Generating Safe Paths in Dynamic Environments By Extracting Minimum Cost Trajectories Using Obstacle Position Probability Distributions and Replanning

University of St Andrews

CS4099: Major Software Project

Author:
Alexander Wallar

Supervisor:
Dr. Michael Weir

March 23, 2015

Abstract

I declare that the material submitted for assessment is my own work except where credit is explicitly given to others by citation or acknowledgement. This work was performed during the current academic year except where otherwise stated.

The main text of this project report is NNN words long, including project specification and plan.

In submitting this project report to the University of St Andrews, I give permission for it to be made available for use in accordance with the regulations of the University Library. I also give permission for the title and abstract to be published and for copies of the report to be made and supplied at cost to any bona fide library or research worker, and to be made available on the World Wide Web. I retain the copyright in this work.

Contents

Chapter 1

Objectives

$$J(C) = \int\limits_C \exp\Big(P_A(x, y, t_0, t_m) + 1\Big)\,\mathrm{d}s \tag{1.1}$$

Where $P_A : U \subset R^4 \to R$, $C \subset U$, $s \in C$, and $g \in C$ where s and g are the starting and goal configurations respectively. The curve C that minimizes J will be the path in space-time that has the smallest chance of coming into contact with an obstacle and is thus the safest possible path through the dynamic environment.

Chapter 2

Design

2.1 Agents

$$P_a(x, y, t_0, t_m) = \int_{t_0}^{t_m} \mathcal{N}(\zeta_a(t), \alpha \cdot (t - t_0)^2 + \beta, x, y) \cdot (t_m - t)^\gamma \, \mathrm{d}t \tag{2.1}$$

Where $\mathcal{N}(\mu, \sigma^2, x, y)$ is the evaluation of a 3D normal distribution centered at (μ_x, μ_y) with a variance of σ^2 at (x, y).

$$P_A(x, y, t_0, t_m) = \frac{\sum\limits_{a \in A} P_a(x, y, t_0, t_m)}{|A|} \tag{2.2}$$

$$\tilde{\zeta_a}(t) = \begin{cases} \tilde{\zeta_a}(t - \delta t) + \zeta_a'(t) \cdot \delta t + \rho & \text{if } t > 0 \\ \zeta_a^{(0)} & \text{if } t = 0 \end{cases} \tag{2.3}$$

Where $\rho \sim \mathcal{U}(-\epsilon, \epsilon)$, $\epsilon > 0$, and $\zeta_a^{(0)}$ is the initial position of the obstacle.

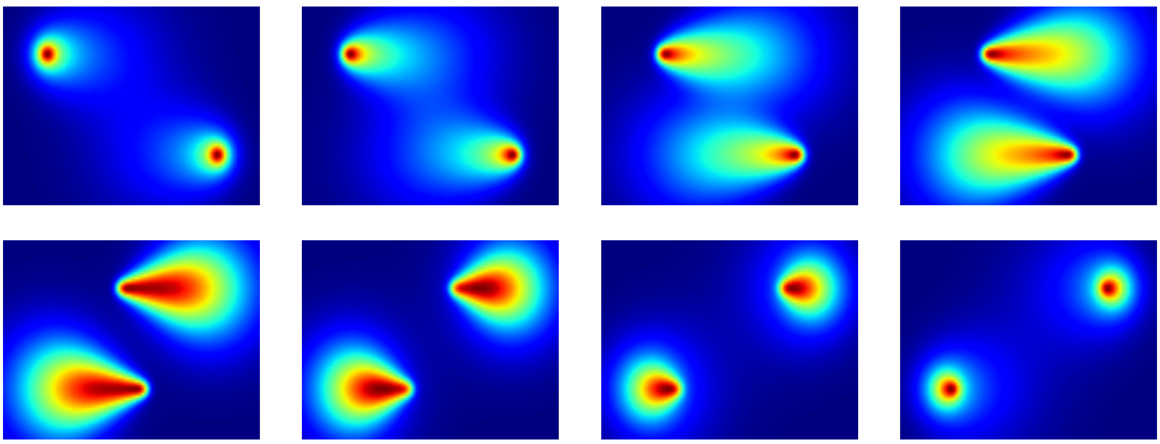

Figure 2.1: Cost distributions indicating the likelihood that an agent will be at a certain location within a given time interval. These figures show how this distribution changes over time (left to right, top to bottom)

