)$, an estimate of the cost of reaching the goal from node n. It is necessary that the estimate $h(n)$ is provided by an *admissible heuristic* - which means that the heuristic must not estimate a cost that is higher than the actual shortest possible distance to the goal. In other words, the heuristic must

make an optimistic, rather than pessimistic guess about how quickly the agent can reach the goal from the node. One common way of doing this in navigation is to make a 'straight line' estimate of the cost of getting to the goal (i.e. ignoring obstacles) - which is likely to be only somewhat realistic, but certainly optimistic enough to suit the demands of A*.

A* executes as follows:

1. An *open* list and a *closed* list are initialised. The open list starts with the *start* node, and the *closed list* starts off empty.
2. while (goal node has not been added to the open list) {
   (a) Select the node n with the lowest $c(n)$ cost from the *open* list.
   (b) Move node n to the closed list.
   (c) All nodes o that are direct neighbours of the node n that are not in the closed list (i.e. that are unvisited) are treated as follows:
      i. If the node o is not already in the open list, put it in the open list.
      ii. If the node o is in the open list, recalculate its $f(n)$ using the new closed list.

   }

Finally, a path is computed backwards from the goal. This is done by ensuring that when a node is visited, its 'parent' from which it was selected (or from which a lower $f(n)$ was calculated) is stored in the data structure with it.

A.2 Why use it?

A* search is *optimal* and *complete*. It will find the shortest path between a start and goal node, and it is guaranteed to find a solution if one exists. It is also perfectly suited to coping with discrete environments that are represented as grids, as they can be treated directly as graphs. These properties have endeared it to the gaming industry.

Unfortunately, in practice A* is extremely expensive both in terms of computational time and space required. The computational cost of evaluating a path grows exponentially with

the path length (except when the $h(n)$ heuristic is exceptionally accurate[1]) as does the amount of memory used to maintain the open and closed lists. Frequently, A* search can run out of memory during execution when trying to navigate between a distant start and goal position.

The effectiveness of A* can also be limited to a large extent by the $h(n)$ heuristic chosen. A good heuristic will produce results with less computational effort than a bad heuristic. However, A* search is nonetheless *optimally efficient* for any admissible heuristic - that is, it considers the fewest possible nodes of any possible search algorithm, given that the other algorithm does not have access to a more accurate heuristic estimate than $h(n)$.

A.3 Where is it used?

A* search is the main approach used for path planning on graphs corresponding to topological maps (Section 2.5.1), or for long-term path planning on the graphs generated by cell decomposition (Section 2.5.3) or the roadmap method (Section 2.5.2) when carried out on metric maps.

When used with topological maps or roadmaps, A* search is usually computationally cheap and requires little storage, since the graphs are usually small and have low branching rates. When used with cell decomposition approaches, this is often no longer the case. In 2-D environments, it is possible for tens of thousands of nodes and edges to be used, with a branching rate of 4 at each node. In 3-D environments, it is plausible that the size of the state space may be measured in millions of nodes and tens of millions of edges, with a branching rate of 8 at each node, for even a low resolution 100x100x100 grid.

A* search is particularly popular in computer games, where non-optimal variants allowing bounded computational cost and bounded memory versions of the approach are almost exclusively used for navigation. Section 2.6.9 has more information on the use of A* in the gaming industry, as well as [8, 75, 192, 230].

A* is not suited to an online approach to navigation, and it is not well suited to navigation in very complex environments (where the heuristic is likely to perform poorly), or in navigational situations involving non-holonomic constraints or moving obstacles. It is useful though in situations where terrain is part of the problem, as this can be modelled

[1] A more detailed and mathematical description of 'exceptionally accurate' can be found in [217].

in a straightforward manner by associating a cost with each edge, corresponding to the difficulty of the terrain.

Appendix B

Email interview with a game AI designer

This appendix provides excerpts taken from a short email interview with Mr. Phil Rutherford of Kuju Games, the AI designer for 'Prisoner of War' [44], the only computer game explicitly advertising itself as having agent navigation AI based on potential fields.

> GBB: "I have heard that agents in this game use potential field based AI. Is this used in your navigation approach?"
>
> PR: " Hi Graeme, I was the AI programmer on POW. It seems like a very long time ago now!
>
> We did use a form of potential fields for steering, combined with a pre-placed A* pathnet for higher-level navigation. It worked very well in the original prototype, which had little coffin-like shapes for characters. The flocking and queueing all worked as expected and looked impressive; however, when we moved to real characters the system did not work so well, as the characters had far greater restrictions placed on their speed and turn-rate in order to look realistic with the animation system. This often lead to scenarios where the AI character wanted to move a short distance but had nowhere to go and couldn't quite work that out - a problem with PF I found - so would end up spinning round on the spot."
>
> GBB: "Briefly, how are you using it? (e.g. standard gradient descent? continuous or discrete technique?)"
>
> PR: "From memory: The system I used was a simple summation of repulsive

forces from objects to avoid with a fall-off function based on distance, and attractive forces to the target point. These were all floating-point continuous functions - discrete functions caused chaos when added to the system I found. Various additions (bodges) were added later on in the project to cope with nasty scenarios, like the spinning on the spot."

GBB: "Where did you learn about the PF method for navigation? Another games programmer, games literature, robotics literature?"

PR: "I don't actually remember. It may have been in Craig Reynolds' steering literature, which was well published at the time."

The author is grateful to Mr. Rutherford and to Kuju Games for taking part in this short, but interesting interview.

Appendix C

A chronology of potential field navigation

This appendix provides a chronology of significant work and recent work in potential field based navigation.

C.1 Landmark publications

- 1978
  - Khatib publishes a paper with La Maitre [128] suggesting the possibility of a potential function based approach to robotic control.
- 1980
  - Khatib publishes his thesis, "Commande dynamique dans l'espace operational des robots manipulateurs en presence d'obstacles" [123], elaborating further on the material in [128]. This is followed by a series of related papers leading up to 1986. The primary contributions are the idea that the negated gradient of the potential field can be used to control the torque of a robotic effector, along with a closed form mapping of navigation problems into potential field functions.
- 1983

  - Andrews and Hogan suggest the idea of imaginary forces acting on a robotic manipulator in [63].

- 1984

  - Arimoto and Miyazaki of Japan and Pavlov and Voronin of the USSR independently come up with similar results to Khatib [183, 197]. Coverage of the differences in how they came to their approaches in contrast with Khatib can be found in [133].

