Using trails to improve map generation for virtual agents in large scale, online environments

by

Katrina Samperi

A thesis submitted to
The University of Birmingham
for the degree of
Doctor of Philosophy (PhD)

School of Computer Science
The University of Birmingham
January 2015

Abstract

This thesis looks at improving the generation of maps for intelligent virtual agents in large scale environments. Virtual environments are growing larger in size and becoming more complex. There is a major challenge in providing agents that are able to autonomously generate their own map representations of the environment for use in navigation. Currently, map generation for agents in large scale virtual environments is performed either by hand or requires a lengthy pre-processing step where the map is built offline. We are interested in environments where this process is not possible, such as those that encourage user generated content.

We look at improving map generation in these environments by using *trails*. Trails are a set of observations of how a user navigates an environment over time. By observing trails an agent is able to identify free space in an environment and how to navigate between points without needing to perform any collision checking. We found that trails in a virtual environments are a useful source of information for an agent's map building process. Trails can be used to improve rapidly exploring randomised tree and probabilistic roadmap generation, as well as being used as a source of information for segmenting maps in very large scale environments.

Publications arising from this Thesis

- Katrina Samperi. Large-scale Mapping and Navigation in Virtual Worlds. Thesis Summary. In Proceedings of the twenty-sixth AAAI conference on Artificial Intelligence, July 2012 (Samperi, 2012)
- Katrina Samperi, Russell Beale, Nick Hawes. Please keep off the grass: Individual norms in virtual worlds. In Proceedings of BCS HCI, 2012 (Samperi et al., 2012)
- Kat Samperi, Nick Hawes and Russell Beale. Improving Map Generation in Large-Scale Environments for Intelligent Virtual Agents. The AAMAS-2013 Workshop on Cognitive Agents for Virtual Environments. May 2013 (Samperi et al., 2013a)
- Katrina Samperi, Nick Hawes and Russell Beale. Towards mapping and segmentation of very large scale spaces with intelligent virtual agents. 13th International Conference on Intelligent Virtual Agents August 2013.(Samperi et al., 2013b)
- Katrina Samperi and Nick Hawes. Improving the generation of rapidly exploring randomised trees (RRTs) in large scale virtual environments using trails. Towards Autonomous Robotic Systems (TAROS), September 2014 (Samperi and Hawes, 2014)

Acknowledgements

I'm not sure I would ever have finished writing this thesis without the friends and family that have supported me throughout. It has been a long few years and they have had to deal with ups and downs, excitement, stress, grumpiness and excessive demands for tea. I would especially like to thank Barry for first putting up with me over the past few years and then proposing and getting married! You will have to deal with me trying to explain how awesome agents are for a few more years to come.

Huge thanks are owed to my parents. My dad for teaching me about computers and encouraging me to learn more. My mum for all the help, support (and proof reading) over the years. I don't think I would even have started university, let alone be reaching the end of my PhD without you guys.

I would like to thank Nick Hawes and Russell Beale for all their supervision, support and guidance over the years. Nick especially for taking a chance on me when I went to his office asking about a complete change in topic in my first year. Thanks are also due to the many members of the School of Computer Science and the Intelligent Robotics Lab who have helped and guided me over the years. I would especially like to mention Mark Lee for all his help with my original thesis proposal, Peter Hancox for encouraging me both to start the PhD and then to continue with it despite the setbacks, and Tom Chothia for always being there if I needed a chat.

Thanks also need to go out to Chris Staite, Vivek Nallur, Andrew Fielder, Ben Woolford-Lim, Olaf Klinke, Sara Al-Azzani, Matt Smart, Chris Novakovic, Catherine Harris and Ian Batten for all the tea, random chats and ideas over the years. Despite our lack of a yucca, 117 was a great office to be in, so thank you to all those I shared the space with!

Finally, this would not be the document it is if it wasn't for Pancakes, the literature review bunny, who delighted in stealing copies of whatever I was trying to read at the time. Or, on one occasion, the pen I was writing with. Her successors, Angus and Shadow have been equally 'helpful' while I've been attempting to proof read this document.

Contents

List of Figures

List of Tables

List of Algorithms

CHAPTER 1

Introduction

The prevalence of virtual worlds presents an interesting challenge for intelligent mobile agents. Online, large-scale, persistent virtual worlds such as Second Life (Linden Research Inc., 2012) and Massive Multiplayer Online (MMO) style games are becoming more popular (Pittman and GauthierDickey, 2007). The term *virtual environment* refers to a digital representation of a geometric or abstract space. A traditional view of virtual environments is virtual reality, a digital representation of a three dimensional world. Users are able to move around these worlds exploring and interacting with their environment. Virtual environments are used for a variety of applications, the largest being the computer game industry, attracting $21 billion in sales for 2012 (Entertainment Software Association, 2013). Other uses for virtual environments include socialising, research, modelling, business and education.

Intelligent agents are a type of program that is able to perceive its environment and then act on the observations made (Russell and Norvig, 2003). We are interested in intelligent embodied agents in this thesis. An intelligent agent makes decisions based on observations of the environment they are in. Embodied agents are those that have a presence in the environment they are interacting with. In the real world agents observe the environment using sensors and act on it using motors. Examples of these agents are robots that interact with the real world such as the mars rover (Maurette, 2003), driverless cars (Thrun, 2010) or the Roomba (irobot.com, 2014). Virtual embodied agents control game characters and avatars in virtual environments. They are able to perceive their environment using the information

provided by the server that controls the virtual world. They then act on it the same way as a human controlled avatar, by sending commands back to this server.

There is a challenge to provide intelligent agents for virtual environments that are able to plan routes and navigate the world even when faced with large scale environments and user generated content. Planning for agents involves linking a series of actions together to achieve a specific task. Mobile agents are required to plan routes and navigate between specific points in an environment. To plan a route over long distances the agent needs an internal representation, a map, of the environment used to identify collision free paths between obstacles in the world. Maps of an environment can be given to the agent or they can generate their own. Currently, game based virtual agents are given a pre-computed set of routes for known environments. These are usually generated by hand, but some can be done automatically (Nieuwenhuisen et al., 2004). This is due to the time required to generate a map in a large scale, unknown environment. The generation time for an agent to create a full map of the environment can be high, and so it is often done in a pre-processing step and given to the agent before the environment is released for use. When the agent needs to be provided with maps in advance there is little a user in the virtual world can do to change their environment without causing the agent problems with navigation.

A couple of examples of environments that encourage user generated content are Second Life, and first person shooter games such as Unreal Tournament (Epic Games Inc., 2012). Second Life is a very large, online, populated environment where users are encouraged to create their own spaces. This virtual world is continually growing and changing as more users engage with the environment. There is a low cost in terms of time and effort for virtual environments to change, so change tends to be vast and rapid. It is not possible for an agent to pre-compute a map of the entire environment in an offline pre-processing step. Second Life changes too rapidly for this to be feasible. First person shooters have a long history of encouraging user generated content in terms of new levels to explore. One of the original first person shooters widely available on the PC, Doom, still has a set of tutorials and a community of people building new content for the game (vd Heiden, 2008). Most new games follow this example and provide toolkits for adding new content. For example: Bethesda Softworks Inc. (2008) provide a toolkit for adding new areas and content to their game worlds (Fallout 3 and New Vegas in this case) and the Open Metaverse Foundation (2012) provide libraries for creating Second Life clones and allowing agents to connect to these environments. When an agent is expected to navigate a user created environment, it will have a limited period of time to try and build a usable map of this environment before use. The current method used is to place the map building burden on the level designer and ask them to generate the navigation maps

and route plans for an agent. Realistic behaviour of agents for these games is preferred. Nieuwenhuisen et al. (2004) note that automatically generating the navigation maps for an agent can lead to low quality maps and repetitive behaviour which is noticeable to an end user.

This thesis looks at improving the generation of map representations used for navigation by intelligent mobile agents in large scale, virtual environments. This problem is interesting as virtual worlds are very large, continuous, ever changing environments. Intelligent mobile agents are embodied agents that inhabit virtual environments and are able to interact with their world in the same way as any human user . It is not trivial for an agent to create a map and be able to use it in a short period of time in large scale, complex environments. In very large worlds like Second Life there is a market for autonomous agents, able to create their own maps of the world and then to use these to navigate on demand. As we are interested in online populated environments, the speed the agent can react to commands and changes in the environment is important. Too long in delaying an action and the environment and situation may have changed dramatically.

To plan navigation between points an agent needs some type of internal map representation of their environment. We look at generating and using *roadmaps* for navigation in this thesis. Roadmaps are a specific map representation that consist of a set of points in an environment and the paths that can be taken to move between these points. We go into more detail in the literature review about mapping and the specific roadmap representations we focus on (Sections 2.4.1 and 2.4.2). Roadmaps are a popular form of representation due to the ease at which they can be constructed in any environment. However, the time required to generate a map in a large scale environment can be very long, especially if the environment contains narrow passages.

To improve the map generation of intelligent mobile virtual agents in large scale online environments we look at an available, if little used, method of gathering information about the world: *trails*. Trails are a set of observations based on the movement of other agents in an environment. Trails in the real world are the traces left by an agent (human, animal or artificial) as it navigates between places in an environment. One example of trails can be seen where people have worn down long grass by their passage (See Figure 2.6). Historically trails have been used to track food and water as well as finding routes between settlements. Animal trails are used by both animal and human hunters when looking for food and water. Trails have been used to follow the movement of people over entire continents throughout history by looking at burial habits and the artefacts left behind (Bradley, 2007). They are still a useful tool in architecture, showing the path people use when crossing grass or scrub land areas rather than

the paths that have been built (Myhill, 2004). More recently, trails have been used as a starting point for agent navigation in real environments and to aid human navigation in virtual environments (Ruddle, 2005; Yuan et al., 2010). A digital trail is an observation of the location of a particular avatar over time.

The specific properties of trails that we are interested in are those that give the agent more information about its environment. Every position an avatar has been observed at is in free space. This information is important as the most costly part of map generation in unknown environments is collision checking. The agent can make an assumption that if another agent has been observed in a location it is free without needing to perform any calculations. Furthermore, if the time between consecutive observations from a single agent is small, then the agent is able to conclude that a straight line path between the two observations will also be possible to navigate. We use these pieces of information to improve roadmap generation as a roadmap is, in the simplest form, a set of nodes and straight lines connecting them. Trails give this information without the need for any calculations and can form a starting point for map generation.

The limitations of using trails is that they only give information about where agents have been observed. If an environment is not visited by other people then no trail information will be available. Virtual environments also can have other forms of transport, such as teleportation and flying. We are interested in an agent that exists at ground level and walks between points in the environment. So if the primary mode of transport in a virtual world is flying, no useful trail information will be available. We look at methods of combining trails with existing map generation methods to mitigate this.

1.1 Contributions of the Thesis

The contributions of this thesis are as follows:

1. Validation that trails are a useful source of information for an agent to build a map (Chapter 3 – 7)

2. A study into the effect of trails being used to generate probabilistic roadmaps in large scale environments.

   We find that trails have a positive effect on roadmap generation, reducing the time needed for map generation, reducing the overall plan length and improving on the success rate of the agent being able to plan routes around the environment. (Chapter 5)

3. A study into the effect of trails on roadmap generation as the environment grows larger. Validation that trails become more important as the size of the environment increases (Chapter 6)

4. A study into the effect of trails on another type of roadmap representation, rapidly exploring randomised trees. Trails also have a beneficial effect on the generation of RRTs, improving their performance (Chapter 4).

5. An investigation into segmentation strategies in large scale virtual environments and the effect that trails can have if they are used as an information source. (Chapter 7)

6. A map trail corpus for intelligent agents in Second Life available at `http://www.katsamperi.co.uk/maps.php`.

1.2 Overview of the Thesis

Chapter 2 looks at the related literature covering virtual environments, agent based map representation, roadmaps and trails. We look at a number of different map representations that are available to agents attempting to navigate virtual environments. Section 2.4 then focuses on roadmap representations, these are the map representations we look at improving the generation of in Chapters 4 – 6. Following that, Section 2.6 looks at how trails have been used in the real world for navigation and their application in other domains.

We did not know initially if trails were a viable source of information for agent map building. Chapter 3 presents our evaluation domain and looks at trails in more detail. This chapter investigates patterns in digital trails in an attempt to see if *meaningful* information can be extracted from the movement of avatars in an environment. After first establishing that it is possible to observe the record the movement of avatars, we studied a particular social norm that emerged from the observations. Not walking on the grass is an established social norm in the United Kingdom. Social pressures ensure that in grass and park areas people tend to keep on the paths when navigating between places. This pattern is also present in a virtual park environment and that it holds over avatars of all 'ages' (calculated from their creation date). This chapter validates the idea that it is possible to gather meaningful trails in virtual environments.

Chapter 4 then investigates using trail information to improve the generation of rapidly exploring randomised trees (RRT). This chapter compares existing RRT generation and point selection methods with those biased by trail points. Biasing the map generation using trails gives a tree that tends to

grow towards more populated areas of the environment. The results show that this reduces the length of planned routes between a given start and end point.

Chapter 5 follows on by exploring the effect trail points have on a different type of roadmap representation, the Probabilistic Roadmap (PRM). In this chapter we look at changing the point selection method part of the PRM generation algorithm. Trails are used either to bias points selected for inclusion in the roadmap, or in the roadmap directly. The results of the experiment were that in both small and large scale environments trails have a positive effect on the generation time of the PRM. Trails also improve the success rate of being able to plan a set of six routes using these maps and reduce the overall planned path length. A combination of quasi-random Halton points and Trail points give the best performance in the environments studied.

Chapter 6 expands on the work done previously. The disadvantage when looking at online virtual environments is finding an area that matches the set of criteria we are interested in. The environment must have unrestricted access, be large enough, mostly flat and highly populated. In this chapter we designed a set of environments to look at the difference made by trails in a specific set of circumstances. This gives us an idea of how trail based PRMs will perform in different situations. The results show that as an environment becomes larger, trails are more important as in limited point scenarios there is no way for a random PRM method to successfully plan routes.

As the size of the environment grows larger, the time required for the agent to generate a map also increases. We want to keep this time as short as possible and so Chapter 7 looks at map segmentation, a standard approach taken in large scale environments. Segmentation splits a map into distinct regions. This chapter looks at whether trails can be used to improve the segmentation methods, speeding up map generation and path planning. The results show that the due to the nature of trails, the spread of points is generally not sufficient to generate a good segmentation over the environment. This then results in trail based segmentation not out outperforming more simple methods of segmentation. However, trails improve the route planning success rate in large scale environments.

Finally, Chapter 8 reviews all our results and looks at future work which can be done to improve map generation in intelligent mobile virtual agents.

CHAPTER 2

Literature Review

2.1 Introduction

This chapter looks at the current literature related to this thesis. The overarching problem we are looking at is how to improve map generation for intelligent virtual agents in large scale, populated virtual environments. We first look at what is meant by the term 'virtual environment'. Section 2.2 looks at how virtual environments have evolved over time and their many uses in entertainment, research, education and industry. This section gives us the background information for virtual environments and why we are choosing to look at agents inhabiting online worlds.

Section 2.3 then looks at current map representations for use in virtual and real world environments. We look at the different classifications of maps, how they are used, and which type is considered most suitable for which type of environment. Section 2.4 then expands on the roadmap representations we focus on in Chapters 4, 5 and 6. After this, in Section 2.5, we look at how maps can be used to represent very large scale environments. We focus on segmentation, the process of splitting a large map into small individual parts.

Finally, Section 2.6 looks at trails. We define digital trails as a set of observations made of the location of a specific avatar over time. This idea is based on trails in the real world, the evidence left as an agent has

navigated between places. We look at how trails have been used for research in the real world and some of their uses in virtual environments. Trails are used throughout this thesis as a source of information for an agent's map building process. Chapter 3 looks at whether meaningful trails can be gathered in our chosen evaluation domain, Chapter 4 – 6 look at using trails as a source of information for roadmap representations of space and Chapter 7 looks at how we can segment a map using the information in trails.

2.2 Virtual Environments

The term virtual environment refers to any digital representation of an area. This area can be based on the real world or entirely abstract. The original virtual environments were text based games. These gave an impression of space through descriptions of the environment and the effect actions had. Virtual environments and virtual reality have long been of interest to researchers and for information visualisation. We primarily look at game based virtual environments for our research.

Game-based virtual environments can be split into two categories: single and multi-player environments. Single player environments started experimenting with graphics moving from static images through to the advanced visuals we have in today's games. Multi-player games became popular first with text based games such as Multi-User Dungeons (MUDs) (Bartle, 1990). MUDs then evolved over time, incorporating graphics to achieve the large online game environments seen today. These come in two flavours:

1. Game worlds, where the player has little control of their environment. e.g.
    - Massively Multi-Player Online Roleplay Games (MMORPG) such as World of Warcraft (Blizzard Entertainment, 2013)
    - First person shooters such as Counterstrike (Valve Corporation, 2013)
2. Collaborative environments where the users can work together to build their environment. e.g.
    - Second Life (Linden Research Inc., 2012)
    - Minecraft (Notch Development AB, 2012)
    - Metaplace (Metaplace Inc, 2010)

In recent years the application of virtual environments has grown wider and not just limited to gaming, they have social, business, education and research platforms as well (Chittaro et al., 2006). Offline virtual worlds are used in design and architecture. They give an opportunity to showcase the appearance of a product from all angles without the expense of building a prototype. Online virtual worlds, such as Second Life (Linden Research Inc., 2012) provide an ever growing large scale, persistent virtual environment. Unlike many game-based virtual environments they encourage users to create and own their own spaces and objects in the world. Many businesses have virtual buildings in Second Life. These have been used to conduct meetings, interviews and recruitment events (Gandhi, 2010). Users can find out information about the business and talk to potential customers in an informal setting. In this way the virtual environment is being used as a cross between an instant messaging service, video conferencing and a 3D website.

In education, virtual environments can be useful for distance learning. Universities also use Second Life for teaching, prototyping and meetings. (Glasgow Caledonian University, 2013; The Open University, 2013). The Open University also use Second Life as a method for delivering lectures and seminars to distance learners (The Open University, 2013). Environments such as WhyVille are aimed at children and used to explain concepts. By using a game world, Whyville was able to explore and explain to school children how diseases spread based on the child's own interest in the subject (Foley and La Torre, 2004).

From a research perspective, virtual environments offer a controlled setting which can be used to consistently replicate experiments for direct comparison (Kaminka et al., 2002; Guimarães et al., 2003; Laird, 2002; Heylen et al., 2001). It is considered a simple matter to gather information about the movement and behaviour of avatars in virtual environments (Chittaro et al., 2006).

Researchers have used virtual environments to investigate a number of ideas:

- The behaviour of people in virtual and real environments (Rakkolainen et al., 1998; Heylen et al., 2001; Yee et al., 2007; Friedman et al., 2007b; Samperi et al., 2012; Loomis et al., 1999)
- Understanding human memory and applying this to agents (Brom et al., 2009; Brom and Lukavský, 2009a,b)
- How people visualise, represent, learn and navigate space (Sadeghian et al., 2006; Ruddle, 2005; Barbosa and Rodrigues, 2006; Ramloll and Mowat, 2001; Ruddle, 2008; Doyle et al., 1997; Chittaro et al., 2006; Dixit and Youngblood, 2008)
- Testing map representation and use in robotics (Mekni and Moulin, 2010; Jarvis, 2010; Yuan et al.,

2010)

- Replicating human behaviour for intelligent systems (Malfaz and Salichs, 2010; Doyle et al., 1997; Heylen et al., 2001)

Linden Research Inc. published a series of statistics about the Second Life environment after its 10th anniversary (Linden Research Inc., 2013). Second Life is home to roughly 36 million accounts, with more than one million of these visiting the virtual world on a monthly basis. In terms of size, the full environment is roughly 700 square miles. Although this is not always continuous space. The environment is ever changing as people are able to purchase land and change their environment with few rules applied. Anecdotally, while undertaking the research for this PhD, the five region area of Second Life we have chosen to use as the basis for our experiments has had significant changes on at least five occasions. This has entirely changed the structure of the environment. Regions have been added to the environment and then removed. Regions have also changed their position relative to those around them. Virtual environments, unlike real world environments, are very flexible about change. The most common type of changes in these environments are on a smaller scale. These are small objects and buildings created and moved around by individual users. For an agent to be able to generate a map of an environment even a fraction of the size of the full Second Life environment it must be able to do so online. We explore how agents are able to connect to Second Life in Section 3.2, where we go into more detail about the evaluation domain used in this thesis.

By looking at virtual agents we can side-step the problems faced by physical agents and focus on how to represent and build maps of increasingly large, complex environments (Laird, 2002). By looking at dynamic and populated environments a mobile agent is faced with a realistic setting, in which objects move, the layout can change, and the agent is able to observe the movement of other avatars in the area.

2.3 Mapping

Mapping for intelligent mobile agents is an active research area with a lot of research still being done. Agents need map representations for planning how to navigate through an environment. In virtual and game environments intelligent agents use maps to orient themselves and make decisions about how to navigate between places. For user controlled agents (such as avatars) maps are used for a similar purpose. They orient the user in an environment showing them what is in the near area and how to navigate to further away points. In games often the map is not given up front to a user as part of the game is

to explore the environment. When looking at intelligent virtual agents we assume that they are not interested in exploration of an environment for fun, merely as a tool to build a map which can then be used to carry out tasks. In this section we briefly look at the different types of map representation available to intelligent mobile virtual agents giving a background to our research. Section 2.4 then goes into more detail about the specific map representations we are using in this thesis.

Map representations can be split into three broad categories: metric, topological and conceptual maps. Combinations of different map representation types are called hybrid maps, these are a combination of different types of information and the means for the agent to link representations together (Simhon and Dudek, 1998; Poncela et al., 2002; Kuipers et al., 2004; Zender et al., 2008; Mekni and Moulin, 2010; Konolige et al., 2011). Hybrid representations are used in an attempt to reduce the limitations of any one map representation. For example, Simmons and Koenig (1995) note that metric maps built from sensors can be prone to inaccuracy, so they use topological maps to try and resolve any ambiguities in the underlying representation.

2.3.1 Conceptual Maps

Conceptual maps attempt to attach semantic meaning to objects and locations in an environment. They are used by an agent to understand how a world fits together and derive meaning from objects found. Cognitive maps are a type of conceptual map. They are one theory as to how humans visualise space as they attempt to navigate between places (Tolman, 1948). Cognitive maps have been used as the basis for agent map generation and representation (Kuipers, 1982; Kuipers and Levitt, 1988; Zender et al., 2008; Pronobis et al., 2009). This theory is that when people are using maps to navigate they do not use just one mental representation, but instead use a hierarchical representation, storing information about landmarks and paths between places in different levels (Tapus and Vasudevan, 2005; Stoffel et al., 2008).

2.3.2 Metric Maps

Metric maps are geometric representations of an environment (Poncela et al., 2002). These are considered to be the simplest and most robust of mapping options giving the location of objects in relation to one another (Kuipers and Levitt, 1988; Filliat, 2003). Metric maps are built directly from sensor information and so a major advantage in using these representations comes from the ease at which the map can be understood by both human users and other agents. The purpose of a metric map is to provide a geometric representation of free and blocked space in an environment (Filliat, 2003). These maps can

be built incrementally in real-time, allowing an agent to detect changes in its environment through new blocked and free areas (Saiki et al., 2009).

The most common metric map representation used is the occupancy grid (Elfes, 1989). This represents the environment as a grid with each cell containing the probability of that area in the environment being blocked or free. Occupancy grids are popular due to their simple nature, they are fast to build and maintain and can describe any environment in a number of dimensions (Poncela et al., 2002). They have been used in dynamic and unknown environments by updating the probability stored in each cell over time. As a new observation is made by the agent the probability of the cell being blocked is calculated and this is combined with any previously stored information in the map (Wolf and Sukhatme, 2005; Yang and Wang, 2011). However, occupancy grids use a lot of memory to store and process (Kraetzschmar et al., 2004; Waqar, 2011; Park et al., 2012) and can be slow to use in path planning (Filliat, 2003) so they are often abstracted into either topological maps or feature-based representations such as the appearance or line map (Zender et al., 2008; Ozkil et al., 2011).

Landmark and feature-based maps are a subset of metric representations. These maps use a metric representation of the environment, with labels added to identify specific places (Thrun et al., 1998). Other metric representations include the barometric map which uses air pressure to determine which floor of a building the agent is on in a three dimensional environment (Ozkil et al., 2011).

One type of map representation that lends itself well to virtual environments is the object map. When using object maps in a real environment, sensor information is used to segment the world into a set of distinct objects which is stored in relation to a coordinate framework (Thrun, 2002). Virtual worlds are presented to the agent in this format directly. Object maps are a more compact representation than occupancy grids. This combined with the ease of building object maps in virtual environments mean that they are considered a good type of representation to use in large-scale or complex virtual environments (Biswas et al., 2002; Wurm et al., 2011). The disadvantage inherent in object maps is the same for all metric representations: Metric maps are resource intensive, explicit localisation is needed at all times and the computational and memory cost of generating, updating and using a metric map for planning is high when compared to more abstract representations, such as the topological map (Poncela et al., 2002; Filliat, 2003; Shi et al., 2010; Konolige et al., 2011; Augustine et al., 2012).

2.3.3 Topological Maps

Topological maps are abstract representations of an environment, built either from metric maps or directly from sensor information (Choset and Nagatani, 2001). They represent the environment as a graph of interconnecting points, only including information relevant to identifying a set of places and instructions on how to move between them (Thrun and Bücken, 1996; Filliat, 2003; Pronobis et al., 2009; Shi et al., 2010; Konolige et al., 2011). Topological maps are usually built when the environment has been fully explored and the metric representation stored (Filliat, 2003). There are two main advantages to using a topological map. The first is that it allows for much faster path planning over an environment as the search space for path planning is reduced to a smaller number of nodes. The second is that they are more robust when sensor noise is a problem, as a topological map does not require explicit localisation at all times (Shi et al., 2010; Konolige et al., 2011). The main disadvantage of topological maps is that they are harder to understand by a human user and harder to share between agents as there is no objective description of the environment. Topological maps rely on the contextual information they were built with (Kuipers and Levitt, 1988; Poncela et al., 2002).

One specific type of topological map we look at in this thesis is the roadmap. Roadmaps are a type of map representation that describes the interconnectivity of free space in an environment (Park et al., 2012). They comprise a set of nodes in free space and arcs describing how to navigate between connected nodes.

In game AI, path planning is often fully scripted, or generated using grid based mapping approaches (Nieuwenhuisen et al., 2004). Fully scripted path planning does not use any type of map representation, instead it relies on someone assigning a set of points that the agent will visit in order. This is time consuming to do by hand and often leads to players being able to recognise repetitive behaviour. To avoid this, roadmaps have been investigated as a potential alternative. In game AI, roadmaps are also known as navigation graphs (Buckland, 2005) or waypoint graphs (Graham et al., 2003; Tozour, 2008; Wardhana et al., 2012). Waypoint graphs consist of points in free space and straight line segments that connect the points (Wardhana et al., 2012). They are not always ideal to use in large scale environments as the number of nodes required in the graph can be very high. Waypoint graphs are usually generated by hand by the level designer and so are computed offline, before the agent is able to use them (Wardhana et al., 2012). This process can take a long period of time (Nieuwenhuisen et al., 2004). Another disadvantage to using waypoint graphs is that the path planned can be very unnatural looking, zig-zagging between points rather than using a straight line (Nieuwenhuisen et al., 2004; Tozour, 2008). Rather than using

Waypoint graph for Halaa, an area of World of Warcraft.

Navigation mesh for the same area.

Figure 2.1: Waypoint graph and Navigation mesh examples for an area of World of Warcraft (Blizzard Entertainment, 2013) created by Tozour (2008).

waypoints, it is often better to use navigation meshes.

Navigation meshes represent the environment as a set of segments an agent can navigate over (See Figure 2.1 for an example of both waypoints and navigation meshes) (Graham et al., 2003). Rather than having single points an avatar can navigate to, the mesh gives an *area* that it can navigate to any point within. Any path inside the mesh is considered valid (Borovikov, 2011) Meshes are more compact than navigation graphs (Buckland, 2005; Epic Games Inc., 2012) and by using them an agent is able to plan a 'more realistic' route using any of the points contained within that segment (Tozour, 2008). The disadvantage to using meshes is the time required to generate. Navigation meshes, like waypoint graphs are generated offline and often by hand (Oliva and Pelechano, 2011). Some research is being done into automatic generation of meshes and making them more reliable (Oliva and Pelechano, 2011; Wardhana et al., 2012), but this does not take into account the time requirement for generating the maps as it is assumed this will be done in a pre-processing step before the map is available to the agent.

We chose to look at roadmaps in this thesis, rather than using navigation meshes, as roadmaps are a popular method of map representation and planning for agents inhabiting both the real and virtual worlds. By using trails our aim is to deal with the speed problem inherent in roadmap and waypoint graph generation. We also want the agent to be able to generate the map autonomously and quickly. An ideal game based agent will be able to generate a map of its environment online, rather than requiring several hours of processing time before before the map is able to be used. We go into more detail about roadmaps and the specific types we look at in Section 2.4.

2.4 Roadmaps

As previously mentioned, roadmaps are a type of topological map representation. They represent the environment as a set of nodes in free space and arcs (or edges) describing collision free paths between linked nodes (Nieuwenhuisen et al., 2004). The most time consuming part of roadmap generation is often the collision checking for nodes and arcs against objects in the environment (Geraerts and Overmars, 2002; Gasparri et al., 2009). When generating roadmaps it is usual for collision checking to happen as each point and arc is included in the roadmap. However, 'lazy' versions of all algorithms exist where collision checking is only performed when a point or arc is selected in the path planning process (Murphy et al., 1999; Marthi, 2010). Accessibility of both arcs and nodes in these maps is updated over time giving a probabilistic confidence value to each path generated (Murphy et al., 1999; Bi et al., 2009).

In this section we look at some different roadmap representations. We look into two types of roadmap in more detail: the probabilistic roadmap and the rapidly exploring randomised tree. These two roadmap representations are used in Chapters 4, 5 and 6.

The earliest type of roadmap used was the potential field map (Barraquand et al., 1992). This builds a roadmap by selecting points for inclusion in the map which are closer to the end goal than the current point. More recent variants of the potential field map include ant colony optimisation (Bi et al., 2009; Mei et al., 2006), hill-climbing (Schmidt and Azarm, 1992) and water-front approaches (Dakulovic et al., 2011).

Ant colony systems were first proposed by Dorigo and Gambardella (1997) and they mimic how ants navigate an environment and apply this to agent navigation. As an ant moves around an environment it leaves a trail of pheromones behind it. When it finds a source of food it takes some and returns to the nest following its own trail. Other ants exploring the environment and trying to find food can sense these pheromone trails and are more likely to chose to walk down a path where the trail is stronger. This leads to a second ant following the first ant's path to the source of food and returning. Pheromones fade over time, but the constant back and forth of ants between a found source of food and their nest reinforces that path causing more ants to switch to following the pheromones. If the route is unsuccessful and no food is found then there will be less return journeys down the path, and so the trail will fade over time. Ant colony approaches for intelligent agents use this idea that as an ant travels between places it leaves a trail of pheromones (Bi et al., 2009). The strength of the pheromones is inversely proportional to the length of the path. When an agent is trying to plan where to travel it takes into account these pheromones and selects the path with the strongest concentration. Hill-climbing and waterfront algorithms use a

similar concept. The goal point is placed at the 'highest' accessible point in an environment. The value of each location in the environment is then calculated based on the distance to the goal point. Waterfront simulates the way water would flow in an environment creating a weighted occupancy grid. Path planning then occurs with each step going towards the 'source' of the water, or the highest point in the environment, giving a path from start to goal (Schmidt and Azarm, 1992; Dakulovic et al., 2011).

All these approaches have local minima problem, where the points selected do not allow the roadmap to escape blocked regions. Different point selection methods have been proposed to try and solve this problem.

A different roadmap-like approach is the Visibility Graph (Huang and Chung, 2004). This calculates bounding boxes for all known objects in the environment and assigns points at the corners of these. A graph is then formed linking the points based on which are 'visible' to one another. This has the advantage that a path should exist around all the objects present in the environment. The main disadvantage is the time required to generate the map is high. As there is no maximum distance after which pairs of points are not considered, in large scale environments the collision calculations for the arc can take a long period of time. Another disadvantage is that as the number of objects in the environment increases, the number of points to be included in the roadmap also increases. This then increases the number of potential arcs that need checking for collisions with objects in the environment.

2.4.1 Rapidly Exploring Randomised Trees

One type of roadmap we want to look at in detail is the rapidly exploring randomised tree (RRT). Chapter 4 explores methods for improving the generation of RRTs by using trails. In this section we look at various RRT generation algorithms.

The RRT algorithm generates a roadmap by simulating the growth of a tree. The map starts from a single *root* point and grows outwards, adding branches to the existing structure as time goes on. A brief outline of the algorithm can be seen in Algorithm 1 (LaValle, 1998). A graphical version of the algorithm can be seen in Figure 2.2. An advantage to the structure of an RRT is that as each branch is grown from an existing part of the tree, the map is always connected. RRTs are considered better than alternative roadmap representations for fast path planning especially in dynamic environments (Kumar and Chakravorty, 2012a). However, the algorithm can still have issues when facing cluttered environments (Park et al., 2010). A disadvantage of the RRT is that it is usually generated as a single use map. The start point of the roadmap is required in advance. If a different start point is used for navigation then a

second tree must be grown.

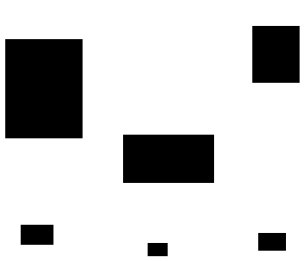
Algorithm 1 The basic RRT generation algorithm

```
1: Require: Start point (s), End point (e), listofpoints, listofbranches
2: Add (s) to list of points
3: while list of points does not contain a point close to (e) do
4:    Select a random point (p)
5:    Find the nearest neighbouring point (n) in the tree to (p)
6:    Calculate the point (p′) a given distance from (n) towards (p).
7:    if n is free AND there is a clear path between the points (n, p′) then
8:       add (p′) to the list of points
9:       add (n, p′) to the list of branches
10:   end if
11: end while
```

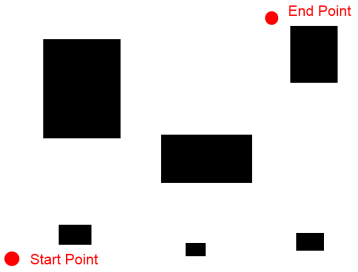
One method of addressing this issue has been proposed by Ferguson et al. (2006). When the environment changes and a branch is no longer accessible that area of the tree is pruned. The blocked branch and any other branches attached to it are removed from the tree. Points are then selected to attempt to regrow the tree with a bias towards the now disconnected regions of the environment. This has the effect of focusing the tree regrowth towards a small section of the environment and allows the tree to recover and identify a route around new obstacles.

The way that the points are selected for branch growth in RRTs can change the tree being grown. Generally the 'cost' of growing a branch is taken into account when deciding how to select points to grow towards (Tahirovic and Magnani, 2011). The standard method is to select points at random (LaValle, 1998). We can also generate a Voronoi diagram from these randomly selected points and bias growth towards the largest Voronoi region at any point (Lindemann and LaValle, 2006). Rodriguez et al. (2006) look at three dimensional RRTs and methods for growing branches based on random selection of a direction to grow in. Their choice of branch growth is based on obstacles in the environment. Krammer et al. (2011) look at biasing the tree growth towards 'useful' areas of the environment. However, this requires some pre-processing of the environment and still does not effectively navigate through narrow corridors. The RRT-Connect algorithm generates two RRTs at the same time, one from the start point and one from the end (Kuffner Jr and LaValle, 2000). This allows the tree to be built faster, finding a solution in a shorter time span. The end point being required in advance has a disadvantage when generating speculative trees. These are trees created ahead of use before an end point is identified. Algorithms such as the RRT-Connect would not be able to function in this way as there is no end point to use as a root for the second tree.

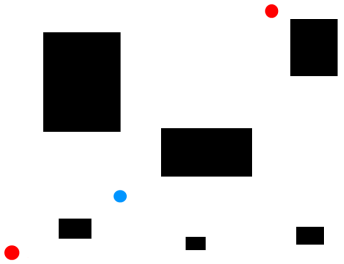
Other variations on the RRT algorithm look at removing some of the randomisation of RRTs. Linde-



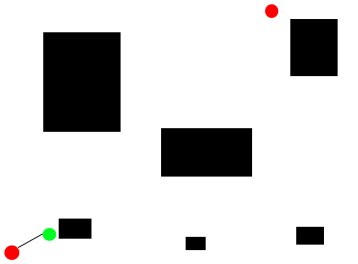
1. The initial environment. Obstacles are black areas and free space is white.



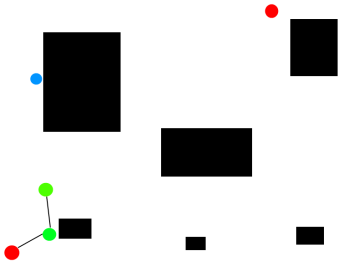
2. The start and end point are added to the roadmap.



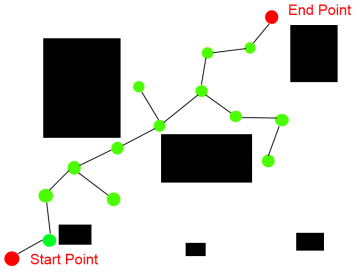
3. A point is selected in the environment. Unlike the PRM this is not added directly to the roadmap and does not need to be in free space.



4. The nearest point on the tree is found and a branch grown towards the selected point. This branch and branch end point (in green) are added to the roadmap. The selected point is discarded.



5. Points continue to be selected and branches grown in the RRT. The algorithm finishes when some criteria is met, usually the end point is able to be connected to the tree.



6. The final roadmap consists of points and arcs connected in a single tree.

Figure 2.2: The steps of the RRT generation process

mann and LaValle (2006) proposed the multi-source RRTs. The multi-source RRT selects a number of random points in the environment. A branch is grown from the node in the tree with the largest number of random points nearby. The disadvantage to this branch growth method is that the RRT takes longer to generate. The authors overcome this by generating a set of uniformly distributed points and adding to this list, rather than reselecting a list of random points each time.

Deciding which point on the existing tree is the 'nearest neighbour' to the selected point can also be calculated differently. Generally this is based on the Euclidean distance between two points. However, it can also be based on environmental factors, such as the roughness of terrain (Tahirovic and Magnani, 2011).

There are two main methods of deciding how to grow a branch from the tree. The standard method is to use a set length for the branch. When the nearest point on the tree is found, a point a given distance away is calculated. That point and the path the branch will take are checked for collisions, and if free they are added to the tree. The other approach is the Greedy RRT, which grows each branch from a neighbouring point until it reaches a maximum distance or an obstacle in the environment (Kuffner Jr and LaValle, 2000; Rodriguez et al., 2006). The Greedy RRT is usually found to be a quicker way of generating a tree. The Greedy RRT algorithm is shown in Algorithm 2. The main difference is that in the original RRT generation algorithm, line 6 calculated a single point a given distance away from the current tree. In the Greedy RRT this is replaced by a loop that calculates all the points on a line between the current tree and selected point, testing each of these in turn until either a point is not in free space, the edge of the environment is reached. Then the final point and branch is added to the tree.

2.4.2 Probabilistic Roadmaps

The other type of roadmap representation we want to look at in more detail is the probabilistic roadmap (PRM). This is the roadmap representation we use in Chapters 5 and 6. PRMs are one of the most popular types of road map representations due to their flexibility (Rodriguez et al., 2006; McMahon et al., 2012). PRMs are abstract representations of the interconnectivity of free space in an environment (Park et al., 2012). They consist of nodes and arcs, linked together to form a graph. PRMs can be generated in any type of environment in any number of dimensions and will eventually allow a route to be planned between any two points in an environment provided that enough nodes are included in the map and that it is possible to do so(Kavraki et al., 1996). While most PRMs are initially generated and used by a single agent, some research has been done on building PRMs using multiple agents (Kumar

Algorithm 2 The Greedy RRT generation algorithm

```
1: Require: Start point (s), End point (e), listofpoints, listofbranches
2: Add (s) to list of points
3: while list of points does not contain a point close to (e) do
4:    p ← Selected point
5:    n ← nearest neighbour in the existing tree
6:
7:    //Using the equation of a line, calculate m and c for the line connecting the points n and p
8:    if p.x != n.x then
9:       m = (n.y - p.y)/(n.x - p.x)
10:   else
11:      m = 0
12:   end if
13:   c = p.y - (m * p.x)
14:
15:   addedSomething = false
16:
17:   //go along the line testing each point one at a time.
18:   newx ← n.x
19:   while not addedSomething do
20:      newx ← oldx + 1
21:      newy ← m * newx + c
22:      if newy is not outside bounds of the environment then
23:
24:         if not isFree(newx,newy) then
25:            //point was blocked, go to previous point and add this as the end point and branch
26:            point ← oldx, oldy
27:            add (point) to the list of points
28:            add (n, point) to the list of branches
29:            addedSomething ← true
30:
31:         else
32:            //x,y coordinate is in free space
33:            if x coordinate at edge of environment then
34:               point ← newx, newy
35:               add (point) to the list of points
36:               add (n, point) to the list of branches
37:               addedSomething ← true
38:            end if
39:
40:         end if
41:
42:      end if
43:      oldx ← newx
44:      oldy ← newy
45:   end while
46: end while
```

and Chakravorty, 2012b). We focus on single agent systems in this thesis.

