Generating Safe Trajectories in Stochastic Dynamic Enviroments

University of St Andrews

CS4099: Major Software Project

Author:
Alexander Wallar

Supervisor:
Dr. Michael Weir

March 24, 2015

Abstract

Contents

List of Figures

Chapter 1

Introduction

Chapter 2

Context Survey

Chapter 3

Ethics

Chapter 4

Software Development Framework

Chapter 5

Objectives

5.1 Primary

The main objective of this work is to develop an algorithm that generates a quantitatively safe trajectory for a robot through an uncertain dynamic environment by utilizing information about how dynamic obstacles are going to move in the future. This can be simply stated as determining the curvature that minimizes the line integral over the dynamic cost distribution for a given set of dynamic obstacles.

$$J(C, A) = \int\limits_{C} \exp\Big(P(x, y, t_0, t_m, A) + 1\Big)\, \mathrm{d}s \tag{5.1}$$

Eq. 5.1 describes the objective function, J, that needs to be minimized with respect to the curvature in order to determine the safest path through the environment. In Eq. 5.1, the function P is the cost surface for a given time interval and set of obstacles. More description about P is given in Sec. 7.1.2 and is formally defined in Eq. 7.2. More precisely, the objective of this work to develop an algorithm that will provide an approximate solution to Eq. 5.2 which will return the minimum cost path through a environment for a given set of obstacles. The solution is described in Sec. 7.2

$$\Gamma(A) = \arg\min_{C}\ J(C, A) \tag{5.2}$$

5.2 Secondary

The main secondary objective for this work is to show that the proposed solution can provide safer paths than standard planners such as potential fields by leveraging information about how the obstacles move through the environment. Quantitative and qualitative experiments have been conducted that provide evidence that the proposed solution does indeed produce safer paths based on the safety metrics that have been devised for this work. These results are shown in Ch. 10.

Chapter 6

Planner Methodology

As with any research project, many different attempts were made to come up with a solution to the objectives state in Ch. 5. Three different techniques were developed in sequence to try and provide a solution that best matched the sought behaviour for the planner. The first attempt was a simple potential field that would take into account the predicted trajectories of the obstacles, and leverage this information to provide safer paths. The second attempt included generating a probabilistic roadmap in relative space-time and using stock graph search algorithms such as Dijkstra's algorithm and Edmond's algorithm to derive low costs paths through the environment. The last attempt, and the most successful is a planner that uses a two dimensional probabilistic roadmap to sample the search space and then uses Best First Search to expand nodes in space-time to determine the minimum cost path through the dynamic environment. These three attempts are described individually and more detail in this section.

6.1 Potential Fields

6.2 Space-time Roadmap

6.3 Probabilistic Roadmap With Best First Search

Chapter 7

Design

7.1 Dynamic Obstacles

As a main component of this work, dynamic obstacles needed to be designed such that one could quantify their trajectories, initial configurations, and their level of uncertainty. This section introduces the definition of a dynamic obstacle used throughout this work along with how it is represented to the planner and its simulated & predicted equations of motion.

7.1.1 Definition

A dynamic obstacle is defined as a 5-tuple, $a = (I, \dot{\zeta}, \epsilon, \xi, T)$ where I is the initial configuration of the obstacle, $\dot{\zeta}$ is a function, $\dot{\zeta} : \mathbb{R}^+ \to \mathbb{R}^2$, representing the velocity of the obstacle, ϵ is used to define a random variable $\rho \sim \mathcal{U}(-\epsilon, \epsilon)$ that is injects noise into an obstacle's trajectory shown in Eq. 7.4 where $\mathcal{U}$ is a uniform distribution, ξ is the current configuration used for prediction, and T is the time that the obstacle was in configuration ξ. The variables ξ and T are dynamic variables and are updated throughout the execution of the algorithm and are used to determine when it is appropriate for the algorithm replan and find a new path through the environment using more up to date information. This is explained in Sec. 7.2. The variables ξ and T are initially set to I and 0 respectively.