- 1985

  - Khatib presents a conference paper, "Real-time obstacle avoidance for manipulators and mobile robotics" at the IEEE International Conference on Robotics and Automation [126]. A journal publication of the same material is made the following year.

- 1986

  - Khatib publishes the landmark journal paper "Real time obstacle avoidance for manipulators and mobile robots" [127]. This paper popularised potential field based navigation and represents the beginning of mainstream research in the area.
  - Krogh and Thorpe create the earliest hybrid approach [135] by using potential fields to travel between 'critical points' generated by Thorpe's earlier offline 'path relaxation' path-planning technique [229].

- 1987

  - Koditschek publishes the second major landmark paper in potential fields, "Exact robot navigation by means of potential functions: Some topological considerations" [132]. The paper introduces the navigational potential fields approach and suggests ways in which the approach could be implemented for a small subset of navigation environments (spherical environments containing spherical obstacles).

- 1989

  - Koren and Borenstein present a technique closely related to potential fields - the VFF technique[73, 94] focuses upon the 'virtual force' metaphor for potential fields. The technique is capable of coping with a level of uncertainty about the positions and shapes of obstacles through a grid discretisation of the environment.
  - Koditschek provides coverage of the history and development of research surrounding the potential fields approach till 1989, in [133].
  - Barraquand and Latombe propose techniques for discrete harmonic potential fields [65] and a random-discrete potential field based path planner [66]. These papers are together a landmark in that they represent the most significant work in discrete potential fields research.

- 1990

  - Connolly et al. publish "Path Planning using Laplace's equation" [84], the third major landmark paper, which introduces the idea of generating functions that satisfy Laplace's equation as a way to build a minima-free navigational potential field.

- 1991

  - Koren and Borenstein publish "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation" [134]. As its name suggests, this paper makes strong criticisms of the potential fields approach, which are discussed earlier in Section 4.5.4. At the time, potential fields were still quite popular, and so this paper represents a final landmark, in the sense that is is the start of a shift away from the area after five years of research efforts.
  - Latombe publishes "Robot Motion Planning" [138]. This book summarises potential fields research up until 1991 (as well as research in a number of other areas of navigation) and expands upon the detail of the RPP approach and discrete harmonic potential field approaches that Barraquand and Latombe proposed in 1989.

- 1992

  - Rimon and Koditschek publish a paper that proves that navigational potential fields exist for further types of navigational environments, and demonstrate techniques for generating fields in these cases.

  - Connolly et al. publish five more papers (including [88, 85]) improving their techniques for building approximations of navigational potential fields from harmonic functions.
  - At the same time, Kim and Khosla publish a technique also using closed form functions to build potential fields, using 'panels' as the basic obstacle type as an attempt to simplify the mathematics involved. Some experimental results are given [130].

- 1993

  - Zelinsky and Yuta suggest a hybrid model using A* for global planning with discrete grid-based potential fields for local planning [241]. The global planner operates on a known map, and the local planner deals with any unmapped objects encountered en route. The technique is implemented on a robotic agent.

- 1995

  - Zelek suggests using coarse approximations to harmonic functions combined with large groups of parallel computers [240], as a way to try and make navigational potential fields feasible for online navigation. The technique is computationally expensive though, and only works well in simple environments.
  - Reif introduces the idea of 'social potential fields' for multi-robot behaviour control [200].
  - The boundary integral equation method [179] is proposed by Mantegh to reduce the computational costs of harmonic potential fields, reducing the complexity in some cases to a boundary integral rather than a surface integral.
  - Connolly publishes his last paper on harmonic functions [87].

- 2000

  - Dudek and Jenkin publish the book "Computational Principles of Mobile Robotics" [92]. The book is particularly worth mentioning as it is the only recent text providing an extremely good overview of potential field navigation techniques and research.

C.2 Recent publications

- 2002
  - Koditschek et al. present a technique for the use of potential fields in the application domain of controlling visual servos; i.e. for automated camera control [89].
  - Loizou and Kyriakopoulus suggest ways of representing harmonic potential fields in the case of multiple robots [174], re-addressing the same problem as [223].
  - Caselli et al. present a technique that uses several parallel attempts to escape from a minima in offline path planning [79].
- 2003
  - Conner et al. present a hybrid technique [83] for carrying out navigation using navigational potential fields generated over fragments of the environment. In essence this is really a cell decomposition hybrid with potential fields only for local navigation.
  - Waydo presents a technique using 'stream functions' from hydrodynamics to generate a navigational potential field that offers smoother paths, more suited to the domain of controlling i.e. autonomous aircraft which operate under rigid non-holonomic constraints [235].
  - Zou and Zhu publish a hybrid technique [243] combining a geometrical approach with potential fields to overcome the local minimum problem. It works in environments containing only trapezoid obstacles.
- 2004
  - Bell and Weir publish two papers describing the 'Forward Chaining' approach to navigation using the traditional non-harmonic potential field representation [71, 70].
  - Laue and Rofer publish a multi-robot potential fields hybrid approach to participating in robocup [169].
- 2005

- Bell and Livesey publish a paper highlighting the previously unrecognised problems presented by saddle points for both traditional and navigational approaches to potential fields [69].
- Shimoda et al. publish a paper on using shaped potential fields to bias steering towards trajectories that suit steering limitations [221].

Appendix D

Novelty of LPCIRCLE

This appendix discusses the novelty of the LPCIRCLE descent technique.

D.1 Introduction

The LPCIRCLE technique described in Section 6.5 seems obvious, yet does not seem to have been suggested elsewhere. Possible reasons for this, as well as differences with existing techniques, are discussed in this Appendix. LPCIRCLE is not the primary research contribution of this thesis and the issue of novelty is discussed here purely out of interest.

Clearly, the technique is dissimilar to the standard continuous and discrete forms of gradient descent that are practiced throughout potential fields. Furthermore, the well known gradient estimator techniques that exist in numerical analysis, (which are based on building up the gradient from numerical estimates of the partial derivative in each axis e.g. Central Difference or Intermediate Difference methods[1]) build up an estimate of the gradient using a few local samples of a function, and are intended for use in cases when the gradient of a function is not available directly.

LPCIRCLE differs from these numerical analysis approaches in that it is based on carrying

[1]There is a helpful summary of some common techniques available online at `http://www.cg.tuwien.ac.at/~theussl/DA/node40.html` and at `http://www.aravind.ca/cs788h_Final_Project/gradient_estimators.htm`.

out exhaustive[2] sampling rather than taking only a few samples; in that it improves in accuracy in its task as samples are taken *further* from the current position rather than *closer* to the current position; and in that it does not aim to approximate the gradient, but rather, attempts to improve upon the gradient, for the specific purpose of field descent.