2.2 Planning Algorithm

$$C_A(i,j) = \int_0^1 \exp\Big(P_A(x(\lambda), y(\lambda), i_t, j_t) + 1\Big) \cdot ||i - j||_2 \,\mathrm{d}\lambda \tag{2.4}$$

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 $\textsc{Roadmap}(n, d, w, h, O)$

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

for $i = 1$ **to** n **do**
    $q \leftarrow \textsc{RandomPoint2D}(w, h)$
    if $\bigwedge_{o \in O} \neg\textsc{Collision}(o, q)$ **then**
        $V \leftarrow V \cup \{q\}$
for all $q_i \in V$ **do**
    for all $q_j \in V$ **do**
        if $q_i \neq q_j \wedge ||q_i - q_j|| \leq d$ **then**
            $E \leftarrow E \cup \{(q_i, q_j)\}$
return (V, E)

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

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:

$(V, E) \leftarrow \textsc{Roadmap}(n, d, w, h, O)$
$\Pi \leftarrow \emptyset$
$q \leftarrow p$
while $||\textsc{Back}(\Pi) - g||_2 > R$ **do**
    $\pi \leftarrow \textsc{SearchGraph}(V, E, R, A, q, g)$
    for all $i \in \pi$ **do**
        $\Pi \leftarrow \Pi \cup \{i\}$
        for all $a \in A$ **do**
            $\textsc{Step}(a)$
        if $\bigvee_{a \in A} ||\tilde{\zeta_a}(i_t) - \zeta_a(i_t)|| > \delta$ **then**
            for all $a \in A$ **do**
                $\textsc{Update}(\zeta_a, \tilde{\zeta_a})$
            $q \leftarrow i$
            break
return Π

Algorithm 3 $\textsc{SearchGraph}(V, E, R, A, p, g)$

$Q \leftarrow \textsc{PriorityQueue}()$
$D \leftarrow \textsc{Dictionary}()$
$\Pi \leftarrow \textsc{Dictionary}()$
$\textsc{Insert}(Q, p, 0)$
while $\neg\textsc{Empty}(Q)$ **do**
    $q, w \leftarrow \textsc{Pop}(Q)$
    if $||q - g||_2 \leq R$ **then**
        return $\textsc{BacktrackPath}(p, g, \Pi)$
    $N \leftarrow \textsc{GetTemporalNeighbours}(V, E, q)$
    for all $n \in N$ **do**
        $\Pi_n \leftarrow q$
        $c \leftarrow \psi \cdot C_A(q, n) + \omega \cdot D_n$
        $D_n \leftarrow D_n + 1$
        $Q \leftarrow \textsc{Insert}(Q, n, c)$

Algorithm 4 $\textsc{GetTemporalNeighbours}(V, E, q)$

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

Algorithm 5 $\textsc{BacktrackPath}(p, g, \Pi)$

$q \leftarrow g$
$S \leftarrow \textsc{Stack}()$
while $\Pi_q \neq p$ **do**
    $S \leftarrow \textsc{Push}(S, q)$
    $q \leftarrow \Pi_q$
$S \leftarrow \textsc{Push}(S, p)$
return S

Chapter 3

Experimental Setup

Algorithm 6 $\mathrm{PF}(q, g, O, A, R)$

$q_{min} \leftarrow q$
$p_{min} \leftarrow \infty$
$\theta \leftarrow 0$
while $\theta \leq 2\pi$ **do**
    $q' \leftarrow q + \delta t \cdot s \cdot \textsc{Rot}(\theta)$
    $p \leftarrow U_{rep}(q', O \cup A) + U_{\mathrm{att}}(q', g)$
    if $p < p_{min}$ **then**
        $p_{min} \leftarrow p$
        $q_{min} \leftarrow q'$
    $\theta \leftarrow \theta + \delta\theta$
if $||q_{min} - g|| < R$ **then**
    return $\{p_{min}\}$
return $\{q_{min}\} \cup \mathrm{PF}(q_{min}, g, O, R)$