7.1.2 Cost Function

Unlike in the previous work, dynamic obstacles are represented by cost distributions that resemble probability density functions. The difference being is that these cost distributions do not have a unit integral. These cost distributions are used to describe where the obstacle is going to be in within a time interval and can be generated by a third party system, such as a motion capture system. There is an assumption that for a given interval, $\mathcal{T} = [t_0, t_m]$, the highest cost with the smallest uncertainty will be at $t = t_0$ and the lowest cost with the highest uncertainty will be at $t = t_m$. Under this assumption, the cost function models how the obstacle may diverge from its current trajectory as time increases. With these assumptions, the cost function, $P_a : \mathbb{R}^2 \times (\mathbb{R}^+)^2 \to \mathbb{R}$, represents represents the cost surface for a given obstacle within a given time interval. Eq. 7.1 formally defines the cost function for a single obstacle.

$$P_a(x, y, t_0, t_m) = \int_{t_0}^{t_m} \mathcal{N}(\zeta_a(t), \alpha \cdot (t - t_0)^2 + \beta, x, y) \cdot (t_m - t)^\gamma \, \mathrm{d}t \tag{7.1}$$

In Eq. 7.1, $\mathcal{N}(\mu, \sigma^2, x, y)$ is the evaluation of a 3D normal distribution centered at (μ_x, μ_y) with a variance of σ^2 at (x, y). This equation models how the uncertainty of obstacle trajectory prediction increases over time by increasing the standard deviation of the Gaussian distribution as the time increases.

Likewise, this function multiplies the Gaussian distribution by a factor of $(t_m - t)^\gamma$ where $\gamma \geq 1$ which gives higher costs to times closer to t_0.

A cost function is also needed that can incorporate the cost distributions for multiple dynamic obstacles within the environment. The cost function used in this work, $P : \mathbb{R}^2 \times (\mathbb{R}^+)^2 \times \mathcal{A} \to \mathbb{R}$ where $\mathcal{A}$ is the set of all possible sets of dynamic obstacles, calculates the average cost at a point (x, y) within a given time interval for a given set of agents. This is shown formally in Eq. 7.2.

$$P(x, y, t_0, t_m, A) = \frac{\sum\limits_{a \in A} P_a(x, y, t_0, t_m)}{|A|} \tag{7.2}$$

An example of how P changes over time is shown in Fig. 7.1. In that example, two dynamic obstacles are placed in the scene and given sinusoidal velocities. In this example the time interval, δt, is kept constant throughout the simulation, i.e. $t_m = t_0 + \delta t$, for all $t_0 \in [0, T - \delta t]$ where T is the length of the simulation. Since the velocity does not remain constant in the example, the cost distribution elongates an shrinks based on the acceleration of the obstacle. For instance, in the first and last images in Fig. 7.1, the cost is contained to a small area due to the velocity equations of the obstacles being at their minimum and in the fourth image,the cost is more spread out through the environment because the velocity is at its maximum.

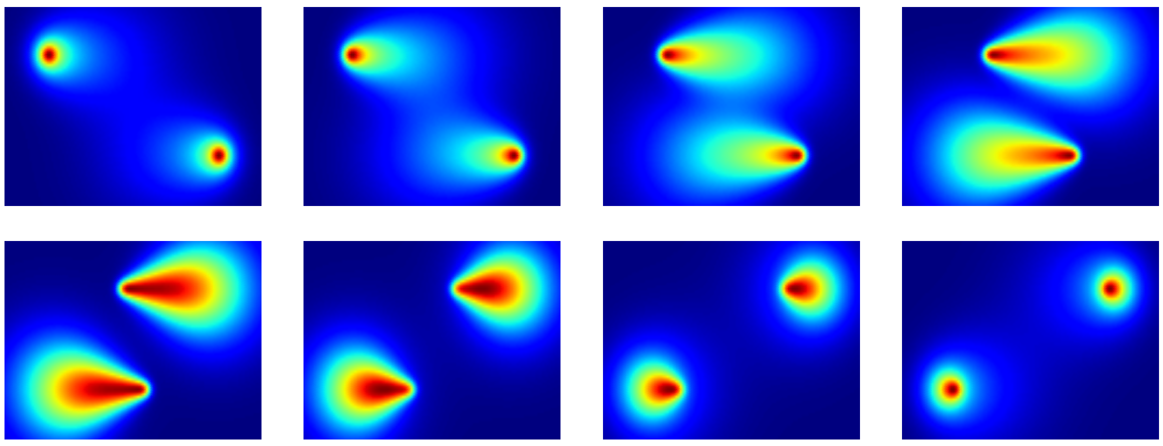

Figure 7.1: Cost distributions indicating the likelihood that an agent will be at a certain location within a given time interval. These figures show how this distribution changes over time (left to right, top to bottom)