In short, LPCIRCLE is not a gradient estimator, it is a potential-based descent technique inspired by gradient descent but different from the existing forms of continuous and discrete descent. At the limit, it is a continuous technique, but it is more usually a 'nearly-continuous' technique when its complete 'infinite samples' circle sampling approach is approximated, by sampling at a finite number of points approximating a circle.

D.2 Comparison table

- Continuous Gradient Descent
  - Number of samples (n-D) : (1).
  - Distance of samples from point being evaluated: zero.
  - Purpose: Variable or fixed step size descent technique based on evaluation of gradient function.
- Discrete Gradient Descent
  - Number of samples (n-D) : ($2n$).
  - Distance of samples from point being evaluated: non-local.
  - Purpose: Fixed step size descent technique based on evaluation of potential function at grid coordinates.
- Intermediate Difference Method
  - Number of samples (n-D) : ($n + 1$).
  - Distance of samples from point being evaluated: local.
  - Purpose: Popular numerical analysis technique for calculating an approximation to the gradient using the potential function.

[2] At its highest resolution, infinitely exhaustive, even in 2-D.

- Central Difference Method
  - Number of samples (n-D) : ($2n$).
  - Distance of samples from point being evaluated: local.
  - Purpose: Popular numerical analysis technique for calculating an approximation to the gradient using the potential function.
- LPCIRCLE
  - Number of samples (n-D) : (∞). Typical approximation : $2.(18^{n-1})$.
  - Distance of samples from point being evaluated: non-local.
  - Purpose: Fixed step size continuous descent technique (continuous at limit, semi-continuous via approximation) based on exhaustive evaluation of potential values on a circle (hypersphere in n-D).

D.3 Possible reasons for non-discovery?

Some possible reasons why this technique may not have been suggested before:

- Increased computational expense even in 2-D and 3-D, and in theory infinite and thus completely incomputable cost when used at full precision. Typically, users and developers of numerical techniques are keen to minimise computational cost as much as possible.
- Unlike other GD techniques, it rapidly becomes intractable for n-D descent, $n > 3$. Proponents of potential field techniques are usually quite proud of the fact that the techniques scale effortlessly into n-D.
- Potential field researchers in physics, neural networks and so on tend not to think so much about fixed step size forms of gradient descent, as variable step-size descent regimes offer an opportunity to reduce the number of descent steps needed, do not suffer an 'overshoot' problem around the global minimum, and more closely resemble the behaviour of particles in potential fields in the real world. Use of fixed step size gradient descent in potential fields research seems limited mainly to potential field based navigation.

- Continuous gradient descent techniques represent an ‘off the shelf’ technique for researchers working in continuous environments; discrete gradient descent techniques are an ‘off the shelf’ approach for researchers with grid-based representations of the environment. There may have seemed to be little need to try to develop a new approach when a straightforward (though unreliable) technique was immediately to hand in both cases.

D.4 Summary

LPCIRCLE differs from existing standard gradient estimators in numerical analysis and from existing continuous and discrete gradient descent techniques.

The cost of LPCIRCLE varies from high to intractable, and it is not immediately applicable in n-D field descent (though Section 8.2.1 discusses some techniques to remedy this), and some aspects of the nature of descent are unusual within potential fields research (the fixed step size). This makes it unlikely that anyone would have considered using such a technique before now.

The LPCIRCLE technique therefore appears to be novel.

Appendix E

Pathological problems for Forward Chaining

This appendix discusses pathological problems for Forward Chaining, and a range of techniques to overcome them.

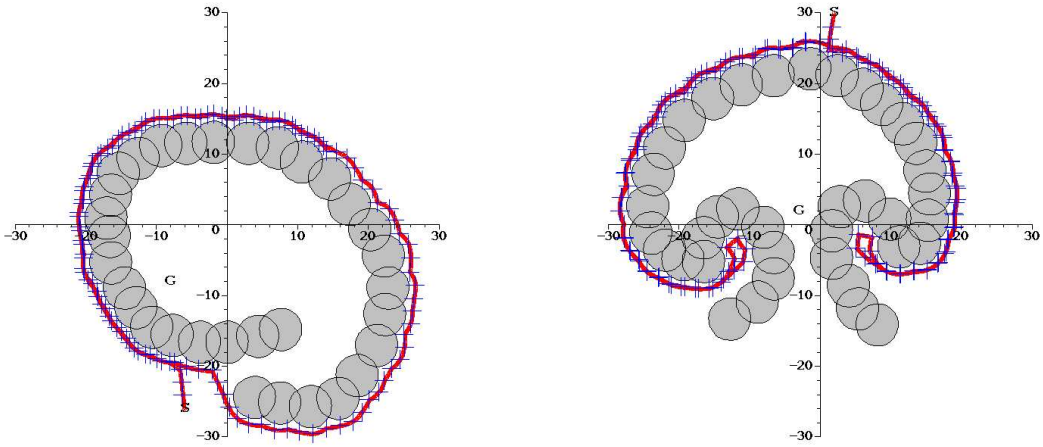
E.1 Introduction

One of the main ideas in this thesis is that the local minima caused by obstacle configurations (or individual obstacles) of concave curvature, which represent the major problem for online potential field based navigation, can be overcome by a purely potential field based heuristic approach. Unfortunately, though very useful, the ability to overcome concave, convex and flat obstacle shapes turns out to be insufficient to overcome all possible problems that potential field based navigation may face. This section will introduce the idea of pathological navigation problems, deliberately designed to frustrate the success of Forward Chaining.

The two main problems that will be discussed are the 'Spiral' problem, and the 'Mushroom' problem. In both cases, the problem hinges on making Forward Chaining's choice of direction lead it eventually back into the same behaviour repeatedly within the same set of obstacle structures.

E.2 The Spiral Problem

In the spiral problem, the agent is trapped in a long term oscillatory cycle that encompasses the goal. An example of this is shown in Figure E.1(a). The spiral problem shown in Figure E.1(a) is a right-handed spiral problem. The mirror-image of the spiral obstacle configuration shown would be a left-handed spiral problem.



(a) The Spiral Problem (b) The Mushroom Problem

Figure E.1: The Mushroom and Spiral Problems.

The problem is not necessarily associated only with a perfect spiral shape, but rather with any shape that causes the problem of rotating around the goal and repeating approximately the same path over and over. The term ‘spiral problem’ is used merely because it was the obstacle configuration explicitly devised in this research to create such behaviour.