The application of a PRM consists of two phases, the construction (also known as the learning phase (Bohlin and Kavraki, 2000)) and the query phase. In the construction phase, points are chosen and connected where a straight line exists in free space between them forming a graph. Connections between points have also been called the visibility set for a node(Sun et al., 2005). The query phase then uses this graph to plan a set of routes between two given points. PRMs usually require an existing map of the environment from which to identify free and blocked space. However, research into how to remove this restriction in uncertain and dynamic environments has been done. These methods generate a basic PRM and update it as new information becomes available (Jaillet and Simeon, 2004; Kneebone and Dearden, 2009; Marthi, 2010). PRMs generated in environments where the agent does not have full knowledge use a probability of the point selected being in free space or not (Missiuro and Roy, 2006). An outline of the basic algorithm can be seen in Figure 3 we use this algorithm in Chapters 5 and 6. A graphical description of the PRM generation process is shown in Figure 2.3.

Algorithm 3 The basic PRM generation algorithm. The connectToNeighbour method changes for each variation on the algorithm. See Algorithm 4 for the basic PRM algorithm that allows loops in the graph, Algorithm 5 for the variation we use in this thesis that does not include loops and Algorithm 6 for the greedy PRM approach.

```
listOfPoints ← Empty
listOfArcs ← Empty
currentPoints ← 0
while (currentPoints <maxPoints) do
  p ← generatePoint
  if isFree(p) then
    neighbouringPoints ← list of points within a minimum distance of p sorted by distance from p
    connectToNeighbour(p, neighbouringPoints)
    listOfPoints.add(p)
    currentPoints ++
  end if
end while
```

Algorithm 4 The connect to neighbour method for the standard PRM. This method allows for loops to be created in the graph. The clear path algorithm method checks if the straight line distance between the two points is free and able to be traversed.

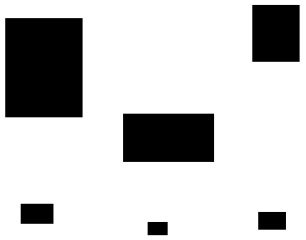
```
connectToNeighbour(p, neighbouringPoints):
for each q in neighbouringPoints do
  if clearPath(p, q) then
    arcs.add(p,q)
  end if
end for
```

Algorithm 5 The connect to neighbour method for the non looping PRM. This is the PRM generation method we used in this thesis. As each arc is calculated the graph is checked to see if the new point is connected to the root node. If this is the case then no other arcs are attempted. This leads to a simpler graph and allows for faster path planning.

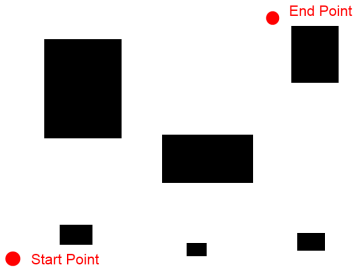
```
connectToNeighbour(p, neighbouringPoints) :
if listOfPoints.isEmpty then
  rootPoint ← p
end if
for each q in neighbouringPoints do
  if clearPath(p, q) then
    arcs.add(p,q)
    if p is connected to rootPoint then
      break out of for loop
    end if
  end if
end for
```

Algorithm 6 The Greedy PRM approach. Note in this case we do not necessarily need to add the point initially generated, instead the last point in free space alone the calculated arc is included.

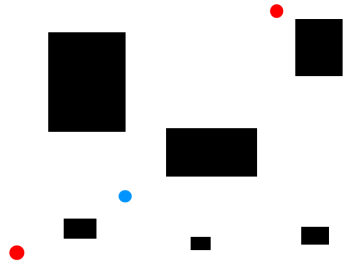
```
connectToNeighbour(p, neighbouringPoints)
for each q in neighbouringPoints do
  if (p.x != q.x) then
    m ← (q.y − p.y)/(q.x − p.x)
  else
    m ← 0
  end if
  //Calculate the equation of the line
  c ← p.y − (m ∗ p.x)
  x ← p.x
  while x<q.x do
    y ← m ∗ x + c
    if NOT isFree(x, y) then
      //go to previous point and add it
      x ← x − 1
      y ← m ∗ x + c
      points.add(x,y)
      arcs.add(p, new Point(x,y))
    end if
  end while
  //reached the point q
  arcs.add(p,q)
end for
```



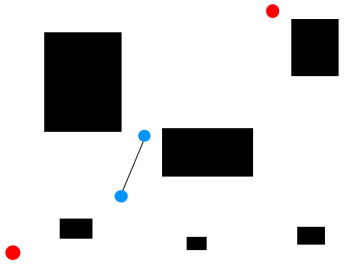
1. The initial environment. Obstacles are black areas and free space is white.



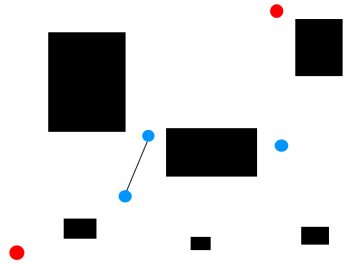
2. The start and end point are added to the roadmap.



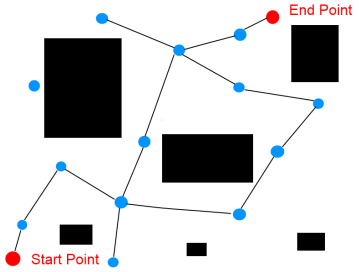
3. A point is selected in the environment, and if it is in free space, it is included in the roadmap.



4. Another point is added to the map. If there is a point in the graph within a given radius of the selected point then an arc is calculated between the two. If this arc is entirely in free space, then we add the arc to the roadmap.



5. Points continue to be added to the roadmap until some criteria is met. This can be after a set number of points, a period of time, or when both the start and end points are connected to the map.



6. The final roadmap consists of points and arcs. Unlike in the RRT, points may not be connected to the graph if there is no free arc available.

Figure 2.3: The steps of the PRM generation process

Many different factors affect the PRM generated (LaValle, 2006).

- Size of the environment
- Number of objects present
- Placement of these objects (clustered together or significant distances apart)
- Number of points
- Distance of arcs
- Method for selecting the points
- Method for determining if a point is free
- Whether the PRM allows for loops in the graph

A good roadmap will have good connectivity and coverage. Connectivity in a roadmap is a measure of whether the various sub-graphs in the PRM are connected to one another. Unlike the RRT algorithm, there is no requirement for each point to be connected to a single root node. This can lead to disconnected regions of the PRM graph. Coverage is the measure of how much of the environment is represented by the map. Good coverage means that the PRM covers all areas of the environment and so a route can be planned between any two points in the environment.

As the size of the environment increases more nodes will be required, or the distance that arcs can cross will need to increase to provide good coverage of the environment. This will increase the time required to generate the roadmap as more collision checks are required, especially when calculating whether a path is free between two nodes. The number of objects in the environment affects the PRM by increasing the number of collision checks required to add each point to the map. As collision checking is one of the most costly operations in roadmap generation (Geraerts and Overmars, 2002), we want to keep this as low as possible. The placement of objects affects the PRM through creating narrow confines the roadmap must find a path between. The narrow passage problem is one addressed by a number of different sampling strategies (Gaussian (Boor et al., 1999), Bridge Obstacle(Hsu et al., 2003; Sun et al., 2005) and OBPRM (Amato et al., 1998) to name a few). Narrow passages are often the cause of connectivity problems in a roadmap.

The method of selecting points changes how PRMs are generated. PRMs that use only point selection methods that generate points near obstacles will have a different shape and connectivity properties than

one that uses uniform sampling. The number of collision checks required for each point selection method is different, for example, a Gaussian point requires at least two points to be checked for collisions compared to a random point which requires only a single point. This leads to a difference in generation time. Whether to allow loops in PRMs changes the route planning stage rather than the generation stage of the roadmap. If loops are permitted in the representation, route planning can take longer.

The advantage to using PRMs is that they work in any environment and will, eventually, find a route between two points if one is possible. A 'complete' PRM algorithm is one that will either always return a path, or state that no path is possible, after a given number of iterations (Bohlin and Kavraki, 2000). As PRMs are an abstraction of an environment they are very fast to use in path planning algorithms.

The disadvantage to this is that if the PRM is to be used for more than one query, it requires good connectivity and coverage over all sections of the environment (Nieuwenhuisen et al., 2004; Sun et al., 2005). If this is not the case the map will be unable to be used to plan routes successfully. To ensure good connectivity the map can take a very long time to generate, especially in large scale environments (Kumar and Chakravorty, 2012a; Bahar et al., 2012). The points selected for inclusion in the road map are important for ensuring connectivity and often the probability of selecting the correct set of points to connect all regions is low (Yang and Brock, 2004; Park et al., 2012; Wardhana et al., 2012). In some cases the roadmap needs to be pre-computed offline before the agent is given use of it. This is usually seen where there are a large number of agents using a single map (van den Berg et al., 2008), or when the environment is very large (Nielsen and Kavraki, 2000; Nieuwenhuisen et al., 2004). Much of the existing research focuses on offline processing of a map, rather than online, due to time constraints (van den Berg et al., 2008; Bi et al., 2009; Marthi, 2010).

While the standard PRM algorithm checks each point and arc are in free space before adding it to the map, not all algorithms perform collision checking at this stage. The lazy PRM builds a map without any initial collision checking and only checks if an arc is viable if it is being used by the path planner (Bohlin and Kavraki, 2000). Variations on the Lazy PRM have been looked at for mapping dynamic environments (Poncela et al., 2002). In a worst case scenario, all the nodes will be linked to their neighbours and in the planning stage the entire road map will need to be checked for collisions. Normally however, only a subset will need these collision checks (Branicky et al., 2001, 2002; Jaillet and Simeon, 2004). The Fuzzy PRM is a similar method of PRM generation, which assigns a probability that an edge is free and then only checks this if the edge is then used by the path planner (Nielsen and Kavraki, 2000). Bahar et al. (2012) look at creating roadmaps using a grid based approach and do not include any arcs at the initial

generation stage. They then use an artificial neural network to connect neighbouring points and learn where not to attempt arc connections based on previous attempts. Other PRM algorithms do not add a fixed number of points, rather they continue to add points until the start and end points both belong to the same sub-graph (Bohlin and Kavraki, 2000). Greedy PRMs are a type of roadmap that add selected points in the environment and attempt to find an arc between that point the nearest point in the PRM, regardless of the distance between them.

McMahon et al. (2012) look at a different adaptation of the PRM algorithm. Rather than connecting each new point in the roadmap to its nearest neighbour they randomise this process. A set of the nearest k points are selected from the roadmap (where k is within a certain distance of the new point or a set number of neighbours). A single point is then selected from this set and an arc from the new point to the selected point is attempted to be included in the map. This reduces the number of duplicate arcs being checked for collisions during the generation phase allowing for a quicker PRM generation.

We consider one of the most important part of the construction phase to be how the points are selected for inclusion in the roadmap. A PRM should have good coverage over the environment, connecting all regions and allowing the agent to plan any path required. In environments where there are narrow corridors or cluttered objects this can be a difficult task for PRMs (Bohlin and Kavraki, 2000). As collision checking is usually one of the most costly operations in a roadmap building process (Geraerts and Overmars, 2002) we want to limit the number of points that are included in some way. Point selection methods that generate a large number of superfluous points will spend more time collision checking than may be necessary and will slow the map building process down. A higher number of points in the roadmap will also slow down path planning, as many more paths need to be considered. Many different point selection methods have been proposed. Each attempts to solve the narrow corridor and superfluous point issues in its own way. Kumar and Chakravorty (2012a) note that, in general, attempts to improve the point selection method in PRMs leads to an increase in the time required to construct the roadmap, but the savings come from needing fewer points overall to plan a single route.

2.4.3 Point Selection Methods

The most time consuming part of roadmap generation is the collision checking for each point and connecting arc against the objects in an environment (Geraerts and Overmars, 2002). This means that while some point selection methods may generate more useful points in an environment, the checking required to include a single point in the roadmap may be too high. Often point selection methods are reported to

be fast, but are only evaluated on smaller environments, with few objects (For example: Gasparri et al. (2009) who look at only a $20m^2$ environment).

Chapter 5 looks at how to mitigate this problem by using digital trails. We compare trails to some existing point selection methods evaluated below.

Random

The simplest method of point selection is to use a random number generator. The advantage to this method is that it is very easy to implement as it requires no environmental information. The disadvantage is that the points selected often lead to bad coverage and disconnected regions in the roadmap (Branicky et al., 2001). No information about the environment is taken into consideration when using random point selection, so in order to connect all regions a large number of points may be added to the road map not all of which will add new connectivity information (Missiuro and Roy, 2006).

Using only random sampling can lead to a low quality roadmap that plans 'ugly' paths that take strange routes when navigating between points. This is a particular problem in game AI, where believability of non-player controlled characters is important (Nieuwenhuisen et al., 2004).

Uniform Sampling / Regular Grid

Many methods of PRM generation use a form of uniform sampling (Missiuro and Roy, 2006; Kumar and Chakravorty, 2012a; Bahar et al., 2012). This generates points at constant intervals over the environment forming a grid. Grid based, regular sampling strategies ensure that the coverage of the environment is good (Geraerts and Overmars, 2002), but if the distance between points being selected is too low then a lot of superfluous points can be included in the roadmap without adding new information (Missiuro and Roy, 2006). This will then slow down path planning algorithms as they have a lot more points to consider. Uniform sampling is also sometimes used as a starting point for point selection in PRMs (Kumar and Chakravorty, 2012a). Additional points are then included or points moved to better reflect the environmental structure (Park and Chung, 2009).

Halton / Hammersley Set

To solve the potential coverage problem in random point selection we can instead use a point generation method based on the Halton and Hammersley set of numbers (Branicky et al., 2001; Geraerts and Overmars, 2002). These sequences use the division of primes to ensure that while the points are

selected unpredictably, they cover the entire space of available numbers more evenly, usually giving a better roadmap (Branicky et al., 2001, 2002). However, this does not solve the issue of how to generate points in narrow passages or around objects in highly cluttered environments.

Gaussian, Bridge Obstacle and Obstacle based point selection

These point selection methods all share a similar idea, they aim to generate points close to obstacles in the environment. Gaussian point selection is based on Gaussian blurring methods used in image processing (Boor et al., 1999) . It selects points for inclusion in the roadmap only in areas of the environment near blocked spaces. To do this two points are selected in the environment, one randomly and one a given distance away. The points are checked to see if they are free or blocked. If both are free or blocked the points are ignored. However, if only point one is free then this shows that this point is close to an object in the map, and so the point is included in the PRM (Boor et al., 1999).

A refinement of the Gaussian point selection method is the Bridge Obstacle method. This looks at three points in a line (Hsu et al., 2003; Sun et al., 2005). If the outer two points are blocked and the mid-point between them is free then this point acts as a 'bridge' between the two restricted areas of the environment and is included in the roadmap.

There are several other methods for point selection that have a bias towards narrow corridors. Kurniawati and Hsu (2004) segment the environment according to its geometric properties and bias point generation to where the generated segments are very small. Yang and Brock (2004) look at placing points along the medial axis of an environment, the points equidistant between objects. Both map generation methods have effect of biasing points to only areas with narrow passages. Nieuwenhuisen et al. (2004) look at something similar for game agent planning. They generate points by finding the mid-point between two obstacles and then attempting to connect this to the roadmap. The many collision checks required for these methods slow map creation.

The Obstacle Based PRM (OBPRM) (Amato et al., 1998) and Uniform OBPRM (Yeh et al., 2012) algorithms generate points near to objects in three dimensions. The standard OBPRM algorithm selects a single point in the environment and casts rays from this origin, adding points to the roadmap where the ray intersects with object boundaries. The Uniform OBPRM takes a similar approach but generates at random a number of origin points, ray directions and lengths, still adding points to the roadmap where the ray intersects with an object. Random point selection was shown to generate PRMs in a shorter period of time compared to Uniform OBPRM, but the authors are more interested in the position of

points being selected, rather than the overall speed of the algorithm.

Other PRM generation methods make use of the points that are found to be blocked, such as the Toggle PRM (Denny and Amato, 2011). This PRM algorithm aims to discover narrow confines by also generating a graph of all the points that are in blocked space. An example of this is when a point is selected for inclusion that is in a narrow corridor. Attempts to connect this point to others in the environment will be unsuccessful as the space is blocked. The Toggle PRM uses this information to try and find a midpoint between obstacles that is in free space, adding this to the roadmap. The Toggle PRM has been used in conjunction with Lazy PRMs to generate maps in complex environments that are able to quickly generate paths between two given points with a minimum of collision detector calls (Gasparri et al., 2009).

A general disadvantage for these biased methods of point selection is that they need to be combined with other methods of point selection to achieve a full coverage (Sun et al., 2005; Missiuro and Roy, 2006). Due to the numerous collision checks required before including a point in the roadmap, the cost of adding each point is increased and the overall map generation can be slower than more quasi-randomised methods.

Visibility PRM

The Visibility PRM generates points at random, but has a set of criteria for whether the point is added to the map (Simeon et al., 2000; Branicky et al., 2002). Points are only added if they either in entirely new areas of the environment, which have no neighbouring connections, or they connect two other points. This ensures that each point added to the map adds new information about the environment. This solves the problem of only adding useful and important points to the map. Only those which add new information are included. The disadvantage to this method is the time required in very large scale environments to find if there is a clear path between two points as all points are considered potential neighbours, not just those within a small radius.

Combinations

Biasing point selection based on the environment can speed up generation of the road map. Park and Chung (2009) use a combination of deterministic point generation methods which select points based on obstacles in the environment. While this approach can be successful in some environments, the deterministic nature of their algorithm may mean that the best solution and road maps may never be

found in environments that do not match the underlying assumptions made.

Hsu et al. (2005) examine an intelligent hybrid approach to PRM generation. Their approach uses a number of different point selection methods and chooses which method to use based on two factors: the average cost of selecting a point using that method and how much new information selected points will add to the map. The ratio of the least cost and most new information is used to decide which point selection method to run. This has a huge advantage in agents that need to adapt to new and changing environments as the most information is gained at each point selection step. Other researchers also combine point selection methods. Sun et al. (2005) combine bridge obstacle with regular point selection methods to improve on the roadmap built over pure random or Gaussian approaches. Thomas et al. (2007) acknowledged the fact that there is no single point selection method that works well in all situations and environments and so defined a set of metrics to evaluate points and deciding if a point should be included in a roadmap. After each point is generated and added to the roadmap it is classified as one of the following:

1. Create: New connected component
2. Merge: Point is a connection between two previously disconnected areas
3. Expand: The point increases the coverage of the roadmap by one sample
4. Oversample: The point adds no new information to the road map

Create and Merge point are considered to be the most useful as these add new information into the roadmap. 'Create'ed points are new areas that are not connected to the existing graph at all. 'Merge'd points are the connections between two or more sub-graphs that were not previously connected. 'Expand' points add some new information, they take an existing sub-graph and add a little more connectivity information to it. These are points that connect with only one or two existing points in the roadmap. 'Oversample' points do not add any new information, they connect points which were already connected in the map and so can be safely discarded.

Roadmaps are generated using a combination of different point selection methods. Each sampling method is evaluated after a point is generated and added to the roadmap. The maps are evaluated by looking at the type of points sampled, the coverage of the roadmap over the entire environment, the ability of the road map to solve a previously defined query and the number of samples generated in narrow confines. The results of these experiments show that a combination of point sampling strategies generates better maps that are able to plan a route between specific points in an environment using a

Table 2.1: Summary of point selection methods, their advantages and disadvantages

Point selection method	**Advantage**	**Disadvantage**
Random	Quick, simple	Bad point coverage
Halton / Hammersley Set	Quick, better point coverage than random point selection	Still may not select correct points to allow for complete coverage
Regular Grid	Simple, very good coverage	Can lead to many superfluous points and so have a high roadmap generation time
Gaussian / Bridge Obstacle	Generates points near and between objects	Does not generate points in empty space, requires more collision checking per point than other methods
(U)/OBPRM	Generates points near objects in the environment	Does not generate points anywhere else, fine in a greedy PRM not as useful for limited distances.
Visibility PRM	Only adds points where they add new information	Very long generation time due to potential neighbours being anywhere in the world

smaller number of points in the roadmap. The combinations of each point selection method are fixed, in contrast to Hsu et al. (2005).

2.5 Segmentation of maps in large-scale environments

The term 'large-scale' has different meanings for different agents, environments and contexts. Most research into map representation for intelligent mobile agents that exist in the real world currently takes place in small-scale, structured office-like environments (Fox et al., 2005). There are some exceptions to this, notably the original DARPA grand challenge to cross the Mojave Desert (Thrun, 2006), self driving cars on the road (Carnegie Mellon News, 2013) or underwater autonomous vehicles (Dearden et al., 2007).

Kuipers (1982) defines the size of the environment in relation to the agent observing it. An environment is considered to be large-scale if it requires more than one observation to gather all available information about the space. Another example can be found in work done by Sibley et al. (2010). The authors use the term 'vast-scale' to refer to their environment, based in the real world. One of the sample routes planned is 121km long, between London and Oxford. For this thesis, we consider a large-scale environment to be a continuous ever-expanding space in a virtual environment. Second Life consists of thousands of 256m^2 regions. In this thesis we look at regions 256m^2, 512m^2, 1024m^2 and 2048m^2 in size.

The scale of these environments can cause problems for conventional map representations. As the size of the environment increases so do the memory and computational requirements to store and process

the map (Tahirovic and Magnani, 2011). In this thesis one of our assumptions is that an agent will not be given infinite memory and time to complete its map representation. Some map representations are more compact than others (for example, object maps rather than occupancy grids), but in general a single monolithic map is not feasible for very large-scale environments without giving an agent unlimited resources (Wurm et al., 2011; Tahirovic and Magnani, 2011). Many systems prefer to do initial planning offline before the agent is in the environment, then adapt and refine this plan online as it travels from one point to another (Wenjian et al., 2009).

For these reasons large-scale environments are often split into individual segments, connected using a topological representation (Thrun, 1998b; Ramloll and Mowat, 2001; Choset and Nagatani, 2001; Kuipers and Levitt, 1988; Stoffel et al., 2008; Kuipers et al., 2004). Segmentation is the process of splitting a single map into several distinct sections (Ozkil et al., 2011). Segmentation divides an environment into more manageable sections allowing an agent to focus on its local surroundings rather than the entire environment reducing the time required to generate, update and use the map (Wurm et al., 2011; Wang et al., 2011). In general, segmentation is based on a set of landmarks in an environment used to identify specific places. A standard example is to look at the placement of doors in an environment to segment the map into specific rooms (Hawes et al., 2011). Other landmarks and cues have also been looked at such as the difference in air pressure on different floors of a building, allowing a 3D map to be generated (Ozkil et al., 2011). The disadvantage to using segmentation is that it can add complexity when switching segments or propagating changes through the map. Segmentation can cause non-optimal routes to be planned when the start and end point are in different segments.

This section looks at some of the various different segmentation methods used with maps of large-scale environments. We look in particular detail at Voronoi and quadtree segmentation as these are two we focus on improving in Chapter 7.

2.5.1 Grid Based / Regular Segmentation

Regular segmentation is one of the most simple segmentation methods. It splits the environment along a specific grid without taking into account any factors in the environment (Marthi, 2010). The main advantage to doing this is that the segmentation is fast, therefore cheap in terms of computing resources. The disadvantages come from the lack of attention paid to the environment. The segmentation may place the boundaries of a section in the middle of a room, meaning the agent needs to consider two segments when navigating that room rather than just one (Marthi, 2010).

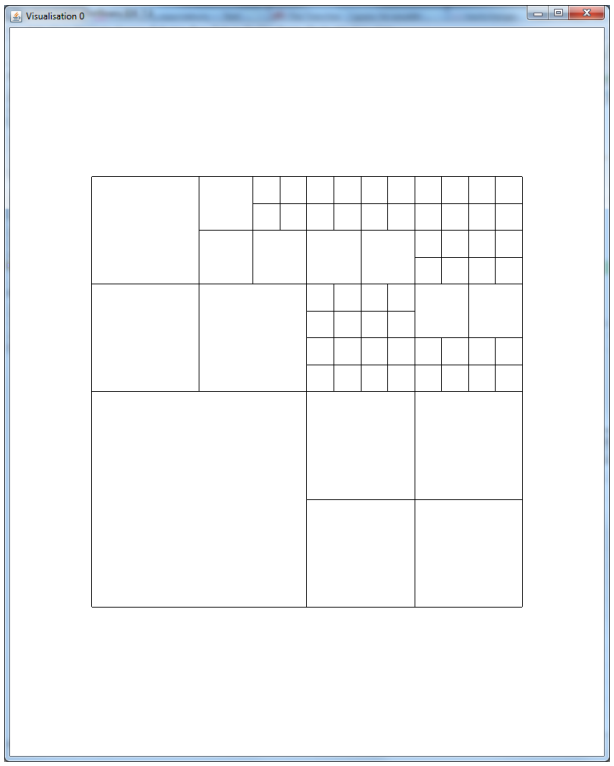


Figure 2.4: Quadtree segmentation method. The length of the side was set to 32m.

2.5.2 Quadtree Segmentation

Quadtrees are a mathematical concept that divides a space recursively. An image of quadtree segmentation can be seen in Figure 2.4. Initially the environment is one large square segment. If this segment is found to have any part of it blocked then it is split into four quarters as child segments. This continues recursively until either a minimum size for the segment is reached, or the entire map consists of segments that are only free/blocked space. The three dimension version of these is called the octree (Wurm et al., 2011). Due to the recursive tree like structure of this map, adding new segments is a trivial matter, as it just requires splitting a parent node into four children. Similarly, nodes will only merge with their immediate parent, making adaptations to the structure simple (Cocaud and Jnifene, 2010).

A variant of quadtree segmentation is the probabilistic quadtree (Kraetzschmar et al., 2004). This segments the map at varying resolutions, rather than always splitting a segment into four quarters each time. The resulting map representation then comprises a large number of very small segments around the boundaries of objects and larger free space segments in the middle. This helps the map scale to very large-scale environments while using less memory than a regular grid approach (Wurm et al., 2011; Zhang et al., 2012). However, in the worst case scenario, probabilistic quadtrees will still require more memory

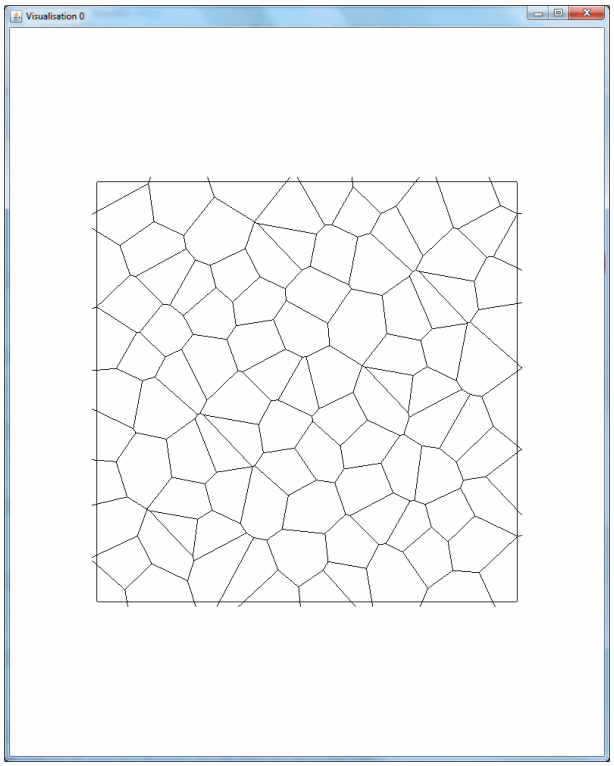


Figure 2.5: Voronoi Diagram generated using 100 Halton seed points.

than a regular grid based approach due to the recursive tree structure.

A general disadvantage of both the standard and probabilistic quadtree methods is the time required to generate the segments. This usually leads to it being used offline, in a pre-processing step before the agent has access to the environment or needs to plan (Shojaeipour et al., 2010). We only look at using the standard quadtree algorithm in Chapter 7.

2.5.3 Voronoi Segmentation

Voronoi segmentation is based on the Voronoi diagram. Voronoi diagrams are built from a set of seed points used to segment the environment such that any point in a segment is closer to the seed point at the centre of that segment than any other seed point in the map (Aurenhammer, 1991). How the seeds are positioned changes the Voronoi segmentation created.As this approach is concerned with relative distance to the center point of a segment as a method of segmentation, it does not matter if the seed point is in free space or not. . Figure 2.5 shows a Voronoi diagram generated by our software using a set of Halton seed points.

Generalised Voronoi Diagrams (GVD) are a map representation generated using a slightly different

method. Paths are generated from points along the medial axis of the environment; the set of points equidistant between two obstacles (Thrun, 1998a; Lindemann and LaValle, 2006; Choset and Nagatani, 2001). GVDs have then been combined with quadtree or octree representations to give hybrid map representations. First the environment is segmented using quadtree segmentation, then within each final segment a GVD is generated finding a path along the medial axis point between all obstacles. The advantages to combining these approaches for map representation is that path planning from start to goal point is very fast and the map can be used in both two and three dimensional environments. However, due to the quadtree segmentation step, the segmentation cannot be generated online as it takes too much time (Shojaeipour et al., 2010).

2.6 Digital Trails

This thesis looks at improving an intelligent virtual agent's map building process by including information gathered by observing and recording trails. A trail is formed when an agent in the real world moves from one place to another, leaving some evidence of their movement (Ruddle, 2005). These can often be seen by worn down paths left in grassy areas (for example Figure 2.6). Trails have been used by people for centuries to learn how to navigate new environments, find specific places and when hunting food. They are used to track the movement of people in history (Bradley, 2007) and they are still used in urban architecture to identify where people desire a path to be built in an environment, rather than where ones are provided (Myhill, 2004).

Digital trails apply this idea to virtual environments. We can observe and record the movement and activities of other agents, human or AI based, and from this identify a set of points in that environment and the path an agent can take to navigate between them.

The idea of using biologically inspired systems to aid agent navigation is not a new one. Li et al. (2013) look at localisation and mapping in dynamic environments using artificial neural networks and a model of short term memory in humans. Ant Colony Optimisation looks to ant pheromone trails for inspiration (Bi et al., 2009; Zhang et al., 2012). This section of the literature review looks at other research performed using trails and gives a more formal definition of the type of trail we will be using in later chapters.

We define a *digital trail* as a list of locations where an individual avatar has been observed over time. Virtual environments suffer from an idea known as 'extreme cleanness syndrome' (Grammenos et al.,

Figure 2.6: The worn down area in the long grass is known as a trail, photograph taken by the author in Toronto, 2012

2002). This means that avatars do not leave marks in their environment as they navigate between places. To identify a set of digital trails an agent needs to observe them in progress, rather than be trying to construct them after the event. Each set of trails is only relevant to the environment they were generated in.

Trails do not have to be generated using the movements of avatars. Antonya et al. (2011) use eye movements and gaze analysis to evaluate where a user intends an avatar to go and build up a set of trail-like paths based on this. The Game Trace Archive (Guo and Iosup, 2012) holds records of all interactions users have with game environments, not just avatar movements. Trails have also been used to try and add 'interaction history' to virtual artefacts and digital information (Wexelblat and Maes, 1999; Ruddle, 2009; Dixit and Youngblood, 2008; Grammenos et al., 2002). Jiang et al. (2010) look at clustering trails together to find similarities between individuals in virtual environments. Their aim is to find groups of people with similar interests, personalities and hobbies comparing trails based on the Euclidean distance of one trail from another and the individual start and end points. Trails have also been looked at in browsing histories (Ruddle, 2009), or when studying how people interact with learning resources (Bouvier et al., 2013) to find out how people navigate and process information.

In research, trails have been used to observe and aid human navigation in virtual and real environments

(Ruddle, 2005; Grammenos et al., 2006; Jiang et al., 2010; Miller and Crowcroft, 2009; Guo and Iosup, 2012; Stoffel et al., 2008). Trails have also been used as a starting point for a robots map building process (Yuan et al., 2010; Alempijevic et al., 2013). By observing the movement of avatars in an environment researchers have been able to study where potential busy locations (Miller and Crowcroft, 2009) and bottlenecks will be in an environment (Chittaro et al., 2006). Research has also been done in an attempt to understand the behaviour of people in virtual environments (Yee et al., 2007; Guo and Iosup, 2012; Alempijevic et al., 2013).

Trails are sometimes called 'player traces', 'human positional traces' or 'virtual footprints' (Grammenos et al., 2002, 2006; Guo and Iosup, 2012; Alempijevic et al., 2013). These have been used to identify how human users move around maps. Visualising game traces and trails is considered important for game and world designers (Joslin et al., 2006). Player traces have been studied to identify if an avatar is player or agent controlled (Chen et al., 2009; Ketkar and Youngblood, 2010; Chittaro et al., 2006). Many game environments do not allow agent controlled avatars to exist and so methods of detecting agent controlled 'bots' is important in these domains. Chen et al. (2009) show that there is a significant difference in how an agent controlled avatar moves through an environment when compared to a human controlled avatar. Humans have a defined set of behaviours that an intelligent mobile agent will rarely follow. Social norms are carried by people from the real world into the virtual (Yee et al., 2007). We explore this idea more in Chapter 3.

A real world application of trails can be seen by looking at GPS traces (Bruntrup et al., 2005) or large datasets such as OpenStreetMap (OpenStreetMap, 2013). These projects record the location of a Global Positioning System (GPS) device over time to produce a list of connected points. The GPS traces have been used to build incremental road maps of real environments and to infer where road sections meet (Fathi and Krumm, 2010). OpenStreetMap merges trails together to form a fully connected map of large outdoor environments. These systems rely on agents, usually vehicles, travelling from one point to another directly. The advantage of using vehicle based GPS trail points is that you will not get the meandering nature of humans on foot, the disadvantage is that in a vehicle the trails are restricted to roads rather than all routes through an environment. Human GPS trails have also been considered for aiding agent navigation in virtual copies of real world locations (Pouke, 2013). This study compared an agent using the human generation trails with an agent using a calculated navigation mesh. They found that when the GPS data was very good then the agent was able to plan a good route, but when the data was only partially available the agent was not able to plan routes as successfully.

2.7 Summary

In summary, this chapter looked at various different research areas linked by the themes of virtual environments, agents, trails and mapping. The open problems that we are looking at in this thesis is how to improve the generation of maps in large scale virtual environments.

We looked into the background behind virtual worlds, giving a background for our research. We then looked at map representation and generation for agents. The type of map representation we are interested in for this thesis are used for navigation. There are many different types of map representation, but there is no single 'best' map to use in all situations that will allow an agent to be able to quickly plan a route in any environment. Some maps are not useful for navigation, such as the conceptual map, which concerns itself more with labelling an environment and understanding the underlying concepts of an area. Path planning in large scale or changing environments is an NP-Hard problem (Bi et al., 2009). Metric map representations are the simplest type of map, built directly from sensor data giving a geometric representation of the environment. However, the problem with these type of maps is that they can be slow to generate and take up a lot of space in memory. Topological maps are generated from metric representations to overcome this problem. As topological maps are abstract representations of the environment they are more compact representations. They are faster for planning navigation, especially in the type of large scale environment we are concerned with. We focus in this thesis on a type of topological map, roadmaps, including the PRM and the RRT.

Roadmap representations are fast to generate and use for route planning when compared with metric representations of the environment. These are the key points we are interested in for our agent. A roadmap representation will, if given unlimited time and resources, be able to be used to plan a route between any two points in an environment if one it is possible to do so. However, having unlimited resources is not a realistic proposition and so in this thesis we look at how to improve the generation of roadmap representations using trails.

It is our hypothesis that we can use trails to overcome this problem, and in Chapters 4 and 5 we look at improving both the RRT and PRM using trails. Trails give the agent a set of points in the environment that are known to be in free space and paths between them that are possible to traverse. By being able to use this information without the need for calculating collisions and accessibility of the paths, our hypothesis is that trails will improve roadmap generation over a variety of metrics.

As the environments we are looking at are large scale, we also looked at some of the literature related to segmenting maps. This is a standard approach used to reduce the map generation and path planning

time in agents. We looked at regular, quadtree and Voronoi segmentation, the methods that we compare and attempt to improve on in Chapter 7. Selecting a good segmentation method, or a good set of seed points for use in a Voronoi segmentation is an open problem in large scale environments.

The next chapter looks in more detail at trails, with the initial aim of ensuring that trails were a viable source of information for intelligent virtual agents.

CHAPTER 3

Trails and social norms in virtual environments

3.1 Introduction

The hypothesis driving this thesis is that trails can be used to help intelligent virtual agents build better maps of large scale virtual environments. As previously stated in Chapter 2, a trail is formed when an agent in the real world moves from one place to another, leaving some evidence of their movement (Ruddle, 2005) (see Figure 2.6 as an example of a trail). Virtual environments do not usually record this information, a so called 'extreme cleanliness syndrome' (Grammenos et al., 2002). Short term exceptions to this rule exist in some game worlds. For example, in the large scale game environment World of Warcraft Blizzard Entertainment (2013) the corpses left from monsters killed by the players stay present in the world for a set period of time allowing others to see the path an avatar has taken. After this has expired the evidence is removed from the environment so to prevent clutter. For this reason trails recording the movement of avatars in a virtual environment are usually only able to be observed while the movement is in progress. It is harder to infer a trail after the fact. We define a *digital trail* as a set of observations of the location of a particular avatar over time. This set of locations can then be collated to form a trail. Digital trails have previously been used to aid human and agent navigation in both real and virtual environments (Ruddle, 2005; Bruntrup et al., 2005; Fathi and Krumm, 2010; Yuan et al., 2010;

Alempijevic et al., 2013).

We want to use trails to aid intelligent virtual agents build maps of their environments. One of the most costly operations in map generation is collision detection. This is the act of deciding if a certain point is available for an agent to move into to, or if the space is blocked by an obstical. By observing another avatar navigating an environment we are able to infer points in the environment that are in free space and the routes that can be used to navigate between them without the need for any collision detection calculations. There is the added advantage that by using human controlled avatar generated trails in map generation, the type of routes planned by the agent will be more human-like. This is important in some applications, but not an effect we assess in this thesis.

To investigate whether trails could be useful for an agent, we first needed to establish there was a pattern to people's movements in virtual environments and that these patterns related to the structure of the world. We can then use the trails observed to identify paths and potential routes through an environment. We are only interested in ground level movements for our agent, if the prevalent mode of transport used in virtual environments is flight then this will not generate useful trails for the agent.

This chapter looks at the evaluation domain for the rest of this thesis, based on Second Life. It goes into detail about the regions used and how we gathered the object and trail information that made up the environments. To establish whether trails are a useful source of information for an agent we look at two movement based patterns from avatars:

1. Did the avatars walk when navigating from one point to another, or use alternative modes of transport?

2. Are the movements the agent is able to observe meaningful?

The contribution of this chapter is a brief study into the movement of avatars in virtual environments and validation that trails are a useful source of information for agent based navigation. This study does not form the main focus of the thesis, but poses some interesting questions about social norms in virtual environments that may be of interest to others. This chapter is based on our paper published in the British Computing Society Human Computer Interaction conference in 2012.

> **Katrina Samperi, Russell Beale, Nick Hawes: Please keep off the grass: individual norms in virtual worlds. BCS-HCI '12 Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers, BCS-HCI 2012, 12-14 September 2012, Birmingham, UK: 375-380** (Samperi et al., 2012)

The remainder of this chapter is structured as follows: First we look at our evaluation domain in Section 3.2. Then we discuss our preliminary study and the results of our observations of trails in virtual environments. We then expand on these observations in Section 3.4. Section 3.5 looks at the results of our study, and finally, Section 3.6 looks at what these results mean in the wider context of this thesis.

3.2 Evaluation Domain

To evaluate our agent's performance in large scale, populated virtual environments we elected to look at one specific virtual world, Second Life (Linden Research Inc., 2012). Second Life is an established virtual environment comprising of thousands of $256m^3$ areas, known as *regions*. Regions can be connected to one another allowing free passage between them, or arranged individually as an island with a virtual sea on all sides. Unlike some virtual environments, Second Life allows and encourages agent based avatars in their world (known as bots). The LibOpenMetaverse library has been created and maintained, allowing agents to interact with the environment, as if they were any other user (Open Metaverse Foundation, 2012).

There are three primary modes of transport in Second Life: walking, flying and teleportation. We are interested in ground level, walking movements for our agent, so we filter the trail and object information made available by the environment to ground level. To filter out flying agents we applied a threshold to the coordinates an avatar was observed at. If the z coordinate was outside a given threshold ($20 < z \leq 25$) then this point was not recorded as part of the trail. To filter out teleportation we looked at the time stamp and distance travelled between points. If the distance was greater than it should be possible to move in that time period, then the trail was split. This led to a set of trails that described individual routes taken by agents, removing any incorrect assumptions made disconnection or teleportation. Using this route information gave us a more accurate picture of how an avatar was moving. Figure 3.1 shows an example of a full trail before it was split into routes. The points are where observations were made and the lines link all the points in the order they were seen. Note the long lines where the avatar has teleported from one side of the environment to another. This is the type of information we removed by filtering the trails.

When connecting to a Second Life region the agent is sent a list of objects that exist in the environment, which can be used to calculate if a point is in free space or not. To collect trail information the agent polls the server for the location of every avatar within the same region it is occupying every half a second.

Figure 3.1: All the points where a single avatar was observed linked by the order in which they were seen. This trail information does not take into account avatars disconnecting from the Second Life grid, or the avatar teleporting.