7.1.3 Equations of Motion

The motion of a dynamic obstacle is defined by the velocity equation, the initial configuration, the amount of uncertainty. Defining the obstacle's trajectory in terms of its velocity makes it easier to model when creating scenes. The equation of motion for the dynamic obstacle is shown in Eq. 7.3.

$$\zeta_a(t) = \begin{cases} \xi_a + \int_{T_a}^{t} \dot{\zeta}_a(\lambda)\, \mathrm{d}\lambda & \text{if } t \geq T_a \\ \tilde{\zeta}_a(t) & \text{if } t < T_a \end{cases} \tag{7.3}$$

In Eq. 7.3, $\tilde{\zeta}_a$ represents the observed trajectory of the obstacle whereas ζ_a corresponds to the predicted trajectory of the obstacle. This disambiguation is needed because the planner needs to be able to extrapolate the future movements of a dynamic obstacle. The variables, ξ_a and T_a are dynamically updated when the planner replans and are initially set to I_a and 0 respectively. The need for this and how it is designed will be discussed in Sec. 7.2

For the experiments, the motion of the obstacles are simulated by adding a random variable, $\rho \sim \mathcal{U}(-\epsilon, \epsilon)$ to the trajectory during the integration of the velocity equation. This form of stochasticity allows the obstacle to diverge from its specified trajectory whilst maintaining the same velocity equation. This means that the obstacle will not exhibit random motion around its specified path, but rather is able to diverge completely. Also, by adding the random variable to the velocity equation during integration, the obstacle will not "jump" to a new location, but will gradually diverge. The definition of $\tilde{\zeta_a}$ is shown in Eq. 7.4. For this equation, it is assumed that the function only computes a value for any given time value, t, only once.

$$\tilde{\zeta_a}(t) = I_a + \int_0^t \dot{\zeta}_a(\lambda) + \rho \, \mathrm{d}\lambda \tag{7.4}$$

7.2 Planning Algorithm

$$C(i, j, A) = \int_0^1 \exp\Big(P(x(\lambda), y(\lambda), i_t, j_t, A) + 1\Big) \cdot ||i - j||_2 \, \mathrm{d}\lambda \tag{7.5}$$

Where $x(\lambda) = (j_x - i_x) \cdot \lambda + i_x$ and $y(\lambda) = (j_y - i_y) \cdot \lambda + i_y$ are the parametric equations of the line from i to j.

Algorithm 1 ROADMAP(n, d, w, h, O)

Input:
n: Maximum number of samples
d: Maximum distance between neighbouring nodes
O: Set of obstacles
Output:
An unweighted graph of points describing the connectivity of the environment

```
1: for $i = 1$ to $n$ do
2:     $q \leftarrow$ RANDOMPOINT2D$(w, h)$
3:     if $\bigwedge_{o \in O} \neg$COLLISION$(o, q)$ then
4:         $V \leftarrow V \cup \{q\}$
5: for all $q_i \in V$ do
6:     for all $q_j \in V$ do
7:         if $q_i \neq q_j \wedge ||q_i - q_j|| \leq d$ then
8:             $E \leftarrow E \cup \{(q_i, q_j)\}$
9: return $(V, E)$
```

Algorithm 2 GetPath($n, d, w, h, \delta, p, g, O, A, R$)

Input:
n: Maximum number of samples for the roadmap
d: Maximum distance between neighbouring nodes in the roadmap
w: Width of the scene
h: Height of the scene
Output:

```
1: $(V, E) \leftarrow \textsc{Roadmap}(n, d, w, h, O)$
2: $\Pi \leftarrow \emptyset$
3: $q \leftarrow p$
4: while $||\textsc{Back}(\Pi) - g||_2 > R$ do
5:    $\pi \leftarrow \textsc{SearchGraph}(V, E, R, A, q, g)$
6:    for all $i \in \pi$ do
7:       $\Pi \leftarrow \Pi \cup \{i\}$
8:       for all $a \in A$ do
9:          $\textsc{Step}(a)$
10:      if $\bigvee_{a \in A} ||\tilde{\zeta_a}(i_t) - \zeta_a(i_t)|| > \delta$ then
11:         for all $a \in A$ do
12:            $\textsc{Update}(\zeta_a, \tilde{\zeta_a})$
13:         $q \leftarrow i$
14:         break
15: return $\Pi$
```