E.3 The Mushroom Problem

In the mushroom problem, the agent is trapped between two concave shapes, each of which is designed to make it very likely that gradient descent will exit the problem configuration on the same side it entered. An example obstacle configuration associated with this problem is shown in Figure E.1(b).

Again, this problem is not necessarily associated with perfect mushroom shapes. Any obstacle formation that consists of two problems that reliably capture and reverse the

direction of the agent, is sufficient to trigger this form of long-term oscillatory behaviour in the Forward Chaining heuristic. The etymological origins of the term 'mushroom problem' are the same as that of the spiral problem.

E.4 Addressing the Mushroom and Spiral problems

Perhaps unsurprisingly, the pathological problems hinge on symmetry. In the case of the spiral problem, rotational symmetry is used to force Forward Chaining to repeatedly face the same problem. In the mushroom problem, a form of lateral symmetry is used.

The key to beating this problem therefore lies in finding some way to break this symmetry reliably in such a way that the problem is not re-encountered. Some possibilities for doing this will now be discussed.

E.4.1 Symmetry breaking: Always go left (or right)

Here, a particular direction in 2-D (either left or right) is picked as the strategy for navigation - for the moment, consider that 'left' is chosen. Upon encountering an obstacle in the sampling region, marked by high, probably infinite potential depending on the obstacle representation[1], the agent immediately places its first subgoal directly to the left, then sets a second subgoal based upon this, as per FORWARDS. Essentially, the agent immediately strongly biases travel towards a particular side on the search range when an obstacle is first encountered.

Consequently, on mushroom problems, the agent now succeeds. On spiral problems, the agent fails half of the time depending on whether the spiral is 'left' or 'right' handed, and depending on the direction the agent is biased towards.

As a side note, this approach is quite inelegant, in that it does not restrict itself to incrementally adjusting the tested behaviour of Forward Chaining, but rather, it completely overwrites the existing strategy in all cases where obstacles are encountered, breaking with the philosophy of 'retain the existing successful strategy to overcome less challenging problems, and augment the strategy under conditions associated with a new class of

[1]Notice that unlike the previous subgoal selection techniques, this technique would unfortunately rely on explicit recognition of obstacles - albeit, still through the potential field.

problems' that was adopted in Chapter 6. The fixed left or fixed right approach is also somewhat unnatural in nature - a person or animal that always turned left when they even brushed past a wall might look quite unusual.

E.4.2 Symmetry breaking: Pick a random direction

Here, upon encountering an obstacle in the sampling region, the agent immediately places its first subgoal randomly either to the left or right. The agent can now overcome mushroom or spiral problems, but it is a matter of statistics as to whether the agent will succeed in any reasonable amount of time. The number of mushroom type and spiral type problems in the environment will determine how long the agent takes.

The key problem is that the agent may repeatedly choose an action at random that begins to undo the existing useful progress towards the goal. This is the similar to one of the problems faced by the RPP technique (see Section 5.4.2). The other problems of the previous heuristic are also present.

E.4.3 Symmetry breaking: Learning approaches

One might wonder, "why doesn't the agent simply remember what it did the last time it encountered the obstacle configuration, and then take a different action if it encounters the obstacle again?". There are unfortunately some problems with such an approach.

Navigation is taking place in a continuous space, so it is highly improbable that the agent will reach the surface of the obstacle configuration at exactly the same point twice. This would mean that some kind of generalisation would be needed, i.e. the agent will try to take a different action at point X, if it is within (e.g. half a step / a step / two steps) of a position Y that it has reached before.

- Storing such information for every position ever reached by the agent may well require a prohibitive amount of storage space per agent (if resources are limited due to the agent being implemented as a computationally limited robotic platform, or because it is one of thousands of simulated agents being run in parallel). Similarly, checking each position being currently reached against the set of positions previously reached might be computationally expensive.

- Generally, the idea that 'turning left at position X' and 'turning right at position Y' (where X and Y are different positions, but nearby each other) will necessarily lead to different behaviour in the long term shape of the agent's path is a very dangerous assumption to make blindly. Geographical proximity does not guarantee that the same choice made at each position will have the same effect - a form of 'butterfly effect' may exist.

  For example, it may be possible to imagine byzantine scenarios designed to deliberately exploit the 'learning' heuristic, and thus trick the agent into oscillatory behaviour[2]. It is also possible such problematic situations may occur quite normally in practice in an environment, without the need for human nature to explicitly produce the problem.

- The agent may later approach the same position from a different angle - it is not clear how a learning heuristic can work if a different forwards range is in use each time the position is reached.

- The technique is unlikely to scale well into 3-D, much less n-D, as there would be an infinite number of directions to choose from.

There may be other unforeseen issues involved in this learning and generalising approach. However, this represents the most appealing alternative of the approaches presented so far. The earlier issue of 'breaking from the design ethos of FWDS' is still present in this heuristic.

E.4.4 Symmetry breaking: Design heuristics in the style of FWDS

A series of obstacle configuration problems in Chapter 6 were overcome by using the limited knowledge available from the agent's path and the potential field. It may be possible to extend the heuristic in a similar manner for these new problem scenarios. Two approaches (which may be combined), based on this idea, are presented below.

[2]In the domain of computer games particularly, human players can be relied upon to develop such exploitative scenarios if it is in their interests to do so to beat computer controlled agents.

E.5 Overcoming the Spiral problem with FWDS-SPIRAL

In the spiral problem, it can be observed that the problem faced by FORWARDS is that the useful knowledge - that the agent's path is being twisted - is being undone by the agent's rotation around the goal. While this is a generally useful aspect of FORWARDS in the frequent case of obstacle circumnavigation, it is not such a useful behaviour in the case where the obstacle is being repeatedly circumnavigated.

Therefore, a new condition can be introduced to overcome this problem when it occurs.

"If the agent's absolute path twist reaches 360 degrees, then irrespective of the *path twist relative to the goal*, the agent should immediately try to untwist its path."

The behaviour of the strategy will not change on existing problems that have been covered in Chapter 6, and will also not change on mushroom problems. It will only change on spiral-type problems. The strategy is defined only in terms of the few existing state variables does not require positions in the environment to be generalised, as the learning approach would.

E.6 Overcoming the Mushroom problem with FWDS-MUSHROOM

It is possible to observe that the agent is not getting any closer to the goal each time it re-enters a part of the mushroom problem. This knowledge can be exploited as follows.