This information is freely available. It is usually used to update a map of the environment showing the position of every avatar in the environment at any given time.

Each region uses its own coordinate frame, between (0,0,0) and (255,255,255). To give the agent larger environments to work with we combine the object and trail maps together, shifting the x and y coordinate of each object or trail point by the required amount to give a single, unified map.

We focus on one area of Second Life in the remainder of this thesis, the "London area". This is an area made up of five individual regions.

1. London Community
2. Hyde Park
3. Kensington
4. Knightsbridge
5. Mayfair

London Community is a sparsely populated, open environment. This region is available seasonally. When we gathered trail information for the roadmap and segmentation experiments this environment was set up as a 'winter wonderland' style area for winter 2012.

Hyde Park is the most populated region, known as a 'new user hub'. When an avatar is created it goes through a tutorial on a separate part of the Second Life grid. After this point the user gets to choose a hub to go to. This area has a high traffic throughput of both brand new and older avatars. It has a large number of objects available. The structure of the environment is an open park like area. There are areas of grass with paths crossing between distinct locations in the environment.

Kensington and Knightsbridge are both a combination of open and more structured space. The structured areas are roads lined with shops. The difference between these environments and Mayfair is that the roads do not necessarily follow a grid structure. These environments contained more objects than Hyde Park but they had fewer avatars active in the area.

The last region is Mayfair. This region is a highly structured environment with a high number of objects available and a higher traffic throughput than Kensington and Knightsbridge. We do not use this environment as part of our four-part Greater London Area. The other four regions make up a square and so together are more directly comparable with a single region environment. The total number of objects and trail points gathered in each region can be seen in table 3.1.

Table 3.1: The total number of objects and trails in each environment:

Environment	Objects	Trail Information Avatars	Total Trail Points
London Community	1476	23	921
Hyde Park	7096	4274	402036
Kensington	11770	61	2730
Knightsbridge	10201	183	10486
Mayfair	11765	1583	84475

We chose to look at this environment as we needed a populated area of Second Life in order to gather enough trails to be meaningful for our evaluation. An added benefit is that these regions are fairly flat, with ground level always being between 20 and 25m, meaning it was easy to tell if an avatar was flying, rather than walking at ground level.

The Greater London Area we specified as the four region area of London Community in the north-west, Hyde Park in the north-east, Kensington in the south-west and Knightsbridge in the south-east. An image of this can be seen in Figure 3.2.

3.3 Preliminary study

The aim of our preliminary study was to look at whether meaningful trails could be gathered from virtual environments. We did not know initially whether the information would be available to request from the server or if we would need to have our agent follow others to learn their movements. We also wanted to look at whether the trails observed had any patterns to them that we could generalise and use for our future agent's map building. We looked at the Hyde Park region of Second Life. We chose this region as it is a high population environment with avatars of all ages. As a park, the environment has a lot of open space as well as paths that cross between distinct locations.

The agent connected to the virtual environment and recorded the location of every avatar[1] within a single region every half a second for thirty hours, starting on the 24th February 2010. We evaluated the preliminary study by visual inspection of the trails. We mapped the trail points gathered onto the overhead map to try and determine if there was any pattern to the movements made by avatars (see Figure 3.3 for a single trail and Figure 3.4 for all the trails gathered).

From this set of images we concluded that we were able to gather trail information in Second Life.

[1]While it is likely that most or all of the avatars observed in Second Life were controlled by humans rather than by agents, we have no way to being sure of this fact. The only indicator given for if an avatar is controlled by an agent is a flag which can be set, however, there is no requirement to do so.

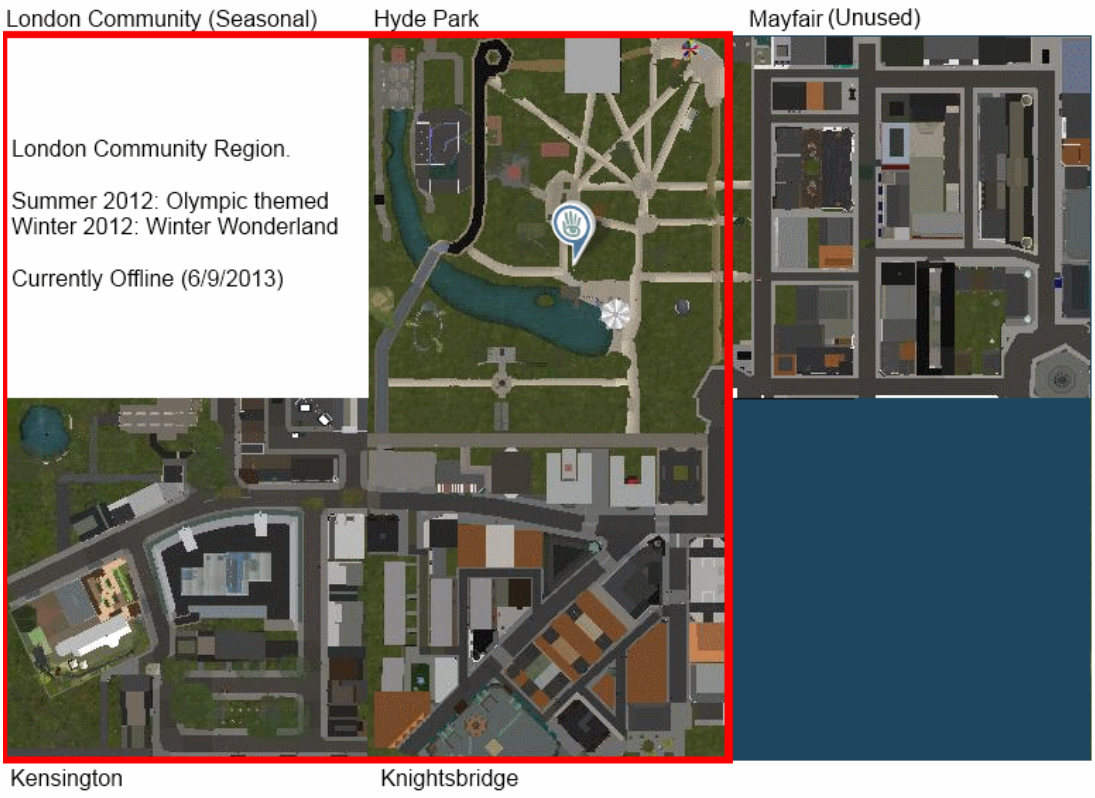


Figure 3.2: Four out of the five regions that make up the "London" area used in our experiments. The Greater London Area used in Chapter 4, 5 and 7 is made up of London Community, Hyde Park, Kensington and Knightsbridge (area in red). We do not have an image of London Community at this time as the region is seasonal and it is not currently available.
As of the 6th September 2013 London Community is offline and Kensington has moved to the bottom right hand position, all the regions have undergone several changes in layout.

We were able to identify the route an avatar took when navigating between places, identify the popular locations in the environment and where paths connected these. The Second Life server and libraries allow an agent to observe and record the routes every avatar in a region takes at once and then use this to infer free space and path information.

One pattern we observed in the preliminary study was that avatars had a habit of using paths to navigate between points in the environment, rather than take a more direct, straight line route. This is a known social convention in the real world. People tend to avoid walking on the grass for many reasons, for example:

- A sign stating that the grass is not to be walked on
- A desire to avoid wet or muddy shoes
- Implied social pressure to follow the default behaviour of the crowd, and so stay on the paths
- To preserve a fragile grass environment for the shared enjoyment of others

These reasons do not hold in virtual environments. There is no sign saying to keep off, and there is no cost associated with 'wet or muddy' shoes. This transference of social norms from the real world into the virtual has been noted in other areas of research. Previous studies, for example Schroeder (2002), Yee et al. (2007) and Friedman et al. (2007a) have shown that social conventions and human norms are carried over into virtual environments. Interpersonal distance between avatars and eye movements when scanning a virtual environment are much the same as habits observed in real world interactions. In game based virtual environments, such as World of Warcraft (Blizzard Entertainment, 2013) analysis of trails can identify where paths exist in a subsection of an environment (Miller and Crowcroft, 2009). This gives us good reason to believe that the movements avatars take in a virtual world would mimic those taken in a similar, real world environment.

We decided to investigate this occurrence further and repeated our trail observations a second time looking at whether the experience in virtual environments, measured by avatar age, changed the probability of an avatar breaking social convention and walk on the grass.

3.4 Social Norms in Virtual Environments

This investigation was based on the following hypothesis: the more time users spent in virtual environments, the more experience they would have, the more likely the user would be to break with social norms.

Figure 3.3: A single trail observed in the Hyde Park environment.

Figure 3.4: All the trails observed in the preliminary experiment.

There is no explicit cost to walking over the grass in virtual worlds, there is no sign in the environment we studied to tell avatars to keep off the grass. There is only the social convention that, unless invited to, you should stay off the grass.

This time when observing the movement of avatars through the environment we also had our agent record each avatar's date of birth. The date of birth relates to when the avatar was created and so gives some insight into how long that user has been interacting with Second Life. This is not an ideal metric, but without the ability to question every avatar in detail about their experience, it has been considered the best metric available (Friedman et al., 2007b).

3.4.1 Evaluation

As stated previously, our preliminary study observed the movements of avatars over a 30 hour period starting the 24th February 2010. Our second study observed 775 avatars over a 60 hour period in the same environment on the 31st May 2012, collating over 69,000 locations.

During both sets of observations the Second Life servers went offline to restart in the early hours of the morning. Connection was re-established as soon as possible for continued observation. Avatars were identified using their unique ID and their location was only recorded when it was different from the previously recorded location. Avatars which chose to hide their date of birth have not been included in this evaluation.

We used the overhead map to identify where paths existed in the environment. Our first inclination was to automate the identification of paths in the environment based on the colours used (see Figure 3.5). However, this did not work as well as expected due to the range of colours provided in the image. We then elected to annotate the image by hand, drawing a block of a single colour over paths identified by sight (see Figure 3.6). It was then possible to match the coordinate given by the avatar location to a specific pixel in the image, and decide whether this location was on or off the path.

Our hypothesis was that as the age increases we should see a positive correlation with the percentage of off-path movements. To investigate this our evaluation calculated the percentage of observed locations on and off the path for each avatar and plotted this information against the age of the avatar.

We also looked at the percentage of locations for each avatar that were above ground level in comparison with the age of the avatar. As flying is a more challenging mode of transport than walking in the environment our hypothesis was that we would see older avatars using flight more than newer avatars.

Figure 3.5: Map of the current Hyde Park region of Second Life

Figure 3.6: Annotated map of the current Hyde Park region of Second Life, areas blocked out in gray are identified as paths

3.5 Results

Our results can be seen in Figure 3.7 and 3.8. . While observing the avatars movements through the environment we confirmed our previous observation. Avatars tend to stay on the paths more often than walk on the grass. An example of this can be seen in Figure 3.9. This demonstrates the route taken by an individual avatar in the space. The yellow lines mark the trail, and most of the movements shown are superimposed on the lighter grey of the paths. The trails show a curved route following the path where a straight line over the grass would clearly have been quicker and there is nothing in the way to prevent it.

Two notable exceptions to this exist. The first is the centre of the environment. This is where all new avatars are teleported to on arrival. In the preliminary study this area was paved over but in this study it has been converted to an area of grass surrounded by a path. This lead to more observed movements off the path. The second is an interesting case, the bottom right corner of the map in Figure 3.5 shows an open grassy area. Visualising the trails we can see a lot of movement on this patch of grass (Figure 3.10). We can find the explanation for this behaviour by looking closely at the environment. Figure 3.11 is a screenshot overlooking this area of the Second Life region. We can see that a stage has been built and is used for Second Life based concerts and parties, inviting avatars onto this apparent grassy area to celebrate and enjoy themselves.

We can see that there is a slight positive correlation between the age of the avatars and the percentage

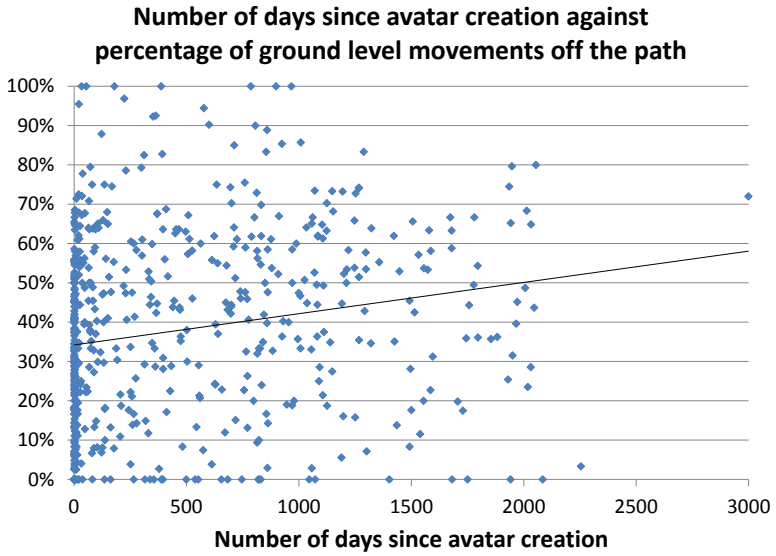


Figure 3.7: Results for the age of the avatar vs the percentage of their ground level movements that are not on a path

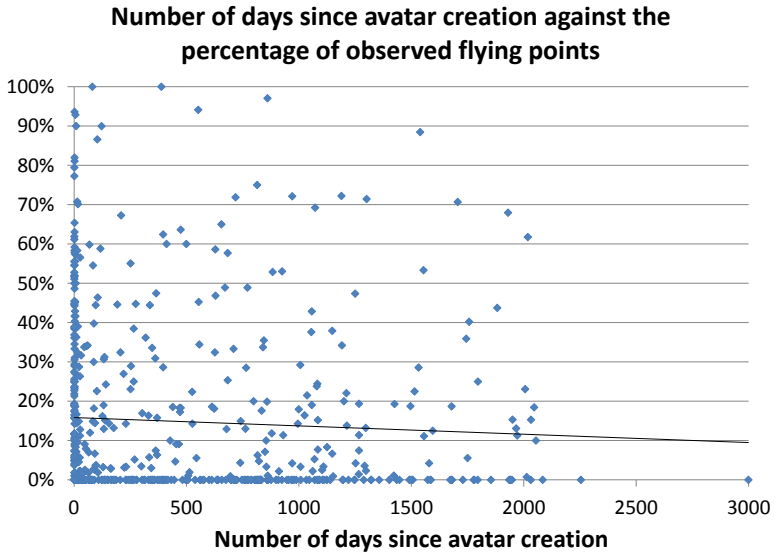


Figure 3.8: Results for the age of the avatar vs the percentage of observed flying points

Figure 3.9: A single avatar's movements in the environment

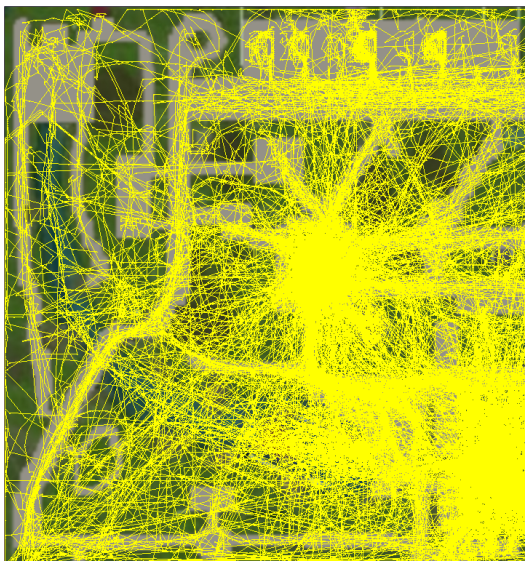


Figure 3.10: The movements of all the avatars in the 2012 observation

Figure 3.11: Second Life screenshot. Stage area on the grass in virtual Hyde Park. Note the overhead map in the top right. The avatar in this case is a cat.

of their movements off the path (see Figure 3.7). This is less apparent when the agents are grouped together in the table. To determine the significance of this result we split the avatars into new avatars (those 6 days old and younger) and old avatars (7 days and older). The average time spend on the path by the new avatars was 60.40% and the average time spent on the path for the older group is 47.38%. The significance of this was assessed using the Wilcoxon rank sum test and was found to be significant ($p<0.04$), suggesting that younger avatars spend more time on the paths than older ones, but this should be treated with caution since the age in Second Life is suspect anyway, and the division used here is somewhat arbitrary, and other choices may give non-significant results. Visual inspection of the collated trail information in Figure 3.10 shows that we can identify free space and popular regions of the environment. Flying was not a popular mode of transport, most movements were at ground level.

3.6 Discussion

This chapter looked at the evaluation domain used in the rest of this thesis. We also set out to look at two patterns in agent movement:

1. Did the avatars walk when navigating from one point to another, or use alternative modes of transport?

2. Are the movements the agent is able to observe meaningful?

Our results showed that the majority of agent movement was walking, at ground level. Flying was found to be an unpopular mode of transport around the environment. The age of the avatar did not affect the movement type used by the avatar. The trails we observed were meaningful and in fact followed known social conventions from human movements in real environments. Our preliminary study established that we could a) gather trails (though the action of doing so), b) that these trails could be used to identify free space in the environment (as there is no way for an avatar to be observed in a location unless it is in free space) and c) suggests that there was a high percentage of avatars staying to the paths when navigating the park.

We discovered that there was a (slight) positive correlation between the age of the avatar and the percentage of their movements off the path. The older an avatar was, the more likely it was to navigate over the grass when travelling between places.

The results gathered by our second study highlight the limitations of our approach. The only measure we had for experience with virtual worlds was the avatar's 'born on' date. This does not give any indication

as to how often the avatar has been used, whether the user behind the avatar has any experience with other virtual worlds or if this is their only avatar. We also do not know the motivation behind the avatar movements. Not stepping on the grass is a social norm for people in the United Kingdom. We had no physical location information for any of the people controlling the avatars, so we have no idea if the same social norms hold. Second Life is also used for role-playing games. So if a person is attempting to mimic real behaviours they may be forcing themselves to stick to social convention. Trails observed are not guaranteed to provide useful information about an environment. Some avatars move directly from one point to another in the environment, giving useful information about the path between two distinct locations. Other avatars may move around only small areas, giving cyclical paths in a corner of the environment. Some avatars make very small movements, generating a large number of points without adding any additional information

For this thesis we only wanted to investigate if trails were possible to gather and whether they would give us any understanding of the structure of the virtual environment they were observed in. We have shown this to be the case. Trails are a reliable way of inferring free space in an environment and even where paths may exist in the world. We can therefore use trails to help guide an agent's map building process. Trail information continues to be meaningful across a variety of avatars of different ages and experiences within the environment and so this allows us to consider maps built by an agent based on trails to be a valid option.

The next chapter will look at map building for intelligent mobile agents and how we can use trails to improve on the generation of roadmaps in large scale virtual environments.

CHAPTER 4

Improving Rapidly Exploring Randomised Tree generation using trails

4.1 Introduction

As the size and complexity of virtual environments increase it becomes more of a challenge for an agent to generate a map of a previously unknown environment in a short period of time. Path planning in large scale environments is considered to be a difficult problem for intelligent agents (Yang and Brock, 2004). To solve this, topological representations such as roadmaps are used to represent the world and for planning. The abstract nature of topological and road maps give an advantage over metric representations for planning due to the simplification of the environment. An environment is reduced to a set of nodes (places in an environment the agent can visit) and arcs (how to navigate from one node to another). This reduces the path planning problem to a simple graph traversal, rather than needing to consider all the objects in the environment as it plans.

In this chapter we look at improving the *generation step* (sometimes called the *construction phase*) of Rapidly Exploring Randomised Trees (RRT). RRTs are a popular type of roadmap used when an agent needs to generate a single plan from a known start configuration (LaValle, 1998). An RRT relies on the fast and random growth of a search tree that spans the environment from a starting root node. Every point in an RRT is connected to the root either directly or by traversing a set of branches, so a path

between any two points in the tree is always possible. Section 2.4.1 goes through in detail the various different algorithm and point selection methods for RRTs.

While the random growth of branches in the RRT allows the tree to explore a whole environment, in time critical situations the length of time needed to generate the tree and use this to plan a path between the root and the goal configuration may be too long to be useful. RRTs are generally used to plan a single route before being discarded. If the environment changes a new tree is required to be regenerated, though some research has been done to try and limit this effect (Ferguson et al., 2006).

We are interested in agents that exist in large scale online environments. Specifically, we look at environments that are based in Second Life, a large scale, persistent and populated environment. It is critical that the agent is able to plan a route quickly and accurately. A prolonged delay may cause the agent to be disconnected from a virtual environment. In Second Life the time an avatar can spend idle before disconnection is 30 minutes. However, we want the agent to be able to plan and navigate routes as quickly as possible when faced with a new environment. The 30 minute time out should be considered a hard maximum time limit, with times between under 2 minutes preferable.. The agent should be capable of planning a route as required while still connected to the environment. The fast growth of an RRT means that this type of map representation is a useful one for our agent. RRTs can also be used in conjunction with other road map representations to connect start and end points to an existing road map.

The hypotheses driving this chapter are:

1. Tails can be used to reduce the generation time needed to build an RRT which can then be used to plan a route between two given points in an environment

2. The routes planned by the trail based RRTs will be shorter than those planned using other RRT point selection methods

Trails provide contextual knowledge about the environment by giving a set of points in free space and the paths taken by avatars between then. They show where avatars tend to congregate, giving information about important or large empty spaces in the environment. We covered trails in more detail in the previous chapter, Chapter 3. It is possible to bias the growth of an RRT into populated areas of the environment by looking at where avatars cluster together. This has the potential to produce an RRT capable of planning a route between two points in a shorter time period and requiring fewer nodes than purely random tree growth. The contribution of this chapter is to examine trails as a new source of information for generating RRTs in both small and large scale virtual environments.

The rest of this chapter is structured as follows: First in Section 4.2 we discuss our approach to using trails in RRT generation, and the different RRT generation algorithms we will be looking at. Section 4.3 looks at how we evaluate our experiment and the experimental setup used. Following that Section 4.4 reports our results and finally, Section 4.5 summarises our findings and future work we can do in this area.

4.2 Approach

We cover the details of the RRT algorithm in Section 2.4.1, Algorithm 1. To improve on the RRT generation phase we need to look at how the basic algorithm works. An initial starting point is required for the tree, this is the *root node*. From this node branches are grown by selecting points in the environment according to some criteria. The nearest point in the existing tree to this selected point is then found. A branch is grown from the tree towards the selected point and if both the branch and the branch end point are collision free they are added to the tree. This process continues until an end criteria is met, usually a branch is grown within a given distance of an end point. The method we use to select points in the environment can therefore affect the growth of the tree.

We look at three types of RRT. The first is the traditional RRT, which selects a point in the environment and grows a branch towards this point from the original tree a predetermined distance. The second type is the Greedy RRT (Kuffner Jr and LaValle, 2000). The branches in a Greedy RRT are not constrained by distance. They continue to grow in a given direction until they either reach the point or find an obstacle. This leads to slower growth, but in an empty environment a single branch is all that is needed to connect any two points. The final type is the RRT-Connect (Kuffner Jr and LaValle, 2000). This approach uses two trees, one growing from the start point and one from the end. At each step a branch is added to both trees until they grow close enough to be connected. This is considered one of the better RRT generation methods available by the authors. However, the RRT-Connect algorithm requires that a single end point is known in advance for the second tree to use as its root, speculative tree growth is not considered.

A good RRT is one that is able to plan on average a *short path* between a given start and end point in the *fastest time* possible. A short path uses the minimum distance required to connect two points. If there are no obstacles between points then this would be a straight line. However, in cluttered environments the shortest path will be longer than this distance. We consider the most important of

these factors to be the time taken for the algorithm to complete. The primary advantage of using RRTs is that it is a quick algorithm for discovering a path between two points. It should add a *minimum number of nodes* to the tree, as this will reduce the number of collision checks that are needed and the time required for path planning. We want to compare the standard, randomised method of RRT growth with a quasi-randomised and a biased approach, looking at trails to bias growth towards more populated areas of the environment.

A standard randomised approach to point selection uses a random number generator to select points in the environment. This has the advantage of being fast, but an even spread of points over the environment is not guaranteed. In comparison we can use a quasi-randomised point selection method, based on the Halton sequence (Geraerts and Overmars, 2002). Using the Halton Sequence ensures that the RRT algorithm uses unpredictable point selection whilst also ensuring a good spread over the environment. If the points generated are not spread over the entire environment, then the tree growth stagnates and the RRT is unable to be used to find a solution to the path planning problem.

The third method of point selection we are looking at uses trails to bias the point selection method. Any number of trail points can be used to bias the growth. A random point is still chosen, but whether this point is used to grow a branch is based on the number of trail points that exist within a certain radius. If the number of neighbouring trail points is higher than a given threshold then we use this random point to grow a tree branch (See Algorithm 7). So the algorithm does not stagnate, there is still a chance that a randomly selected point without any neighbouring trail points will be used. The probability of keeping a point is 10% plus an additional 10% for each trail point within a 10m radius. If a point is selected with no neighbours there is only a 10% chance it is kept, but if the point has nine trail points nearby then the probability of keeping the point is 100%. We vary the number of trail points used in the experiment as it is not clear if there will be a single 'best' number of trail points to include, or if there is a saturation point, after which the inclusion of more trail points does not help the RRT growth. A higher number of trail points may in fact be detrimental to the RRT growth due to algorithm needing to check every trail point to see if it is a neighbour of the randomly selected point in question.

The type of environment we are interested in looking at is large scale, populated and online. The agent should be given a set of objects that make up the environment and trails observed there. The agent should be able to calculate free and blocked space from the objects and detect clusters from the trail information in a variety of ways. Collision detection for points in the environment is needed for the agent to identify if a location is in free space, and so is available for using when planning a route. The

specifics of how free space is identified is an implementation requirement, rather than a scientific one. We cover the environment and how trails are gathered and selected in Section 3.2.

Algorithm 7 Algorithm for biasing point selection in an environment near to trail points

Require: $listoftrailpoints$, iterations i
for $x = 1 \rightarrow i$ **do**
  p = random point
  $neighboursList$ = list of trail points within 10m of (p)
  $n = neighboursList.size()$
  r = random number between 0 and 100
  $chanceToKeepPoint = 10 + (n * 10)$ # (10% Change of keeping point plus 10% for each neighbour)

  if ($r < chanceToKeepPoint$) **then**
    return p
  end if
end for

4.3 Evaluation

We wanted to investigate the effect that different RRT generation methods have on environments of different sizes. To do this we looked at two environments discussed previously in Section 3.2, the small scale, Hyde Park and a large scale environment, the Greater London Area. These are the two environments described in the previous chapter. Hyde Park is a 256m^2 environment that contains both free open areas of grass and some more structured space such as paths and buildings. The Greater London Area is an environment twice as large at 512m^2. It consists of Hyde Park and three other regions surrounding it. Images for the two environments can be see in Figure 4.1, more details can be found in Chapter 3 on page 42

We used three RRT algorithms: the standard RRT, the Greedy RRT and the RRT-Connect algorithms. These algorithms were first described in Chapter 2.4.1 on pages 17 and 20. We have repeated the algorothms here for clarity within this chapter. The standard algorithm had a maximum branch distance of 10m, and can be seen in Algorithm 8.

The Greedy RRT was allowed to continue growing branches until it reached the point selected or an obstacle, the algorithm for this approach can be seen in Algorithm 9.

The third algorithm, the RRT connect, grows two trees at the same time. One from the start point and one at the end, the algorithm ending when the trees meet. This also had a maximum branch distance of 10m applied. This algorithm can be seen in Algorithm 10.

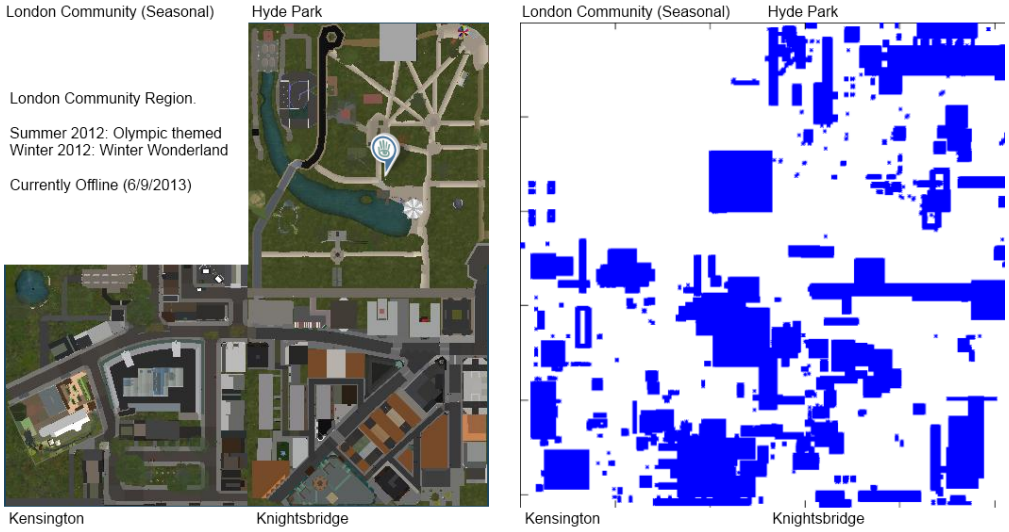


Figure 4.1: The environments used for the experiments. The smaller, Hyde Park region only uses the top, right hand corner of the environment. This is a 256^2 environment. The Greater London Area includes Hyde Park and three other regions. At the time of writing there is no image available for the London Community Region. The image on the right hand side is the collision map for the environments. This shows the free (white) and blocked (blue) space in the environment.

Algorithm 8 The basic RRT generation algorithm

Require: Start point (s), End point (e), $listofpoints$, $listofbranches$
Add (s) to list of points
while list of points does not contain a point close to (e) **do**
  Select a random point (p)
  Find the nearest neighbouring point (n) in the tree to (p)
  Calculate the point (p') a given distance from (n) towards (p).
  if n is free AND there is a clear path between the points (n, p') **then**
    add (p') to the list of points
    add (n, p') to the list of branches
  end if
end while

Algorithm 9 The Greedy RRT generation algorithm

```
1: Require: Start point (s), End point (e), listofpoints, listofbranches
2: Add (s) to list of points
3: while list of points does not contain a point close to (e) do
4:    p ← Selected point
5:    n ← nearest neighbour in the existing tree
6:
7:    //Using the equation of a line, calculate m and c for the line connecting the points n and p
8:    if p.x != n.x then
9:       m = (n.y - p.y)/(n.x - p.x)
10:   else
11:      m = 0
12:   end if
13:   c = p.y - (m * p.x)
14:
15:   addedSomething = false
16:
17:   //go along the line testing each point one at a time.
18:   newx ← n.x
19:   while not addedSomething do
20:      newx ← oldx + 1
21:      newy ← m * newx + c
22:      if newy is not outside bounds of the environment then
23:
24:         if not isFree(newx,newy) then
25:            //point was blocked, go to previous point and add this as the end point and branch
26:            point ← oldx, oldy
27:            add (point) to the list of points
28:            add (n, point) to the list of branches
29:            addedSomething ← true
30:
31:         else
32:            //x,y coordinate is in free space
33:            if x coordinate at edge of environment then
34:               point ← newx, newy
35:               add (point) to the list of points
36:               add (n, point) to the list of branches
37:               addedSomething ← true
38:            end if
39:
40:         end if
41:
42:      end if
43:      oldx ← newx
44:      oldy ← newy
45:   end while
46: end while
```

Algorithm 10 The RRT Connect generation algorithm

Require: Start point (s), End point (e), $ListOfPointsInTreeOne$, $ListOfBranchesInTreeOne$, $ListOfPointsInTreeTwo$, $ListOfBranchesInTreeTwo$
Add (s) to $ListOfPointsInTreeOne$
Add (e) to $ListOfPointsInTreeTwo$
$connected \leftarrow false$
while not connected **do**
  Select a random point (p)

  // Grow branch in Tree One
  Find the nearest neighbouring point (n) in TreeOne to (p)
  Calculate the point (p') a given distance from (n) towards (p).
  if n is free AND there is a clear path between the points (n, p') **then**
    add (p') to the list of points for TreeOne
    add (n, p') to the list of branches for TreeOne
  end if

  // Grow branch in Tree Two
  Find the nearest neighbouring point (n') in TreeTwo to (p)
  Calculate the point (p'') a given distance from (n') towards (p).
  if n' is free AND there is a clear path between the points (n', p'') **then**
    add (p'') to the list of points for TreeTwo
    add (n', p'') to the list of branches for TreeTwo
  end if
  for each point in ListOfPointsInTreeOne **do**
    if there is a point in TreeTwo within a given distance **and** there is a clear path between the two **then**
      $connected \leftarrow true$
      $FinalListOfPoints = ListOfPointsInTreeOne + ListOfPointsInTreeTwo$
      $FinalListOfArcs = ListOfBranchesInTreeOne + ListOfBranchesInTreeTwo + ConnectingArc$
    end if
  end for
end while

The same approach for point selection was then applied to all three algorithms. The first point selected for the tree to grow towards was the planning end point. Then points were selected according to some method for use in the algorithm. After every 200 points the planning end point was again selected. The choice of 200 points is arbitrary and any number could be used. By including the end point in the selected points we ensured that there was at least some bias towards the planning end point, and so the algorithm successfully being able to plan a route. After each point is used to grow a branch in the tree a check is made as to whether there is a branch end point within a given radius of the planning end point. If this is the case the generation phase of the algorithm is completed, and a path planned using the tree. If not then further points are selected and more branches grown. After the generation phase has been completed, the agent plans a path between the start and end point.

The start point for the RRT in the small scale environment was (0,0) and the end point (255,255). A tree was then grown from one corner of the environment to the opposite corner. In the larger environment we used (0,0) and (492, 453) as there was no available route between the two corners in this environment. A* route planning Russell and Norvig (2003) is then used to determine the final route.

The biased RRTs were given a number of trail points with which to bias the growth of the tree. To test whether there was a saturation point for the usefulness of trails we compared the difference 1000, 2500, 5000, 7500, 10,000, 12,500, 15,000, 17,500 and 20,000 trail points made to the tree growth.

The trail information was observed and gathered previously. How the trail information was gathered and split into individual sets for use is covered in Section 3.2.

All the experiments are run on a single computer, under as close to the same circumstances as possible. The machine was an Intel(R) Core(TM)2 Quad CPU (2.40GHz) running Scientific Linux version 2.6.32 with 2GB RAM.

4.4 Results

The full results tables can be seen in Appendix A. We focus on the best result for each set of experiments in this section, comparing the three different point section methods for each of the three algorithms.

4.4.1 Hyde Park

The results for these experiments can be seen in Tables 4.1, and the graph in Figure 4.2.

Trees built using the trail bias method planned the *shortest* path for all three RRT generation algo-

Normal	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	**3234.50**	429.88	**74.60**	7.72	1.75	1.41	388.10	14.21
Halton	3269.15	639.08	75.35	16.76	1.75	1.26	392.09	22.69
Trail Bias - 7500 Points	3303.50	420.92	75.65	10.01	1.60	1.16	389.87	26.22
Trail Bias - 10000 Points	3527.35	765.33	79.35	16.32	1.70	1.45	**382.81**	16.03
Greedy	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	**3376.60**	539.62	77.55	11.43	1.55	1.24	390.25	20.06
Halton	3549.15	723.07	80.85	12.82	1.60	1.74	389.44	20.04
Trail Bias - 10000 Points	3420.15	692.76	**75.65**	15.89	1.90	1.41	**382.05**	15.91
Trail Bias - 15000 Points	3387.45	682.33	77.65	11.85	1.60	1.56	389.98	19.73
RRT-Connect	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	3325.65	590.44	76.00	12.09	1.55	1.40	389.09	16.20
Halton	**3244.55**	558.87	**75.20**	13.39	1.55	1.28	380.97	17.81
Trail Bias - 5000 Points	3374.70	724.22	75.70	14.98	1.60	1.77	388.03	17.52
Trail Bias - 7500 Points	3490.05	1042.46	79.90	21.23	1.65	1.74	**380.10**	20.30

Table 4.1: Results of the RRT experiments in Hyde Park. We only report two trail bias methods, the one with the shortest generation time and the one with the shortest planned path. Full results can be found in Appendix A. All times are in milliseconds, the length is in meters. Values in bold text are the best for that metric.

Normal	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	161761.45	38257.79	756.35	180.80	26.75	22.81	905.92	55.61
Halton	138370.25	16129.70	665.25	78.03	19.05	11.91	891.36	54.00
Trail Bias - 15000	**81622.00**	17986.10	**345.80**	82.61	7.45	4.73	**883.84**	65.51
Greedy	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	93520.90	79735.06	240.15	195.31	6.30	6.24	1038.31	150.03
Halton	74502.80	53991.81	**190.95**	132.17	5.20	5.10	1109.75	239.86
Trail Bias - 15000	**51103.80**	27110.77	237.10	79.88	5.25	2.72	981.27	133.59
Trail Bias - 17500	62167.80	32290.47	310.95	187.36	5.40	5.07	**957.42**	105.97
RRT-Connect	Total Time	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length	Std Dev
Random	59404.90	22046.29	257.85	90.10	6.35	4.19	937.58	89.38
Halton	**49410.05**	11083.17	**222.50**	49.09	4.90	2.88	876.21	47.63
Trail Bias - 1000	69209.80	29290.38	294.10	112.13	7.05	4.68	873.97	75.84
Trail Bias - 10000	85835.40	40193.31	372.90	176.33	7.40	5.67	**870.65**	73.93

Table 4.2: Results of the RRT experiments in the large scale Greater London Area. We only report two trail bias methods, the one with the shortest generation time and the one with the shortest planned path. Full results can be found in Appendix A. All times are in milliseconds, the length is in meters. Values in bold text are the best for that metric.

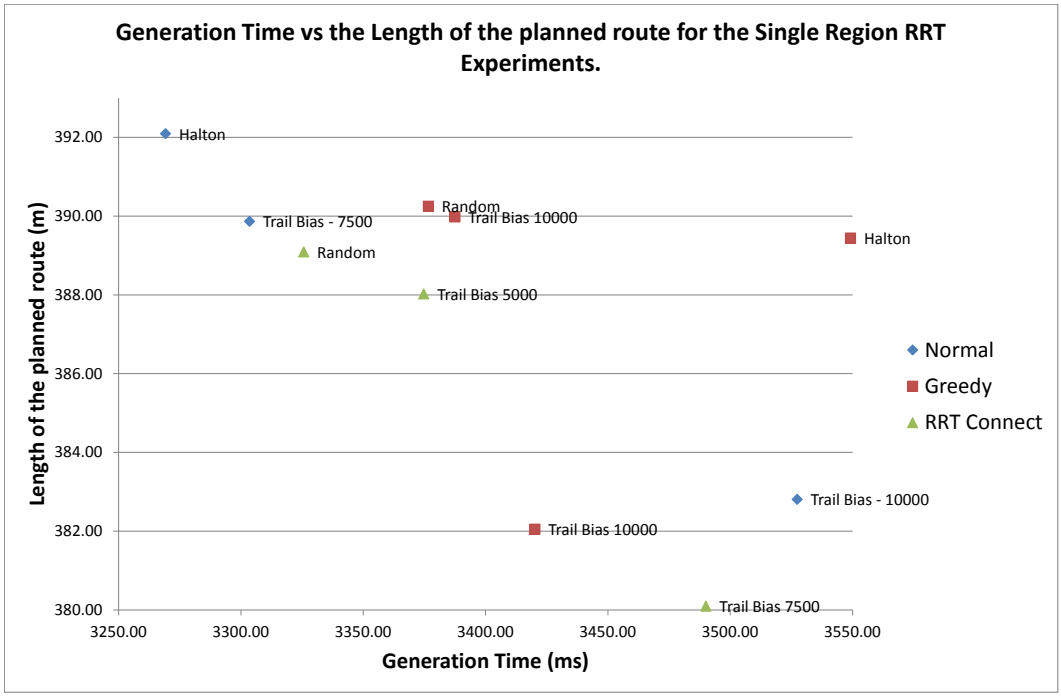


Figure 4.2: Scatter plot of the length of the planned route v.s. the generation time required in the single region (Hyde Park) experiments. The further towards the bottom a point is, the shorter the route planned. The further towards the left hand side, the faster the generation time. Ideal results will be in the bottom left hand corner of the graph.

rithms. This is due to the human optimisation effect inherent in trails. However, these methods were not always the fastest. Random point selection generated the tree in the shortest period of time for both the normal and greedy RRT algorithms. These both planned longer routes than the trail biased RRT. When using the RRT-Connect algorithm the fastest point generation method used Halton generated points. The better spread of points enabled the two trees to grow together faster and generate reasonably short route. Generally, the planning time correlated with the number of nodes in the tree. Where fewer points were required linking the root and end goal, the planning time was short.

4.4.2 The Greater London Area

In a large scale environment it is important to select points with care, to ensure that the tree is able to reach its maximum potential. Trail based point selection approaches worked better here than in the single region environment. The results can be seen in Table 4.2 and the graph in Figure 4.3.

When using normal or greedy point selection the fastest tree generation method biased the growth using 15,000 trail points. For the normal RRT algorithm, 15,000 trail points also required the least nodes to be added to the tree and generated the shortest path. However, when using the greedy RRT algorithm, the shortest route planned used 17,500 trail points. This shows that trail points are beneficial to tree growth when using these two algorithms. The exact number of points required to get both the fastest generation time and the shortest route can fluctuate, depending on the trails available.