Algorithm 3 SearchGraph(V, E, R, A, p, g)

```
1: $Q \leftarrow \textsc{PriorityQueue}()$
2: $D \leftarrow \textsc{Dictionary}()$
3: $\Pi \leftarrow \textsc{Dictionary}()$
4: $\textsc{Insert}(Q, p, 0)$
5: while $\neg\textsc{Empty}(Q)$ do
6:    $q, w \leftarrow \textsc{Pop}(Q)$
7:    if $||q - g||_2 \leq R$ then
8:       return $\textsc{BacktrackPath}(p, g, \Pi)$
9:    $N \leftarrow \textsc{GetTemporalNeighbours}(V, E, q)$
10:   for all $n \in N$ do
11:      $\Pi_n \leftarrow q$
12:      $c \leftarrow \psi \cdot C_A(q, n) + \omega \cdot D_n$
13:      $D_n \leftarrow D_n + 1$
14:      $Q \leftarrow \textsc{Insert}(Q, n, c)$
```

Algorithm 4 GetTemporalNeighbours(V, E, q)

```
1: $S \leftarrow \emptyset$
2: $N \leftarrow \textsc{Neighbours}(V, E, q)$
3: for all $n \in N$ do
4:    $t \leftarrow ||q - n||_2/s + q_t$
5:    $S \leftarrow S \cup \{(n_x, n_y, t)\}$
```

Algorithm 5 BacktrackPath(p, g, Π)

```
1: $q \leftarrow g$
2: $S \leftarrow \textsc{Stack}()$
3: while $\Pi_q \neq p$ do
4:    $S \leftarrow \textsc{Push}(S, q)$
5:    $q \leftarrow \Pi_q$
6: $S \leftarrow \textsc{Push}(S, p)$
7: return $S$
```

Chapter 8

Implementation

Chapter 9

Experimental Setup

Algorithm 6 PF(q, g, O, A, R)

```
1: $q_{min} \leftarrow q$
2: $p_{min} \leftarrow \infty$
3: $\theta \leftarrow 0$
4: while $\theta \leq 2\pi$ do
5:     $q' \leftarrow q + \delta t \cdot s \cdot \textsc{Rot}(\theta)$
6:     $p \leftarrow U_{rep}(q', O \cup A) + U_{att}(q', g)$
7:     if $p < p_{min}$ then
8:         $p_{min} \leftarrow p$
9:         $q_{min} \leftarrow q'$
10:    $\theta \leftarrow \theta + \delta\theta$
11: if $||q_{min} - g|| < R$ then
12:     return $\{p_{min}\}$
13: return $\{q_{min}\} \cup \text{PF}(q_{min}, g, O, R)$
```

9.1 Metrics

$$\mathit{MinDist}(\Pi) = \min_{t \in \mathcal{T}} \min_{a \in A} ||\zeta_a(t) - \Pi(t)|| \tag{9.1}$$

$$\mathit{MaxCost}(\Pi) = \max_{t \in \mathcal{T}} P_A(\Pi(t)) \tag{9.2}$$

$$\mathit{AvgCost}(\Pi) = \int_{\mathcal{T}} P_A(\Pi(t))\,\mathrm{d}t \tag{9.3}$$