1. In FWDS-MUSHROOM, if at any point the agent is sampling potential at a point closer to the goal than it ever has before, the agent will mark the new closest distance to the goal at which potential has been sampled.
2. If
   (a) the agent has untwisted completely (relative goal-path twist has returned to zero) and
   (b) the potential value at one steps distance, in the direction of the goal, suggests an obstacle surface is present, and

(c) the point at which the potential is sampled *is not closer to the goal* than the 'closest to the goal' distance stored in the agent's state.

3. then the agent should immediately 'flip' direction by twisting 180 degrees in the direction it was last twisted. The agent should then try to untwist as per FORWARDS.

4. else the agent should behave as FORWARDS or FWDS-SPIRAL.

E.7 Do nothing?

A final approach is to recognise that the spiral problem and mushroom problem exist, as well as other perhaps other types of pathological problem currently unimagined, and accept that Forward Chaining with FORWARDS is not a *complete* planner, but that it is a *good* planner that should work on many problems in a straightforward and computationally cheap way. In situations where completeness is required, alternative approaches based on this research can be found in Chapter 9.

E.8 Conclusion

The problems represented by the Mushroom and Spiral problems should be overcome by adopting one of the approaches above. These problems should be uncommon and it is feasible to consider using Forward Chaining with the final FORWARDS subgoal selection heuristic despite its potential for failure in the scenarios outlined above.

Appendix F

Examples from the Bigmap experiments

This appendix gives examples of graphical output (Figures F.1 to F.8) corresponding to experimental runs of Forward Chaining and Gradient Descent on the Bigmap navigation environment.

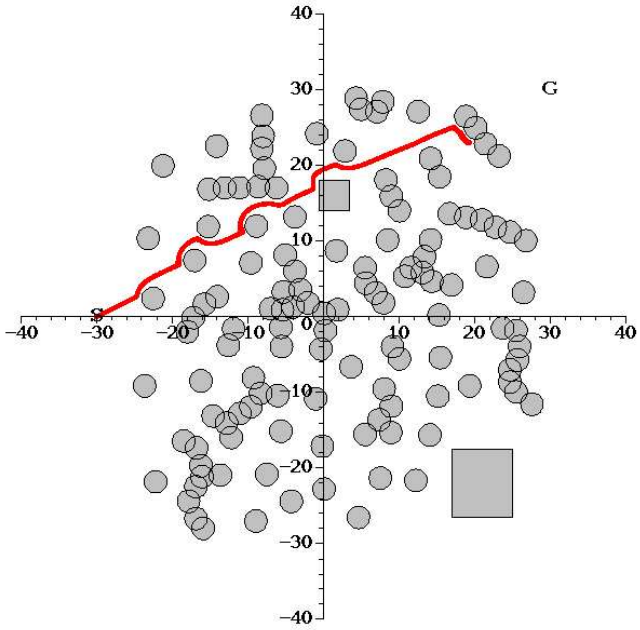


Figure F.1: Gradient Descent navigating from (-30,0) to (30,30).

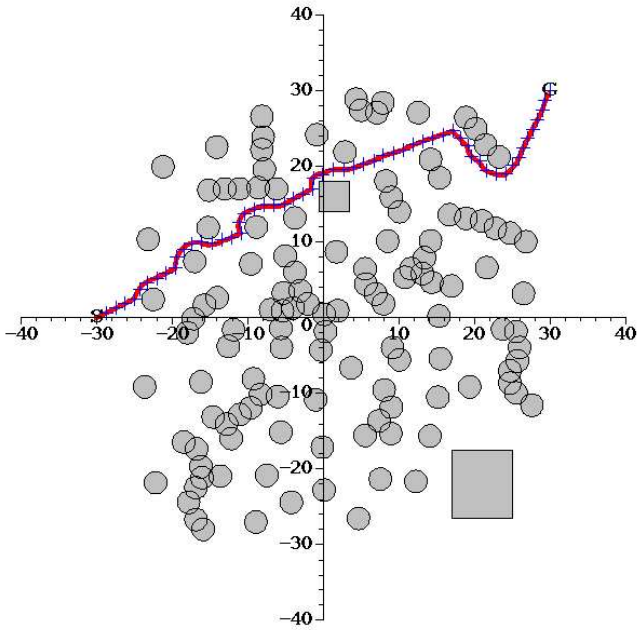


Figure F.2: Forward Chaining navigating from (-30,0) to (30,30).

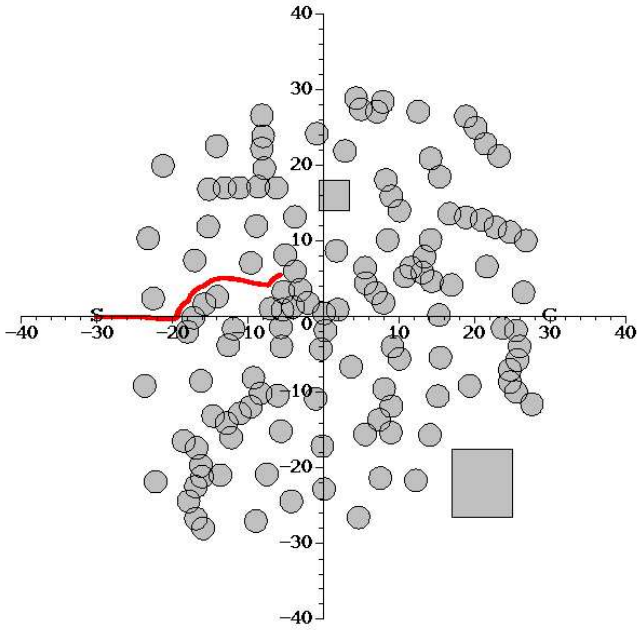


Figure F.3: Gradient Descent navigating from (-30,0) to (30,0).

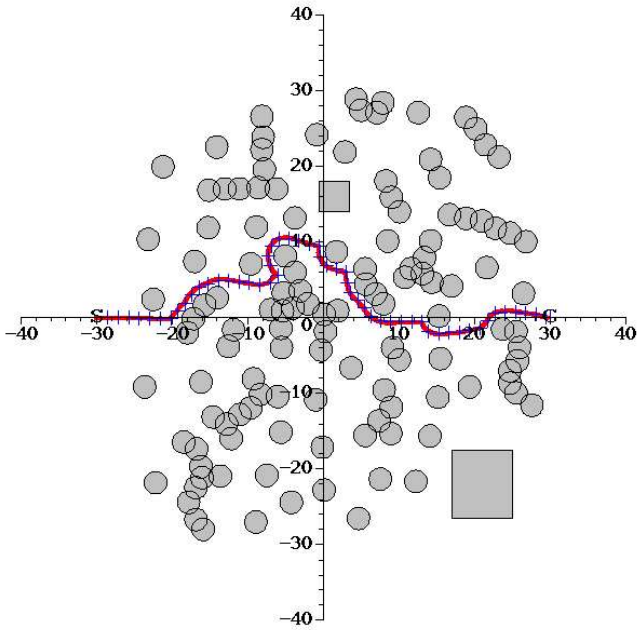


Figure F.4: Forward Chaining navigating from (-30,0) to (30,0).