When using the RRT connect method of tree generation the quasi-random Halton sequence performed the best. A better spread of points over the environment ensured that the tree was able to be generated in a short period of time, with a low planning cost. However, the shortest planned route was still achieved by a trail biased tree using 10,000 points.

The success of the RRT-Connect algorithm comes from being able to grow two trees at once. When doing this it is essential that the points selected are spread evenly around the environment. Using trails to bias these points still caused the trees to stagnate rather than enabling them to grow towards common areas from different directions. When using trail points there is always the chance that that concentration of trails may not be in areas where the agent needs to plan a route between. In this case the agent will need to select points carefully from the trail set available to grow a tree towards. Trails again planned the shortest path in all three RRT generation algorithms.

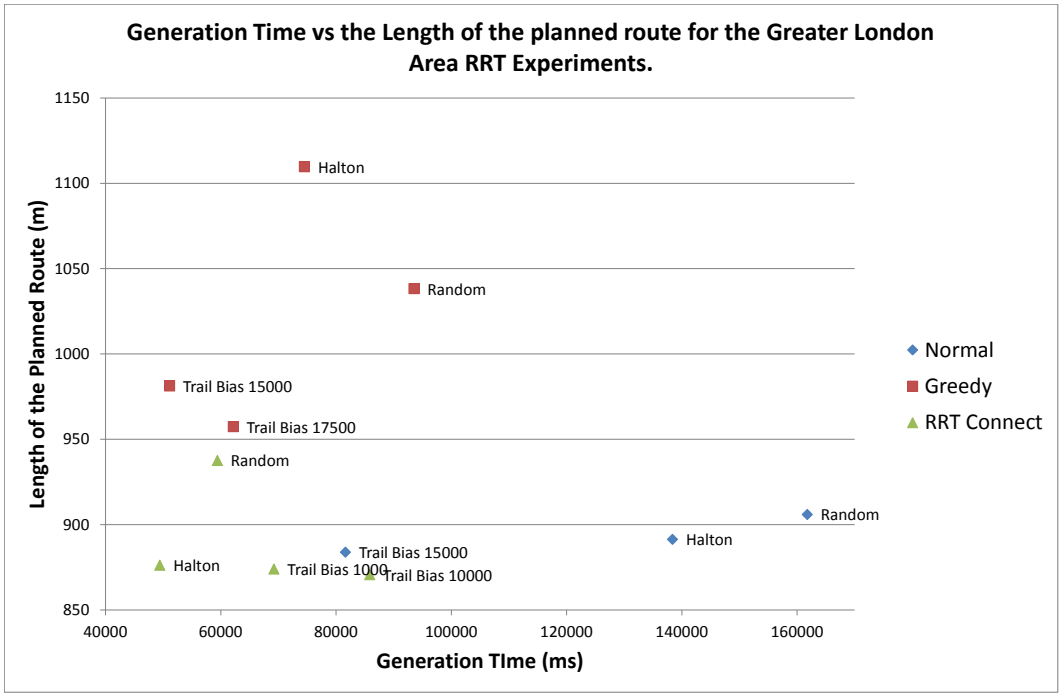


Figure 4.3: Scatter plot of the length of the planned route v.s. the generation time required in the large scale (Greater London Area) experiments. The further towards the bottom a point is, the shorter the route planned. The further towards the left hand side, the faster the generation time. Ideal results will be in the bottom left hand corner of the graph.

4.5 Discussion

The hypotheses behind this chapter were that:

1. Tails can be used to reduce the generation time needed to build an RRT which can then be used to plan a route between two given points in an environment

2. The routes planned by the trail based RRTs will be shorter than those planned using other RRT point selection methods

In general we found that trails did improve the generation of RRTs. We used trails to bias the growth of the tree towards popular areas and paths in the environment. In small scale environments the shortest generation time is achieved using random or Halton point selection. The better spread of points selected over the environment gave an advantage to these point selection methods. A set of trail points nowhere near the start or end point may detract from the tree growth process as it causes the RRT to expand in an unnecessary direction. The goal of the RRT algorithm is to use randomness to grow the tree, enabling branches to spread around obstacles in the environment. Trails guide growth towards the popular areas, but not always towards the end point required. Despite this, the shortest planned path in all methods used the trail bias method. The human optimisation effect in trails means that the trees generated allowed for shorter paths to be planned using all three RRT generation methods.

In large scale environment trails outperform random methods when using normal or greedy RRT generation. It is important that as the size of the environment increases that the correct set of points are selected when growing the tree. When using the RRT-Connect algorithm the Halton point selection method led to faster generation of the tree. Trails again planned the shortest path in all three RRT generation algorithms.

The first hypothesis was partially correct. In some cases trails did reduce the time required to generate an RRT capable of planning a route between two points in both environments. This holds more accurately for large scale environments than small scale ones. Hypothesis two always held. The shortest routes were planned by trail biased RRTs, this held for both small and large scale environments, and whichever RRT generation algorithm was looked at. There is further work to do in this area. One option we want to investigate is to look at ERRT waypoints (Bruce and Veloso, 2002). These bias tree growth towards a set of points in an environment identified before the tree is grown. There is potential for the cluster points in trails, the areas where many avatars have been observed, to be useful in selecting waypoints.

We also only used the raw trail information during this experiment. One method of improving upon this is to look again at clustering methods for the trail points. By merging or clustering trail points together we keep only the useful information from the trails, reducing large groups of points down to a single important location (we explore this idea more in Chapter 5 for Probabilistic Roadmap generation). These locations may be better used to bias the tree growth towards the important areas in the environment and so generate trees better in cluttered and restricted environments. Combining point selection methods may also give better results, combining the better spread of random methods with the biasing of trails towards known paths in the environment.

CHAPTER 5

Improving Probabilistic Roadmap generation using trails

5.1 Introduction

In Chapter 4 we looked at improving RRTs using trails. This chapter looks at a different type of roadmap representation, the Probabilistic Roadmap (PRM). The primary difference between the PRM algorithm and the RRT is that the PRM generation phase does not need to take into consideration the start or end point as the map is built. PRMs are designed with reuse in mind. They were first proposed as a method of describing the interconnectivity of free space in an environment. An agent can then use this connectivity information to plan a path between two points in an environment (Kavraki et al., 1996). The application of a PRM consists of two phases, the *construction phase* and the *query phase*.

In the construction phase, points are chosen and connected to their neighbours if a path (typically a straight line) through free space exists between them. A point is considered to be a neighbour to another point in the roadmap if they are within a certain distance of one another. This produces a graph of interlinked nodes. A PRM does not need to be fully connected, there is no requirement for every point to connect to a single *root* point like in the RRT. The query phase then uses this graph to plan a path between any two points in the environment. A brief outline of the construction phase can be seen in Section 2.4.2, Algorithm 3.

One significant disadvantage to using PRMs in large scale or restricted environments is the time required to generate a map capable of planning a route between a given set of start and end points. In restricted environments, such as those with narrow passages, the correct selection of points is required to ensure all parts of the environment are connected. This process can take a long period of time. Also, large scale and complex environments contain many obstacles, increasing the number of collision checks required to add a point. This again can take a long period of time. We require our agent to be able to generate a map and plan routes using this as quickly as possible.

In this Chapter we examine the effect of using trails points on the generation of PRMs in different virtual environments. Our goal is to determine the effects of using raw and processed trail information on the point selection method of the road map. We formed the following hypotheses for investigation:

1. By using trails we can reduce the time required to generate a PRM more than other point selection methods

2. By using trails we can increase the success rate of the agent at planning routes in an environment. It will be able to plan more of a given set of routes when using trails compared with when using standard PRM point selection methods.

3. By using trails when generating the PRM the agent will then be able to plan shorter routes than if it was using a PRM generated using other point selection methods.

These hypotheses were formed based on the idea thatOne of the reasons we believe these hypotheses will be true is that including a trail point in the roadmap and connections between two subsequent points in a trail requires no collision checking, and this reduces one of the major sources of complexity in a standard PRM.

This chapter is based on our paper published in 2013:

> **Katrina Samperi, Nick Hawes, Russell Beale: Improving Map Generation in Large-Scale Environments for Intelligent Virtual Agents. The AAMAS-2013 Workshop on Cognitive Agents for Virtual Environments, Springer, May 2013** (Samperi et al., 2013a)

The contribution of this chapter is a new source of information for quickly building PRMs. We looked at how the combination of a number of different point selection methods changed and improved the overall map generation and discovered that, in both small and large scale environments, trail based PRMs outperformed current methods over a variety of metrics.

The rest of this chapter is structured as follows: Section 5.2 describes our analysis and approach to investigating generating PRMs using trails. We cover different methods of using trails in the PRM generation algorithm and the results of our preliminary experiments. Section 5.3 covers the experiment details and how we evaluate our results. Following this Section 5.4 looks at the results of our experiments in small and large scale environments. Finally, Section 5.5 discusses our findings and future work in this area

5.2 Approach

A good PRM is one that is able to be generated in a *short period of time* and can *successfully* and *quickly plan short routes* between a given set of points. It will be well connected, ensuring that all areas of the environment are covered by the graph. The application of a PRM is a two stage process, the first stage is the generation stage. The second is to use the map generated to plan routes. If the map size increases then the generation time also increases, we want to limit the effect the size of the environment has on the map by changing the point selection approach used. A short path in this situation is the minimum distance that the agent will travel when navigating between two points. This is not the same as the straight line distance between the points as obstacles will change the route the agent must use. We want to minimise the distance and so compare the results against the shortest generated path in each case.

As we want to compare the difference made by the point selection step of the PRM, we chose to look at the standard PRM generation algorithm. This is the algorithm first described in Section 2.4.2 and has been reproduced here as Algorithm 11. While the lazy PRM can be faster to generate, it relies more heavily on the planning algorithm used when selecting which points and arcs to test when planning a path between two points (see Section 2.4.2 for more details).

The agent is given an object map of the environment, a set of trail points and a set of points to plan paths between. Using these, the agent is required to generate a PRM and use it to plan a set of routes. The agent uses A* route planning Russell and Norvig (2003) over this entire chapter. We measure the time taken to build the roadmap, the time taken to plan the path, the success the agent has at planning and the length of the subsequent path. We want to compare a number of different point generation methods, and combinations of trail and non-trail based methods.

Algorithm 11 The basic PRM generation algorithm. The connectToNeighbour method changes for each variation on the algorithm. See Algorithm 4 for the basic PRM algorithm that allows loops in the graph, Algorithm 5 for the variation we use in this thesis that does not include loops and Algorithm 6 for the greedy PRM approach.

```
listOfPoints ← Empty
listOfArcs ← Empty
currentPoints ← 0
while (currentPoints <maxPoints) do
  p ← generatePoint
  if isFree(p) then
    neighbouringPoints ← list of points within a minimum distance of p sorted by distance from p
    connectToNeighbour(p, neighbouringPoints)
    listOfPoints.add(p)
    currentPoints ++
  end if
end while
```

5.2.1 Using Trails in PRMs

Trails can be used to help generate PRMs in a number of ways. The simplest is to use trail points directly for point selection in the PRM algorithm. The advantage of doing this is that during the generation phase of the PRM no collision checking is required to ensure that points are in free space; the act of observing an avatar at that location tells us it is free. It is also possible to connect contiguous trail points without collision checking against known obstacles, thus including the full route an avatar was observed taking. The disadvantage to using only trail points is that the map generated will only show the connectivity of space where other agents have been observed, this can cause problems for the agent when planning routes (Pouke, 2013). We solve this by combining PRM point selection methods to try and cover all scenarios. This technique has been looked at by Hsu et al. (2003), the authors compare the combination of numerous methods of point selection to find the best combination to use in any situation.

We combine different PRM point selection methods by choosing a certain percentage of points according to each different selection method and then building the roadmap using these. The order in which points were added to the map is the order given when describing the point selection methods used. Trail or Merged Trail points were added to the roadmap first, followed by Gaussian then Halton points. Combining several methods of point selection should, in theory, provide all the advantages of using trails, but will still allow paths to be planned through sparsely visited areas of the environment.

Another method of using trail information is to use it to bias whether or not a randomly generated point is added to the road map. This is similar to the tree growth bias we looked at for RRTs. First a point is randomly selected, the probability of adding this point in the map is directly correlated with the

number of known trail points nearby. All points have a small chance of being added to the map, but the more trail points in the neighbouring region, the higher the probability that the selected point will be included. This produces a map where points are biased towards popular regions of the environment, and much more sparsely generated elsewhere. The disadvantage to this method is we lose any benefit gained by not needing to perform collision checks when adding nodes and arcs to the roadmap.

5.2.2 Preliminary Experiments

We ran a set of preliminary experiments to look at which of the various factors in our experiment were important. We looked at PRMs generated with a range of maximum number of points being selected for inclusion in the roadmap. During these investigations we discovered that on occasion using the raw trail information could lead to very high PRM generation times. This is due to the nature of trails. Avatar movements are not always between two distinct points in space, generating many points in a small area, while not adding any new information. It was clear that we needed to filter some of the trail information, keeping the important points and arcs, but removing some of the clutter that slows down the map generation.

To overcome this, we developed a method to merge trail points together, and thus only include points that add new information. Our method for doing this is similar to that used by methods such as the Visibility PRM (Simeon et al., 2000), the Hybrid PRM approaches (Hsu et al., 2005) or path averaging (Pouke, 2013). The trail information is processed before being included in the roadmap. This involves iterating over the trail list and if two points are within a given small distance of one another, all references to the points are merged together. This results in a single point in the graph with all the arcs still linking it to neighbouring points giving a skeletonised version of the trail map with fewer points and links, but these points all contain information about connectivity and free space in the environment. Figure 5.1 shows a PRM generated using the raw trail information, while Figure 5.2 shows the same set of trails post-merger. Algorithm 12 shows the algorithm used to merge the trail points. In our results we include this pre-processing time in the PRM generation time for a fair comparison.

There is a disadvantage to merging trail points. When reducing and the number of points and arcs some information will be lost. Points selected for inclusion in the roadmap that do not use trails will need collision checks performed. These points may have been original trail points, but removed in the merging process. Likewise, some connectivity from the arcs is lost as points are merged together.

Algorithm 12 The algorithm used to merge trail points

Input: List of routes comprising all the trail information
Input: Distance at which two points will be considered neighbours

```
for all routes in the trail information do
  for all trail point (p) in the route do
    listOfNeighbours ← all points within a radius of r to p from the trail information

    if listOfNeighbours > 1 then
      nearestNeighbour ← the closest neighbour to p
      // Replace all references to the neighbouring point with p
      for all route in trail information do
        for all trail point (q) in the route do
          if q =nearestNeighbour then
            q = p
          end if
        end for
      end for
    end if

  end for
end for
```

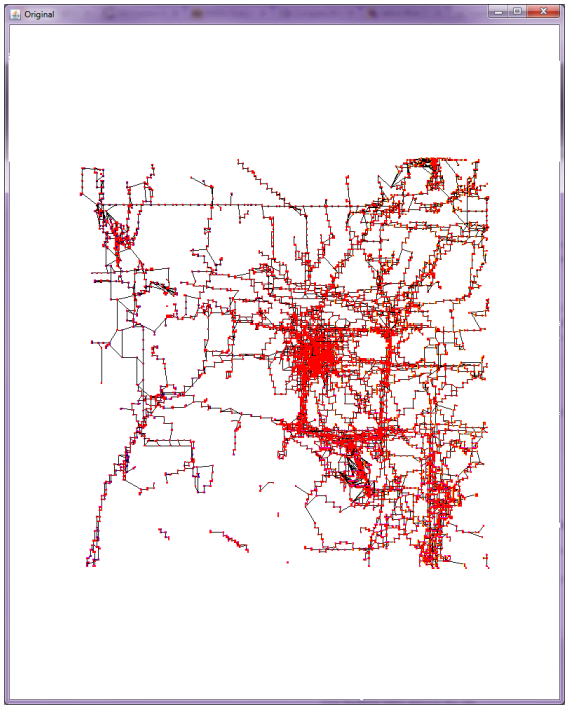


Figure 5.1: PRM generated from 100% raw trail data. The raw data produced a map with 6991 unique points and 15689 arcs.

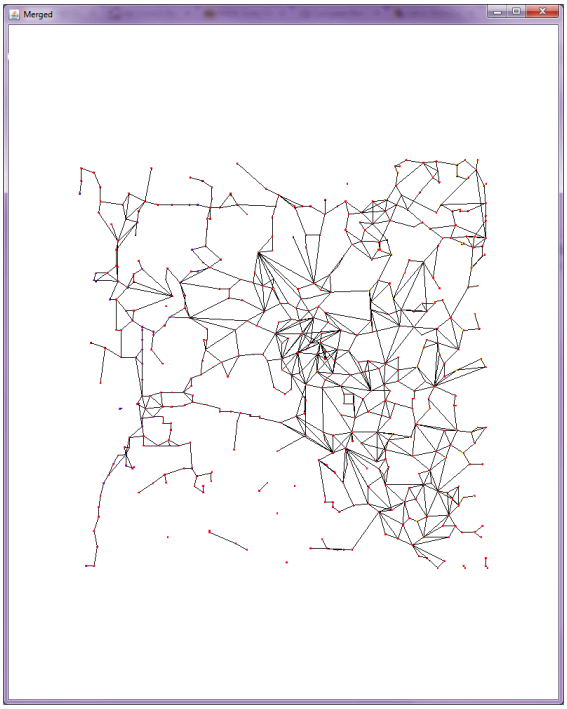


Figure 5.2: PRM generated from the same trail set as Figure 5.1 but merging the points together to include only the useful information from the trails. This map retained 523 unique points and 9262 arcs.

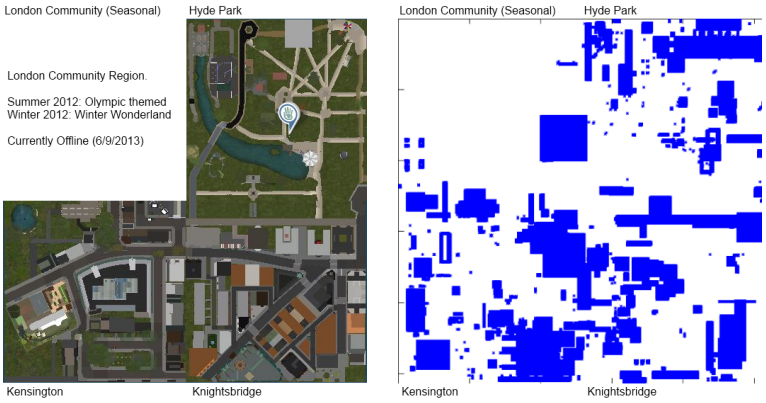


Figure 5.3: The environments used for the experiments. The smaller, Hyde Park region only uses the top, right hand corner of the environment. This is a 256^2 environment. The Greater London Area includes Hyde Park and three other regions. At the time of writing there is no image available for the London Community Region. The image on the right hand side is the collision map for the environments. This shows the free (white) and blocked (blue) space in the environment.

5.3 Evaluation

To test out hypothesis we looked looking at the two regions outlined in Section 3.2, Hyde Park and the Greater London Area. Hyde Park is a single region, 256m^2 in size. The Greater London Area consists of four regions, 512m^2 in size. Both these regions have been described in detail in Section 3.2 on Page 3.2. Images for the two regions can be seen in Figure 5.3.

To measure the effect that using trails has on generating PRMs, we compare our trail-based approach to PRMs generated with the alternative point generation methods. Our comparison is performed using the approach taken by Geraerts and Overmars (2002), Park and Chung (2009) and Gasparri et al. (2009). In these approaches, a fixed limit is applied to the number of points selected and added to the PRM. Once this limit is reached the PRM is then used in the planning task. The planning task requires the agent to plan a route between six given start and end points. The distance at which two points can be considered neighbours is also fixed at 10m. When merging trail points together we reduced this distance to 5m, to ensure that the merged trail points more accurately represented the cluster that existed previously. The point selection methods are then compared on the percentage of routes they successfully find. A failure means the map does not allow the agent to find a path between the start and end points.

A set of trail information is also given to the agent at the same time as the object map for the world. These are both described in more detail in Section 3.2. Processing of the trails for the merged trail PRMs is included in the generation time of the road map. Like previously, for RRTs, we were only interested

in generating maps at ground level for a walking avatar. The trail information was filtered so the agent only had access to ground level movements without flying or teleportation involved.

To compare the different PRM point selection methods, we generated twenty PRMs using each of the following point generation methods. The details of how each point selection method works can be found in Section 2.4.2.

Standard Methods:

- Random point selection
- Halton point selection
- 25%, 50%, 75%, 100% Gaussian points Combined with Halton Points

New Methods:

- Raw Trail information
  - Randomly selected points biased by Trail points (10% chance of keeping points, increasing by 10% for each neighbouring trail point detected)
  - 25%, 50%, 75%, 100% Trail points combined with Halton points
  - 25%, 50%, 75% Trail points combined with Gaussian generated points
  - 25%, 25%, 50% split for Trail, Gaussian and Halton points in all combinations
- Merged Trail information
  - 25%, 50%, 75% 100% Merged trail points combined with Halton points
  - 25%, 50%, 75% Merged Trail points combined with Gaussian generated points
  - 25%, 25%, 50% split for Merged Trail, Gaussian and Halton points in all combinations

For each PRM the generation phase is limited to 1000, 5000 or 10,000 points. Not all the points generated will be included in the map. Points may be rejected if they have already been included, or are not in free space. Once a roadmap has been generated, the agent must complete a planning task using this map. The agent must attempt to plan a set of routes between six different start and end points. The paths are split into two groups. The first group connected popular locations in the environment, places we had observed many avatars moving between. The second group were in locations less frequently or never visited, so less trail information would be available for use.

For each approach we measure the following criteria:

1. Success at the planning task

2. Length of the paths found between start and end points

3. Generation time for the PRM

4. Planning time averaged over all routes

The most important of these factors is the success rate. We judge a higher success rate finding longer routes as better than a lower success rate finding shorter routes. A failed map is one that is unable to plan a route between the given points. This may occur if the number of points included in the roadmap is not high enough, or the placement of points doesn't allow for paths to be found around obstacles.

We report the length of the planned routes as a single number. This was achieved by taking the ratio of each individually planned route to the shortest route planned for that set of points. The closer this number was to one, the shorter, on average, the planned route was. We then averaged this over the six planned routes for each point selection type. This gives us a single comparable number for the length of the planned routes.

Point selection methods can be assigned a r. This is the relative 'effort' it takes for a point to be added to the roadmap and has been used by the hybrid PRM (Hsu et al., 2005) to calculate which point selection method is best used when generating PRMs. We calculate the cost by looking at the number of collision checks required to add each point to the PRM. We omit the collision checks required to find a path between a point and its neighbour as every point added will have these checks. While trail points have some connections added without collision checking, an attempt is still made to connect these points to the main road map. The basic cost per point for methods can be seen in Table 5.1. Combinations of point selection methods are weighted by how many of each type of point they use. For example, a "25% trail, 25% Gaussian and 50% Halton" point selection method would have a combined cost of: $(0.25 * 0) + (0.25 * 2) + (0.5 * 1) = 1$. The cost metric here is a useful comparison showing the relative 'effort' a PRM algorithm would have to go through to add a node to the road map. These statistics are more useful for adaptive roadmap generation methods, as the algorithm can adapt its point selection method over time. However, that is outside the scope of this thesis.

All the experiments are run on a single computer, under as close to the same circumstances as possible. The machine was an Intel(R) Core(TM)2 Quad CPU (2.40GHz) running Scientific Linux version 2.6.32 with 2GB RAM.

Cost	Point generation type	Rational
0	Trail Points	No collision checking required as we know this place to be free. Another avatar cannot stand there if it is blocked
1	Random / Halton Trail Bias	A single collision check is required to add this point A single point is selected and checked, the bias happens before the collision checking
2	Gaussian	Gaussian checks the point in question and at least one neighbouring point. Only if that neighbour is blocked is the point added
3	Bridge Obsticle	3 points are required for checking if a point should be included

Table 5.1: Cost (the number of collision checks required) per point for the standard point generation methods

5.4 Results

The full results for the PRM experiments are presented in Appendix B. We focus on a subsection of the results here, where the success rate of each method was 100%. Results were averaged over 20 runs of each point selection method or combination in each environment.

In general, we found that including trails improved the generation of probabilistic roadmaps. Trails worked best when combined with other point generation methods.

Point Generation Method	Gen Time (mins)	Std.Dev.	Length	Plan Time (ms)	Std.Dev.
100% Halton	**1.52**	0.01	**4.41**	**115.28**	77.04
25% Trails	2.58	0.10	4.94	692.21	414.12
50% Gaus 50% Halton	3.63	0.07	8.67	153.33	89.33
75% Gaus 50% Halton	4.63	0.08	8.83	174.28	106.24
25% Trail 25% Gaus 50% Halton	1.76	0.06	5.79	196.73	113.37
25% Trail 50% Gaus 25% Halton	3.16	0.09	5.82	218.54	136.00
50% Trail 25% Gaus 25% Halton	1.94	0.07	5.28	267.24	171.98
25% Merge 50% Gaus 25% Halton	2.92	0.05	5.03	135.63	94.51

Table 5.2: PRM generation results for maps built using 5000 points in the Hyde Park environment. These results only show methods that achieved a 100% success rate. The results in bold text are the best for that metric.

5.4.1 Hyde Park

1000 points were not enough to generate a map with a high enough success rate to be included in our results. The most successful method achieved only a 67% success rate when using 100% Halton points.

Point Generation Method	Gen Time (mins)	Std.Dev.	Length	Plan Time (ms)	Std.Dev.
100% Halton	2.70	0.07	5.54	492.95	290.80
25% Trails 75% Halton	2.58	0.10	4.94	692.21	414.12
50% Trails 50% Halton	2.40	0.11	4.37	930.76	566.45
75% Trails 25% Halton	2.09	0.14	4.10	1313.70	881.76
25% Gaus 75% Halton	4.90	0.07	9.25	520.48	318.64
50% Gaus 50% Halton	7.03	0.08	9.16	576.88	346.49
75% Gaus 25% Halton	9.01	0.16	8.79	594.42	409.23
25% Trail 25% Gaus 50% Halton	3.36	0.10	4.90	745.30	502.30
25% Trail 50% Gaus 25% Halton	6.02	0.16	4.89	785.13	579.61
50% Trail 25% Gaus 25% Halton	3.83	0.17	4.44	1130.91	776.65
50% MergeTrails 50% Halton	1.66	0.01	3.58	291.23	192.81
75% MergeTrails 25% Halton	**1.09**	0.01	**3.50**	**180.83**	117.44
50% MergeTrails 25% Gaus 25% Halton	3.12	0.07	3.64	346.23	261.27

Table 5.3: PRM generation results for maps built using 10,000 points in the Hyde Park environment. These results only show methods that achieved a 100% success rate. The results in bold text are the best for that metric.

Point Generation Method	Gen Time (mins)	Std.Dev.	Length	Plan Time (ms)	Std.Dev.
100% Halton	10.29	0.18	2.30	101.96	60.79

Table 5.4: PRM generation results for maps in the Greater London Area built using 5,000 points. These results only show methods that achieved a 100% success rate. The results in bold text are the best for that metric.

Point Generation Method	Gen Time (mins)	Std.Dev.	Length	Plan Time (ms)	Std.Dev.
100% Halton	17.82	0.32	2.39	549.24	311.15
50% Trails 50% Halton	14.81	0.63	1.93	924.83	505.57
25% Gaus 75% Halton	28.48	0.77	2.67	647.91	300.66
50% Gaus 50% Halton	38.26	0.95	2.82	1236.54	617.57
25% MergeTrails 75% Halton	15.32	0.53	2.01	384.78	195.61
50% MergeTrails 50% Halton	**11.89**	0.31	**1.55**	**278.47**	169.34
25% MergeTrails 25% Gaus 50% Halton	26.18	0.54	2.00	707.70	372.12

Table 5.5: PRM generation results for maps in the Greater London Area built using 10,000 points. These results only show methods that achieved a 100% success rate. The results in bold text are the best for that metric.

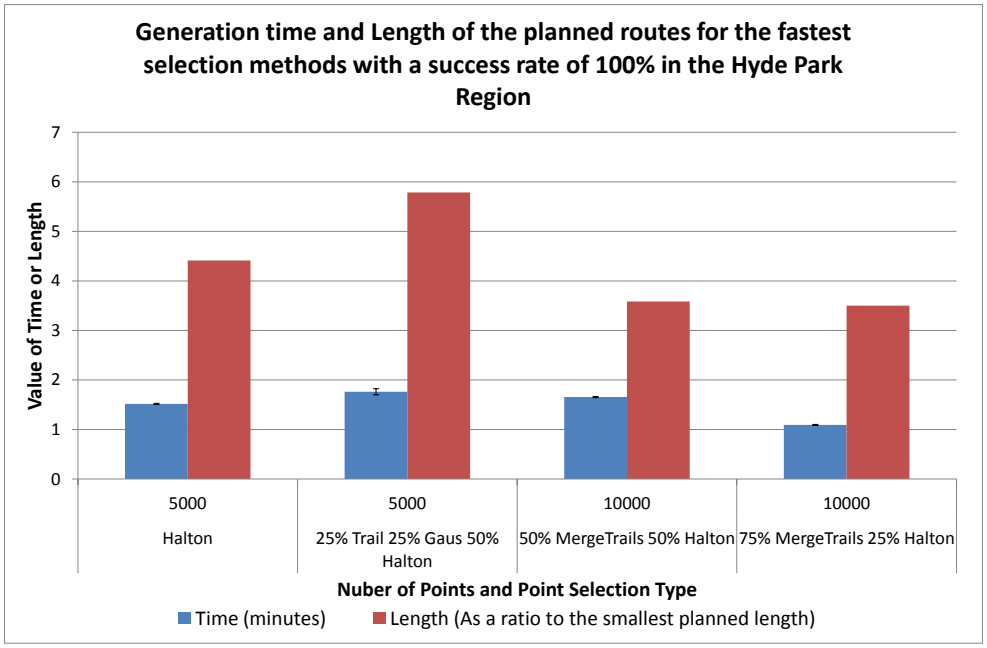


Figure 5.4: Graph showing the generation time and length of the subsequent planned route for the four fastest point selection methods in the single region, Hyde Park, environment. The error bars here show the standard deviation for the time.

This map was able to plan the 'easy' set of routes, but had a much lower success rate for the harder routes, due to a lack of information in the roadmap.

Maps generated using 5000 points had a higher success rate. Seven achieved a 100% success rate. Table 5.2 shows the results for these roadmaps. The shortest generation time, shortest path and shortest planning time all used the 100% Halton point selection method.

When using 10,000 points a higher proportion of the point selection methods had a 100% success rate. The table of results for 10,000 points can be seen in Table 5.3. Of these, the best method used 75% Merged trails and 25% Halton selected points. This had the lowest generation time, the shortest planned path and the shortest planning time of all the methods. Figure 5.4 shows a graph of the generation time and length of the planned route for the four fastest PRM generation methods. The generation time, at 1.09 minutes, was shorter than the PRMs using 5000 Halton points previously. The length of the route was also shorter when using 10,000 points. The short routes resulted from the human optimisation effect of trails; human controlled avatars tend to find shorter paths when travelling between points. Merged trail points were much more successfully used than the raw trails. The clutter of raw trails detracted from the map building process.

The planning time for some maps generated using trails is higher than the planning time when not using trails. Trails are not collision checked as they are added to the roadmap. The Gaussian and random points are checked for collisions and so may not be included in the map at all. This led to trail based maps having a higher number of points included at the end of the generation phase, despite the same number being selected. This had an effect on the planning time as more points in the map lead to more paths needing consideration and so an increase in time. The overall generation time included the planning time in these statistics, so this effect was taken into account when looking at the results. Planning time is an interesting metric if we want to consider map re-use, as the generation phase only needs repeating when the environment changes

5.4.2 Greater London Area

Merged trails were again very good for planning routes in a short period of time, but unless the trail points existed then the roadmap was unable to plan in that area. The larger size of the environment meant that a good spread of trail points is important.

The size of the environment meant that PRMs generated using 1000 points were not able to plan any of the routes required. Every method had a 0% success rate.

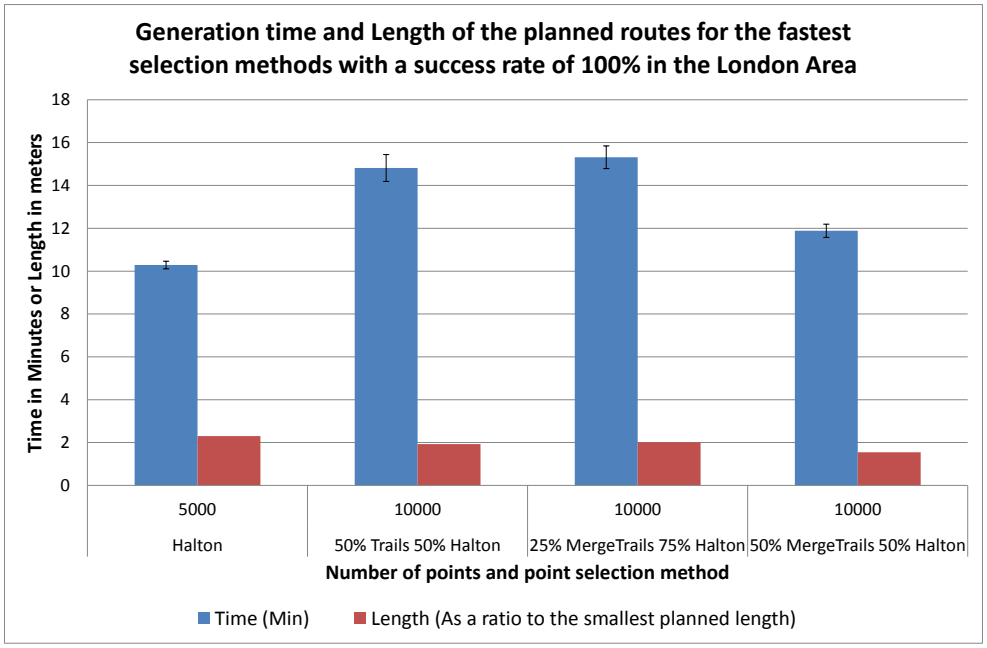


Figure 5.5: Graph showing the generation time and length of the subsequent planned route for the four fastest point selection methods in the four part Greater London Area environment. The error bars here show the standard deviation for the time.

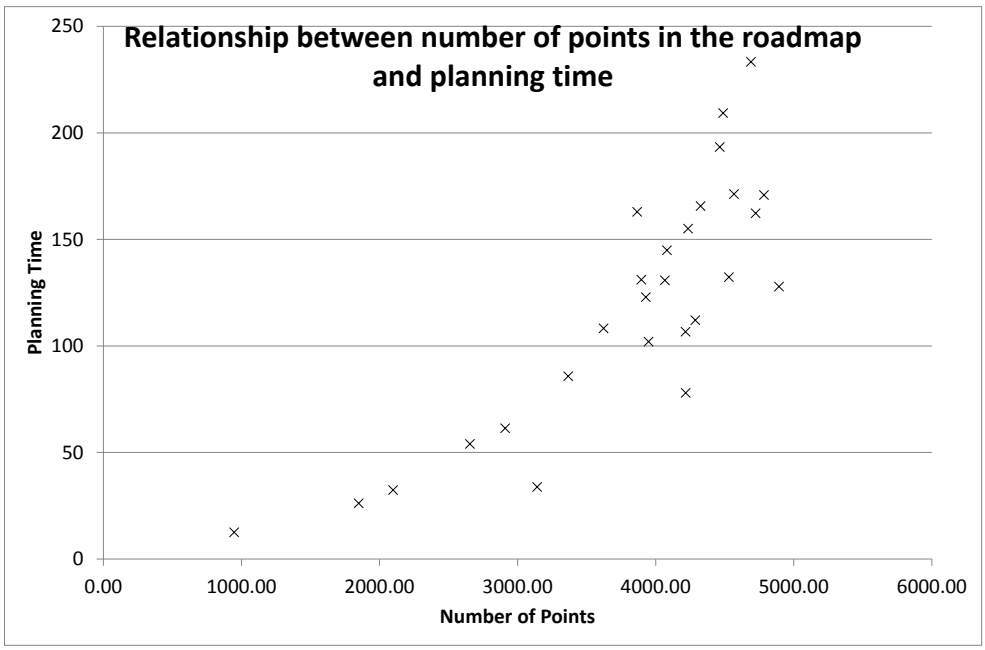


Figure 5.6: Number of included points in the 5000 point PRM vs the planning time in the subsequent map.

When we used 5000 points only Halton point selection achieved a 100% success rate (seen table 5.4). The generation time for this method was over ten minutes, which is excessive when compared to the small scale environment.

Looking at 10,000 point roadmaps, a larger number achieved a 100% success rate. These results are reported in Table 5.5. The best point selection method here is using 50% Merged Trail points and 50% Halton points. This generated the roadmap in the shortest period of time, had the shortest planning time and also planned the shortest routes. The generation time was slightly higher than the 5000 Halton point roadmap, however, the route planned was shorter (see the graph in Figure 5.5). There is a positive correlation between the planning time and the number of points in the roadmap (see Figure 5.6) so in general as the number of points included in the PRM increases, so does the planning time.

Initially we intended to generate maps with 20,000 points believing this to be necessary in large scale environments, but our results showed that we did not need to increase the number of points included in the roadmap beyond 10,000.

In both environments we discovered that some point selection methods were not successful. This is

due to the number and the way the points are selected. If the number of points isn't high enough, such as the 1000 point maps, or the placement of the points is not sufficient to plan routes around obstacles, then these maps are considered failures. Failure can be overcome in most cases by increasing the number of points selected for inclusion in the roadmap, which is why we saw a larger number of point selection methods included in the results for both Hyde Park and the Greater London Area.

5.5 Discussion and future work

The contribution of this section was to look at whether the generation of probabilistic roadmaps could be improved by using trails as a source of information. The hypotheses we formed to test this were:

1. By using trails we can reduce the time required to generate a PRM more than other point selection methods
2. By using trails we can increase the success rate of the agent at planning routes in an environment. It will be able to plan more of a given set of routes when using trails compared with when using standard PRM point selection methods.
3. By using trails when generating the PRM the agent will then be able to plan shorter routes than if it was using a PRM generated using other point selection methods.

In small scale environments we showed that a combination of 75% merged trail points and 25% Halton points were best for roadmap generation. These PRMs were generated in a short period of time, with a high success rate and a short overall planned route. The cost of merging the points was included in the generation time, so this result is comparable with non-trail based approaches.

In the large scale environment the best combination of points to use when generating the roadmap was 50% Merged Trails and 50% Halton points. Of all the methods in the 10,000 point experiments that achieved a 100% success rate this was the roadmap with the fastest generation and planning time as well as the shortest planned routes.

We have shown our hypotheses to be true. The generation of PRMs can be improved by using trail information. They are able to be generated in a shorter period of time and plan shorter routes. Using trails helped improve the success rate over some more limited point selection methods, such as Gaussian. The spread of points is very important in the roadmap and as the size of the map increases it becomes more important. The Halton sequence was so successful when generating roadmaps and planning routes

as a result of there being a good spread of points over the environment. This was more of a critical factor in the large scale map where longer distances need to be covered. Trails only exist where the environment is populated, so as we moved to a more sparsely visited map the number of Halton points required to 'fill in' the roadmap needed to be increased.

In the future we aim to investigate this further by looking at how to generate these maps while the agent is connected to the Second Life environment. A skeletonised PRM can be generated and trails are added to this map as they are observed. As some randomisation is needed to improve the maps generated only with trails, it will be interesting to see how these maps develop over time. This type of 'on demand' PRM would be comparable to the RRT algorithm as it would take into account the start point of the agent and where it is trying to navigate towards. Some method of identifying useful trails and merging them into currently known paths will be required. There are also other ways we could evaluate these roadmaps. One option is to keep adding points to the roadmap until the paths can all be planned and evaluate these based on the number of points required and the time taken to generate the map. These would be useful in time limited situations or when successful path planning is necessary.

We also wanted to compare trail-based PRMs with the Visibility PRM algorithm outlined by Simeon et al. (2000). However, the time required to generate a Visibility PRM far outstripped any of the other methods, owing to how the Visibility PRM is generated. Normally points are only connected to neighbouring points. In our experiment the maximum distance two points could be apart and still considered neighbours was 10m. However, the Visibility PRM does not have this limit, so connections are attempted to every other point in the map, meaning that a lot of collision checking was required when trying to connect two points in the road map. This increased the length of time needed to add a single point into the roadmap, vastly outstripping other point selection methods. The Visibility PRM seems better suited to smaller environments, where fewer collision checks are needed to build the map.

It is always possible to extend this work by adding more methods of point generation into the comparison. Some that are of particular interest are octree, or adaptive grid methods for generating the grid points (LaValle, 2006). These would work well when combined with the segmentation of the environment we look at in Chapter 7.

CHAPTER 6

Roadmaps in larger environments

6.1 Introduction

Chapter 4 and 5 have shown that trails can be used to improve the generation of roadmaps in small and larger scale virtual environments. The environment we have been using for these experiments is an existing virtual environment that uses one or four distinct regions from Second Life. To increase the size of the environment further we would need to find a suitable sixteen or thirty two region in Second Life, with a good avatar population that is flat and also covers a range of different environment types. This is difficult to achieve.

In some cases it is also difficult to see whether the change in the effectiveness of trail based roadmaps is due to the trails observed or other factors in the environment. This chapter is designed to look at trails in a controlled environment which we can extend to be of a much larger size. We are able to control the exact trails that the agent has access to and the layout, for instance, we can design environments with narrow corridors to test trail inclusion in this situation. We can then see if trails have a positive effect on an agents ability to plan routes between a given set of points.