Chapter 10

Results

10.1 Safety

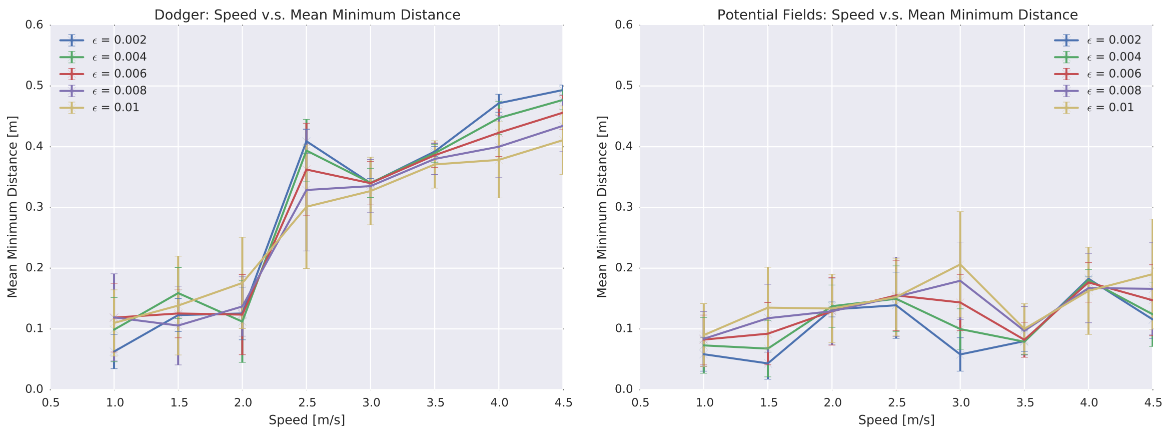

Figure 10.1: Plots showing how the average minimum distance to the obstacles changes as the speed increases for various amounts of obstacle position uncertainties

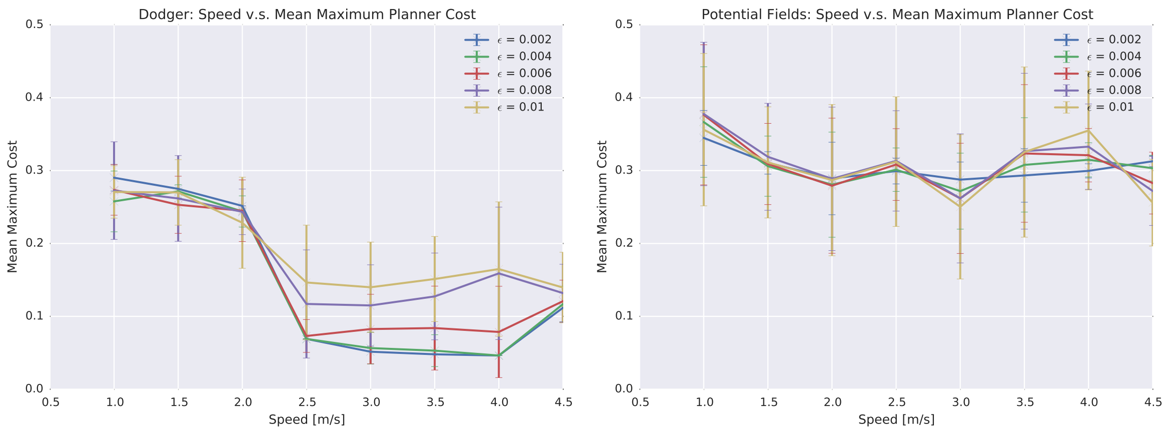

Figure 10.2:

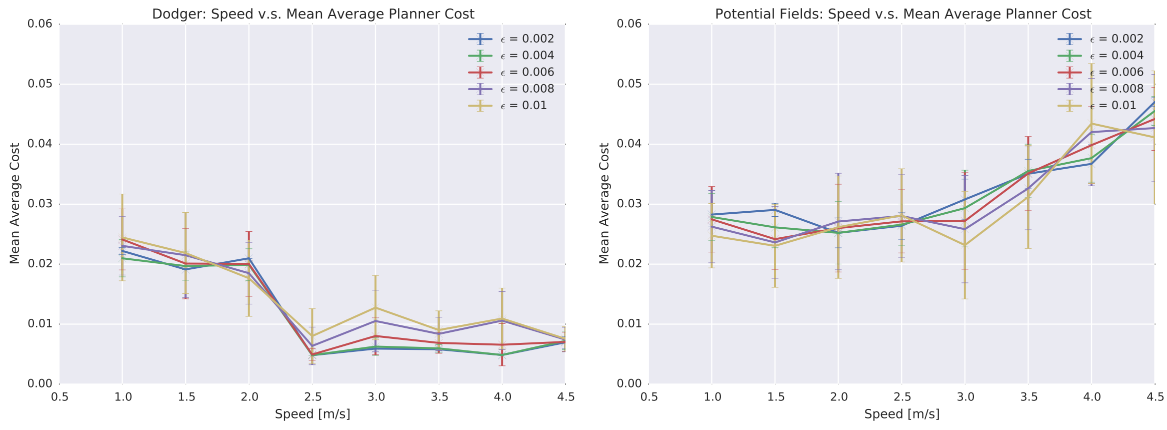

Figure 10.3:

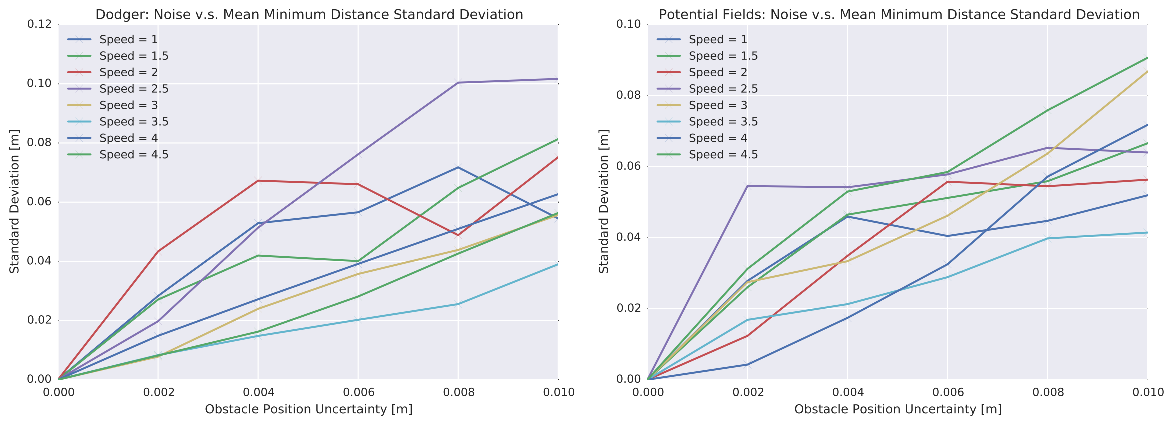

Figure 10.4:

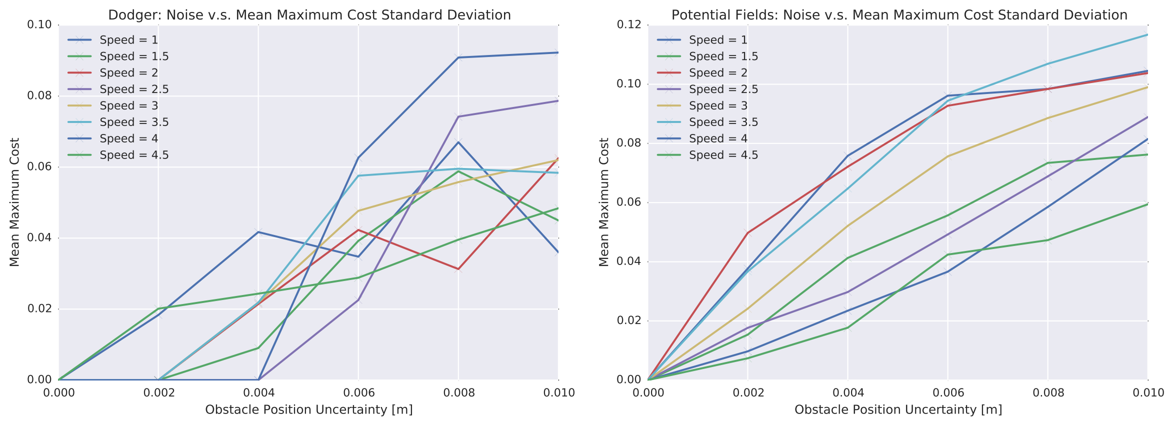

Figure 10.5:

10.1.1 Variance

10.2 Computational Time

10.2.1 Variance

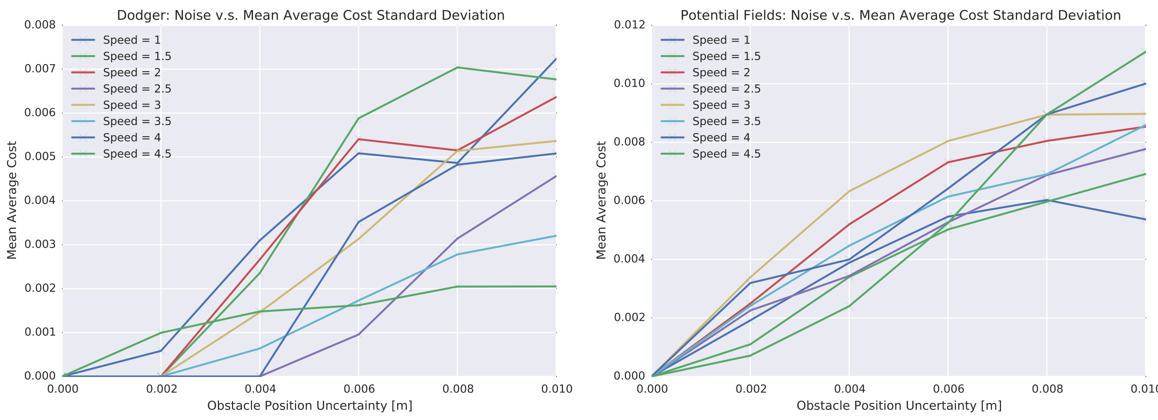

Figure 10.6:

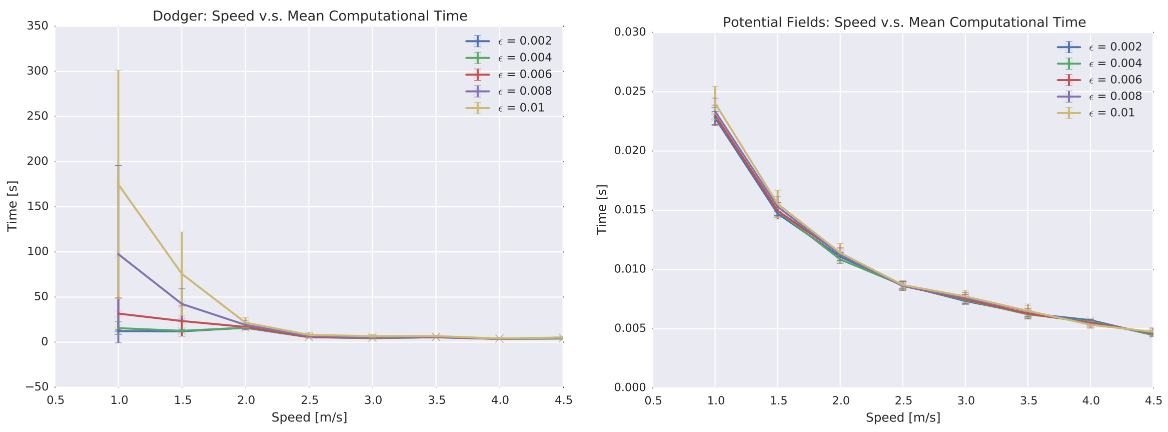

Figure 10.7: Plots showing how the computational time changes as the speed increases for various amounts of obstacle position uncertainties

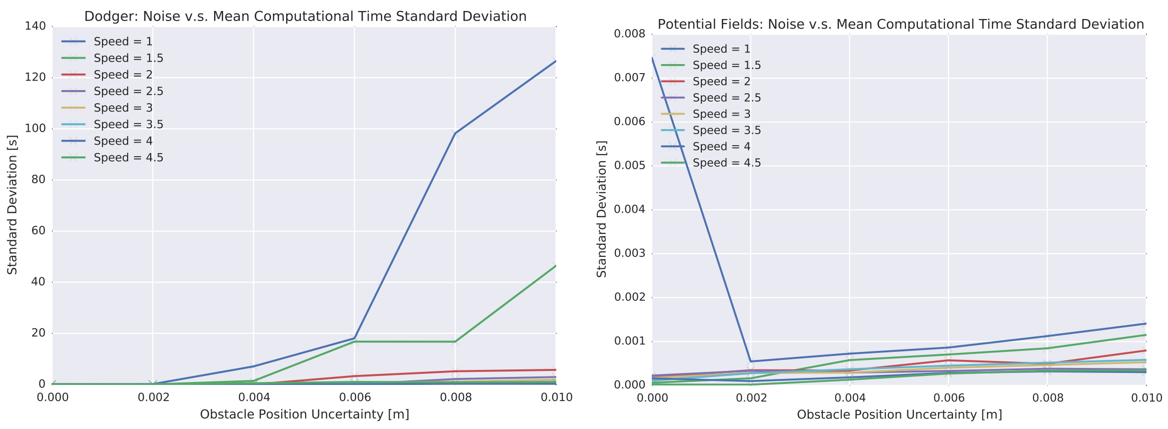

Figure 10.8:

Chapter 11

Discussion

Chapter 12

Acknowledgements

The author would like to thank Dr. Michael Weir and Dr. Erion Plaku for their valuable insight throughout the development of this project and the School of Computer Science for a truly phenomenal undergraduate education. The author would also like to thank his wife, Jessica, for putting up with the late nights and limited contact whilst this project slowly enveloped his life.