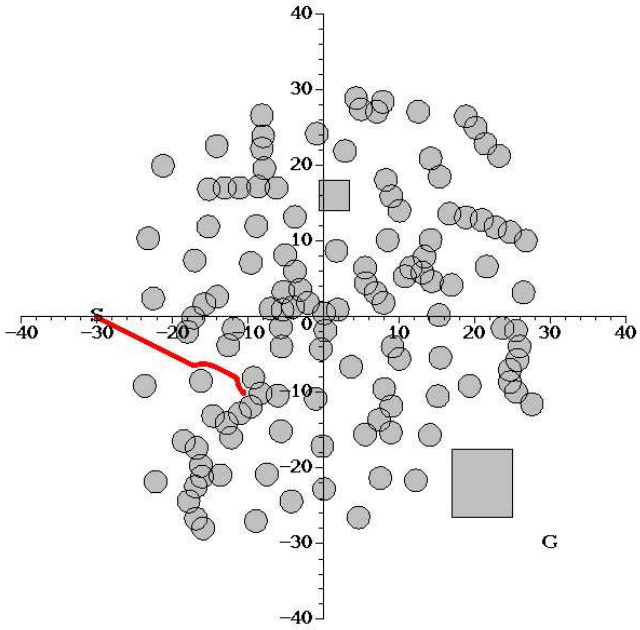


Figure F.5: Gradient Descent navigating from (-30,0) to (30,-30).

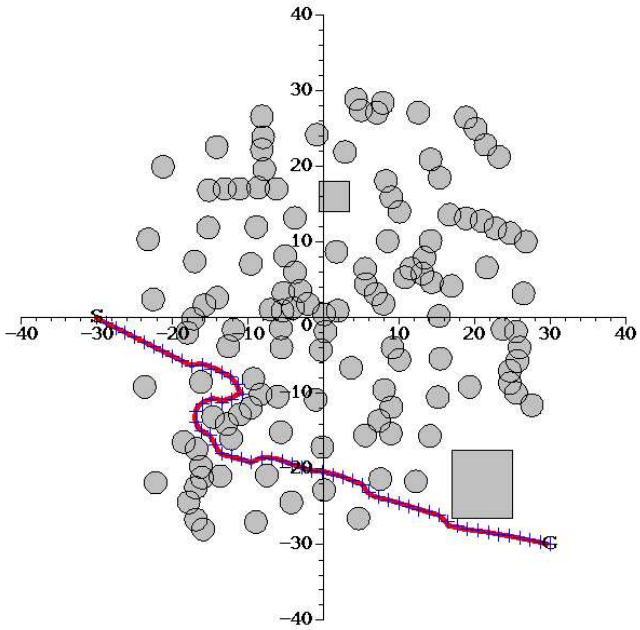


Figure F.6: Forward Chaining navigating from (-30,0) to (30,-30).

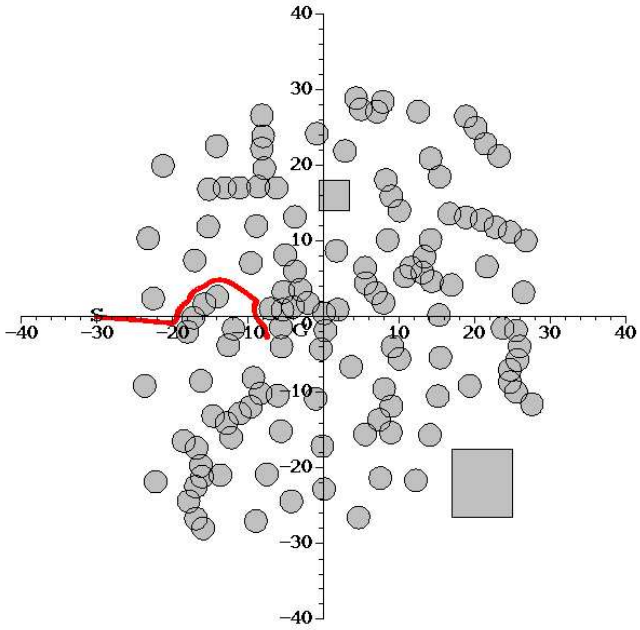


Figure F.7: Gradient Descent navigating from (-30,0) to (-3,2).

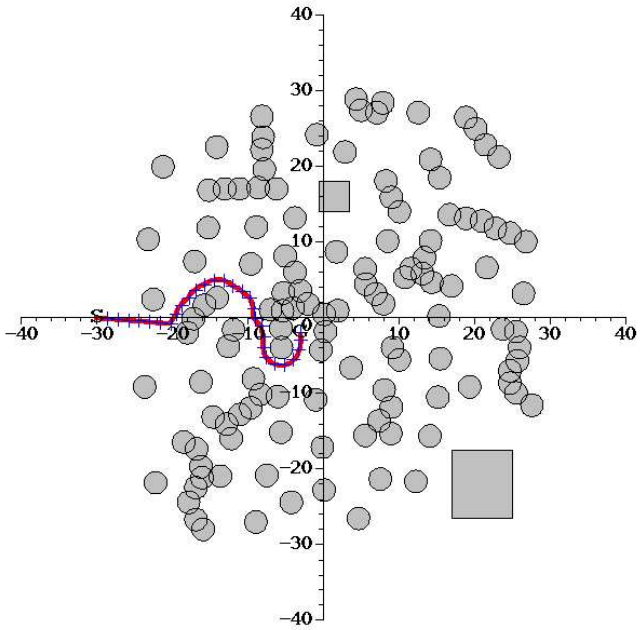


Figure F.8: Forward Chaining navigating from (-30,0) to (-3,2).