The hypotheses we are looking at in this chapter are:

1. Trails can be used to improve the generation of PRMs in a set of test environments

2. Trails which do not connect the start and end point in an environment can still be useful when included in the roadmap

The contribution of this chapter is an investigation into the effect trails have on generating PRMs in very large scale environments. The area of Second Life we have trail and object information for is small compared to the overall size of the environment. By looking at generated environments we can extend the environment size without needing to find a different area of the Second Life virtual environment to use.

The rest of this chapter is structured as follows. Section 6.2 looks at the problem in more detail and at the environments we will be looking at. After that, Section 6.3 describes our evaluation method for these experiments. Section 6.4 follows with the results of our experiments. Finally, Section 6.5 discusses our results and future work we can do in this area.

6.2 Analysis

We designed six environments based on different criteria. These environments are available in four different sizes: $256m^2$, $512m^2$, $1024m^2$, and $2048m^2$. This corresponds to one, four, sixteen and thirty-two regions in Second Life.

1. Empty Environment

   The empty environment has no objects. A route should be easy to plan for the agent as there are no obstacles to consider.

2. Single Gap Environment

   The single gap environment has a wall in the middle of it with a single gap allowing the roadmap to connect one half of the room to the other. This environment shows an example of when a roadmap algorithm may struggle to fully connect its graph when specific points are required in the map building process. Sometimes the corridor is changed to have turns incorporated, for example:

3. Long Corridor Environment

   The environment consists of two rooms with a narrow corridor between them. This is similar to the single gap environment, but the length of the corridor makes this a more difficult problem as the agent will need to select several points within the corridor to ensure a path can be planned from one end to the other.

4. Sparse Random Clutter

   This environment is not structured and has objects placed around the environment. There is room to move around all the objects.

5. Dense Random Clutter

   A densely cluttered environment restricting the free space the agent can access. This is a typical limitation test for roadmap building methods used by a number of previous studies, for example: (Geraerts and Overmars, 2002) and (Thomas et al., 2007)

6. Zig Zag Maze

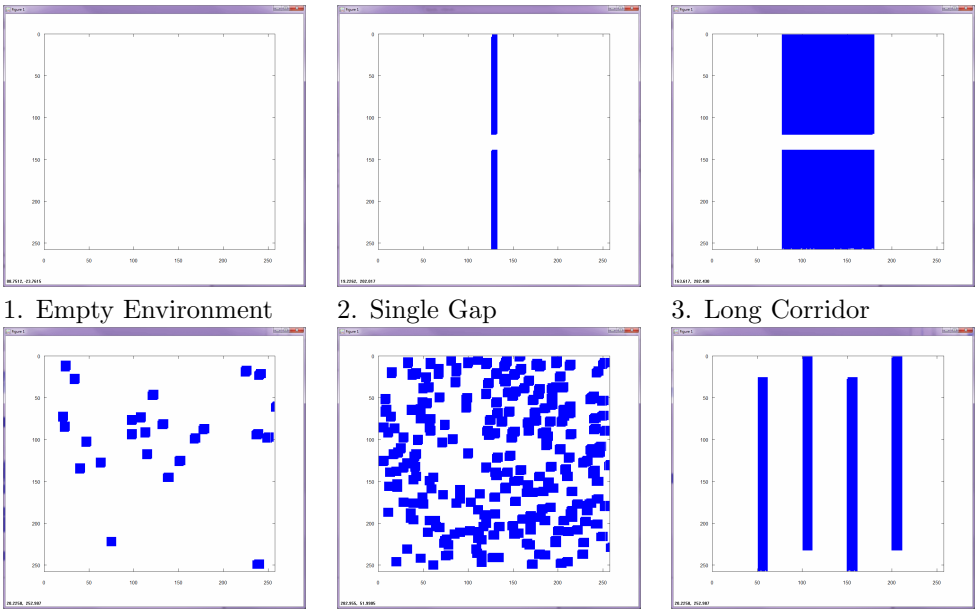
   This is an environment with many narrow passages to navigate to get from start to end. This will require selecting a correct set of points to ensure that the agent is able to navigate through all the corridors one after another.

Some of the environments were inspired by environments used to assess other PRM algorithms and point selection methods. Boor et al. (1999) use environments with narrow gaps and corridors for an agent to plan a route over. Geraerts and Overmars (2002), Yang and Brock (2004) and Thomas et al. (2007) all evaluate the effectiveness of various PRM algorithms and point selection methods on a three dimensional version of a single gap environment. These environments have a wall splitting the two sides of the environment with a narrow hole the agent has to navigate through. Denny and Amato (2011) use a two dimensional version of the single gap environment to evaluate their PRM algorithm, and also environments with long narrow corridors. Images of all the environments can be seen in Figure 6.1. These images are generated from the 256m^2 environments. Sparse and dense clutter was generated randomly, ensuring that the start and end point could be connected in the roadmap and a route planned between all the start and end points.

The agent may have no trail information given to it, or a single trail that can be of the following type:

- A full trail from the start to end point
- Partial trails, made up from 25%, 50% or 75% of the full trail
- A 'misleading' trail, connecting two points in the environment which are not always near the start or end point.

The trails were created by calculating the shortest path between two points, either the bottom left and top right corner for the useful trail or two different points for the misleading trail. This was done by



1. Empty Environment
2. Single Gap
3. Long Corridor
4. Sparse Clutter
5. Dense Clutter
6. Zig Zag Maze

Figure 6.1: The six test environments created. Areas in white are free space, areas in blue were blocked by objects. These environments are all 256m^2, the placement of objects in the sparse and dense clutter environments varies for each of the sizes.

creating a using the PRM generation algorithm as an implementation shortcut. A PRM was generated that added points in a regular, grid like fashion, 10m apart. This meant that every point could connect to the eight neighbouring points around it. Using this mesh of points and connections we were able to determine the shortest path between any two points to use as the trail. We did not include a random trail in the evaluation as this would be equivalent to running the standard PRM algorithm (selecting points in space then connecting them based on distance). This would not add any useful information. The misleading trail was included to assess whether trails would still be of use even when they do not join the start or end point. Recently, Alempijevic et al. (2013) have also investigated this approach for agents acting in the real world.

We chose to look at Halton point selection in combination with trails. Halton point selection gives a good spread of points over an entire environment. It gave a better performance than the randomly generated points in Chapter 5. It was also the best point selection method when combined with trails for both small and large environments. Given more time we would like to extend this work to look at more point selection methods.

First the trail points will be included in the roadmap. After that Halton points will be generated and added to the roadmap.

6.3 Evaluation

The evaluation of PRMs in this chapter is different to the evaluation method used in Chapter 5. Previously we had a large number of trails we could include in a roadmap, and so the percentage of points selected from trails rather than other methods had an effect on the time required to generate the roadmap. In this chapter, the agent only has access to one trail.

In all the environments the useful trails were planned between (0,0) and (maxsize, maxsize), where maxsize is the maximum size of the environment (256, 512, 1024 or 2048).

In the 256 and 512m environments the misleading trail was different for the long corridor environment compared to the other environments. This is because the trail provided information about how to traverse the narrow corridor. in the 1024 and 2048m environments the change is in the zig zag maze environments. This is due to the initial end point being blocked in those environments.

The start and end points for the misleading trails are as follows:

Table 6.1: Start and end points for the misleading trails in the environments. Only changes from the general start or end point are shown. Each point is given as an X,Y coordinate pair.

Size of Environment	256		512		1024		2048	
	Start	End	Start	End	Start	End	Start	End
General	10,10	90,250	468,309	340,504	10,1000	300,256	10,1000	300,256
H Corridor		25,250		440,504				
Zig Zag Maze						295,256		295,256

As the environments were scaled up the various size of features in each environment also changed. In the single gap environment the gap the agent had to navigate between was set at 20m, placed in the center of the wall. The gap had the same size no matter the size of the environment. This is also true of the narrow corridor environment. The gap between the two walls was 20m no matter the size of the overall environment. The length of the corridor was calculated as (size / 2.5)m, this gave a long corridor that scaled with the size of the environment. In the sparse clutter environment the number of blocks included in the map was (size / 10). For dense clutter this was simply (size). For the zig zag maze the environment the number of walls going along horizontally was calculated by taking the size of the environment and placing a wall every 50m. The top blocks were positioned first, then the bottom blocks were positioned half way between each existing pair. The gap between the end of the block and the edge of the environment was calculated as (maxsize * 0.1).

The PRMs are generated by including the trail (if necessary) before including other points in the environment. We generate 10,000 Halton points for inclusion in the map. Unlike in previous chapters,

the generation time for the maps is unimportant. As we are not concerned with the number of obstacles in the environment, or varying the number of points being included in the PRM then the generation time does not give us any new information. The same number of Halton points are always included in the roadmap, the difference is whether the single generated trail was used, whether it allowed the agent to plan a route between two points, and the length of the planned route .

As before, the distance two nodes can be from one another and still considered neighbours is 10m. A straight line is used to connect points in the PRM. The full algorithm for how to generate a PRM can be seen in Chapter 2, Algorithm 3 (page 21) . The agent is given two points to plan between in the environment. In all cases this is from the bottom left corner (0,0) to the top right corner (256,256 or 512,512 etc.).

To evaluate the roadmaps generated we will measure two factors: the success rate of the roadmap and the length of the planned route. The success rate is whether the roadmap is able to be used to plan a route between the bottom left and top right corner of the environment. A good roadmap will have a *high success rate* and a *low route length*. As with all of our experiments the success rate of the map being used to plan a route is the most important of the metrics. The exact specification for which type of roadmap generation approach is better in which situation will change depending on the requirements of the system. The hypotheses mentioned earlier are that trails will be beneficial to PRM generation, so in the results we are looking for a high success rate and low route length when using full or partial trails in combination with the Halton points.

Results for the experiments are averaged over twenty runs. The environment and trail used are the same for each set of experiments.

All the experiments are run on a single computer, under as close to the same circumstances as possible. The machine was an Intel(R) Core(TM)2 Quad CPU (2.40GHz) running Scientific Linux version 2.6.32 with 2GB RAM.

6.4 Results

The results can be seen in Tables 6.2 – 6.7. Results have been omitted where the roadmap achieved a 0% success rate. When the environment size was smaller, specifically the $256m^2$ and the $512m^2$ environments, all the point selection methods achieved a 100% success rate. In the empty and single gap environment the inclusion of trails usually led to a reduction in the length of the planned route. This was true even

Table 6.2: Empty Environment Results

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	657.30	138.56
	25% of Trail	100%	624.19	110.99
	50% of Trail	100%	520.88	54.02
	75% of Trail	100%	422.82	25.62
	100% of Trail	100%	**370.11**	1.29
	Misleading	100%	582.36	89.52
512	None	100%	1961.63	376.97
	25% of Trail	100%	1698.70	282.27
	50% of Trail	100%	1339.95	315.29
	75% of Trail	100%	1457.67	312.33
	100% of Trail	100%	**736.58**	0.95
	Misleading	100%	1714.18	300.32
1024	None	70%	2753.89	1858.53
	25% of Trail	100%	3936.24	573.91
	50% of Trail	100%	3191.65	276.06
	75% of Trail	100%	2148.31	473.45
	100% of Trail	100%	**1455.04**	0.60
	Misleading	70%	2185.44	1449.04
2048	100%	100%	**2908.27**	0.00

Table 6.3: Single Gap Environment Results

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	618.15	76.99
	25% of Trail	100%	552.83	76.00
	50% of Trail	100%	458.00	69.88
	75% of Trail	100%	364.74	1.46
	100% of Trail	100%	**364.09**	1.18
	Misleading	100%	585.75	108.01
512	None	100%	2035.16	378.47
	25% of Trail	100%	1827.50	345.11
	50% of Trail	100%	1322.82	207.68
	75% of Trail	100%	978.23	129.95
	100% of Trail	100%	**725.59**	0.44
	Misleading	100%	2042.29	302.20
1024	None	55%	2273.51	2101.72
	25% of Trail	100%	3835.26	408.40
	50% of Trail	100%	2674.67	300.62
	75% of Trail	100%	1742.21	156.78
	100% of Trail	100%	**1449.77**	0.60
	Misleading	65%	2708.60	2072.08
2048	100% of Trail	100%	**2897.72**	0.00

Table 6.4: Narrow Corridor Environment Results

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	615.78	54.65
	25% of Trail	100%	639.69	67.24
	50% of Trail	100%	439.26	13.97
	75% of Trail	100%	423.67	5.63
	100% of Trail	100%	**416.52**	0.83
	Misleading	100%	576.01	61.30
512	None	100%	1907.48	219.65
	25% of Trail	100%	1624.57	244.19
	50% of Trail	100%	1445.92	249.60
	75% of Trail	100%	1064.44	182.75
	100% of Trail	100%	**841.92**	0.90
	Misleading	100%	1660.53	217.61
1024	75% of Trail	100%	2022.43	140.39
	100% of Trail	100%	**1681.54**	0.00
2048	100% of Trail	100%	**3372.21**	0.00

Table 6.5: Sparse Clutter Environment Results

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	657.30	138.56
	25% of Trail	100%	624.19	110.99
	50% of Trail	100%	520.88	54.02
	75% of Trail	100%	422.82	25.62
	100% of Trail	100%	**370.11**	1.29
	Misleading	100%	582.36	89.52
512	None	100%	1961.63	376.97
	25% of Trail	100%	1698.70	282.27
	50% of Trail	100%	1339.95	315.29
	75% of Trail	100%	1457.67	312.33
	100% of Trail	100%	**736.58**	0.95
	Misleading	100%	1714.18	300.32
1024	None	70%	2753.89	1858.53
	25% of Trail	100%	3936.24	573.91
	50% of Trail	100%	3191.65	276.06
	75% of Trail	100%	2148.31	473.45
	100% of Trail	100%	**1455.04**	0.60
	Misleading	70%	2185.44	1449.04
2048	100% of Trail	100%	**2908.27**	0.00

Table 6.6: Dense Clutter Environment Results. Please note, the length for the 1028m and 2048m^2 environments are correct. Due to the random nature by which the environments were generated, objects may appear in configurations that make the free space more restricted and so the planned path longer.

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	598.36	72.02
	25% of Trail	100%	549.03	55.84
	50% of Trail	100%	588.19	92.20
	75% of Trail	100%	464.49	17.90
	100% of Trail	100%	**386.57**	0.65
	Misleading	100%	606.92	73.71
512	None	100%	1961.45	531.62
	25% of Trail	100%	1319.35	240.23
	50% of Trail	100%	1188.12	73.35
	75% of Trail	100%	852.70	47.29
	100% of Trail	100%	**799.40**	0.44
	Misleading	100%	1560.31	120.01
1024	100% of Trail	100%	**16459.87**	0.00
2048	100% of Trail	100%	**2945.17**	0.00

Table 6.7: Zig Zag Maze Environment Results

Size	Percentage of trail used	Success Rate	Length (m)	Std. Dev.
256	None	100%	2006.48	55.31
	25% of Trail	100%	1909.14	45.92
	50% of Trail	100%	1756.54	44.44
	75% of Trail	100%	1372.09	90.71
	100% of Trail	100%	**1239.04**	1.38
	Misleading	100%	1847.26	51.27
512	None	100%	6491.13	132.36
	25% of Trail	100%	6235.67	111.99
	50% of Trail	100%	5800.72	122.97
	75% of Trail	100%	4884.56	75.65
	100% of Trail	100%	**4125.51**	0.60
	Misleading	100%	6089.63	74.24
1024	100% of Trail	100%	**16459.87**	0.00
2048	100% of Trail	100%	**65790.36**	0.00

when the trail was considered misleading. In the narrow corridor environment the shortest paths were planned using 50% of the full trail, 75% of the full trail or the misleading trail. Using 25% of the trail seemed to have little effect on the path planning process.

In the 1024m^2 environments we see different results. In the empty and single gap environments, trail based PRMs had either the same or a higher success rate at planning the routes than using no trails at all. The length of the planned route was again generally shorter when using trail points.

The narrow corridor environment was designed to be more of a challenge to build a PRM in that was able to connect the two corners of the environment. There was a lower success rate for planning the routes in the 1024m^2 environment. Only PRMs built using 75% or 100% of the full trail had any success at all. The full trail allowed for the shorter route to be planned.

The 1024m^2 sparse clutter environment had success rates for trail based PRMs the same or higher than PRMs built using no trails. The length of the planned route for the 75%, full trail and misleading trail PRMs is shorter than when using no trail points.

The dense clutter environment had more objects and so free space was more limited, this shows in the results. Again the success rate was 100% for the smaller two environments. The shortest routes were planned using trail points. In the 256m^2 environment, the misleading trail increased the length of the route planned by 0.1%. When using a number of points from the full trail the length was reduced. This indicates that sometimes, in complex environments, a misleading trail can change the PRM being generated and cause parts of a planned route to take a longer route around objects. In the 512m^2 environment the misleading trail led to a reduction in the route planning length. In the 1024 environment only the PRM that used 100% of the full trail was able to plan a route between the two corners in the environment.

The Zig Zag environment showed a similar pattern to the dense clutter. In the 256 and 512m^2 environments all the methods had a 100% success rate. The shortest routes were planned using trails. In this case the misleading trail was still beneficial to the PRM, and allowed for a shorter route to be planned than when using no trails. In the 1024 environment only the PRM generated using the full route was able to plan a route between the start and end points.

The 2048m^2 environments all achieved the same result. The roadmap was only able to successfully plan routes when using the full trail. The full trail roadmaps had a 100% success rate and the same route length. There was a standard deviation of zero for these results as the planned path was the exact trail returned connecting the start and end point. No Halton selected points had any effect on the planned

path. Using only 10,000 points when generating the roadmap was not sufficient in an environment this size. We attempted to increase the number of points included in the road map and preliminary tests show that around 50,000 to 60,000 points may be required.

6.5 Discussion

We can see from these results that trails have a beneficial effect on PRMs generated in our six environments in all sizes. In the smaller environments, 256 and 512m^2, it was possible to generate roadmaps using no trail points and have a 100% success rate for path planning. The length of the route was generally reduced by including some trail points.

The one exception to this was the long corridor, 256m environment. Here using 25% of the full trail points in the roadmap led to a slight increase in the planned route length. The graph in Figure 6.2 shows the length of the planned paths vs the number of arcs used that were originally from the trails. We can see a definite trend that as the number of arcs used from the trail increases, the overall length of the path decreases. The order in which points are added to the roadmap changes how the arcs connect to one another. Trail points being added to the roadmap first can change how the roadmap is connected as more Halton points are included. This changes the shape of paths planned, and can cause the overall length to increase. Using 15 or more arcs from the trail has a beneficial effect.

In the 1024m^2 environments we saw a similar pattern. The success rate was either the same as, or better when using trail based PRMs than when the PRM is generated using only Halton points. The length of the planned path was generally shorter when using trails, even when the trail was considered 'misleading'. There were three cases where using trails may lead to a slightly higher route length. This was in the empty , single gap and sparse clutter environments. In larger environments the trails available may bear no relationship to the routes being planned, and so inclusion of trails in PRM generation can affect the performance of the resulting roadmapThis is why the inclusion of a random point selection method is important. In the dense clutter and zig zag maze environments, PRMs built using the trail information were the only ones to be successful at all. While in smaller versions of these environments it is possible to avoid using trails to generate the roadmap, as the size of the environment increases, the need for trails to allow the agent to plan a route also increases.

The 2048m^2 environments all had the same result. They all had a 100% success rate when using the full route in the PRM and a 0% success rate when using any other PRM point selection method. While

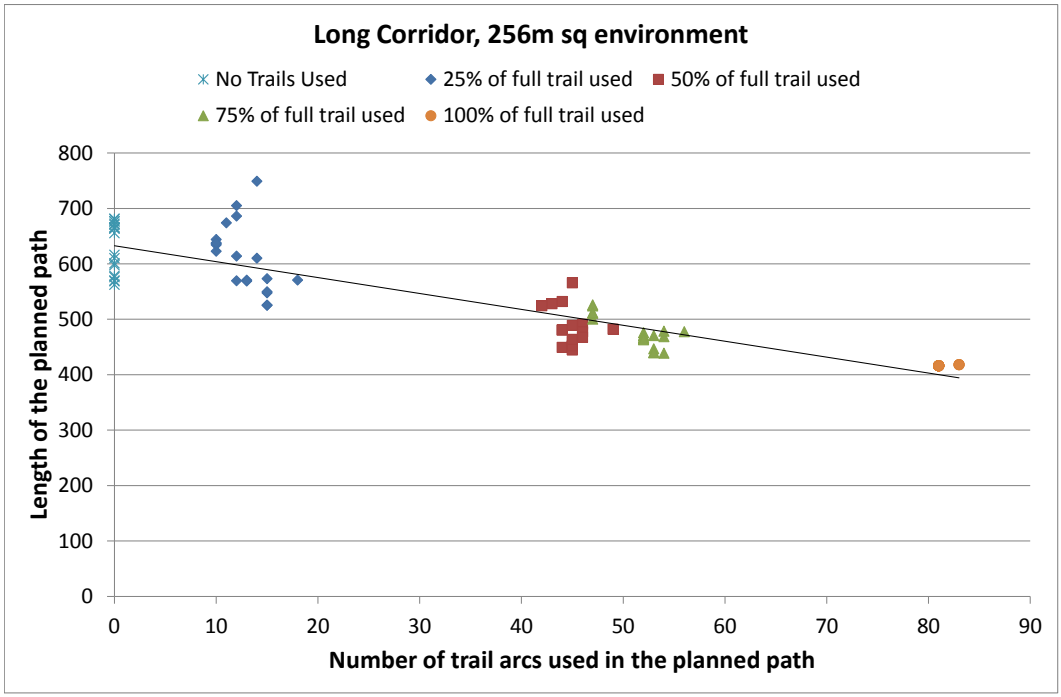


Figure 6.2: The length of the planned path against the number of arcs used in that plan that came from trail points. These results were calculated in a separate set of experiments (on a different computer setup) to collect new information. These results show the same pattern to the results reported in the table previously.

the roadmaps generated in the other sized environments were able to be used to plan routes with 10,000 point PRMs, more points are required in a 2048m^2 environment to generate a successful roadmap.

Our hypotheses were:

1. Trails can be used to improve the generation of PRMs in a set of test environments
2. Trails which do not connect the start and end point in an environment can still be useful when included in the roadmap

We have shown both hypotheses to be true. Trails do improve the generation of PRMs in these environments. They reduce the length of the planned route in all but a few cases. These cases could be improved by using more trail points in the map. Trails either had the same, or improved on the success rate of the roadmaps for planning routes in the environment. In many cases the 'misleading trails', trails that did not connect the start or end point, also improved the generation of the roadmap by providing more information in the roadmap. This is because even if a small part of the trail is useful to the route being planned, it can add useful arcs to the roadmap for the planner to utilise, even when the trail as a whole does not appear to be useful. It also shows that we cannot rely on arbitrary decisions as to whether a trail will be useful in the roadmap.

There is still future work to be done in this area. We would like to investigate the 1024 and 2048m^2 environments with a higher number of PRM points being allowed in the roadmap to see if the pattern continues to hold. We can also change the evaluation criteria and generate a PRM until the route can be successfully planned. In this case the number of points included in the roadmap could be recorded and used as a measure of success for each point selection method. Finally, we want to investigate RRTs in these large scale environments to see if a similar pattern emerges.

CHAPTER 7

Segmenting large scale environments using trails

7.1 Introduction

In the previous chapters we have been looking at generating roadmaps for navigation in large scale virtual environments. Throughout this thesis we have been increasing the size of the environment and we have found that as we do so, the time required to generate a roadmap in these environments has also increased. One example of this can be seen in Chapter 5. To generate a PRM with 5000 points in a $256m^2$ environment using only Halton points took 1.52 minutes. In a $512m^2$ environment, the same number of points took 10.29 minutes to complete (See table 5.2 and 5.4). As the size of the environment increases a single map will also become too large to fit in the memory available to the agent. A standard solution to this problem is to *segment* the map to improve performance.

Segmentation is the process of splitting a map into individual sections. This allows an agent to look at a smaller section of the environment when generating, updating or using the map (Ozkil et al., 2011). There are many different methods of doing this, we go into more details about segmentation types and approaches in Section 2.5. We look at three different types of segmentation methods in this chapter: regular, quadtree and Voronoi segmentation.

By reducing the size of the environment being considered, segmentation should reduce the time

required to generate a roadmap of an environment which is then used in path planning. The path planning is performed at two different stages in the mapping process. The first path is planned using the topological map, generated by segmenting the environment. This gives a list of segments the agent will need to visit. The agent can then generate roadmaps only within the required segments, rather than needing to cover the entire environment.

Our hypothesis is that we can improve the segmentation of the map by using trails. Trails have been previously explained in Section 2.6, they are a set of observations of how other avatars navigate an environment. In our previous chapters we have shown that trails can provide information about the structure of the environment (Chapter 3) and that trails can be used to improve roadmap generation (Chapters 4 and 5). We can use trails to identify cluster points, the places where many avatars have gathered. They represent popular and important locations in the environment according to the avatars observed. Cluster points then can be used to guide segmentation. We go into more detail about the specific hypotheses we are investigating in Section 7.2.

The work presented in this chapter is based on our published short paper:

> **Katrina Samperi, Nick Hawes, Russell Beale. Towards mapping and segmentation of very large scale spaces with intelligent virtual agents. Thirteenth International Conference on Intelligent Virtual Agents (IVA 2013). Springer-Verlag, 2013** (Samperi et al., 2013b)

The contributions for this chapter are: a new method of selecting seed points for Voronoi segmentation, a study into various segmentation methods and an evaluation of the best method of segmentation available for an intelligent virtual agent in a large scale environment.

The rest of this chapter is organised as follows. In Section 7.2 and 7.3 we define our problem in more detail and explain how will evaluate our results. Following that, section 7.4 gives the results of our initial investigations. Finally in section 7.5 we discuss our findings and future work.

7.2 Analysis

The task the agent has been given is to plan a route between two points in a large scale environment. The environment must first be segmented. This will create a topological map, with each node containing the objects and trail points that are present in an individual segment. The arcs show the connectivity between segments, that is, which segments are next to one another in the environment. The agent will

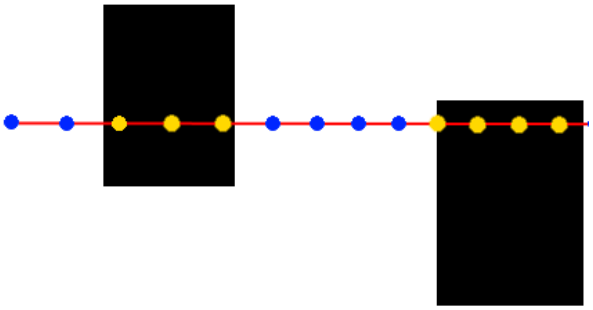


Figure 7.1: Image showing an example of gateway points. The red line is the boundary between two segments. Points in blue are in free space, and so would be added to the list of gateway points that can be used to cross from one segment to another. Points in yellow lie in objects, and so are not able to be used in this way.

then need to identify which segments the start and end point belong to and use this information to plan a route in the topological level, giving a list of segments to visit in order.

Inside each of these segments a roadmap will be built, connecting a set of 'gateway' points. Gateway points are points along the boundary between two segments that are not blocked by an object. These are free points that the agent can use to move from one segment to another. Image 7.1 shows a simplified example of this in an environment. The black rectangles are objects and the white areas are free space. The red line represents the boundaries between two segments. Along this line, every point is tested to see if it is collision free or not. The blue points in the image would be added to the set of gateway points for this boundary. The yellow points all fall inside an obstacle, so these would not be included as the agent cannot use them for traversing between segments.

The roadmap within each segment is built with a high number of points being included so that the chance of a route not being found across a segment is small. When this does happen, or if the agent cannot find a route between the start and end points after generating all the roadmaps, the agent returns to the topological level and re-plans. This continues until either a route has been found linking the start and end point, or there is no viable route available. Algorithm 13 shows the pseudo-code for the experiments we have run.

The roadmap representation we have chosen to use is the PRM. This will be generated using 75% trail and 25% Halton points. While in Chapter 5 we showed that in a larger environment a combination of 50% trail and Halton points was most effective, the segments the agent will be generating PRMs within will be smaller than 512m^2. We chose the PRM over the RRT as a PRM can have any number of start and end points. This means it is able to link any number of the 'gateway points' that allow the agent to

cross from one segment to another. By contrast, the RRT selects a single root node and creates a tree from that point, stopping when it reaches one of the gateway points on the other side of the segment. As there is only ever one root node in a RRT, the potential for the planning step to fail is a lot higher. A PRM is more suited to this type of environment. The point selection method used by the PRM was chosen based on the results of Chapter 5. This is the point selection method that had the highest success rate in the shortest period of time in smaller environments.

Algorithm 13 Pseudo-code algorithm for the segmentation experiments

```
1: Initialise Start and End points for Route Planner
2: ObjectMap ← loaded from file
3: TrailInformation ← loaded from file
4: TopologicalMap ← Segmented Object map and trail information
5: StartSegment ← Segment the start point is inside
6: EndSegment ← Segment the end point is inside
7: SuccessfulRoutePlanned ← Plan Route in topological map from StartSegment to EndSegment
8: if SuccessfulRoutePlanned then
9:    for Every node in the topological map visited by planned route do
10:      if If a PRM exists in the node then
11:         Add the gateway points for the node and its neighbour to the existing PRM.
12:      else
13:         Generate a PRM in this node, store in topological map node
14:      end if
15:   end for
16:   Attempt to plan the final route from start to end using topological map and local PRMs.
17:   if Successful route planned then
18:      Report length and final plan; finish with a success
19:   else
20:      Replan route in Topological Map (line 7)
21:   end if
22: else
23:   No route possible in the topological map; finish with a failure
24: end if
```

We will be comparing the effect of using three different types of segmentation against the base case, where a single map is used. The base case is useful to see whether segmentation is improving the map generation by allowing for a faster generation and path planning time. Segmentation should improve the generation time, by allowing the agent to generate PRMs and plan only in some segments of the environment. This smaller section of space should reduce the time required to generate the PRM. However, segmentation may cause the length of the planned route to increase.

We also want to look at a range of environments, both small and large scale. In small scale environments we will look at a relatively empty environment, a highly structured environment and one which combines both structure and free space.

The hypotheses we are working with for this experiment are:

1. That trail based segmentation will allow the agent to generate a map and plan a route between two given points in a shorter time period than using no segmentation at all.

2. Trail based segmentation will also outperform other methods of segmentation. It will generate the map and plan a route in a shorter time period than either using regular or quadtree segmentation.

3. Regular and quadtree segmentation will perform better in structured environments, compared to Voronoi segmentation.

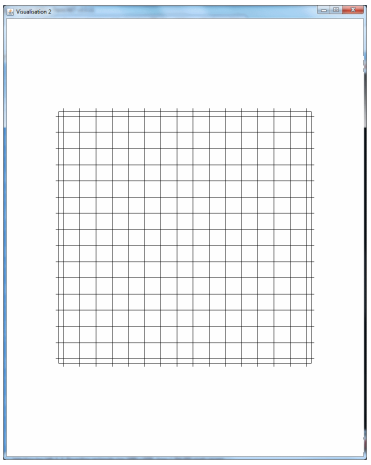
4. The length of the planned route in any environment using segmentation will be longer than a route planned using a single map, but the time taken to finish planning this route will be shorter.

7.2.1 Segmentation Methods

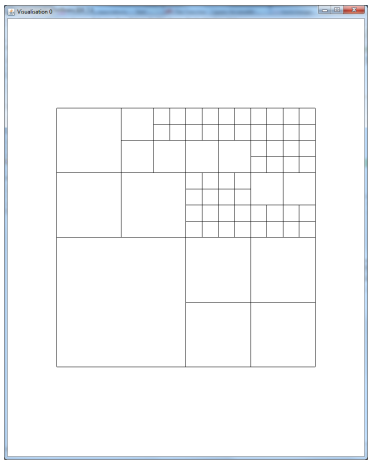
The three types of segmentation we are comparing are: Quadtree, Voronoi and regular segmentation. Images of the three types of segmentation when applied to the 256m^2 environment of Hyde Park can be seen in Figure 7.2. Objects were sorted into segment by testing the center point for the object and assigning it to which segement it belonged within. Trails were similarly split. Each of the segmentation approachs took the same form. First we generated the topological map for the environment, defining the segment boundaries and calculating the arcs connecting neighbouring segments. Then the object and trail information for the environment was segmented, assigning each object and trail point to the segment it belongs to. Finally the gateway points were calculated between segments, this was stored in the arc information.

Quadtree Segmentation

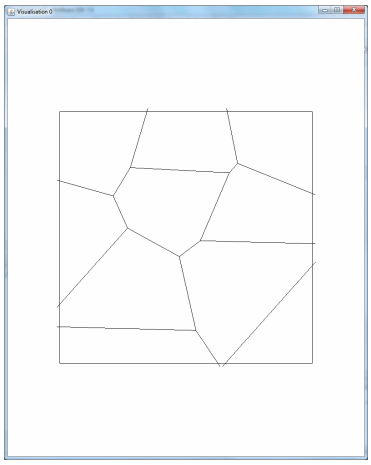
Quadtree segmentation is a recursive form of segmentation. The environment is split into quarters, then within each quarter each point is tested to see if it is free or blocked. If the environment is a mix of free and blocked space then it is segmented into quarters again. This process continues until either a minimum size for the segment is reached, or all the segments contain only either free space, or blocked space.



1. Regular segmentation method. The length of the side was set to 32m.



2. Quadtree segmentation method. The length of the side was set to 32m.



3. Voronoi segmentation method. This was generated using 10 seed points selected using the Halton sequence.

Figure 7.2: Images for the different segmentation types generated in the Hyde Park region of Second Life.

Voronoi Segmentation

Voronoi segmentation is based on the concept of Voronoi Diagrams (Canny and Donald, 1988). These are generated from a set of seed points. The boundaries of each segment are calculated as the set of points equidistant between two seeds. Any point within a Voronoi segment is therefore closer to that seed point than any other in the map. To implement this we adapted and used the Mesh processing library by Lee Byron (Byron, 2008) . This generates Voronoi diagrams based on the QuickHull3D library (Lloyd, 2014), with an O(n log(n)) complexity. The Voronoi algorithm relies on a set of seed points being provided, the number and placement of these will affect the segmentation generated. .

We look at four types of seed point selection, three using a single type of point selection method: random, the Halton sequence and density clustering point selection using trail points. The fourth point selection method combines density clustering and Halton seed points, selecting 50% of the points from each method.. More details about the three segmentation methods can be found in Section 2.5 on page 31. We have used the Halton sequence previously to generate a more even spread of quasi-random points over an environment. The details of how these are generated can be found in Section 2.4.3 on page 27.

Trail points show where avatars have been observed in the environment over time. We can use these points to identify *cluster points*. These are popular locations where many avatars have been observed standing in, or near, over time. Clustering is a method of grouping objects together based on how similar they are (Andrienko et al., 2009). The density clustering algorithm we use is a variation of the *k Nearest*

Neighbour (k-NN) algorithm (Russell and Norvig, 2003) . This groups points according to their Euclidean distance from one another. We use it to identify cluster points from the trail set which can then be used seed points for Voronoi segmentation. K-NN iterates through each point in the dataset (in this case, the set of trail points) and finds all the neighbouring points within a given distance of the initial point. If the number of neighbours is over a threshold *(k)*, then the centre of the cluster is calculated, added as a point in the dataset, and the neighbouring points removed.

Regular segmentation

Regular segmentation splits the environment using a grid. The size of the segments can be altered by changing the length of the sides in the grid. This is a simple form of segmentation that does not take into account any factors about the environment. To generate a regular segmentation over the entire environment we used the same algorithm as the Voronoi segmentation. The seed points used were set a regular distance apart over the entire environment, generating segments in a grid formation over the entire environment.

7.3 Evaluation

We evaluated segmentation in four environments. Three were single region environments, London Community, Hyde Park and Kensington. We chose these three as the number of objects, and therefore obstacles, in each was different. London Community is the most empty region, Hyde Park is a mix between open space and blocked areas and Kensington is a region with a lot of objects. The fourth environment we looked at was a four region environment, the 'Greater London Area'. Details about these environments can be found in Section 3.2. The agent is given a full object map for each environment and a set of trails that correspond to that space.

Several variables in our experiment were set in advance. As before, in Chapter 5, the distance at which two nodes in the PRM could be considered neighbours was set at 10m. We also set the *k-threshold* for clustering at ten points. Voronoi seed points were taken from the clustered trail information at random.

In the regular and quadtree environments we varied the minimum length of the sides in a segment. This started at half the size of the environment (128m in the $256m^2$ environment) and was divided by two for each subsequent set of experiments. In the Voronoi environment we changed both the number of seed points (starting at ten and increasing to one thousand) and the method the seed points were selected by.

In all the environments we also generated a roadmap using no segmentation. This was our 'base case' to compare against.

The number of points generated for inclusion in the PRM was either 10,000 for the smaller environments or 20,000 in the larger environment. These numbers were selected to ensure that the PRM would be able to find all available paths across a segment. The failure of a segmentation method should not be due to the PRM not having enough points to cross between segments. In all the environments the agent was required to plan a route from the bottom left corner (0,0) to the top right hand corner (256, 256 or 512, 512).

Our main criteria for evaluation are:

1. The overall time taken to generate a map and plan a route
2. The time taken to segment the map
3. The success rate at planning the route
4. The length of the successfully planned route

A good segmentation method will have an overall *generation and planning time* shorter than using no segmentation at all. It should have a *high success rate* at planning paths between the two given points. The routes planned should be as *short* as possible.

All the experiments are run on a single computer, under as close to the same circumstances as possible. The machine was an Intel(R) Core(TM)2 Quad CPU (2.40GHz) running Scientific Linux version 2.6.32 with 2GB RAM.

Table 7.1: Results for the London Community segmentation experiments. This is the sparsely cluttered environment. Values in bold text are the best for that metric

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		10441.55	247.18	2.60	11.11	1.00	0.00	494.12	54.81	**100%**
Regular	128 m	**2243.05**	260.16	**754.40**	191.79	1.00	0.00	**710.52**	50.62	**100%**
Quadtree	128 m	5160.50	247.14	2757.40	189.56	1.00	0.00	715.28	71.31	**100%**
Voronoi-Random	10 pts	2414.00	489.74	1101.00	186.72	1.05	0.22	775.38	103.84	**100%**
Voronoi-Halton	10 pts	2391.90	316.63	1123.35	182.09	1.00	0.00	755.43	90.45	**100%**
Voronoi-Density	10 pts	2505.05	438.30	1205.30	199.31	1.10	0.30	735.90	67.62	**100%**
Voronoi-Density and Halton	10 pts	2528.40	396.55	1202.10	236.96	1.00	0.00	735.63	88.52	**100%**

Table 7.2: Results for the Hyde Park segmentation experiments. This environment is a mix of empty and structured space. Values in bold text are the best for that metric

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		62793.30	438.78	5.45	23.30	1.00	0.00	490.20	47.02	**100%**
Regular	64 m	**4598.05**	267.15	3402.15	215.73	1.00	0.00	948.41	56.17	**100%**
Quadtree	32 m	16640.50	762.84	15104.05	706.56	1.00	0.00	**613.44**	29.01	**100%**
Voronoi-Random	10 pts	5658.75	1493.78	**2379.00**	244.67	1.30	0.78	823.49	441.41	95%
Voronoi-Halton	10 pts	5110.60	1784.00	2628.75	1458.28	1.15	0.36	904.58	741.99	**100%**
Voronoi-Density	10 pts	5562.05	1739.90	2573.60	575.00	1.35	0.79	773.10	335.78	90%
Voronoi-Density and Halton	10 pts	5963.45	1428.87	2513.95	357.11	1.05	0.22	976.50	766.50	**100%**

Table 7.3: Results for the Kensington segmentation experiments. This environment is highly structured compared to the other two small scale environments we are looking at. Values in bold text are the best for that metric

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		104728.95	1152.50	2.75	10.16	1.00	0.00	493.70	25.99	**100%**
Regular	64 m	**5922.00**	218.67	4703.50	148.96	1.00	0.00	809.05	33.98	**100%**
Quadtree	64 m	51453.80	1269.72	48886.75	1098.05	2.40	0.49	**669.12**	42.80	**100%**
Voronoi-Random	50 pts	12075.65	809.85	10036.50	226.87	2.50	1.40	736.32	73.08	**100%**
Voronoi-Halton	10 pts	8851.55	1450.83	**4044.75**	270.05	1.05	0.22	708.56	248.60	**100%**
Voronoi-Density	10 pts	10824.95	3177.81	4154.90	357.49	1.25	0.43	781.74	474.06	**100%**
Voronoi-Density and Halton	10 pts	11334.55	2385.00	4327.55	427.63	1.40	0.80	736.50	255.49	95%

Table 7.4: Results for the Greater London Area environment. This is a four region, 512m^2 space. Values in bold text are the best for that metric

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		660369.60	5219.01	2.35	9.79	1.00	0.00	1014.08	66.84	**70%**
Regular	64 m	**25722.70**	1948.93	18970.70	282.49	3.90	0.99	1810.92	109.90	40%
Quadtree	128 m	100981.15	932.91	84734.30	881.07	2.90	0.70	**1222.83**	66.39	**70%**
Voronoi-Random	50 pts	30759.85	7986.82	19050.05	718.11	5.55	4.54	1487.25	330.06	45%
Voronoi-Halton	50 pts	27622.30	3433.34	**18650.95**	315.04	4.60	2.46	1330.61	55.16	50%
Voronoi-Density	50 pts	29794.70	4671.94	20066.50	495.99	4.70	2.93	1873.41	796.49	55%
Voronoi-Density and Halton	50 pts	29603.45	4923.59	19727.90	518.95	4.65	2.15	2613.60	2657.54	45%

7.4 Results

The full tables for the results can be seen in Appendix C. Below we look at the best of each type of segmentation for comparison.

7.4.1 Small scale environments

We looked at three 256m^2 environments, London Community, Hyde Park and Kensington. London community comprised mostly open empty space with very few obstacles. This region contained only 1477 objects, whereas Hyde Park contained 7097 objects and Kensington contained 11771 objects. In the London Community environment, regular segmentation with a minimum segment side of 128m performed the best of all the segmentation methods looked at. As there were not many objects in the environment, larger segments were faster and allowed the agent to plan a route successfully in this environment. Table 7.1 shows the results for the London Community region.

Hyde Park had a mixture of open and structured space. There was no overall best segmentation method for this environment. The fasted overall method was regular segmentation using a minimum segment size of 64m. However, the fastest segmentation time used Voronoi segmentation with random seed points. The segmentation method that planned the shortest route was different again, it used the quadtree segmentation with a minimum segment size of 32m. Table 7.2 shows the results for the Hyde Park region.

Kensington was the most structured of the three environments we looked at. The fastest overall method of segmentation was again regular segmentation with a minimum segment size of 64m. The fastest segmentation time used Voronoi segmentation and Halton seed points. However, the shortest path was planned using quadtrees with a minimum segment size of 64m. Table 7.3 shows the results for the Kensington region.

The graph in Figure 7.3 shows the length of the planned routes in all three small scale environments. There is a lot of variation in the length of the route planned in the Hyde Park region. This will be due to the mixture of open and structured space. In general, regular and quadtree segmentation had the least variation in the length of the route planned. This is because of their deterministic nature. Voronoi segmentation generates a different set of segments each time, based on which seed points were selected. As the objects and the size of the environment stay the same, quadtree and regular segmentation will always return the same set of segments and topological map. This can be detrimental in some cases,

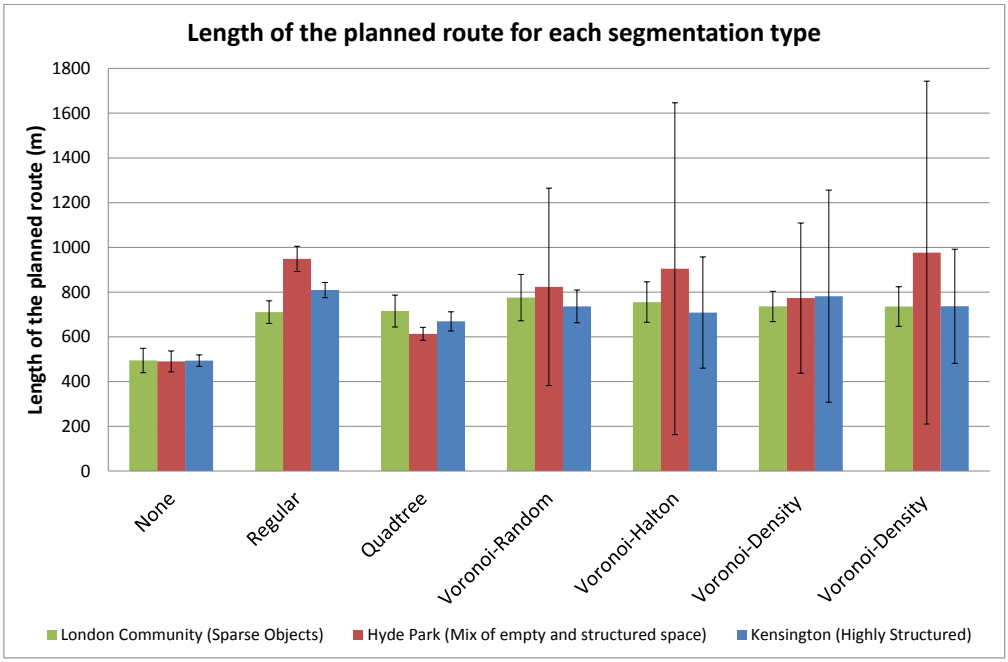


Figure 7.3: Average length of the planned routes in the small scale environments. The error bars show the standard deviation for each average result.

In the Greater London Area environment, using quadtree segmentation and a minimum segment size of 64m, the resulted in a map which was unable to plan the route at all. The positioning of objects in the environment and the segments created meant that there was never an available free path.

7.4.2 Greater London Area

There was no one clear best segmentation method in the larger, four region environment. The results can be seen in Table 7.4. The fastest overall time was achieved by regular segmentation using a minimum segment size of 64m. The fastest segmentation time was again Voronoi segmentation using Halton seed points. The shortest route was planned using quadtree segmentation, with a minimum segment size of 128m. The success rate in large scale environments at planning the route was lower than in small scale environments. Using no segmentation at all, the base case only had a success rate of 70%. Quadtree segmentation matched this. After this the next highest success rate was Voronoi segmentation using trail points as seeds.

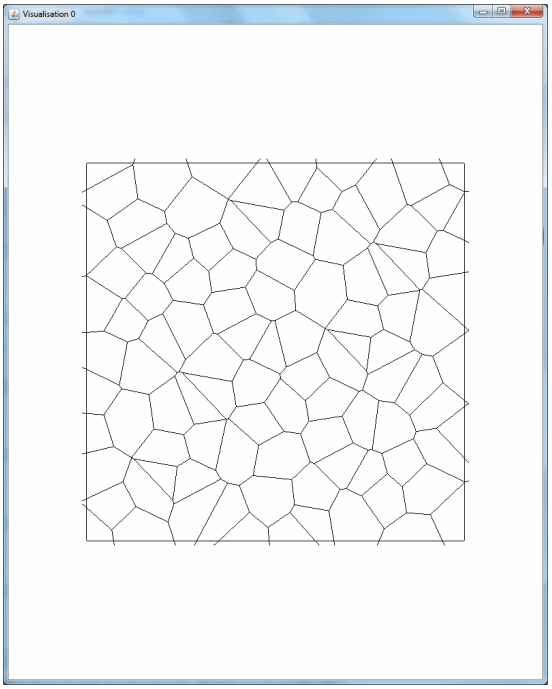
In the larger environment the number of seed points required to generate Voronoi segmentation in the shortest time period increased to 50 seeds. This had the lowest generation time of all the variations of seed points.

All the segmentation methods required more than one plan at the topological map level. This will be due to how the segmentation has split the objects in the environment. We set the number of points for including in segment based PRM to be high, at 20,000, so that the only point when a route would fail to be planned was when objects in that segment were blocking it. This led to some methods requiring a lot of plans in the topological level, the highest of which was an average of 13.05 plans (with a standard deviation of 3.67) in the 16m Quadtree segmentation map. On average, quadtree required more re-planning steps in the topological level than any other segmentation type (8.46 plans over all variations of the quadtree algorithm).

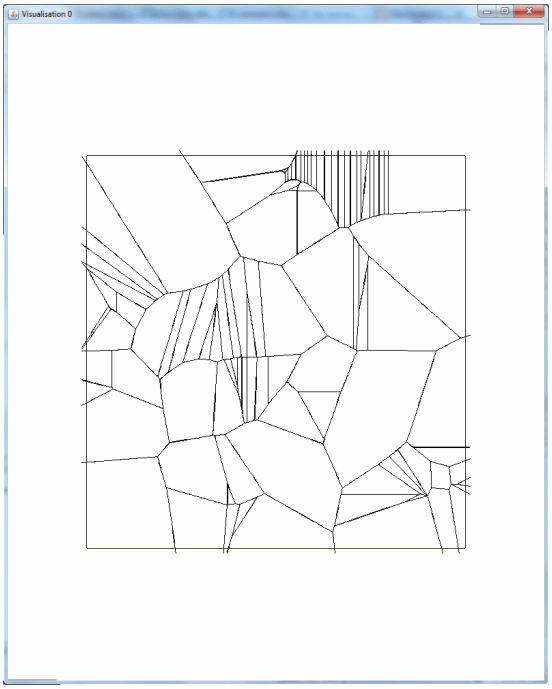
7.5 Discussion

Our first and second hypotheses were that trail based segmentation would outperform other methods of segmentation over a variety of metrics. This is not the case in either small or large scale environments we looked at. Regular segmentation was the overall fastest in all four environments. Either using Voronoi segmentation or regular had the shortest segmentation time. The shortest routes were planned using Quadtrees or regular segmentation.

Trails were in fact detrimental to the map building and path planning process in some cases. In Hyde Park using seed points generated from density clustering of trails decreased the success rate of the maps at planning routes. Using trails in the large-scale environment had a higher success rate than the other methods of segmentation, except for quadtree segmentation. Trail based Voronoi segmentation required a shorter period of time overall then quadtree segmentation. One reason for this may be that trails are, by their nature, not evenly spread over the environment. They represent where avatars have been observed and so the points selected by clustering this information lead to uneven segments, some being very small in popular areas and other segments very large. Figure 7.4 shows the difference made when generating a Voronoi segmentation using 100 seed points. The seed points generated using the Halton points are more evenly placed over the environment leading to segments of roughly the same size. In contrast to this the seed points selected from trail points create segments of random shapes and sizes. This does not have a positive impact on the map generated.



Seed points generated using the Halton sequence



Seed points generated from density clustering using trails

Figure 7.4: Image of the difference between Halton and Trail points when used as seeds in Voronoi Segmentation. Both maps used 100 seed points.

In general, using either ten or fifty seed points was also better for Voronoi segmentation than any other number. As the number of seed points increased, the overall time required to generate and use the map increased. However, a higher number of segments can reduce the length of the planned route in the regular and quadtree segmented environments.

Voronoi segmentation in all three environments often required more than one plan in the topological map level. This results from the placement of objects in the environment. Sometimes it was not possible to generate a path between two sides of a segment if there is a large obstacle in the way. Quadtree segmentation also sometimes required the same re-planning in the topological map as it found segments that were unable to be navigated.

The third hypothesis stated that regular and quadtree segmentation would perform better in structured environments, compared to Voronoi segmentation. We found this to be true. Voronoi had a shorter segmentation time than other methods but the overall time, including map building and planning, was longer than the regular and quadtree methods of segmentation.

Our final hypothesis was that routes planned using segmentation would be longer than those planned using no segmentation at all. This we found to be true in all cases.

These results have focused on segmentation and planning time as the most important criteria for our evaluation. In the large scale Greater London Area environment the success rate was low for all of our methods. We can look instead at the methods of segmentation with the highest success rate at planning the route. This was not looked at as part of the criteria for evaluation in the small environments as all methods could achieve a 100% success rate with enough points. By looking at success rate in the large scale environment we see a more positive picture for trail based segmentation. Four methods of segmentation had a success rate of 95% or 100%. Table 7.5 shows the results for these methods. The fastest overall time used Voronoi segmentation with seed points based on trails. The shortest segmentation time used a combination of seed points from trails and the Halton sequence. These methods both achieved a 95% success rate, higher than using no segmentation, and a shorter overall planning time.

From these results we can see that there is potential for trails to be useful as the size of the map increases. If success at planning the route is more important than simply speed at generating the map, then trails and Voronoi segmentation are a better option than other segmentation methods.

There is a general limitation with trail based Voronoi seed points in that the density clustering method returned a maximum number of points. This was sometimes less than the number of seed points we required. An example of this can be seen in Table C.2 where the full results table for the London

Table 7.5: Methods with the highest success rate at planning the route in the large scale, Greater London Area environment

Type	Variation	Total Time (s)	Std Dev	Seg. Time (s)	Std Dev	Length (m)	Std Dev	Success Rate
Quadtree	32	187.81	1.88	178.99	187.81	**1224.09**	294.36	95%
Voronoi-Random	1000	336.48	27.74	276.85	336.48	1362.99	78.43	100%
Voronoi-Density	500	**167.63**	20.17	136.64	167.63	1396.50	337.09	95%
Voronoi-Density and Halton	500	168.21	17.76	**136.62**	168.21	1379.71	329.50	95%

Community region show that there is very little difference in the results for the Voronoi segmentation performed with 250, 500, 750 or 1000 points from the trail density algorithm. In this environment there was a maximum of 168 cluster points available. For maps where more points are required we would need to use a second seed point selection method, such as the Halton sequence to generate the correct number. These results do not impact on our findings for these chapters as the fastest method of Voronoi segmentation in each case used less than the maximum number of trail based seed points available.

There is a lot of future work in this area we could do. We want to evaluate our segmentation methods using the test environments we developed for Chapter 6. This would give us a better idea about whether segmentation becomes more necessary as the size of the environment increases. We also only look at three methods of segmentation, there are many others that we could take a look at, such as image skeletonisation which we can apply to the available overhead map for any Second Life environment.

CHAPTER 8

Discussion

This thesis has looked at improving map representation and generation for intelligent virtual agents in large scale environments using trails. Virtual environments are becoming larger and more complex over time. A challenge exists to provide agents that are able to generate a map of these environments and use it for path planning. We want the map generation process to be achievable online, in a short period of time. To investigate this we looked a previously under-used source of information, trails. Digital trails are the routes we observe other agents navigating in a virtual environment. These trails show free space in the environment, popular locations where avatars cluster together and routes taken between specific locations.

The contributions of this thesis were outlined originally in Chapter 1. They were:

The contributions of this thesis are as follows:

1. Validation that trails are a useful source of information for an agent to build a map (Chapter 3 – 7)

2. A study into the effect of trails being used to generate Probabilistic Roadmaps in large scale environments. We find that trails have a positive effect on roadmap generation, reducing the time needed for map generation, reducing the overall plan length and improving on the success rate of the agent being able to plan routes around the environment. (Chapter 5)

3. Following this, a study into the effect of trails as the environment grows larger. Validation that trails become more important as the size of the environment increases (Chapter 6)

4. A study into the effect of trails on another type of roadmap representation, rapidly exploring randomised trees. Trails also have a beneficial effect on the generation of RRTs, improving their performance (Chapter 4).

5. An investigation into segmentation strategies in large scale virtual environments and the effect that trails can have if they are used as an information source. (Chapter 7)

6. A map trail corpus for intelligent agents in Second Life available at `http://www.katsamperi.co.uk/maps.php`.

The rest of this chapter will summarise our investigations in this thesis, discuss our findings and look at potential future research for each area.

8.1 Trails in virtual environments

We first looked at whether trails were a viable source of information for agents in virtual environments. Chapter 3 described the environments we used in the rest of the thesis, how we gathered objects and trails describing these environments and then looked at a side study, investigating two hypotheses: First, did the avatars walk between places in the environment, and second, were the movements considered to be meaningful.

If avatars chose to fly or teleport between locations (both modes of transport allowed in our evaluation domain) then the trails gathered from their movements would be of no use to a ground level, walking agent. Whether the trails were meaningful is harder to define. We looked at whether a particular social habit carried from the real world into virtual environments; whether avatars chose to walk on the path or if they chose a straight line route between places.

We found from our investigations that the majority of locations an avatar was observed at were at ground level. We filtered out points where avatars teleported between places, or where the observed location was above ground level. We were also able to conclude that the social convention of not walking on the grass held for a variety of avatars with different levels of experience in the environment.

The limitations of our study were that we calculated experience based on the age of the avatar. Each avatar reports its 'born on' date. This only tells us when the avatar was created, not any indication of

how often the user has logged into the environment, or whether this is the only virtual world they visit. We also do not know anything of the background of users behind avatars.

Despite these limitations we were still able to conclude that avatars behaved in a useful manner for route planning. By observing the trails formed by avatars we were able to identify free space, popular locations and routes between specific parts of the environment. We use trails for the remainder of the thesis as a source of information for map generation in agents. This chapter is a cornerstone in showing that trails are a viable approach.

To expand on this work we would interview the people behind the avatars, and find out more about their background, experience, motivations and any social norms they believe to be holding to. This would help us analyse avatar behaviour with more certainty.

8.2 Improving RRT generation using trails

Chapter 4 built on the idea that we could use trails as a source of information for a virtual agent that must navigate its world. The hypotheses we investigated were:

1. Trails could be used to improve the generation step of RRTs in small and large-scale populated virtual environments.

2. By using trails we can reduce the generation time needed to create a tree able to plan a route between two given points, and

3. The routes planned by the trail based RRTs will be shorter than those planned using other RRT point selection methods

RRTs are built from a single 'root' point. Branches are then added to the existing tree by selecting a point in the environment and growing a branch from the tree towards it. Either the branches can be limited in size (as per the standard algorithm) or continue until they reach an obstacle (using a greedy approach). We varied how points were selected in the environment to see whether biasing them to areas where trails had been observed improved the map generation.

We found that in the small scale environments the better spread of points selected either randomly or using the Halton sequence allowed a tree to be generated that connected the start and end point in the shortest period of time. In large scale environments, if we use the normal or greedy RRT algorithm

this result changes so that trail biased RRTs are completed in the shortest period of time. In both small and large scale environments the shortest paths were planned using trail biased trees.

It will depend on what the specific requirements of an RRT are as well as the size of the environment to choose which method is the best. The better spread of points from random and the Halton sequence helped the tree growth in small environments as this allowed the tree to expand in all areas, not just popular areas identified using trails. Biasing the tree to where trail points have been observed may not cover all parts of the environment. As the environment grows larger, it is important to have some guide for the tree towards the end point. In the large-scale Greater London Area environment there was a higher concentration of trails near the end point selected this helped the tree expand in the correct direction.

The limitations of this approach are that we only ever used raw trail points. In Chapter 5 we looked at merging trail points together for use in PRMs to reduce the trails to a smaller set of points in the environment. We did not have time to fully explore that with RRTs. We were also only able to use the trails we observed. While we attempted to ensure that an even spread of trails was used over the four regions making up the large scale environment, Hyde Park in the top right corner of the environment had a higher number of users, so there was some bias towards this area.

As part of our future work we want to look at generating the roadmaps fully online. That is, as the agent is connected to the virtual world, observing trails as they occur and generating a roadmap without a full set of information. We also want to look at other methods of RRT generation for comparison, such as the waypoints discussed by the ERRT (Bruce and Veloso, 2002), as well as looking at whether using a more even spread of trail points over the entire environment may help the RRT-Connect algorithm. This algorithm grew a tree from both the start and end points given, so required a bias towards the middle of the environment to be successful.

8.3 Improving PRM generation using trails

Following on, Chapter 5 looked at improving the generation of Probabilistic Roadmaps using trails. PRMs are a different type of roadmap representation. Where an RRT needs a starting 'root' point for the tree to grow from, a PRM does not need this information. PRMs are generated by selecting points in an environment and then attempting to connect each new point to its nearest neighbour. Given enough time, and enough points being included in the map, a PRM will be able to find a path between any two

points in an environment if it is possible to do so. The drawback to PRMs is that in large environments or those with a lot of obstacles and narrow passages the generation time can be high. This chapter looked at improving the generation of limited point PRMs in large scale virtual environments.

We compared several point selection methods, generating PRMs with either 1000, 5000 or 10,000 points in 256m^2 or 512m^2 environments. We looked at three standard point selection methods: random, the Halton sequence and Gaussian. We compared these with trail based point selection methods using either the full trail information, or a merged set of trails where sets of points were grouped together until a skeletonised version of the trail information was achieved. Trails were used either on their own or in combination with other point selection methods.

We evaluated each PRM on the time needed to generate the roadmap, its success rate at planning a set of routes and the length of each route. Our hypothesis was that we could improve the generation of PRMs in large scale environments. Specifically, that trails would improve the success rate of maps, while reducing the generation time and planned route length. We found that in the 256m^2 environment a combination of using 75% merged trail points and 25% Halton points when generating the roadmap performed best. The time needed to process and merge the trail points was included in the generation time to give a fair comparison of point selection methods. In the 512m^2 environment the best combination of point selection methods used 50% merged trail points and 50% Halton points.

Trails give the agent information about free space in the environment and free paths which can be used to navigate between these points. By using this information less collision checking is required from the agent when generating the map. These routes are known to be available as another avatar has been observed using them. If the agent was to use only the raw trail information the roadmap became cluttered. A high number of arcs and nodes were included in the roadmap, but these did not always add new information for path planning. The merging process reduced the trails to just the useful nodes and arcs, information that was new in the roadmap. The Halton sequence then added a good spread of random points to the roadmap, which helped connect trails together, and add paths to any areas where avatars had not been observed.

This spread of points is why in the larger environment the better PRM point selection methods used less trail points. While we attempted to have an even spread of trails in all four quadrants that made up the large scale environment, some parts were less frequently visited than others. The London Community region had fewer trail points to use as reference, while Hyde Park had an abundance to choose from. Even so, trails had a positive effect on the generation. The human optimisation effect of

trails allowed for shorter path to be planned in both cases. This is also observed when looking at RRTs in the previous chapter. If the goal of an agent is to find the shortest path, no matter the roadmap method, it is better to use trails than ignore them.

These results form the contributions for this chapter, a study into how trails can be used to improve the generation of roadmap representations in large scale environments and how different point selection methods can be combined for greater effect. The limitations of our approach is similar in all the chapters we have looked at so far. We gave the agent a set of objects and trails to build a map from in advance. In a true online situation the agent would be receiving pieces of information as it generated the map, rather than up front. Our future work in this area is to look at how we can resolve this issue. How can PRMs and RRTs be generated online, and how can we include trail information as it is received? One option we want to look at is generating a basic roadmap in the environment, similar to methods used for the Lazy PRM (Bohlin and Kavraki, 2000). This can then be adapted and improved as more trail and obstacle information becomes available, and as each arc is selected in the path planning algorithm.

We also want to look into developing some kind of heuristic that states which type of point selection method is best for the environment we are looking at. This may be based on the available information at the start of the map building process, the availability of trails as generation happens and also any method we can use to determine how many narrow obstacles are present in the environment. When doing this we can also look at the many other point selection methods available, such as the Visibility PRM.

8.4 Roadmaps in very large scale environments

A limitation of Chapter 4 and 5 is the environments being used. We chose to use the Greater London Area of Second Life as it is a fairly flat, populated environment that we could easily gather information in. It is much more difficult to find somewhere that matches the same criteria in Second Life that would allow us to look at $1024m^2$ or $2048m^2$ environments. These areas would comprise 16 or 32 regions, would need to be connected to one another and have unrestricted access. Without this our agent would not be able to gather information about the objects and trails in these regions. The contribution from this chapter is the validation that as the size of the environment increases there is a need for trails in order to generate maps that are able to successfully plan routes.

To address this problem we designed a set of environments for this chapter. We created object maps for six different types of environment, based on various characteristics of the environments we were

looking at. These were presented to the agent in the same way as region based object maps were. Trails were generated automatically and these were also presented to the agent in the same way the real trail information was. We looked at the difference made by including different type of trail information in PRM generation.

We generated the roadmap using 10,000 Halton points, sometimes in combination with different types and amounts of trail information. The potential trails given were: a full trail connecting the start and end point, 25, 50 or 75% of the full trail, with nodes and arcs selected at random, or a misleading trail that connected points in the environment other than the start and end point.

We found that as the size of the environment increased, trails became essential for generating a roadmap that was capable of planning a route from one corner to another. In smaller environments trails improved the length of the planned route, but do not change the success rate. In larger environments trails both improve the success rate and reduce the planned route length. These results also held when we looked at the misleading trails. The presence of some trail points could improve the roadmap by including additional information in the roadmap.

The limitation of our approach here was the 10,000 point maximum for PRM generation. In 2048m^2 environments only the roadmaps that included the full trail information had any success at all when planning the route. The way to address this is to increase the number of points included in the roadmap as the size of the map increases. This will increase the generation time for the roadmaps, but we can look at these large environments in conjunction with segmentation to attempt a time reduction again. It will also be interesting to look at changing the evaluation of the roadmap from the inclusion of a set number of points to a generation approach that continues until the path has been planned. We could then compare the time taken or the number of points required to achieve this roadmap. It would also give us a closer comparison when looking at RRT maps generated in the same space. Other future work in this area includes looking at the various different point selection methods when using PRMs and also looking at RRT generation in these environments.

8.5 Segmenting the map in very large scale environments

Chapters 4 to 6 looked at generating a single roadmap in each environment, no matter the size. This leads to the time required to generate the roadmap representation increasing as the size of the environment does. One method for reducing this is to segment the environment, splitting it back into smaller sections.

We do not want to assume that all environments will be aligned the same way Second Life is, with each region being a set size. We want to consider continuous large scale environments.

Our hypothesis in this chapter was that we could improve segmentation of an environment by using trails and also speed up map generation in general. As trails show the places where avatars have been observed it is possible to identify cluster points, popular locations in the environment where many avatars have gathered together. Specifically, our hypothesis was that we could use these cluster points to split the environment, generating segments that focus on a single popular location and its surroundings.

We also looked at a second hypothesis. Two of the segmentation methods were deterministic segmentation types: regular segmentation splits the environment according to a grid and quadtree generates a recursive grid structure. Our hypothesis was that these methods of segmentation would outperform Voronoi segmentation in highly structured environments.

Our final hypothesis was that the length of the planned route of any segmented map will be longer than a route planned using a single map, but the time taken to complete the map and route plan will be shorter. Segmentation forces the roadmaps to connect roadmap segments only along 'boundary points'. These are the points that are in free space on the boundary between two segments. This restricts the roadmaps more than if a single map was used.

Our results for these experiments were mixed. Our first hypothesis was incorrect, trail based segmentation was not faster than regular segmentation in any of the environments we looked at. Regular segmentation allowed for the fastest map generation and route planning time in all the environments we looked at. This is due to the simplicity of the segmentation. The shortest routes were found using quadtree segmentation. The second hypothesis was correct. Regular and quadtree methods did outperform Voronoi segmentation in structured environments. Our third hypothesis was also correct.

This chapter shows some improvement on segmentation as a high success rate is important for map generation. Unless the map created is able to be used to plan routes then it is of no use for the agent. When trail based Voronoi segmentation was used the success rate increased for large scale environments and the time taken to generate the map and plan the route was still shorter than using no segmentation at all.

One of the limitations of our approach is that we had a static number of trail points used to identify clusters, set at 10 points within a 10m radius. Changing this number could have an effect on the segmentation performed as a different set of seed points would be available to the agent. We also did not get the opportunity to look at the 1024 or 2048m^2 environments, segmentation in larger environments is

something we hope to do in our future work. Finally, we want to look at other segmentation methods available to split the map.

8.6 Comments and other future work

In general we can state that trails are a good source of information for an agent attempting to autonomously generate a map in a previously unknown environment. Trails shortened the paths planned and increased the success rate at finding routes, when the agent was using a single roadmap. Trails only give information about populated parts of an environment, so to cover an environment that is sparsely visited in places trails would need to be combined with other methods of point selection and segmentation.

There is still more work that can be done regarding trails and their use in map building for intelligent agents. One thread that has been present in several chapters is that there is no single best way of selecting points for use in generating maps or segmentation in all situations. We first need to formally define what makes a good point for each roadmap algorithm, environment type and route type. Then based on this develop a heuristic that can be used to identify which point selection and which map generation algorithm works best for the situation the agent is in. The requirements of an agent that must successfully plan a route are different to those that must plan a path in a specific period of time.

We also could look at obstacle density or the structure of the environment when identifying the best combination of algorithm and point selection. Point selection methods such as Gaussian work well in heavily restricted spaces, but need to be combined with other methods when there are areas of open space.

A general limitation that applies to all the experiments we have done is that the agent has been given a full set of object and trail information to build its map from. When connected to a virtual environment, this information will not be initially available to the agent. The trail information was gathered over a long time period to ensure that we had enough diverse trails to evaluate our methods. We are interested in extending the system and looking at how the agent can build maps online, improving the current representation as more information becomes available. One example using a PRM could be that the agent generates a small map of its immediate environment and expands on this as trail information is observed and as the agent moves to new areas.

Another direction is to look at machine learning. Our agent has no previous knowledge of maps generated in any environment. If the agent could draw on its previous knowledge then it may be able

to develop its own set of heuristics for generating maps in new environments. Looking at trails we can already extract a set of popular places and the routes between them. This information can be transferred to another environment, for example, identifying paths and including them as free space automatically in a roadmap. This would allow an agent to focus on fully exploring areas that are less well known and build more comprehensive representations of an environment.

In summary, this thesis has looked at improving the map generation for agents in large scale, populated virtual environments. We found that trails are a valid, if underused, source of information about an environment and they can be used to improve road map generation in small and large scale virtual environments. Trails can also be used to improve the success rate of segmented maps when route planning as the size of the environment increases.

APPENDIX A

Full RRT Results

This appendix holds the full table of results for the experiments performed in Chapter 4. The results for the experiments performed in small scale environments can be seen in Tables A.1, A.2 and A.3. For large scale environments see Tables A.4, A.5 and A.6.

Table A.1: Small scale environment, Normal RRT generation. All times are in ms. Bold text shows the best value for that metric.

Small Scale		Non Greedy							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		**3234.50**	429.88	**74.60**	7.72	1.75	1.41	388.10	14.21
Halton		3269.15	639.08	75.35	16.76	1.75	1.26	392.09	22.69
Trail Bias	1000	3609.55	772.67	83.35	18.49	1.85	1.31	391.96	16.00
	2500	3691.00	684.18	83.90	15.43	1.80	1.44	393.10	23.04
	5000	3884.60	989.86	91.15	23.61	1.85	1.42	387.95	30.49
	7500	3303.50	420.92	75.65	10.01	1.60	1.16	389.87	26.22
	10000	3527.35	765.33	79.35	16.32	1.70	1.45	**382.81**	16.03
	12500	3456.15	711.64	79.45	13.37	1.65	1.56	394.92	19.47
	15000	3926.95	683.57	89.90	17.32	1.90	1.30	396.08	21.94
	17500	3339.50	812.32	75.70	13.16	**1.65**	1.74	386.14	17.93
	20000	3663.60	628.97	83.10	15.30	1.70	1.23	390.48	17.07

Table A.2: Small scale environment, Greedy RRT generation. All times are in ms. Bold text shows the best value for that metric.

Small Scale		Greedy							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		**3376.60**	539.62	77.55	11.43	**1.55**	1.24	390.25	20.06
Halton		3549.15	723.07	80.85	12.82	1.60	1.74	389.44	20.04
Trail Bias	1000	3623.95	593.72	82.35	9.59	1.70	1.52	392.13	17.99
	2500	3689.85	742.72	82.30	15.26	1.65	1.53	390.49	22.40
	5000	3656.90	582.62	83.80	15.76	1.75	1.48	388.25	27.82
	7500	3485.20	667.49	79.25	14.74	1.80	1.33	383.40	16.07
	10000	3420.15	692.76	**75.65**	15.89	1.90	1.41	**382.05**	15.91
	12500	3881.00	686.46	90.70	14.78	2.05	1.83	389.46	18.45
	15000	3387.45	682.33	77.65	11.85	1.60	1.56	389.98	19.73
	17500	3737.75	509.41	84.50	11.39	1.80	1.50	395.70	24.20
	20000	3704.45	805.49	86.80	18.17	1.75	1.34	388.16	21.56

Table A.3: Small scale environment, RRT-Connect generation. All times are in ms. Bold text shows the best value for that metric.

Small Scale		RRT-Con							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		3325.65	590.44	76.00	12.09	**1.55**	1.40	389.09	16.20
Halton		**3244.55**	558.87	**75.20**	13.39	**1.55**	1.28	380.97	17.81
Trail Bias	1000	3537.65	582.55	81.10	14.17	1.85	1.15	390.24	22.85
	2500	3680.75	650.22	85.05	15.60	1.65	1.35	398.52	23.71
	5000	3374.70	724.22	75.70	14.98	1.60	1.77	388.03	17.52
	7500	3490.05	1042.46	79.90	21.23	1.65	1.74	**380.10**	20.30
	10000	3752.50	1055.16	85.20	22.07	1.80	2.23	390.24	30.95
	12500	3451.85	809.94	80.65	18.74	1.65	1.59	386.93	23.46
	15000	3655.85	807.79	84.40	13.86	1.75	1.95	385.26	15.20
	17500	3826.85	917.79	87.70	20.68	1.90	1.34	386.80	24.69
	20000	3391.35	624.21	77.15	11.56	1.70	1.79	384.81	17.96

Table A.4: Large scale environment, Normal RRT generation. All times are in ms. Bold text shows the best value for that metric.

Large Scale		Non Greedy							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		161761.45	38257.79	756.35	180.80	26.75	22.81	905.92	55.61
Halton		138370.25	16129.70	665.25	78.03	19.05	11.91	891.36	54.00
Trail Bias	1000	94978.65	24846.02	430.30	104.11	9.55	6.32	910.11	70.29
	2500	102460.55	39433.91	446.60	167.11	10.85	7.44	899.24	96.53
	5000	84574.80	20105.06	374.20	95.33	9.20	7.76	913.91	78.82
	7500	91031.05	35205.71	389.30	160.59	8.75	8.43	918.03	81.85
	10000	84344.10	18939.29	365.00	80.21	7.95	5.99	899.54	51.45
	12500	81907.75	26016.98	352.45	100.05	7.95	7.16	886.22	63.60
	15000	**81622.00**	17986.10	**345.80**	82.61	**7.45**	4.73	**883.84**	65.51
	17500	88588.40	25550.19	358.95	81.12	7.75	6.02	908.55	76.40
	20000	84900.55	19594.63	358.70	82.32	7.70	5.12	895.26	52.87

Table A.5: Large Scale Environment, Greedy RRT Generation. All times are in ms. Bold text shows the best value for that metric.

Large Scale		Greedy							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		93520.90	79735.06	240.15	195.31	6.30	6.24	1038.31	150.03
Halton		74502.80	53991.81	**190.95**	132.17	5.20	5.10	1109.75	239.86
Trail Bias	1000	53936.70	20706.10	208.00	99.64	5.25	4.18	1099.00	189.61
	2500	58264.95	47362.78	224.40	184.74	5.45	7.32	1026.70	234.90
	5000	59453.35	24974.10	222.70	102.03	5.45	4.63	1020.89	146.98
	7500	55645.30	48067.60	226.15	149.90	**4.55**	4.97	1034.68	187.92
	10000	57280.90	23882.60	224.55	136.05	5.65	4.38	1017.16	217.19
	12500	59433.20	32658.77	222.30	154.40	**4.55**	5.57	1048.68	201.70
	15000	**51103.80**	27110.77	237.10	79.88	5.25	2.72	981.27	133.59
	17500	62167.80	32290.47	310.95	187.36	5.40	5.07	**957.42**	105.97
	20000	61889.30	29916.60	210.35	96.32	5.25	3.77	1008.05	158.08

Table A.6: Large Scale Environment, RRT-Connect Generation. All times are in ms. Bold text shows the best value for that metric.

Large Scale		RRT Connect							
		Total Time (ms)	Std Dev	Points	Std Dev	Plan Time	Std Dev	Length (m)	Std Dev
Random		59404.90	22046.29	257.85	90.10	6.35	4.19	937.58	89.38
Halton		**49410.05**	11083.17	**222.50**	49.09	**4.90**	2.88	876.21	47.63
Trail Bias	1000	69209.80	29290.38	294.10	112.13	7.05	4.68	873.97	75.84
	2500	75379.10	31148.88	322.90	131.49	6.55	6.24	890.10	65.06
	5000	104757.20	56445.05	453.20	245.03	9.70	10.76	902.21	68.55
	7500	100948.70	40324.82	431.25	175.07	8.35	7.36	915.56	73.21
	10000	85835.40	40193.31	372.90	176.33	7.40	5.67	**870.65**	73.93
	12500	108432.00	56447.97	461.75	231.24	9.60	8.07	887.51	54.26
	15000	103907.60	50304.12	443.00	212.58	9.65	10.96	913.43	59.97
	17500	111487.70	73472.38	466.15	306.47	12.85	22.10	909.42	89.29
	20000	83227.30	25971.52	356.90	111.31	7.75	10.23	900.61	58.76

APPENDIX B

Full PRM Results

The full results of our PRM experiments are presented here. This is the data collected for the experiments run in Chapter 5.