Appendix G

Source Code Listing for FORWARDS

This Appendix provides Maple source code taken from the experimental implementation of Forward Chaining used in this thesis. It is provided here for the purpose of assisting future implementations.

```
#FORWARDS Subgoal Placement Heuristic

#Import the Gradient Descent code.
read "alg/gd.mpl";

#Define procedure
subgoalchain:= proc(startX,startY,goalX,goalY,gdTolerance,gdStepSize,
gdStepLimit,forwardAngleRangeDegrees,forwardAngleSearchSep,
GoalStrengthConstant,goalTolerance,stepLimit,SGDistance,
gdFrameSkip)

#parameter meanings
#startX - the initial X position of agent in 2-D space.
#startY - the initial Y position of agent in 2-D space.
#goalX - the desired (goal) X position of agent in 2-D space.
```

```
#goalY - the desired (goal) Y position of agent in 2-D space.
#gdtolerance - tolerance value to be used during gradient descent -
# e.g. how close gradient descent should be to the goal in
# distance units for the GD phase to be considered successful.
#gdStepSize - size of step to be taken during gradient descent.
# Smaller than subgoal placement step size.
#gdStepLimit - how many steps gradient descent can take before the
# attempt at achieving the subgoal is classed as failure.
#forwardAngleRangeDegrees - The size of the circle arc in degrees
# that the agent scans across for a subgoal candidate.
# 180' in all experiments in Chapter 7 of thesis.
#forwardAngleSearchSep - quality of search. e.g. search every 1 degree
# along the circle or 2 degrees, etc. integer value measured
# in degrees
#GoalStrengthConstant - constant setting 'attractiveness' of the goal
#goalTolerance - how close the agent must be in distance units to be
# considered as having achieved the goal during Forward Chaining.
#stepLimit - maximum number of steps to attempt
#SGDistance - Distance between subgoals, measured in distance units.
#gdFrameSkip - allows scaling down of framecount in the animations.

# Global variables for graphics functions.
global gifcount,XSGt,YSGt,CSGt;

#local variables
#curX,curY - current agent location
#gdSGX,gdSGY = current subgoal location.
#animFrames,goalDiag, baseDiag, startDiag - used to create graphics
#gdSurface - potential field that gets passed to gradient descent for
# inter-subgoal travel - local field.
#SGXtm1,SGXtm2,SGYtm1,SGYtm2 - last positions of subgoals, used to track
# Forwards direction.
```

```
#gifcount - total number of frames generated in the animation
#goalAngle - direction towards goal rel. to positive horizontal x axis.
#newSearchSurface - the FWDS2/FORWARDS temporary goal potential surface
#gdDx,gdDy - potential gradient functions in the x and y axis.
#temporaryGoalAngle - indicates position of temporary goal, after goal
# bias has been applied
#totalAngle - tracks accumulated path twist.
#cur(X,Y)old(0,1,2),angleTwist - variables for calculating path twist.
#totalGoalAngle - accumulated goal twist.
#pathTwist - path twist on current step
#goalAngleTwist - goal twist on current step
#relAngle - relative twist between acc. path twist and acc. goal twist.

local animFrames, startDiag, goalDiag, baseDiag, totalPathTwist,
totalGoalAngle, curX, curY, curXold0,curXold1,curXold2,curYold0,
curYold1,curYold2, SGXtm1, SGXtm2, SGYtm1, SGYtm2, SGXtm3, SGYtm3,
iteration, FWDSAngle, goalAngle, temporaryGoalAngle, pathTwist,
goalAngleTwist, relAngle, GXtmp, GYtmp, GoalDistance, gdSurface,
gdDx, gdDy, newSearchSurface, bestAngle, gdSGX, gdSGY;



#########################################################################

# Initialise graphics variables
CSGt:=1; gifcount:=0; animFrames:=0;
startDiag:=textplot([startX,startY,"S"]):
goalDiag:=textplot([goalX,goalY,"G"]);
baseDiag:=[startDiag,goalDiag,op(obstacleDiagram)];

#initialise path and subgoal history variables.
totalPathTwist:=0; totalGoalAngle:=0;
curX:=startX;  curXold0:=startX;
```

```
curY:=startY;  curYold0:=startY;
SGXtm1:=startX; SGYtm1:=startY;

#initialise forwards direction, path history to point towards the goal.
SGXtm2:=SGXtm1+startX-goalX;
SGYtm2:=SGYtm1+startY-goalY;
SGXtm3:=SGXtm2+startX-goalX;
SGYtm3:=SGYtm2+startY-goalY;
curXold1:=curXold0+startX-goalX;
curYold1:=curYold0+startY-goalY;
curXold2:=curXold1+startX-goalX;
curYold2:=curYold1+startY-goalY;

#while distance is > "goalTolerance" from the goal,
#having taken < "stepLimit" subgoal steps:
for iteration from 0 while
( evalf(sqrt((goalX-curX)^2+(goalY-curY)^2)) > goalTolerance
  and iteration<stepLimit) do

print("Subgoal step number: ",iteration);

# Calculate Forward Direction based on previous subgoals
FWDSAngle:=calcAngle(SGXtm2,SGYtm2,SGXtm1,SGYtm1);
print("Forwards Direction (FWDSAngle)",FWDSAngle);

#Calculate direction of Goal from current position.
goalAngle:=calcAngle(curX,curY,goalX,goalY);
print("Goal angle",goalAngle);

#FWDS2:    Calculate goal-bias direction for temp. goal, +/- 75' bias.
temporaryGoalAngle:=pickGoalBiasedAngle(goalAngle,FWDSAngle,150);
print("temporaryGoalAngle:=",temporaryGoalAngle);
```

```
#FORWARDS: Check path twist, and if necessary, set the angle to untwist,
#          overriding the FWDS2 selection.

pathTwist:=calcAngleTwist(curXold2,curYold2,curXold1,curYold1,
  curXold0,curYold0);
print("Path twist:",pathTwist);

totalPathTwist:=totalPathTwist+pathTwist;
print("Total path twist: ",totalPathTwist);

#calculate twist around the goal

goalAngleTwist:=calcGoalTwist(SGXtm2,SGYtm2,goalX,goalY,SGXtm1,SGYtm1);

# Accumulate the total angle rotated around the goal.
totalGoalAngle:=totalGoalAngle+goalAngleTwist;
print("Total goal angle",totalGoalAngle);

# Calculate relative path twist
relAngle:=totalPathTwist-totalGoalAngle;

#note we now use relAngle not total angle in making decisions