The results for the PRMs generated in small scale environments are presented in the tables below:

- Small scale environment
  - 1000 Point tables: B.1 and B.2
  - 5000 Point tables: B.3 and B.4
  - 10000 Point tables: B.5 and B.6
- Larse scale environment
  - 1000 Point tables: B.7. Table two is not present as there was a 0% success rate at planning all routes for all methods.
  - 5000 Point tables: B.8 and B.9
  - 10000 Point tables: B.10 and B.11

Table B.1: 256m^2 environment, 1000 points PRM experiment full results (Table 1 of 2) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	0.37	0.01	841.30	12.23	811.30	13.00	1.00
Halton	0.44	0.01	847.55	1.32	812.75	1.51	1.00
25% Trails 75% Halton	0.37	0.01	889.60	3.40	1062.80	11.44	0.25
50% Trails 50% Halton	0.30	0.02	909.45	9.23	1263.75	22.34	0.50
75% Trails 25% Halton	0.22	0.02	931.60	11.16	1467.00	34.62	0.75
100% Trails	0.19	0.02	934.65	35.89	1746.35	47.45	0.00
Trail Bias	0.32	0.01	827.65	21.13	775.95	23.01	1.00
100% Gaussian	1.24	0.03	978.30	3.51	954.85	6.13	2.00
25% Gaus 75% Halton	0.64	0.02	892.95	3.49	868.60	4.49	1.25
50% Gaus 50% Halton	0.83	0.01	923.80	2.46	892.50	4.49	1.50
75% Gaus 25% Halton	1.02	0.03	953.35	3.13	886.60	15.17	1.75
25% Trail 75% Gaus	0.98	0.02	992.45	3.89	1546.55	12.73	1.50
50% Trail 50% Gaus	0.72	0.02	983.35	9.12	1355.00	22.33	1.00
75%Trail 25% Gaus	0.46	0.02	972.70	11.90	1546.55	27.72	1.50
25% Trail 25% Gaus 50% Halton	0.37	0.01	502.90	3.03	648.30	13.44	1.00
25% Trail 50% Gaus 25% Halton	0.68	0.02	748.55	4.15	916.60	12.40	1.25
50% Trail 25% Gaus 25% Halton	0.41	0.02	737.70	8.91	1091.75	23.69	0.75
MergeTrails	0.05	0.01	177.70	28.30	1002.90	35.59	0.00
25% MergeTrails 75% Halton	0.36	0.00	710.60	17.03	888.95	18.09	0.75
50% MergeTrails 50% Halton	0.25	0.01	530.05	12.59	893.70	16.16	0.50
75% MergeTrails 25% Halton	0.13	0.01	358.10	17.27	901.85	27.01	0.25
25% MergeTrails 75% Gaus	0.97	0.03	813.45	16.50	997.05	19.26	1.50
50% MergeTrails 50% Gaus	0.66	0.02	603.55	12.68	985.90	20.99	1.00
75% MergeTrails 25% Gaus	0.37	0.02	398.25	17.70	980.30	24.34	0.50
25% MergeTrails 25% Gaus 50% Halton	0.33	0.02	324.05	16.20	466.55	21.09	1.00
25% MergeTrails 50% Gaus 25% Halton	0.65	0.02	569.95	16.02	738.85	17.90	1.25
50% MergeTrails 25% Gaus 25% Halton	0.34	0.02	358.05	12.36	715.75	20.91	0.75

Table B.2: 256m^2 environment, 1000 points PRM experiment full results (Table 2 of 2) (n=20) (* Length is as a ratio to the shortest average planned path)

Point Generation Type	Overall Success Rate	Easy Route	Hard Route	Length*	Planning Time (ms)	Std Dev
Random	36%	55%	17%	7.75	7.82	4.39
Halton	67%	100%	33%	5.83	4.78	3.64
25% Trails 75% Halton	29%	58%	0%	2.22	10.82	4.12
50% Trails 50% Halton	33%	60%	7%	6.78	9.04	5.55
75% Trails 25% Halton	17%	33%	0%	10.67	12.16	3.16
100% Trails	16%	32%	0%	7.73	17.24	5.82
Trail Bias	15%	28%	2%	8.47	4.77	1.96
100% Gaussian	0%	0%	0%	N/A	N/A	N/A
25% Gaus 75% Halton	33%	65%	2%	2.01	9.87	3.78
50% Gaus 50% Halton	32%	62%	2%	2.23	11.08	6.24
75% Gaus 25% Halton	3%	5%	0%	1.09	0.67	0.00
25% Trail 75% Gaus	11%	22%	0%	2.43	13.29	3.91
50% Trail 50% Gaus	18%	35%	0%	1.58	12.39	2.21
75%Trail 25% Gaus	15%	30%	0%	1.14	11.18	3.18
25% Trail 25% Gaus 50% Halton	27%	53%	0%	2.03	10.22	3.32
25% Trail 50% Gaus 25% Halton	13%	27%	0%	1.99	9.45	4.05
50% Trail 25% Gaus 25% Halton	16%	30%	2%	1.41	8.75	5.01
MergeTrails	10%	20%	0%	9.69	4.55	0.55
25% MergeTrails 75% Halton	28%	57%	0%	2.01	10.08	4.63
50% MergeTrails 50% Halton	33%	60%	7%	5.94	4.28	2.65
75% MergeTrails 25% Halton	14%	28%	0%	9.67	2.81	0.81
25% MergeTrails 75% Gaus	6%	12%	0%	2.08	10.92	3.58
50% MergeTrails 50% Gaus	11%	22%	0%	1.33	6.85	2.15
75% MergeTrails 25% Gaus	14%	28%	0%	8.65	3.22	0.87
25% MergeTrails 25% Gaus 50% Halton	28%	55%	0%	1.87	8.59	4.21
25% MergeTrails 50% Gaus 25% Halton	8%	15%	0%	1.24	5.33	3.00
50% MergeTrails 25% Gaus 25% Halton	15%	30%	0%	1.19	4.63	1.88

Table B.3: 256m^2 environment, 5000 points PRM experiment full results (Table 1 of 2) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	1.30	0.02	4076.40	17.25	4072.70	17.41	5.00
Halton	1.52	0.01	4241.70	1.27	4218.45	1.50	5.00
25% Trails 75% Halton	2.58	0.10	8567.75	40.20	10685.10	77.00	1.25
50% Trails 50% Halton	1.24	0.06	4337.25	40.36	6453.00	76.14	2.50
75% Trails 25% Halton	1.13	0.08	4242.55	70.77	7461.20	91.39	3.75
100% Trails	0.88	0.07	4033.45	103.89	8348.45	133.23	0.00
Trail Bias	1.32	0.02	3976.20	31.84	3965.55	32.21	5.00
100% Gaussian	5.57	0.09	4517.75	17.69	4503.80	17.61	10.00
25% Gaus 75% Halton	2.59	0.04	4405.45	4.55	4392.85	4.95	6.25
50% Gaus 50% Halton	3.63	0.07	4499.50	6.38	4491.20	6.06	7.50
75% Gaus 25% Halton	4.63	0.08	4526.70	12.65	4522.05	12.60	8.75
25% Trail 75% Gaus	4.48	0.11	4635.25	31.77	7593.30	44.31	7.50
50% Trail 50% Gaus	3.35	0.08	4583.85	43.98	6667.05	75.38	5.00
75%Trail 25% Gaus	2.18	0.11	4402.75	67.81	7593.30	89.15	7.50
25% Trail 25% Gaus 50% Halton	1.76	0.06	2385.45	25.77	3427.65	39.05	5.00
25% Trail 50% Gaus 25% Halton	3.16	0.09	3541.80	33.13	4583.95	43.14	6.25
50% Trail 25% Gaus 25% Halton	1.94	0.07	3431.55	39.30	5532.55	74.83	3.75
MergeTrails	0.21	0.01	440.55	32.19	4801.00	56.53	0.00
25% MergeTrails 75% Halton	1.24	0.02	3391.60	33.65	4470.70	40.17	3.75
50% MergeTrails 50% Halton	0.96	0.01	2449.60	31.37	4618.70	45.49	2.50
75% MergeTrails 25% Halton	0.63	0.01	1439.95	26.48	4706.90	35.92	1.25
25% MergeTrails 75% Gaus	4.25	0.07	3677.05	39.39	4755.50	45.40	7.50
50% MergeTrails 50% Gaus	3.05	0.05	2697.60	32.41	4863.30	47.42	5.00
75% MergeTrails 25% Gaus	1.66	0.03	1597.65	24.94	4858.55	34.64	2.50
25% MergeTrails 25% Gaus 50% Halton	1.56	0.03	1425.30	32.72	2500.40	39.65	5.00
25% MergeTrails 50% Gaus 25% Halton	2.92	0.05	2577.00	37.53	3656.30	42.70	6.25
50% MergeTrails 25% Gaus 25% Halton	1.61	0.02	1545.20	30.16	3708.65	48.68	3.75

Table B.4: 256m^2 environment, 5000 points PRM experiment full results (Table 2 of 2) (n=20) (* Length is as a ratio to the shortest average planned path)

Point Generation Type	Overall Success Rate	Easy Route	Hard Route	Length*	Planning Time (ms)	Std Dev
Random	87%	100%	73%	5.86	214.08	130.19
Halton	100%	100%	100%	4.41	115.28	77.04
25% Trails 75% Halton	100%	100%	100%	4.94	692.21	414.12
50% Trails 50% Halton	99%	100%	98%	4.98	239.27	138.47
75% Trails 25% Halton	93%	97%	88%	4.44	307.13	199.59
100% Trails	53%	73%	33%	4.51	410.36	230.12
Trail Bias	70%	95%	45%	6.13	158.27	114.25
100% Gaussian	23%	43%	2%	2.35	202.31	142.86
25% Gaus 75% Halton	98%	100%	97%	7.32	129.23	73.04
50% Gaus 50% Halton	100%	100%	100%	8.67	153.33	89.33
75% Gaus 25% Halton	100%	100%	100%	8.83	174.28	106.24
25% Trail 75% Gaus	54%	68%	40%	7.13	245.46	146.82
50% Trail 50% Gaus	62%	73%	50%	5.15	288.83	203.89
75%Trail 25% Gaus	67%	78%	55%	4.39	338.32	244.48
25% Trail 25% Gaus 50% Halton	100%	100%	100%	5.79	196.73	113.37
25% Trail 50% Gaus 25% Halton	100%	100%	100%	5.82	218.54	136.00
50% Trail 25% Gaus 25% Halton	100%	100%	100%	5.28	267.24	171.98
MergeTrails	41%	63%	18%	3.80	29.12	14.80
25% MergeTrails 75% Halton	91%	100%	82%	4.55	101.76	65.58
50% MergeTrails 50% Halton	97%	97%	97%	4.30	80.06	49.23
75% MergeTrails 25% Halton	87%	97%	77%	3.73	52.77	34.64
25% MergeTrails 75% Gaus	50%	68%	32%	6.81	166.96	106.56
50% MergeTrails 50% Gaus	57%	67%	47%	3.56	108.43	88.30
75% MergeTrails 25% Gaus	62%	77%	47%	3.88	68.63	49.05
25% MergeTrails 25% Gaus 50% Halton	99%	100%	98%	5.02	120.69	72.67
25% MergeTrails 50% Gaus 25% Halton	100%	100%	100%	5.03	135.63	94.51
50% MergeTrails 25% Gaus 25% Halton	83%	97%	68%	4.31	96.26	66.29

Table B.5: 256m^2 environment, 10000 points PRM experiment full results (Table 1 of 2) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	2.25	0.07	7837.95	38.48	7836.60	38.59	10.00
Halton	2.70	0.07	8477.65	1.80	8456.20	1.50	10.00
25% Trails 75% Halton	2.58	0.10	8567.75	40.20	10685.10	77.00	2.50
50% Trails 50% Halton	2.40	0.11	8282.30	103.96	12604.30	133.51	5.00
75% Trails 25% Halton	2.09	0.14	7766.45	123.69	14292.90	164.67	7.50
100% Trails	1.70	0.22	7056.95	150.11	15724.95	239.64	0.00
Trail Bias	2.35	0.07	7656.20	73.07	7653.65	73.12	10.00
100% Gaussian	10.58	0.19	8226.25	33.24	8211.10	32.40	20.00
25% Gaus 75% Halton	4.90	0.07	8728.55	8.67	8718.85	9.30	12.50
50% Gaus 50% Halton	7.03	0.08	8760.45	20.82	8753.55	20.98	15.00
75% Gaus 25% Halton	9.01	0.16	8582.30	23.59	8576.45	23.75	17.50
25% Trail 75% Gaus	8.90	0.25	8673.25	48.23	14431.25	94.28	15.00
50% Trail 50% Gaus	6.71	0.23	8547.95	102.78	12747.25	149.67	10.00
75%Trail 25% Gaus	4.31	0.14	8011.35	126.38	14431.25	155.65	15.00
25% Trail 25% Gaus 50% Halton	3.36	0.10	4586.20	44.33	6672.55	68.39	10.00
25% Trail 50% Gaus 25% Halton	6.02	0.16	6724.45	42.34	8787.20	96.06	12.50
50% Trail 25% Gaus 25% Halton	3.83	0.17	6413.55	101.83	10670.00	128.90	7.50
MergeTrails	0.41	0.02	542.00	27.92	9323.05	76.67	0.00
25% MergeTrails 75% Halton	2.26	0.04	6679.90	31.02	8853.10	44.95	7.50
50% MergeTrails 50% Halton	1.66	0.01	4690.05	32.21	9090.95	53.91	5.00
75% MergeTrails 25% Halton	1.09	0.01	2627.75	35.67	9222.90	78.56	2.50
25% MergeTrails 75% Gaus	8.33	0.14	6793.00	41.06	8956.75	54.50	15.00
50% MergeTrails 50% Gaus	5.85	0.09	4962.70	35.55	9349.85	55.61	10.00
75% MergeTrails 25% Gaus	3.11	0.05	2874.70	32.50	9459.95	78.05	5.00
25% MergeTrails 25% Gaus 50% Halton	2.99	0.04	2697.85	28.76	4863.45	46.10	10.00
25% MergeTrails 50% Gaus 25% Halton	5.65	0.05	4832.35	42.07	6995.50	56.05	12.50
50% MergeTrails 25% Gaus 25% Halton	3.12	0.07	2818.70	35.83	7211.35	55.83	7.50

Table B.6: 256m^2 environment, 10000 points PRM experiment full results (Table 2 of 2) (n=20) (* Length is as a ratio to the shortest average planned path)

Point Generation Type	Overall Success Rate	Easy Route	Hard Route	Length*	Planning Time (ms)	Std Dev
Random	98%	100%	97%	5.29	740.13	505.10
Halton	100%	100%	100%	5.54	492.95	290.80
25% Trails 75% Halton	100%	100%	100%	4.94	692.21	414.12
50% Trails 50% Halton	100%	100%	100%	4.37	930.76	566.45
75% Trails 25% Halton	100%	100%	100%	4.10	1,313.70	881.76
100% Trails	84%	92%	77%	3.91	1,752.47	1,224.32
Trail Bias	87%	100%	73%	5.60	681.29	420.39
100% Gaussian	27%	52%	2%	9.44	687.22	410.23
25% Gaus 75% Halton	100%	100%	100%	9.25	520.48	318.64
50% Gaus 50% Halton	100%	100%	100%	9.16	576.88	346.49
75% Gaus 25% Halton	100%	100%	100%	8.79	594.42	409.23
25% Trail 75% Gaus	65%	73%	57%	5.05	823.58	626.85
50% Trail 50% Gaus	80%	87%	73%	4.50	1,238.64	889.76
75%Trail 25% Gaus	87%	92%	82%	4.08	1,628.45	1,224.83
25% Trail 25% Gaus 50% Halton	100%	100%	100%	4.90	745.30	502.30
25% Trail 50% Gaus 25% Halton	100%	100%	100%	4.89	785.13	579.61
50% Trail 25% Gaus 25% Halton	100%	100%	100%	4.44	1,130.91	776.65
MergeTrails	70%	83%	57%	3.43	68.70	42.82
25% MergeTrails 75% Halton	98%	97%	100%	4.44	426.13	265.18
50% MergeTrails 50% Halton	100%	100%	100%	3.58	291.23	192.81
75% MergeTrails 25% Halton	100%	100%	100%	3.50	180.83	117.44
25% MergeTrails 75% Gaus	62%	68%	55%	4.06	481.09	416.32
50% MergeTrails 50% Gaus	78%	83%	72%	3.62	403.86	338.91
75% MergeTrails 25% Gaus	82%	88%	75%	3.53	259.76	203.68
25% MergeTrails 25% Gaus 50% Halton	98%	97%	100%	4.20	437.02	315.56
25% MergeTrails 50% Gaus 25% Halton	98%	97%	100%	4.06	467.90	377.01
50% MergeTrails 25% Gaus 25% Halton	100%	100%	100%	3.64	346.23	261.27

Table B.7: 512m^2 environment, 1000 points PRM experiment full results (Table 1 of 1) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	1.44	0.05	802.40	11.46	432.75	14.78	1.00
Halton	1.23	0.08	810.25	7.25	290.70	26.65	1.00
25% Trails 75% Halton	1.14	0.07	854.70	6.78	591.45	27.52	0.25
50% Trails 50% Halton	1.05	0.06	882.80	12.92	926.60	28.45	0.50
75% Trails 25% Halton	1.00	0.09	896.25	24.86	1261.40	41.72	0.75
100% Trails	1.12	0.06	902.35	27.38	1625.00	48.32	0.00
Trail Bias	1.42	0.06	788.60	12.78	426.25	19.44	1.00
100% Gaussian	5.94	0.16	1005.10	2.32	728.15	10.69	2.00
25% Gaus 75% Halton	2.38	0.09	864.40	5.08	375.35	18.98	1.25
50% Gaus 50% Halton	3.49	0.11	912.70	4.26	476.70	12.00	1.50
75% Gaus 25% Halton	4.75	0.15	961.10	4.06	590.05	11.22	1.75
25% Trail 75% Gaus	2.26	0.11	944.40	22.82	1332.35	39.58	1.50
50% Trail 50% Gaus	3.51	0.12	980.45	12.60	1107.00	27.96	1.00
75%Trail 25% Gaus	2.26	0.11	944.40	22.82	1332.35	39.58	1.50
25% Trail 25% Gaus 50% Halton	2.32	0.09	904.00	7.43	685.10	22.05	1.00
25% Trail 50% Gaus 25% Halton	3.46	0.11	951.90	6.63	783.05	22.92	1.25
50% Trail 25% Gaus 25% Halton	2.29	0.11	932.95	14.03	995.10	32.83	0.75
MergeTrails	0.55	0.07	343.30	33.78	1081.60	38.13	0.00
25% MergeTrails 75% Halton	0.99	0.09	727.45	13.71	470.90	33.54	0.75
50% MergeTrails 50% Halton	0.81	0.08	618.35	23.54	669.05	33.72	0.50
75% MergeTrails 25% Halton	0.59	0.06	475.90	26.29	850.00	32.79	0.25
25% MergeTrails 75% Gaus	4.62	0.16	871.90	13.26	788.05	21.80	1.50
50% MergeTrails 50% Gaus	3.11	0.08	714.55	22.05	845.75	26.97	1.00
75% MergeTrails 25% Gaus	1.77	0.11	522.85	25.24	918.35	30.57	0.50
25% MergeTrails 25% Gaus 50% Halton	2.16	0.10	777.05	13.85	563.55	27.71	1.00
25% MergeTrails 50% Gaus 25% Halton	3.38	0.11	825.65	12.60	657.60	26.53	1.25
50% MergeTrails 25% Gaus 25% Halton	1.94	0.08	666.15	24.08	736.25	30.45	0.75

Table B.8: 512m^2 environment, 5000 points PRM experiment full results (Table 1 of 2) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	9.29	0.21	3928.70	31.93	3865.90	35.00	5.00
Halton	10.29	0.18	3948.75	10.14	3871.20	10.79	5.00
25% Trails 75% Halton	9.65	0.21	4067.05	38.44	4974.35	48.97	1.25
50% Trails 50% Halton	8.53	0.20	4235.60	45.94	6100.75	67.93	2.50
75% Trails 25% Halton	6.47	0.24	4287.05	57.95	7015.70	85.57	3.75
100% Trails	5.24	0.30	4217.55	75.81	8209.55	90.93	0.00
Trail Bias	9.12	0.18	3896.30	29.28	3831.60	29.76	5.00
100% Gaussian	30.50	0.74	4893.20	12.71	4831.30	15.68	10.00
25% Gaus 75% Halton	15.72	0.46	4217.20	7.14	4151.85	8.28	6.25
50% Gaus 50% Halton	20.86	0.46	4464.95	10.17	4413.85	10.92	7.50
75% Gaus 25% Halton	25.45	0.58	4690.45	9.82	4538.15	17.73	8.75
25% Trail 75% Gaus	24.67	0.73	4784.55	41.75	7416.15	53.48	7.50
50% Trail 50% Gaus	19.10	0.46	4724.60	47.07	6604.15	70.11	5.00
75%Trail 25% Gaus	12.22	0.49	4531.40	58.24	7416.15	78.86	7.50
25% Trail 25% Gaus 50% Halton	14.84	0.46	4325.95	38.79	5237.30	50.13	5.00
25% Trail 50% Gaus 25% Halton	19.33	0.66	4568.15	38.57	5388.05	51.04	6.25
50% Trail 25% Gaus 25% Halton	13.47	0.30	4488.95	44.25	6258.35	66.13	3.75
MergeTrails	1.78	0.09	947.85	33.05	5007.75	51.30	0.00
25% MergeTrails 75% Halton	8.95	0.23	3366.80	44.89	4303.15	55.93	3.75
50% MergeTrails 50% Halton	6.87	0.17	2655.05	39.30	4591.90	47.39	2.50
75% MergeTrails 25% Halton	4.08	0.14	1849.75	44.76	4625.30	71.55	1.25
25% MergeTrails 75% Gaus	23.23	0.47	4082.50	43.83	5019.35	53.10	7.50
50% MergeTrails 50% Gaus	17.19	0.46	3142.00	37.69	5089.70	45.90	5.00
75% MergeTrails 25% Gaus	9.76	0.29	2099.05	46.28	5038.75	60.58	2.50
25% MergeTrails 25% Gaus 50% Halton	14.04	0.23	3623.35	47.51	4568.60	56.60	5.00
25% MergeTrails 50% Gaus 25% Halton	18.86	0.47	3866.60	48.00	4715.85	54.32	6.25
50% MergeTrails 25% Gaus 25% Halton	11.88	0.28	2909.75	37.35	4753.00	47.67	3.75

Table B.9: 512m^2 environment, 5000 points PRM experiment full results (Table 2 of 2) (n=20) (* Length is as a ratio to the shortest average planned path)

Point Generation Type	Overall Success Rate	Easy Route	Hard Route	Length*	Planning Time (ms)	Std Dev
Random	52%	20%	83%	1.89	122.85	104.22
Halton	100%	100%	100%	2.30	101.96	60.79
25% Trails 75% Halton	91%	83%	98%	2.58	130.72	69.42
50% Trails 50% Halton	42%	0%	83%	3.05	155.01	64.87
75% Trails 25% Halton	19%	0%	38%	1.37	112.05	47.42
100% Trails	18%	0%	37%	1.19	77.90	24.90
Trail Bias	63%	37%	88%	1.97	131.02	102.45
100% Gaussian	18%	7%	28%	1.98	127.81	26.69
25% Gaus 75% Halton	87%	73%	100%	2.40	106.62	65.24
50% Gaus 50% Halton	81%	62%	100%	2.87	193.29	92.71
75% Gaus 25% Halton	40%	28%	52%	3.13	233.31	105.51
25% Trail 75% Gaus	20%	2%	38%	2.74	170.81	110.99
50% Trail 50% Gaus	20%	0%	40%	2.24	162.18	39.43
75%Trail 25% Gaus	19%	0%	38%	1.68	132.22	27.74
25% Trail 25% Gaus 50% Halton	76%	53%	98%	2.82	165.59	83.48
25% Trail 50% Gaus 25% Halton	40%	15%	65%	2.74	171.24	88.01
50% Trail 25% Gaus 25% Halton	28%	10%	47%	2.06	209.23	136.00
MergeTrails	18%	0%	35%	1.12	12.50	4.50
25% MergeTrails 75% Halton	88%	77%	98%	2.28	85.77	44.63
50% MergeTrails 50% Halton	39%	0%	78%	2.89	53.99	17.75
75% MergeTrails 25% Halton	18%	2%	35%	1.12	26.12	12.49
25% MergeTrails 75% Gaus	21%	0%	42%	3.38	144.85	71.19
50% MergeTrails 50% Gaus	18%	0%	37%	1.30	33.78	11.21
75% MergeTrails 25% Gaus	19%	0%	38%	1.20	32.35	21.25
25% MergeTrails 25% Gaus 50% Halton	67%	38%	95%	2.26	108.20	60.56
25% MergeTrails 50% Gaus 25% Halton	38%	18%	58%	2.52	162.91	83.93
50% MergeTrails 25% Gaus 25% Halton	23%	3%	42%	1.88	61.30	15.55

Table B.10: 512m^2 environment, 10000 points PRM experiment full results (Table 1 of 2) (n=20)

Point Generation Type	Time (minutes)	Std. Dev.	Points	Std. Dev.	Edges	Std. Dev.	Cost
Random	15.73	0.42	7763.70	44.27	7734.85	45.30	10.00
Halton	17.82	0.32	7885.70	5.62	7809.05	6.62	10.00
25% Trails 75% Halton	16.74	0.53	8157.50	45.21	10068.85	66.33	2.50
50% Trails 50% Halton	14.81	0.63	8160.10	80.68	12189.00	92.22	5.00
75% Trails 25% Halton	13.46	0.49	8555.40	54.44	14652.65	79.59	7.50
100% Trails	10.53	0.39	8449.85	102.32	16641.90	124.41	0.00
Trail Bias	15.97	0.34	7717.95	30.10	7688.50	30.66	10.00
100% Gaussian	60.13	1.54	9563.75	25.44	9519.40	24.70	20.00
25% Gaus 75% Halton	28.48	0.77	8384.75	8.23	8338.15	8.49	12.50
50% Gaus 50% Halton	38.26	0.95	8831.75	13.14	8797.40	12.75	15.00
75% Gaus 25% Halton	49.03	1.32	9232.10	15.61	9195.80	14.97	17.50
25% Trail 75% Gaus	24.06	0.62	9044.10	56.34	15149.55	90.31	15.00
50% Trail 50% Gaus	35.30	0.83	9103.65	70.74	13129.60	86.29	10.00
75%Trail 25% Gaus	24.06	0.62	9044.10	56.34	15149.55	90.31	15.00
25% Trail 25% Gaus 50% Halton	27.30	0.40	8661.95	49.89	10591.35	70.16	10.00
25% Trail 50% Gaus 25% Halton	37.70	0.64	9116.70	45.61	11046.35	69.31	12.50
50% Trail 25% Gaus 25% Halton	25.33	0.94	8672.30	73.50	12706.85	84.19	7.50
MergeTrails	2.65	0.14	1282.05	42.18	9626.90	62.82	0.00
25% MergeTrails 75% Halton	15.32	0.53	6576.05	37.96	8563.80	42.49	7.50
50% MergeTrails 50% Halton	11.89	0.31	4888.30	35.25	9034.60	47.22	5.00
75% MergeTrails 25% Halton	8.04	0.23	3166.75	55.97	9424.70	68.21	2.50
25% MergeTrails 75% Gaus	45.83	0.73	7907.70	41.30	9904.65	43.75	15.00
50% MergeTrails 50% Gaus	31.37	0.76	5831.80	30.74	9974.55	43.10	10.00
75% MergeTrails 25% Gaus	18.20	0.37	3654.40	52.27	9922.50	62.12	5.00
25% MergeTrails 25% Gaus 50% Halton	26.18	0.54	7079.90	34.58	9088.70	41.84	10.00
25% MergeTrails 50% Gaus 25% Halton	36.37	0.99	7536.60	39.54	9548.90	45.27	12.50
50% MergeTrails 25% Gaus 25% Halton	22.52	0.46	5399.95	35.22	9560.60	47.49	7.50

Table B.11: 512m^2 environment, 10000 points PRM experiment full results (Table 2 of 2) (n=20) (* Length is as a ratio to the shortest average planned path)

Point Generation Type	Overall Success Rate	Easy Route	Hard Route	Length*	Planning Time (ms)	Std Dev
Random	98%	95%	100%	2.23	881.41	675.16
Halton	100%	100%	100%	2.39	549.24	311.15
25% Trails 75% Halton	99%	100%	98%	2.40	669.02	349.53
50% Trails 50% Halton	100%	100%	100%	1.93	924.83	505.57
75% Trails 25% Halton	55%	10%	100%	1.86	1504.86	748.28
100% Trails	33%	18%	48%	1.97	2706.11	1332.66
Trail Bias	99%	98%	100%	1.96	701.75	594.83
100% Gaussian	25%	17%	33%	2.36	1277.35	418.45
25% Gaus 75% Halton	100%	100%	100%	2.67	647.91	300.66
50% Gaus 50% Halton	100%	100%	100%	2.82	1236.54	617.57
75% Gaus 25% Halton	83%	67%	100%	3.18	1773.47	967.98
25% Trail 75% Gaus	38%	7%	70%	1.78	2403.89	1310.74
50% Trail 50% Gaus	44%	28%	60%	1.93	2283.86	1573.43
75%Trail 25% Gaus	38%	7%	70%	1.78	2403.89	1310.74
25% Trail 25% Gaus 50% Halton	98%	98%	98%	2.48	1282.41	647.03
25% Trail 50% Gaus 25% Halton	83%	67%	100%	2.86	1441.38	895.00
50% Trail 25% Gaus 25% Halton	83%	67%	100%	2.26	1472.43	855.48
MergeTrails	27%	12%	42%	1.77	108.74	46.28
25% MergeTrails 75% Halton	100%	100%	100%	2.01	384.78	195.61
50% MergeTrails 50% Halton	100%	100%	100%	1.55	278.47	169.34
75% MergeTrails 25% Halton	56%	12%	100%	1.36	173.33	89.69
25% MergeTrails 75% Gaus	40%	28%	52%	2.36	1451.47	871.14
50% MergeTrails 50% Gaus	40%	20%	60%	1.96	775.12	448.73
75% MergeTrails 25% Gaus	36%	12%	60%	1.67	375.70	173.87
25% MergeTrails 25% Gaus 50% Halton	100%	100%	100%	2.00	707.70	372.12
25% MergeTrails 50% Gaus 25% Halton	83%	67%	100%	2.44	990.85	543.23
50% MergeTrails 25% Gaus 25% Halton	81%	62%	100%	1.96	593.80	335.14

APPENDIX C

Full Segementation Results

This appendix reports the full table of results for the experiments done in Chapter 7.

- 256m^2 London Community Region: C.1 and C.2
- 256m^2 Hyde Park Region: C.3 and C.4
- 256m^2 Kensington Region: C.5 and C.6
- 512m^2 Greater London Area: C.7 and C.8

Table C.1: Full table of results for the 256m^2 London Community Environment (Table 1 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		10441.55	247.18	2.60	11.11	1.00	0.00	494.12	54.81	100%
Regular	2	4537761.95	51633.58	3691404.55	18104.70	1.00	0.00	510.41	0.00	100%
	4	608307.50	6851.07	500681.95	6820.59	1.00	0.00	510.00	0.00	100%
	8	116698.35	1312.10	100896.25	1314.63	1.00	0.00	510.00	0.00	100%
	16	28562.95	352.67	25477.80	340.04	1.00	0.00	722.59	25.86	100%
	32	7796.80	252.97	6816.30	217.79	1.00	0.00	740.83	31.65	100%
	64	2713.30	238.00	2033.05	193.15	1.00	0.00	818.22	49.39	100%
	128	2243.05	260.16	754.40	191.79	1.00	0.00	710.52	50.62	100%
Quadtree	2	10445.15	274.34	10013.55	249.63	1.00	0.00	615.78	22.88	100%
	4	9915.90	231.02	9494.85	199.87	1.00	0.00	612.93	29.64	100%
	8	8939.15	248.05	8570.45	216.04	1.00	0.00	623.12	37.21	100%
	16	7901.60	259.36	7540.30	214.32	1.00	0.00	630.24	43.62	100%
	32	7131.70	243.11	6711.80	214.20	1.00	0.00	670.97	42.95	100%
	64	5729.15	244.80	4849.40	201.65	1.00	0.00	750.89	66.44	100%
	128	5160.50	247.14	2757.40	189.56	1.00	0.00	715.28	71.31	100%

Table C.2: Full table of results for the 256m^2 London Community Environment (Table 2 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
Voronoi	10	2414.00	489.74	1101.00	186.72	1.05	0.22	775.38	103.84	100%
- Random	50	6909.95	323.46	6038.10	246.52	1.15	0.36	736.30	84.83	100%
	100	13459.25	420.60	12295.15	318.33	1.25	0.43	742.96	87.29	100%
	250	34843.45	1026.29	32233.75	609.45	1.45	1.02	679.08	54.21	100%
	500	71989.05	2065.87	66296.50	1232.90	1.25	0.54	615.72	39.73	100%
	750	111360.70	2539.55	102426.75	1907.98	1.30	0.56	564.10	36.06	100%
	1000	152466.50	4467.72	138880.60	1542.21	1.45	0.97	551.42	27.89	100%
Voronoi	10	2391.90	316.63	1123.35	182.09	1.00	0.00	755.43	90.45	100%
- Halton	50	6693.00	267.84	5959.90	212.31	1.00	0.00	725.89	60.49	100%
	100	13403.65	388.96	12374.90	249.74	1.20	0.60	736.48	61.04	100%
	250	34469.85	738.02	32247.45	584.78	1.20	0.40	659.03	44.56	100%
	500	71427.00	921.30	66555.60	827.20	1.05	0.22	582.62	29.46	100%
	750	112719.00	3045.36	103447.45	1486.44	1.50	0.59	520.66	23.81	100%
	1000	155394.65	3397.42	142942.55	1547.12	1.15	0.36	480.59	13.75	100%
Voronoi	10	2505.05	438.30	1205.30	199.31	1.10	0.30	735.90	67.62	100%
- Trail	50	6779.10	289.53	6004.70	254.06	1.10	0.44	761.59	86.89	100%
Density	100	13891.90	662.40	12711.75	410.58	1.35	0.79	742.53	84.07	100%
Clustering	250	23141.60	914.08	21546.45	759.77	1.05	0.22	678.80	63.96	100%
	500	23476.10	881.08	21886.90	664.16	1.05	0.22	678.29	69.67	100%
	750	23215.80	889.59	21628.10	701.08	1.05	0.22	677.88	66.92	100%
	1000	23201.75	775.61	21614.55	639.81	1.05	0.22	681.91	58.41	100%
Voronoi	10	2528.40	396.55	1202.10	236.96	1.00	0.00	735.63	88.52	100%
- Trail	50	6854.00	316.54	5964.95	208.67	1.35	0.79	799.86	135.88	100%
Density	100	13108.20	681.43	12056.75	426.39	1.15	0.65	746.54	86.11	100%
Clustering	250	31279.05	1840.28	29179.55	1589.73	1.25	0.54	705.30	78.01	100%
And	500	60494.80	4835.11	56107.95	4764.71	1.25	0.54	629.00	36.84	100%
Halton	750	95892.50	8184.81	88708.55	7697.94	1.25	0.43	581.82	28.00	95%
	1000	138060.60	9369.23	126326.00	7857.99	1.25	0.54	560.63	26.51	100%

Table C.3: Full table of results for the 256m^2 Hyde Park Environment (Table 1 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		62793.30	438.78	5.45	23.30	1.00	0.00	490.20	47.02	100%
Regular	2	4785518.25	75214.82	3969151.10	73719.62	1.00	0.00	509.78	0.00	100%
	4	666837.90	7623.57	558895.45	7579.01	1.00	0.00	511.24	0.39	100%
	8	135103.90	1929.04	118949.05	1958.49	1.00	0.00	510.37	0.95	100%
	16	34781.90	608.70	31522.45	603.93	1.00	0.00	768.52	34.35	100%
	32	11825.05	885.49	9155.80	174.30	2.65	0.96	822.46	50.88	100%
	64	4598.05	267.15	3402.15	215.73	1.00	0.00	948.41	56.17	100%
	128	8957.20	305.98	2232.55	217.27	1.00	0.00	683.78	63.53	100%
Quadtree	2	74738.05	972.55	73694.70	989.62	1.00	0.00	603.43	25.78	100%
	4	61466.75	832.42	60592.05	832.03	1.00	0.00	571.44	19.61	100%
	8	43425.00	768.64	42520.55	759.79	1.00	0.00	559.68	30.01	100%
	16	25512.40	453.79	24737.30	404.62	1.00	0.00	564.97	26.69	100%
	32	16640.50	762.84	15104.05	706.56	1.00	0.00	613.44	29.01	100%
	64	19786.60	1518.06	10496.10	350.48	1.55	0.50	621.75	144.02	100%
	128	70777.40	987.89	7895.15	315.10	1.00	0.00	493.13	30.52	100%

Table C.4: Full table of results for the 256m^2 Hyde Park (Table 2 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
Voronoi	10	5658.75	1493.78	2379.00	244.67	1.30	0.78	823.49	441.41	95%
- Random	50	9852.95	704.71	8392.15	558.54	1.60	0.86	748.55	78.89	100%
	100	17773.35	720.01	15814.20	358.32	1.95	1.02	719.54	72.22	100%
	250	42891.70	1734.49	39061.55	874.44	2.00	0.95	669.01	47.65	100%
	500	84946.10	2116.65	78517.70	1124.01	1.40	0.66	599.74	43.46	100%
	750	129927.85	4108.12	120590.60	1582.16	1.30	0.78	561.94	25.12	100%
	1000	177294.90	4783.89	163495.55	2054.14	1.45	0.80	543.13	31.75	95%
Voronoi	10	5110.60	1784.00	2628.75	1458.28	1.15	0.36	904.58	741.99	100%
- Halton	50	9714.85	1332.75	8431.55	1316.48	1.75	0.89	644.60	44.21	80%
	100	17908.80	1465.65	16043.95	1420.08	2.00	1.05	689.52	40.05	100%
	250	42654.95	1788.28	39392.70	1722.05	1.75	0.89	651.08	41.68	100%
	500	86078.40	2122.61	80394.50	1802.06	1.20	0.51	589.09	21.03	100%
	750	132109.35	3511.77	122155.65	1810.26	1.40	0.73	520.16	20.86	100%
	1000	178318.50	2105.12	166134.15	1921.07	1.10	0.30	492.75	13.31	100%
Voronoi	10	5562.05	1739.90	2573.60	575.00	1.35	0.79	773.10	335.78	90%
- Trail	50	9864.45	814.30	8272.15	548.82	1.85	1.06	745.22	83.52	85%
Density	100	17782.80	1064.06	16001.90	465.86	1.90	1.45	729.01	78.29	95%
Clustering	250	43704.95	1396.47	39572.45	991.03	2.25	0.94	702.81	63.61	100%
	500	59884.60	5180.64	55372.85	4563.14	1.75	0.94	702.36	60.55	100%
	750	59792.60	4954.78	55253.35	4474.57	1.75	0.94	704.10	57.88	100%
	1000	59955.50	5137.18	55423.75	4710.44	1.75	0.99	701.33	63.04	100%
Voronoi	10	5963.45	1428.87	2513.95	357.11	1.05	0.22	976.50	766.50	100%
- Trail	50	10159.45	792.85	8367.00	251.18	2.35	1.35	734.67	90.31	95%
Density	100	18154.15	1478.19	15980.40	302.69	2.35	2.13	714.39	104.32	100%
Clustering	250	42358.05	1176.51	39444.10	932.32	1.35	0.79	692.38	49.06	100%
And	500	84499.05	1892.24	78853.20	1627.93	1.20	0.40	612.71	24.42	100%
Halton	750	128481.25	2098.21	119932.05	1789.41	1.10	0.30	571.24	21.91	100%
	1000	177844.10	4786.90	162575.75	1671.25	1.50	0.67	539.74	24.90	100%

Table C.5: Full table of results for the 256m^2 Kensington Environment (Table 1 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		104728.95	1152.50	2.75	10.16	1.00	0.00	493.70	25.99	100%
Regular	2	4945896.75	76547.81	4129241.30	76061.41	1.00	0.00	509.78	0.00	100%
	4	721583.85	10303.91	613313.80	10298.21	1.00	0.00	510.00	0.00	100%
	8	154182.85	5475.01	137763.20	5494.76	1.00	0.00	509.53	0.59	100%
	16	39973.50	765.06	36513.10	739.00	1.00	0.00	768.77	25.41	100%
	32	13682.85	340.80	11158.40	235.29	2.75	0.43	834.73	53.33	100%
	64	5922.00	218.67	4703.50	148.96	1.00	0.00	809.05	33.98	100%
	128	22247.05	450.93	4915.50	220.47	1.00	0.00	630.60	31.36	100%
Quadtree	2	153215.90	977.73	152297.70	1003.94	1.00	0.00	588.99	23.90	100%
	4	143775.60	864.25	142878.70	881.98	1.00	0.00	585.98	33.94	100%
	8	122592.75	922.13	121755.00	927.60	1.00	0.00	586.98	20.88	100%
	16	90808.05	736.07	90045.55	742.37	1.00	0.00	609.96	22.78	100%
	32	63981.60	1319.32	62192.10	1243.40	3.00	0.00	617.57	27.61	100%
	64	51453.80	1269.72	48886.75	1098.05	2.40	0.49	669.12	42.80	100%
	128	62954.70	518.25	39452.35	448.89	2.00	0.00	544.43	37.34	100%

Table C.6: Full table of results for the 256m^2 Kensington Environment (Table 2 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
Voronoi	10	13160.65	4687.24	4026.20	310.32	1.35	0.65	850.78	520.61	95%
- Random	50	12075.65	809.85	10036.50	226.87	2.50	1.40	736.32	73.08	100%
	100	21063.70	1183.95	18626.30	376.25	2.45	1.43	734.75	107.72	100%
	250	48614.30	1226.64	44888.45	768.81	1.95	0.86	684.50	38.93	100%
	500	96450.25	3217.25	89181.80	1350.05	1.80	1.08	623.54	37.58	100%
	750	147656.80	5173.51	135916.40	1742.69	1.85	0.96	580.17	32.64	100%
	1000	202374.55	6106.04	186893.90	5738.59	1.55	0.67	545.40	21.80	100%
Voronoi	10	8851.55	1450.83	4044.75	270.05	1.05	0.22	708.56	248.60	100%
- Halton	50	11627.10	971.69	9968.95	239.64	2.20	1.81	725.80	65.26	90%
	100	21202.45	2244.56	19191.85	1469.95	2.05	1.40	698.96	61.74	100%
	250	48301.10	1729.61	44424.85	764.89	2.25	1.13	672.18	45.09	100%
	500	96973.30	2021.49	90747.35	1079.45	1.55	0.80	596.80	25.37	100%
	750	147797.75	4509.14	136949.95	2249.51	1.95	1.20	526.28	20.24	100%
	1000	202419.25	4380.29	188308.35	3321.88	1.45	0.92	491.14	17.75	100%
Voronoi	10	10824.95	3177.81	4154.90	357.49	1.25	0.43	781.74	474.06	100%
- Trail	50	12045.75	891.80	10318.80	208.89	2.15	1.62	741.44	136.79	100%
Density	100	21752.35	1928.15	18823.00	500.87	3.20	2.56	700.66	53.68	90%
Clustering	250	51258.75	5995.23	46300.10	3489.55	2.85	2.29	656.49	49.38	100%
	500	54274.55	3970.14	49366.55	3765.08	2.85	1.46	669.81	40.44	100%
	750	54314.70	4418.48	49322.65	3996.57	2.90	1.48	667.74	42.24	100%
	1000	54166.25	4044.86	49289.10	3806.44	2.85	1.46	665.02	48.50	100%
Voronoi	10	11334.55	2385.00	4327.55	427.63	1.40	0.80	736.50	255.49	95%
- Trail	50	12351.35	1577.18	10532.40	860.90	1.85	1.35	725.06	94.47	95%
Density	100	21168.40	1277.13	18440.90	434.76	3.00	1.76	803.91	283.10	95%
Clustering	250	47044.65	1722.26	43596.55	1106.69	1.75	1.44	681.16	46.67	100%
And	500	91758.10	3628.02	84459.75	1963.77	1.90	1.09	635.63	44.80	100%
Halton	750	141391.25	9346.81	129652.45	8146.16	2.20	1.17	591.80	33.56	100%
	1000	189975.45	7850.17	175097.65	3443.45	1.75	1.37	557.86	30.76	100%

Table C.7: Full table of results for the 512m^2 Greater London Area Environment (Table 1 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
None		660369.60	5219.01	2.35	9.79	1.00	0.00	1014.08	66.84	70%
Regular	2	60655289.65	600875.50	45308203.70	170726.97	0.95	0.00	1021.24	222.57	95%
	4	6585620.10	105193.05	4977931.60	92676.29	1.00	0.00	1024.70	0.87	100%
	8	1065745.40	6855.61	851656.60	6850.19	1.00	0.00	1022.40	1.02	100%
	16	263796.75	19496.45	201580.30	2825.82	4.05	1.47	1486.67	62.50	100%
	32	82252.10	1587.13	55200.50	925.07	9.60	0.92	1615.56	43.64	15%
	64	25722.70	1948.93	18970.70	282.49	3.90	0.99	1810.92	109.90	40%
	128	46395.50	11995.11	13454.95	341.34	3.95	1.36	3476.85	3796.99	40%
Quadtree	2	532605.50	9237.38	512312.55	4215.27	9.80	3.93	1289.92	58.27	80%
	4	486941.60	9350.92	468357.50	3209.93	9.05	5.15	1258.03	53.57	85%
	8	408022.35	8548.34	388445.60	3672.20	10.55	4.54	1302.26	97.95	85%
	16	294809.40	5866.41	276355.55	1702.81	13.05	3.67	1334.30	62.90	80%
	32	187808.15	1879.78	178989.45	953.42	8.60	1.36	1288.57	89.76	95%
	64	122678.20	1559.45	117424.95	1090.19	5.30	1.31	N/A	N/A	0%
	128	100981.15	932.91	84734.30	881.07	2.90	0.70	1222.83	66.39	70%

Table C.8: Full table of results for the 512m^2 Greater London Area Environment (Table 2 of 2)

Type	Variation	Total Time (ms)	Std Dev	Segmentation Time (ms)	Std Dev	Number of Top. Plans	Std Dev	Length (m)	Std Dev	Success Rate
Voronoi	10	75037.50	23734.96	12637.45	1242.08	2.45	1.69	2407.46	2282.56	65%
- Random	50	30759.85	7986.82	19050.05	718.11	5.55	4.54	1487.25	330.06	45%
	100	40198.00	3788.54	31768.25	997.73	5.15	3.21	1601.64	288.43	50%
	250	82701.35	8848.74	69046.10	1668.96	6.10	4.46	1492.60	131.26	60%
	500	161592.70	11088.26	135709.45	2735.05	5.85	2.76	1469.40	99.54	85%
	750	246425.45	14777.28	204845.00	3077.04	6.10	3.19	1449.78	76.31	85%
	1000	336483.60	27744.45	276846.90	4230.98	6.75	4.12	1362.99	78.43	100%
Voronoi	10	69841.60	18332.45	12758.55	1036.05	2.60	1.56	2898.93	2945.83	60%
- Halton	50	27622.30	3433.34	18650.95	315.04	4.60	2.46	1330.61	55.16	50%
	100	37607.30	4203.22	30323.35	468.14	5.15	3.24	1598.04	262.96	50%
	250	82525.20	5222.22	69340.10	1574.11	6.05	3.06	1541.28	127.67	55%
	500	166812.55	10880.97	136481.20	2152.71	8.40	4.00	1421.15	107.58	70%
	750	250408.25	18397.01	206985.15	2982.17	8.00	4.55	1314.48	91.84	65%
	1000	332737.85	26040.52	279714.95	6040.12	5.90	4.79	1317.43	62.65	85%
Voronoi	10	73203.45	24769.73	14227.50	949.24	2.75	1.48	1367.06	164.33	45%
- Trail	50	29794.70	4671.94	20066.50	495.99	4.70	2.93	1873.41	796.49	55%
Density	100	40565.65	3213.25	32288.75	681.93	5.45	2.64	1716.86	658.89	45%
Clustering	250	87962.95	7250.01	70277.95	1000.70	7.40	4.09	1523.45	110.79	75%
	500	167625.60	20174.87	136643.60	3313.12	7.20	6.19	1470.06	106.81	95%
	750	253224.80	17219.62	205145.70	3701.95	7.05	2.99	1424.51	100.43	85%
	1000	349754.40	25155.86	281188.55	3252.38	6.50	3.31	1445.86	105.49	75%
Voronoi	10	77268.35	25562.06	13528.80	1423.66	2.70	1.82	3106.69	5159.75	70%
- Trail	50	29603.45	4923.59	19727.90	518.95	4.65	2.15	2613.60	2657.54	45%
Density	100	39718.60	2732.52	32144.15	847.64	4.75	2.09	1488.46	171.47	55%
Clustering	250	87097.60	8599.72	69832.00	1087.48	7.60	4.92	1446.60	110.18	65%
And	500	168212.60	17756.78	136619.25	1875.50	7.10	5.57	1452.38	93.12	95%
Halton	750	254512.85	18818.55	204957.85	3353.05	7.90	4.23	1404.60	69.55	85%
	1000	339432.10	22149.45	274516.70	3654.43	6.65	3.37	1383.04	73.62	75%

List of References

Alen Alempijevic, Robert Fitch, and Nathan Kirchner. Bootstrapping navigation and path planning using human positional traces. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1242–1247. IEEE, 2013.

Nancy M. Amato, O. Burchan Bayazit, Lucia K. Dale, Christopher Jones, and Daniel Vallejo. Obprm: An obstacle-based prm for 3d workspaces. *In Proc. Int. Wkshp. on Alg. Found. of Rob. (WAFR)*, pages 155–168, March 1998.

Gennady Andrienko, Natalia Andrienko, Salvatore Rinzivillo, Mirco Nanni, Dino Pedreschi, and Fosca Giannotti. Interactive visual clustering of large collections of trajectories. In *Visual Analytics Science and Technology, 2009. VAST 2009. IEEE Symposium on*, pages 3–10. IEEE, 2009.

C. Antonya, F.G. Barbuceanu, and Z. Rusak. Path generation in virtual reality environment based on gaze analysis. In *AFRICON, 2011*, pages 1 –4, sept. 2011. doi: 10.1109/AFRCON.2011.6072169.

M. Augustine, F. Ortmeier, E. Mair, D. Burschka, A. Stelzer, and M. Suppa. Landmark-tree map: A biologically inspired topological map for long-distance robot navigation. In *Robotics and Biomimetics (ROBIO), 2012 IEEE International Conference on*, pages 128–135, 2012. doi: 10.1109/ROBIO.2012.6490955.

Franz Aurenhammer. Voronoi diagrams – a survey of a fundamental geometric data structure. *ACM Computing Surveys (CSUR)*, 23(3):345–405, 1991.

M.R.B. Bahar, A.R. Ghiasi, and H.B. Bahar. Grid roadmap based {ANN} corridor search for collision free, path planning. *Scientia Iranica*, 19(6):1850 – 1855, 2012. ISSN 1026-3098. doi: 10.1016/j.scient.2012.02.028. URL `http://www.sciencedirect.com/science/article/pii/S1026309812001046`.

Rafael Garcia Barbosa and Maria Andréia Formico Rodrigues. Supporting guided navigation in mobile virtual environments. In *VRST '06: Proceedings of the ACM symposium on Virtual reality software and technology*, pages 220–226, New York, NY, USA, 2006. ACM. ISBN 1-59593-321-2. doi: http://doi.acm.org/10.1145/1180495.1180541.

J. Barraquand, B. Langlois, and J.-C. Latombe. Numerical potential field techniques for robot path planning. *Systems, Man and Cybernetics, IEEE Transactions on*, 22(2):224–241, 1992. ISSN 0018-9472. doi: 10.1109/21.148426.

R. Bartle. Early mud history. `http://www.mud.co.uk/richard/mudhist.htm`, Nov 1990.

Bethesda Softworks Inc. Navmesh generation - geck. http://geck.bethsoft.com/index.php?title=NavMesh_Generation, November 2008. Last Accessed: 2nd January 2014.

Zeng Bi, Yang Yimin, and Xu Yisan. Mobile robot navigation in unknown dynamic environment based on ant colony algorithm. In *GCIS '09: Proceedings of the 2009 WRI Global Congress on Intelligent Systems*, pages 98–102, Washington, DC, USA, 2009. IEEE Computer Society. ISBN 978-0-7695-3571-5. doi: http://dx.doi.org/10.1109/GCIS.2009.274.

Rahul Biswas, Benson Limketkai, Scott Sanner, and Sebastian Thrun. Towards object mapping in non-stationary environments with mobile robots. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 1, pages 1014–1019. IEEE, 2002.

Blizzard Entertainment. World of warcraft. http://eu.battle.net/wow/en/, May 2013. EU Battle.Net world of warcraft website. Last Accessed: 8th January 2014.

R. Bohlin and L.E. Kavraki. Path planning using lazy prm. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 1, pages 521–528. IEEE, 2000.

V. Boor, M.H. Overmars, and A.F. van der Stappen. The gaussian sampling strategy for probabilistic roadmap planners. *Proc. Robotics and Automation, IEEE International Conference on*, 2:1018 –1023, 1999. doi: 10.1109/ROBOT.1999.772447.

Igor Borovikov. Navigation graph generation. http://www.gamedev.net/page/resources/_/technical/artificial-intelligence/navigation-graph-generation-r2805, June 2011. Last Accessed: January 2014.

Patrice Bouvier, Karim Sehaba, Elise Lavoué, and Sébastien George. Using traces to qualify learner's engagement in game-based learning. In *Advanced Learning Technologies (ICALT), 2013 IEEE 13th International Conference on*, pages 432–436. IEEE, 2013.

R Bradley. *The Prehistory of Britain and Ireland.* Cambridge University Press, 2007.

M.S. Branicky, S.M. LaValle, K. Olson, and Libo Yang. Quasi-randomized path planning. In *Robotics and Automation, ICRA '01. IEEE International Conference on*, volume 2, pages 1481 – 1487 vol.2, 2001. doi: 10.1109/ROBOT.2001.932820.

M.S. Branicky, S.M. LaValle, K. Olson, and L. Yang. Deterministic vs. probabilistic roadmaps. *submitted to the IEEE Transactions on Robotics and Automation*, 2002.

Cyril Brom and Jiří Lukavský. Towards more human-like episodic memory for more human-like agents. In *Proceedings of the 9th International Conference on Intelligent Virtual Agents*, IVA '09, pages 484–485, Berlin, Heidelberg, 2009a. Springer-Verlag. ISBN 978-3-642-04379-6. doi: 10.1007/978-3-642-04380-2_54. URL http://dx.doi.org/10.1007/978-3-642-04380-2_54.

Cyril Brom and Jiří Lukavský. Towards virtual characters with a full episodic memory ii: The episodic memory strikes back. In *Proc. Empathic Agents, AAMAS workshop*, pages 1–9, 2009b.

Cyril Brom, Tomáš Korenko, and Jiří Lukavský. How do place and objects combine? "what-where" memory for human-like agents. In *IVA '09: Proceedings of the 9th International Conference on Intelligent Virtual Agents*, pages 42–48, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 978-3-642-04379-6. doi: http://dx.doi.org/10.1007/978-3-642-04380-2_8.

J. Bruce and M. Veloso. Real-time randomized path planning for robot navigation. In *Intelligent Robots and Systems, 2002. IEEE/RSJ International Conference on*, volume 3, pages 2383 – 2388 vol.3, 2002. doi: 10.1109/IRDS.2002.1041624.

R. Bruntrup, S. Edelkamp, S. Jabbar, and B. Scholz. Incremental map generation with gps traces. In *Intelligent Transportation Systems, IEEE*, sept. 2005. doi: 10.1109/ITSC.2005.1520084.

Mat Buckland. *Programming game AI by example*. Jones & Bartlett Learning, 2005.

Lee Byron. Lee byron ¿¿ else ¿¿ mesh - a processing library. `http://www.leebyron.com/else/mesh/`, January 2008. URL `http://www.leebyron.com/else/mesh/`. Last Accessed: January 2014.

J Canny and B Donald. Simplified voronoi diagrams. *Discrete & Computational Geometry*, 3(3):219–236, 1988. ISSN 0179-5376.

Carnegie Mellon News. Press release: Carnegie mellon creates practical self-driving car using automotive-grade radars and other sensors. `https://www.cmu.edu/news/stories/archives/2013/september/sept4_selfdrivingcar.html`, September 2013. Last Accessed: 3rd January 2013.

K.T. Chen, A. Liao, H.K. Pao, and H.H. Chu. Game bot detection based on avatar trajectory. *Entertainment Computing-ICEC 2008*, pages 94–105, 2009.

L. Chittaro, R. Ranon, and L. Ieronutti. Vu-flow: a visualization tool for analyzing navigation in virtual environments. *Visualization and Computer Graphics, IEEE Transactions on*, 12(6):1475–1485, 2006.

H. Choset and K. Nagatani. Topological simultaneous localization and mapping (slam): toward exact localization without explicit localization. *Robotics and Automation, IEEE Transactions on*, 17(2): 125 –137, apr 2001. ISSN 1042-296X. doi: 10.1109/70.928558. URL `http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=928558`.

C. Cocaud and A. Jnifene. Environment mapping using probabilistic quadtree for the guidance and control of autonomous mobile robots. In *Autonomous and Intelligent Systems (AIS), 2010 International Conference on*, pages 1 –6, june 2010. doi: 10.1109/AIS.2010.5547019.

Marija Dakulovic, Sanja Horvatic, and Ivan Petrovic. Complete coverage d* algorithm for path planning of a floor-cleaning mobile robot. In Sergio Bittanti, Angelo Cenedese, and Sandro Zampieri, editors, *Proceedings of the 18th IFAC World Congress, 2011*, volume 18, pages 5950–5955, 2011. doi: 10.3182/20110828-6-IT-1002.03400. URL `http://www.ifac-papersonline.net/Detailed/49439.html`.

Richard W Dearden, Zeyn A Saigol, Jeremy L Wyatt, and Bramley J Murton. Planning for auvs: Dealing with a continuous partially-observable environment. In *Workshop on Planning and Plan Execution for Real-World Systems, ICAPS 2007*, 2007.

Jory Denny and Nancy M Amato. Toggle prm: Simultaneous mapping of c-free and c-obstacle-a study in 2d. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 2632–2639. IEEE, 2011.

P.N. Dixit and G.M. Youngblood. Understanding playtest data through visual data mining in interactive 3d environments. In *12th International Conference on Computer Games: AI, Animation, Mobile, Interactive Multimedia and Serious Games (CGAMES)*, pages 34–42, 2008.

Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *Evolutionary Computation, IEEE Transactions on*, 1(1):53–66, 1997.

Patrick Doyle, Patrick Doyle, Barbara Hayes-roth, and Barbara Hayes-roth. Guided exploration of virtual worlds. In *Network and Netplay: Virtual Groups on the Internet*, pages 243–264. MIT Press, 1997.

A. Elfes. Using occupancy grids for mobile robot perception and navigation. *Computer*, 22(6):46–57, 1989. ISSN 0018-9162. doi: http://dx.doi.org/10.1109/2.30720.

Entertainment Software Association. The entertainment software association - sales and genre data. `http://www.theesa.com/facts/salesandgenre.asp`, 2013. Last Accessed: 20.12.2013.

Epic Games Inc. Udn - three - navigationmeshreference. `http://udn.epicgames.com/Three/NavigationMeshReference.html`, 2012. Last Accessed: 7th January 2014.

Alireza Fathi and John Krumm. Detecting road intersections from gps traces. In *Proceedings of the 6th international conference on Geographic information science*, GIScience'10, pages 56–69, 2010. ISBN 3-642-15299-6, 978-3-642-15299-3. URL http://dl.acm.org/citation.cfm?id=1887961.1887966.

D. Ferguson, N. Kalra, and A. Stentz. Replanning with rrts. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1243–1248. IEEE, 2006.

D. Filliat. Map-based navigation in mobile robots: I. a review of localization strategies. *Cognitive Systems Research*, 4(4):243–282, December 2003. ISSN 13890417. doi: 10.1016/S1389-0417(03)00008-1. URL http://dx.doi.org/10.1016/S1389-0417(03)00008-1.

B.J. Foley and D. La Torre. Who has why-pox: A case study of informal science education on the net. In *Proceedings of the 6th international conference on Learning sciences*, pages 598–598. International Society of the Learning Sciences, 2004.

Dieter Fox, Jonathan Ko, Kurt Konolige, and Benjamin Stewart. A hierarchical bayesian approach to the revisiting problem in mobile robot map building. In Paolo Dario and Raja Chatila, editors, *Robotics Research*, volume 15 of *Springer Tracts in Advanced Robotics*, pages 60–69. Springer Berlin / Heidelberg, 2005. URL http://dx.doi.org/10.1007/11008941_7.

Doron Friedman, Anthony Steed, and Mel Slater. Spatial social behavior in second life. In Catherine Pelachaud, Jean-Claude Martin, Elisabeth Andr, Grard Chollet, Kostas Karpouzis, and Danielle Pel, editors, *Intelligent Virtual Agents*, volume 4722 of *Lecture Notes in Computer Science*, pages 252–263, Springer Berlin Heidelberg, Berlin, Heidelberg, 2007a.

Doron Friedman, Anthony Steed, and Mel Slater. Research bots in second life. In *Workshop on Simulation and Second Life, Second Life*, 2007b.

Shruti Gandhi. Ibm dives into second life. http://www.ibm.com/developerworks/opensource/library/os-social-secondlife/, January 2010. Last Accessed 31/5/13.

Andrea Gasparri, Gabriele Oliva, and Stefano Panzieri. Path planning using a lazy spatial network prm. In *Control and Automation, 2009. MED'09. 17th Mediterranean Conference on*, pages 940–945. IEEE, 2009.

Roland Geraerts and Mark H. Overmars. A comparative study of probabilistic roadmap planners. In *Workshop on the algorithmic foundations of robotics*, 2002.

Glasgow Caledonian University. Cu there on second life — glasgow caledonian university — scotland, uk. http://www.gcu.ac.uk/theuniversity/learningteaching/cuthereonsecondlife/, August 2013. Last access date: 8th January 2014.

Ross Graham, Hugh McCabe, and Stephen Sheridan. Pathfinding in computer games. *ITB Journal*, 8, 2003.

D. Grammenos, M. Filou, P. Papadakos, and C. Stephanidis. Virtual prints: leaving trails in virtual environments. In *Proceedings of the workshop on Virtual environments 2002*, pages 131–ff. Eurographics Association, 2002.

D. Grammenos, A. Mourouzis, and C. Stephanidis. Virtual prints: Augmenting virtual environments with interactive personal marks. *International journal of human-computer studies*, 64(3):221–239, 2006.

E. Guimarães, A. Maffeis, J. Pereira, B. Russo, E. Cardozo, M. Bergerman, and M.F. Magalhães. Real: A virtual laboratory for mobile robot experiments. *Education, IEEE Transactions on*, 46(1):37–42, 2003.

Yong Guo and A. Iosup. The game trace archive. In *Network and Systems Support for Games (NetGames), 2012 11th Annual Workshop on*, pages 1–6, 2012. doi: 10.1109/NetGames.2012.6404027.

Nick Hawes, Marc Hanheide, Jack Hargreaves, Ben Page, Hendrik Zender, and Patric Jensfelt. Home alone: Autonomous extension and correction of spatial representations. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA '11)*, May 2011.

Dirk Heylen, Anton Nijholt, and Mannes Poel. Embodied agents in virtual environments: The aveiro project. In *European Symp. on Intelligent Technologies, Hybrid Systems and their implementation on Smart Adaptive Systems*, pages 110–111, 2001.

D. Hsu, Tingting Jiang, J. Reif, and Zheng Sun. The bridge test for sampling narrow passages with probabilistic roadmap planners. In *Robotics and Automation, 2003. Proceedings. ICRA '03. IEEE International Conference on*, volume 3, pages 4420 – 4426 vol.3, sept. 2003. doi: 10.1109/ROBOT.2003.1242285.

D. Hsu, G. Sanchez-Ante, and Zheng Sun. Hybrid prm sampling with a cost-sensitive adaptive strategy. In *Robotics and Automation, ICRA*, pages 3874 – 3880, 2005. doi: 10.1109/ROBOT.2005.1570712.

Han-Pang Huang and Shu-Yun Chung. Dynamic visibility graph for path planning. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 3, pages 2813–2818 vol.3, 2004. doi: 10.1109/IROS.2004.1389835.

irobot.com. irobot roomba vacuum cleaning robot. `http://www.irobot.com/For-the-Home/Vacuum-Cleaning/Roomba`, 2014. Last Accessed November 2014.

L. Jaillet and T. Simeon. A prm-based motion planner for dynamically changing environments. In *Intelligent Robots and Systems, 2004. (IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1606 – 1611 vol.2, sept.-2 oct. 2004. doi: 10.1109/IROS.2004.1389625.

Ray Jarvis. Terrain-aware path guided mobile robot teleoperation in virtual and real space. In *Advances in Computer-Human Interactions, 2010. ACHI'10. Third International Conference on*, pages 56–65. IEEE, 2010. ISBN 978-1-4244-5693-2. doi: 10.1109/ACHI.2010.23.

Jehn-Ruey Jiang, Ching-Chuan Huang, and Chung-Hsien Tsai. Avatar path clustering in networked virtual environments. In *ICPADS*, pages 845–850. IEEE, 2010.

Simon Joslin, Ross Brown, and Penny Drennan. Modelling quest data for game designers. In *Proceedings of the 2006 international conference on Game research and development*, pages 184–190. Murdoch University, 2006.

Gal A. Kaminka, Manuela M. Veloso, Steve Schaffer, Chris Sollitto, Rogelio Adobbati, Andrew N. Marshall, Andrew Scholer, and Sheila Tejada. Gamebots: a flexible test bed for multiagent team research. *Commun. ACM*, 45(1):43–45, 2002. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/502269.502293.

L.E. Kavraki, P. Svestka, J.-C. Latombe, and M.H. Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation*, 12(4): 566–580, 1996. ISSN 1042296X. doi: 10.1109/70.508439.

N.S. Ketkar and G.M. Youngblood. A largest common subsequence-based distance measure for classifying player motion traces in virtual worlds. In *Twenty-Third International FLAIRS Conference*, 2010.

Michael Kneebone and Richard Dearden. Uncertain probabilistic roadmaps with observations. *Proceedings of the 19th International Conference on Automated Planning and Scheduling (ICAPS09)*, 2009.

K. Konolige, E. Marder-Eppstein, and B. Marthi. Navigation in hybrid metric-topological maps. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 3041 –3047, may 2011. doi: 10.1109/ICRA.2011.5980074.

Gerhard K. Kraetzschmar, Guillem Pags Gassull, Klaus Uhl, Guillem Pags, and Gassull Klaus Uhl. Probabilistic quadtrees for variable-resolution mapping of large environments. In *Eds.), Proceedings of the 5th IFAC/EURON*, 2004.

L. Krammer, W. Granzer, and W. Kastner. A new approach for robot motion planning using rapidly-exploring randomized trees. In *Industrial Informatics (INDIN), 2011 9th IEEE International Conference on*, pages 263 –268, july 2011. doi: 10.1109/INDIN.2011.6034885.

J.J. Kuffner Jr and S.M. LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Robotics and Automation, 2000. Proceedings. ICRA'00. IEEE International Conference on*, volume 2, pages 995–1001. IEEE, 2000.

B. Kuipers. The "map in the head" metaphor. *Environment and Behavior*, 14(2):202–220, 1982.

Benjamin Kuipers, Joseph Modayil, Patrick Beeson, Matt Macmahon, and Francesco Savelli. Local metrical and global topological maps in the hybrid spatial semantic hierarchy. In *in IEEE Int. Conf. on Robotics and Automation (ICRA-04)*, pages 4845–4851, 2004.

Benjamin J. Kuipers and Tod S. Levitt. Navigation and mapping in large-scale space. *AI MAGAZINE*, 9:25–43, 1988.

S. Kumar and S. Chakravorty. Adaptive sampling for generalized probabilistic roadmaps. *Journal of Control Theory and Applications*, 10(1):1–10, 2012a.

S. Kumar and S. Chakravorty. Multi-agent generalized probabilistic roadmaps: Magprm. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 3747–3753, 2012b. doi: 10.1109/IROS.2012.6385678.

H. Kurniawati and D. Hsu. Workspace importance sampling for probabilistic roadmap planning. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 2, pages 1618–1623. IEEE, 2004.

J.E. Laird. Research in human-level ai using computer games. *Communications of the ACM*, 45(1):32–35, 2002.

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

Steven M. LaValle. Rapidly-exploring random trees: A new tool for path planning. Technical report, Iowa State University, 1998.

Yangming Li, Shuai Li, and Yunjian Ge. A biologically inspired solution to simultaneous localization and consistent mapping in dynamic environments. *Neurocomputing*, 104(0):170 – 179, 2013. ISSN 0925-2312. doi: 10.1016/j.neucom.2012.10.011. URL `http://www.sciencedirect.com/science/article/pii/S0925231212008466`.

S. Lindemann and S. LaValle. Steps toward derandomizing rrts. *Robot Motion and Control*, pages 287–300, 2006.

Linden Research Inc. Second life official site. `http://secondlife.com`, 2012. Last Accessed 8th January 2014.

Linden Research Inc. Infographic: 10 years of second life. `http://www.lindenlab.com/releases/infographic-10-years-of-second-life`, 2013. Last Accessed August 2014.

John Lloyd. Quickhull3d: A robust 3d convex hull algorithm in java. `http://www.cs.ubc.ca/~lloyd/java/quickhull3d.html`, 2014. URL `http://www.cs.ubc.ca/~lloyd/java/quickhull3d.html`. Last Accessed: August 2014.

J.M. Loomis, J.J. Blascovich, and A.C. Beall. Immersive virtual environment technology as a basic research tool in psychology. *Behavior Research Methods*, 31(4):557–564, 1999.

M. Malfaz and M. Salichs. Using muds as an experimental platform for testing a decision making system for self-motivated autonomous agents. *Artificial Intelligence and Simulation of Behaviour Journal*, 2 (1):21–44, 2010.

B. Marthi. Navigation in partially observed dynamic roadmaps. In *POMDP Practioners Workshop, International Conference on Automated Planning and Scheduling*, 2010.

M. Maurette. Mars rover autonomous navigation. *Auton. Robots*, 14(2-3):199–208, 2003. ISSN 0929-5593. doi: http://dx.doi.org/10.1023/A:1022283719900.

T. McMahon, S. Jacobs, B. Boyd, L. Tapia, and N.M. Amato. Local randomization in neighbor selection improves prm roadmap quality. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 4441–4448, October 2012.

H. Mei, Y. Tian, and L. Zu. A hybrid ant colony optimization algorithm for path planning of robot in dynamic environment. *International Journal of Information Technology*, 12(3):78–88, 2006.

M. Mekni and B. Moulin. Hierarchical path planning for multi-agent systems situated in informed virtual geographic environments. In *Information, Process, and Knowledge Management, 2010. eKNOW '10. Second International Conference on*, pages 48 –55, feb. 2010. doi: 10.1109/eKNOW.2010.11.

Metaplace Inc. Metaplace. `[Unavailable]www.metaplace.com`, 2010. Website shut down 1st January 2010.

J.L. Miller and J. Crowcroft. Avatar movement in world of warcraft battlegrounds. In *Proceedings of the 8th annual workshop on Network and systems support for games*, page 1. IEEE Press, 2009.

Patrycja E Missiuro and Nicholas Roy. Adapting probabilistic roadmaps to handle uncertain maps. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 1261–1267. Ieee, May 2006. doi: 10.1109/ROBOT.2006.1641882. URL `http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1641882`.

K. Murphy et al. Bayesian map learning in dynamic environments. *Advances in Neural Information Processing Systems (NIPS)*, 12:1015–1021, 1999.

Carl Myhill. Commercial success by looking for desire lines. In *Computer Human Interaction*, pages 293–304. Springer, 2004.

Christian L Nielsen and Lydia E Kavraki. A two level fuzzy prm for manipulation planning. In *Intelligent Robots and Systems, 2000.(IROS 2000). Proceedings. 2000 IEEE/RSJ International Conference on*, volume 3, pages 1716–1721. IEEE, 2000.

Dennis Nieuwenhuisen, Arno Kamphuis, Marlies Mooijekind, and Mark H Overmars. Automatic construction of roadmaps for path planning in games. In *International Conference on Computer Games: Artificial Intelligence, Design and Education*, pages 285–292, 2004.

Notch Development AB. Minecraft. `https://minecraft.net/`, 2012. Last Accessed: 8th January 2014.

Ramon Oliva and Nuria Pelechano. Automatic generation of suboptimal navmeshes. In *Motion in Games*, pages 328–339. Springer, 2011.

Open Metaverse Foundation. Libopenmetaverse. `http://www.openmetaverse.org/projects/libopenmetaverse`, 2012. Last Accessed: 8th January 2014.

OpenStreetMap. Openstreetmap: The free wiki world map. http://www.openstreetmap.org/, May 2013. Last Accessed: 8th January 2014.

A.G. Ozkil, Z. Fan, J. Xiao, S. Dawids, J.K. Kristensen, and K.H. Christensen. Mapping of multi-floor buildings: A barometric approach. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, pages 847–852. IEEE, 2011.

B. Park, J. Choi, and W.K. Chung. Path reconstruction method for sampling based planners. In *Advanced Robotics and its Social Impacts (ARSO), 2010 IEEE Workshop on*, pages 81–86. IEEE, 2010.

B. Park, J. Choi, and W.K. Chung. An efficient mobile robot path planning using hierarchical roadmap representation in indoor environment. In *Robotics and Automation (ICRA), 2012 IEEE International Conference on*, pages 180–186. IEEE, 2012.

Byungjae Park and Wan Kyun Chung. Adaptive node sampling method for probabilistic roadmap planners. In *Intelligent Robots and Systems*, pages 4399–4405, 2009.

Daniel Pittman and Chris GauthierDickey. A measurement study of virtual populations in massively multiplayer online games. In *Proceedings of the 6th ACM SIGCOMM workshop on Network and system support for games*, pages 25–30. ACM, 2007.

Alberto Poncela, Eduardo J Perez, Antonio Bandera, Cristina Urdiales, and Francisco Sandoval. Efficient integration of metric and topological maps for directed exploration of unknown environments. *Robotics and Autonomous Systems*, 41(1):21–39, 2002.

Matti Pouke. Using gps data to control an agent in a realistic 3d environment. In *Next Generation Mobile Apps, Services and Technologies (NGMAST), 2013 Seventh International Conference on*, pages 87–92. IEEE, 2013.

A. Pronobis, K. Sjoo, A. Aydemir, A.N. Bishop, and P. Jensfelt. A framework for robust cognitive spatial mapping. In *Advanced Robotics, 2009. ICAR 2009. International Conference on*, pages 1 –8. IEEE, June 2009.

Ismo Rakkolainen, Simo Pulkkinen, and Arttu Heinonen. Visualizing real-time gps data with internet's vrml worlds. In *Proceedings of the 6th ACM international symposium on Advances in geographic information systems*, pages 52–56. ACM, 1998.

Rameshsharma Ramloll and Darren Mowat. Wayfinding in virtual environments using an interactive spatial cognitive map. In *IV2001 Proceedings (London), IEEE*, pages 2–0. Press, 2001.

S. Rodriguez, X. Tang, J.M. Lien, and N.M. Amato. An obstacle-based rapidly-exploring random tree. In *Robotics and Automation, 2006. ICRA 2006. Proceedings 2006 IEEE International Conference on*, pages 895–900. IEEE, 2006.

R A Ruddle. How do people find information on a familiar website. In *Proceedings of the 23rd BCS Conference on Human- Computer Interaction (HCI'09)*, pages 262–268, 2009.

Roy Ruddle. The effect of trails on first-time and subsequent navigation in a virtual environment. In *VR '05: Proceedings of the 2005 IEEE Conference 2005 on Virtual Reality*, pages 115–122, 321. IEEE Computer Society, 2005. ISBN 0-7803-8929-8. doi: http://dx.doi.org/10.1109/VR.2005.76.

Roy A. Ruddle. Generating trails automatically, to aid navigation when you revisit an environment. *Presence: Teleoper. Virtual Environ.*, 17(6):562–574, 2008. ISSN 1054-7460. doi: http://dx.doi.org/10.1162/pres.17.6.562.

Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003. ISBN 0137903952. URL http://portal.acm.org/citation.cfm?id=773294.

Pedram Sadeghian, Mehmed Kantardzic, Oleksandr Lozitskiy, and Walaa Sheta. The frequent wayfinding-sequence (fws) methodology: Finding preferred routes in complex virtual environments. *International Journal of Human-Computer Studies*, 64(4):356 – 374, 2006. ISSN 1071-5819. doi: DOI:10.1016/j.ijhcs.2005.08.014. URL `http://www.sciencedirect.com/science/article/B6WGR-4H98T9Y-1/2/5191a84d66b6aba1ab2344192624945e`.

Yoichi Morales Saiki, Alexander Carballo, Eijiro Takeuchi, Atsushi Aburadani, Takashi Tsubouchi, and Takashi Tsubouchi. Autonomous robot navigation in outdoor cluttered pedestrian walkways. *J. Field Robotics*, pages 609–635, 2009.

K. Samperi, N. Hawes, and Russell Beale. Improving map generation in large-scale environments for intelligent virtual agents. In *The AAMAS-2013 Workshop on Cognitive Agents for Virtual Environments*, Lecture Notes in Computer Science. Springer, May 2013a.

Katrina Samperi. Large-scale mapping and navigation in virtual worlds: Thesis summary. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

Katrina Samperi and Nick Hawes. Improving the generation of rapidly exploring randomised trees (rrts) in large scale virtual environments using trails. In *TAROS 2014, The 15th Conference Towards Autonomous Robotic Systems*, Birmingham, UK, 2014.

Katrina Samperi, Russell Beale, and Nick Hawes. Please keep off the grass: individual norms in virtual worlds. In *Proceedings of the 26th Annual BCS Interaction Specialist Group Conference on People and Computers, BCS-HCI 2012*, pages 375–380. British Computer Society, September 2012.

Katrina Samperi, Nick Hawes, and Russell Beale. Towards mapping and segmentation of very large scale spaces with intelligent virtual agents. In *Intelligent Virtual Agents*, page 436. Springer, 2013b.

Gunther K Schmidt and Kianoush Azarm. Mobile robot navigation in a dynamic world using an unstead diffusion equation strategy. *Proceedings of the lEEE/RSJ International Conference on Intelligent Robots and Systems*, 1:642–647, 1992. doi: 10.1109/IROS.1992.587412.

Ralph Schroeder. *The Social Life of Avatars: Presence and Interaction in Shared Virtual Environments*. Springer, 2002. URL `http://books.google.com/books?hl=en&lr=&id=d-7vegnIgJsC&pgis=1`. ISBN 1852334614.

Chaoxia Shi, Yanqing Wang, and Jingyu Yang. Online topological map building and qualitative localization in large-scale environment. *ROBOTICS AND AUTONOMOUS SYSTEMS*, 58(5):488–496, MAY 31 2010. ISSN 0921-8890. doi: 10.1016/j.robot.2010.01.009.

S. Shojaeipour, SM Haris, K. Khalili, and A. Shojaeipour. Motion planning for mobile robot navigation using combine quad-tree decomposition and voronoi diagrams. In *Computer and Automation Engineering (ICCAE), 2010 The 2nd International Conference on*, volume 1, pages 90–93. IEEE, 2010.

Gabe Sibley, Christopher Mei, Ian Reid, and Paul Newman. Vast-scale outdoor navigation using adaptive relative bundle adjustment. *The International Journal of Robotics Research*, 29(8):958–980, JUL 2010. ISSN 0278-3649. doi: 10.1177/0278364910369268. 5th Robotics Science and Systems Conference, Seattle, WA, JUN, 2009.

T. Simeon, J.P. Laumond, and C. Nissoux. Visibility-based probabilistic roadmaps for motion planning. In *Advanced Robotics Vol. 14 No. 6*, number 6 in 14, pages 477–493, 2000.

S. Simhon and G. Dudek. A global topological map formed by local metric maps. In *Intelligent Robots and Systems, 1998. Proceedings., 1998 IEEE/RSJ International Conference on*, volume 3, pages 1708 – 1714 vol.3, oct 1998. doi: 10.1109/IROS.1998.724844. URL `http://ieeexplore.ieee.org.ezproxye.bham.ac.uk/stamp/stamp.jsp?tp=&arnumber=724844`.

Reid Simmons and Sven Koenig. Probabilistic robot navigation in partially observable environments. In *IJCAI'95: Proceedings of the 14th international joint conference on Artificial intelligence*, pages 1080–1087, San Francisco, CA, USA, 1995. Morgan Kaufmann Publishers Inc. ISBN 1-55860-363-8.

Edgar-Philipp Stoffel, Korbinian Schoder, and Hans Jürgen Ohlbach. Applying hierarchical graphs to pedestrian indoor navigation. In *GIS '08: Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, pages 1–4, New York, NY, USA, 2008. ACM. ISBN 978-1-60558-323-5. doi: http://doi.acm.org/10.1145/1463434.1463499.

Z. Sun, D. Hsu, T. Jiang, H. Kurniawati, and J.H. Reif. Narrow passage sampling for probabilistic roadmap planning. *Robotics, IEEE Transactions on*, 21(6):1105–1115, 2005.

Adnan Tahirovic and Gianantonio Magnani. A roughness-based rrt for mobile robot navigation planning. In *Proceedings of the 18th IFAC World Congress*, pages 5944–5949, 2011.

Adriana Tapus and Shrihari Vasudevan. Towards a multilevel cognitive probabilistic representation of space. In *In Proc. of the International Conference on Human Vision and Electronic Imaging X, part of the IST-SPIE Symposium on Electronic Imaging*, 2005.

The Open University. Virtual worlds : The ou in second life : Recent history of the ou in second life. `http://www.open.ac.uk/virtualworlds/content/ou-second-life`, May 2013. Last Accessed: 8th January 2014.

Shawna L. Thomas, Marco Morales, Xinyu Tang, and Nancy M. Amato. Biasing samplers to improve motion planning performance. In *ICRA*, pages 1625–1630. IEEE, 2007.

S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002.

S. Thrun and A. Bücken. Integrating grid-based and topological maps for mobile robot navigation. In *Proceedings of the AAAI Thirteenth National Conference on Artificial Intelligence*, Portland, Oregon, 1996.

Sebastian Thrun. Learning metric-topological maps for indoor mobile robot navigation. *Artif. Intell.*, 99 (1):21–71, 1998a. ISSN 0004-3702. doi: http://dx.doi.org/10.1016/S0004-3702(97)00078-7.

Sebastian Thrun. Finding landmarks for mobile robot navigation. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pages 958–963, 1998b.

Sebastian Thrun. Winning the darpa grand challenge: A robot race through the mojave desert. In *ASE 2006: 21st IEEE International Conference on Automated Software Engineering, Proceedings*, IEEE International Conference on Automated Software Engineering, page 11, 10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1264 USA, 2006. IEEE COMPUTER SOC. ISBN 0-7695-2579-2. 21st IEEE International Conference on Automated Software Engineering, Tokyo, JAPAN, SEP 22, 2006.

Sebastian Thrun. Toward robotic cars. *Communications of the ACM*, 53(4):99–106, 2010.

Sebastian Thrun, Dieter Fox, and Wolfram Burgard. Probabilistic mapping of an environment by a mobile robot. In *In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA*, pages 1546–1551, 1998.

EC Tolman. Cognitive maps in rats and men. *Psychological Review*, 55(4):189–208, 1948. ISSN 0033-295X.

Paul Tozour. Fixing pathfinding once and for all. `http://www.ai-blog.net/archives/000152.html`, July 2008. Last Accessed: 2nd January 2014.

Valve Corporation. Counterstrike: Global offensive. http://store.steampowered.com/css, May 2013. Last Accessed: 8th January 2014.

J. van den Berg, S. Patil, J. Sewall, D. Manocha, and M. Lin. Interactive navigation of multiple agents in crowded environments. In *Proceedings of the 2008 symposium on Interactive 3D graphics and games*, pages 139–147. ACM, 2008.

Pascal vd Heiden. Doom builder map editor. http://www.doombuilder.com/, 2008. Last Accessed: 10th January 2014.

Wen-fei Wang, Rong Xiong, and Jian Chu. Mapbuilding for dynamic environments using grid vectors. *Journal of Zhejiang University - Science C*, 12:574–588, 2011. ISSN 1869-1951. URL http://dx.doi.org/10.1631/jzus.C1000255http://www.springerlink.com.ezproxyd.bham.ac.uk/content/n059226q7w5p2218/fulltext.pdf. 10.1631/jzus.C1000255.

A. Waqar. Mobile robot navigation using monte carlo localisation. *Ubiquitous Computing and Communication Journal*, 6(3), June 2011.

Nicholas Mario Wardhana, Henry Johan, and Hock Soon Seah. Automatic generation of enhanced waypoint graph for volumetric path planning. In *Proceedings of the 2012 International Workshop on Advanced Image Technology (IWAIT 2012)*, pages 450–455, 2012.

Ying Wenjian, Sun Fuchun, Liu Huaping, and Song Yu. A collaborative navigation system for autonomous vehicle in flat terrain. In *Control and Automation, 2009. ICCA 2009. IEEE International Conference on*, pages 1615 –1620, dec. 2009. doi: 10.1109/ICCA.2009.5410433.

A. Wexelblat and P. Maes. Footprints: history-rich tools for information foraging. In *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, pages 270–277. ACM, 1999.

D.F. Wolf and G.S. Sukhatme. Mobile robot simultaneous localization and mapping in dynamic environments. *Autonomous Robots*, 19(1):53–65, 2005.

KM Wurm, D. Hennes, D. Holz, RB Rusu, C. Stachniss, K. Konolige, and W. Burgard. Hierarchies of octrees for efficient 3d mapping. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. IEEE, 2011.

S.W. Yang and C.C. Wang. Feasibility grids for localization and mapping in crowded urban scenes. In *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pages 2322–2328. IEEE, 2011.

Yuandong Yang and Oliver Brock. Adapting the sampling distribution in prm planners based on an approximated medial axis. In *Robotics and Automation, 2004. Proceedings. ICRA '04. IEEE International Conference on*, pages 4405–4410. IEEE, 2004.

Nick Yee, Jeremy N. Bailenson, Mark Urbanek, Francis Chang, and Dan Merget. The unbearable likeness of being digital: The persistence of nonverbal social norms in online virtual environments. *Cyberpsychology & behavior : the impact of the Internet, multimedia and virtual reality on behavior and society*, 10 (1):115–121, 2007. doi: 10.1089/cpb.2006.9984.

Hsin-Yi Yeh, S. Thomas, D. Eppstein, and N.M. Amato. Uobprm: A uniformly distributed obstacle-based prm. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 2655–2662, 2012. doi: 10.1109/IROS.2012.6385875.

Fang Yuan, Lukas Twardon, and Marc Hanheide. Dynamic path planning adopting human navigation strategies for a domestic mobile robot. In *Intelligent Robots and Systems*, pages 3275–3281. IEEE, 2010. ISBN 978-1-4244-6674-0.

H. Zender, O. Martínez Mozos, P. Jensfelt, G. J. M. Kruijff, and W. Burgard. Conceptual spatial representations for indoor mobile robots. *Robot. Auton. Syst.*, 56(6):493–502, 2008. ISSN 0921-8890. doi: http://dx.doi.org/10.1016/j.robot.2008.03.007.

Qi Zhang, Jiachen Ma, and Qiang Liu. Path planning based quadtree representation for mobile robot using hybrid-simulated annealing and ant colony optimization algorithm. In *Intelligent Control and Automation (WCICA), 2012 10th World Congress on*, pages 2537–2542, 2012. doi: 10.1109/WCICA.2012.6358300.