#note also that the FWDS2 subgoal selection is only over-written
#if the agent's path has twisted too much.

print("Relangle",relAngle);

#if twisted too much anti-clockwise, twist clockwise
if relAngle>180 then
temporaryGoalAngle:=FWDSAngle-forwardAngleRangeDegrees/2;
```

```
#if twisted too much clockwise, twist anti-clockwise
else if relAngle<-180 then
temporaryGoalAngle:=FWDSAngle+forwardAngleRangeDegrees/2;

fi;
fi;

print("temporaryGoalAngle chosen: ", temporaryGoalAngle);

# Now set up a potential field for the temporary goal + obstacles
GXtmp:= curX+evalf(cos(temporaryGoalAngle*2*Pi/360)*SGDistance);
GYtmp:= curY+evalf(sin(temporaryGoalAngle*2*Pi/360)*SGDistance);
newSearchSurface:=(x,y)->oP(x,y)+
  CONE(x,y,GXtmp,GYtmp,GoalStrengthConstant,0);

#Sample for the lowest point along the forward hemisphere
#on 'newSearchSurface' to use as the gradient descent subgoal.
bestAngle:=findLowestPoint(newSearchSurface,FWDSAngle,
   forwardAngleRangeDegrees, SGDistance,
   curX,curY,forwardAngleSearchSep);

print("Subgoal angle picked",bestAngle);

#now tell gradient descent to go for this target.
gdSGX:= curX+evalf(cos(bestAngle*2*Pi/360)*SGDistance);
gdSGY:= curY+evalf(sin(bestAngle*2*Pi/360)*SGDistance);

# Finally, when within one step's distance of the goal,
# set the gradient descent target to be the goal.
GoalDistance:=evalf(sqrt((goalX-curX)^2+(goalY-curY)^2));
if (GoalDistance<=SGDistance) then gdSGX:=goalX; gdSGY:=goalY; fi;
print("Subgoal position:",gdSGX, gdSGY);
```

```
# Set up subgoal-obstacle potential surface for gradient descent.
gdSurface:=unapply(oP(x,y)+
   CONE(x,y,gdSGX,gdSGY,GoalStrengthConstant,0),(x,y));
gdDx := unapply(dxOp(x,y)+
dxCONE(x,y,gdSGX,gdSGY,GoalStrengthConstant,0),(x,y));
gdDy := unapply(dyOp(x,y)+
dyCONE(x,y,gdSGX,gdSGY,GoalStrengthConstant,0),(x,y));

# Update records for image rendering.
XSGt[CSGt]:=gdSGX;
YSGt[CSGt]:=gdSGY;
CSGt:=CSGt+1;

#record the position reached
(curX,curY):=grad_descent(curX,curY,gdStepLimit,
gdStepSize,gdSurface,gdDx,gdDy,gdSGX,gdSGY,gdTolerance,
baseDiag,gdFrameSkip);
print("Position after gradient descent travel", curX, curY);

#update records of recent subgoals and agent positions.
curXold2:=curXold1; curYold2:=curYold1;
curXold1:=curXold0; curYold1:=curYold0;
curXold0:=curX; curYold0:=curY;
SGXtm2:=SGXtm1; SGYtm2:=SGYtm1;
SGXtm1:=gdSGX; SGYtm1:=gdSGY;

od;

#generate subgoal chaining diagram
read "lib/plotSGD.mpl";
printDiagram(baseDiag);
```

```
print ("Goal reached or step limit reached!");
end proc:



##########
#pickGoalBiasedAngle is used by FWDS2 and FORWARDS for 'goal-bias'.
#It picks out the angle pointing at the goal if the goal lies
#within the bias range, otherwise it picks out the side of the
#bias range that is closest to the goal. e.g. imagine a car
#speedometer which is currently pointing at '40' and is allowed
#to vary by + or - 10 mph. If you wanted it to point as close
#to 35 as possible, you could make it point straight at '35'.
#If you wanted it to point at 25, then '30' is the best you
#could manage within the range. Similarly if you wanted it to
#point at '75', then '50' is the best you could manage.

pickGoalBiasedAngle:=proc(goalAngle,FWDSAngle,
  angleRangeDegrees)

local diffAngle,halfRange;
halfRange:=angleRangeDegrees/2;
diffAngle:=goalAngle-FWDSAngle;
if diffAngle>=180 then diffAngle:=diffAngle-360; fi;
if diffAngle<-180 then diffAngle:=diffAngle+360; fi;
if (diffAngle>=0) then return(FWDSAngle+min(diffAngle,halfRange));
else return (FWDSAngle+max(diffAngle,-halfRange));
fi;
end proc:
```

```
##########
#FindLowestPoint is used to find a point of lowest potential on
#the forwards semi-circle. It can in fact use other parts of the
#circle surrounding the forwards position - such as smaller
#circle arcs than 180' of the circle.
#The parameters specify the potential surface to use when
#searching, the angle to start from relative to the positive
#horizontal axis, the range (degreeLimit), the radius of the circle
#along which the search is being conducted, the current X and Y
#co-ordinates, and the coarseness of the search (i.e. the increment
#applied between checking different angles, such as +1 degree each time,
#or +5 degrees each time).

findLowestPoint:=proc(surface,initAngle,degreeLimit,distance,
      X,Y,angleSep)

local theta,height,bestTheta,bestHeight,angleMin,angleMax;

#initialise values.
bestHeight:=evalf(surface(evalf(X+cos(initAngle*2*Pi/360)*distance),
                          evalf(Y+sin(initAngle*2*Pi/360)*distance)));
bestTheta:=initAngle;
angleMin:=round(initAngle-degreeLimit/2);
angleMax:=round(initAngle+degreeLimit/2);
theta:=angleMin;

#scan for lowest point
while theta<angleMax do

height:=evalf(surface(X+evalf(cos(theta*2*Pi/360)*distance),
```

```
                    Y+evalf(sin(theta*2*Pi/360)*distance)));
if (height<bestHeight) then
bestHeight:=height;
bestTheta:=theta;
fi;
theta:=theta+angleSep;

od;

return(bestTheta);
end proc:
##########
```

Appendix H

Guide to the DVD

This appendix is a guide to the contents of the DVD that can be found at the back of this thesis.

The DVD is structured into 4 directories;

- /experimentalresults/ - this directory contains graphics and timing information corresponding to all of the experiments in Chapter 7.
- /animations/ - this directory contains animations for some of the paths generated using the Bigmap environment.
- /webcache/ - this directory contains a cached copy of all material referenced from the World Wide Web. This is provided for convenience because of the transient nature of many web-based resources. Content is organised according to reference number in the bibliography.
- /pdf/ - this directory contains the PDF file corresponding to this thesis document.