3.1 Metrics

$$MinDist(\Pi) = \min_{t \in T} \min_{a \in A} ||\zeta_a(t) - \Pi(t)|| \tag{3.1}$$

$$MaxCost(\Pi) = \max_{t \in T} P_A(\Pi(t)) \tag{3.2}$$

$$AvgCost(\Pi) = \int_T P_A(\Pi(t))\,\mathrm{d}t \tag{3.3}$$

Chapter 4

Results

4.1 Safety

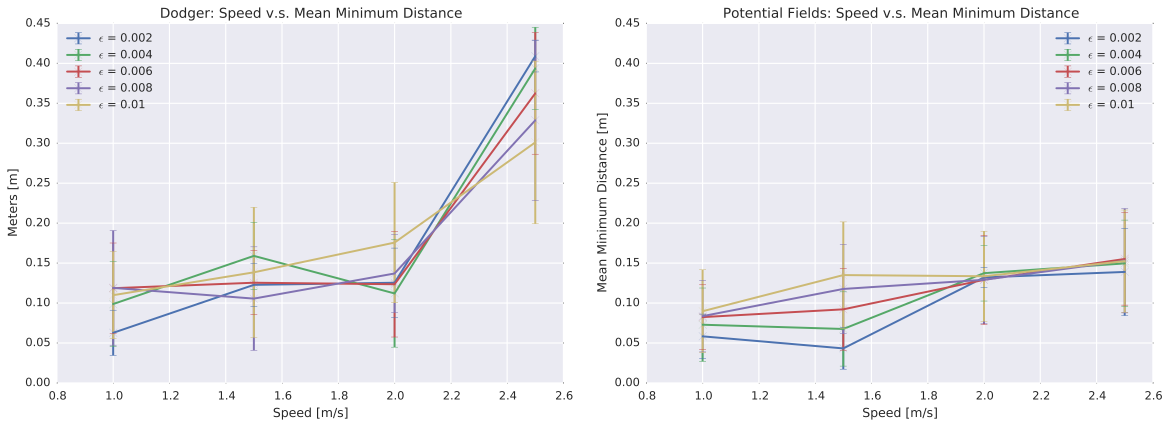

Figure 4.1: Plots showing how the average minimum distance to the obstacles changes as the speed increases for various amounts of obstacle position uncertainties

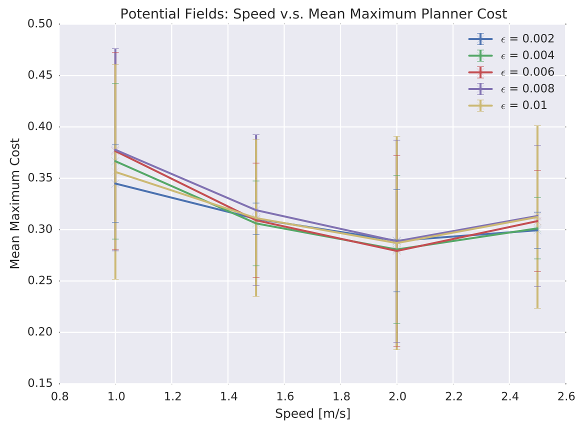

Figure 4.2:

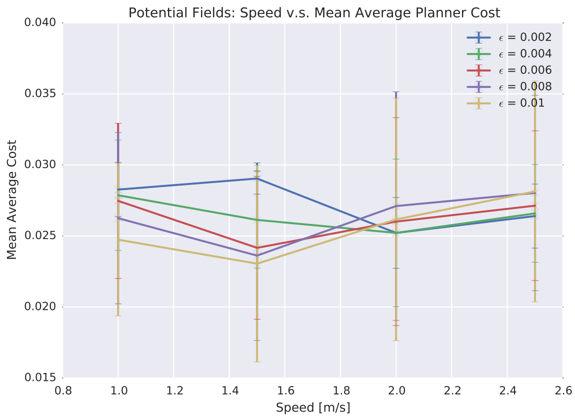

Figure 4.3:

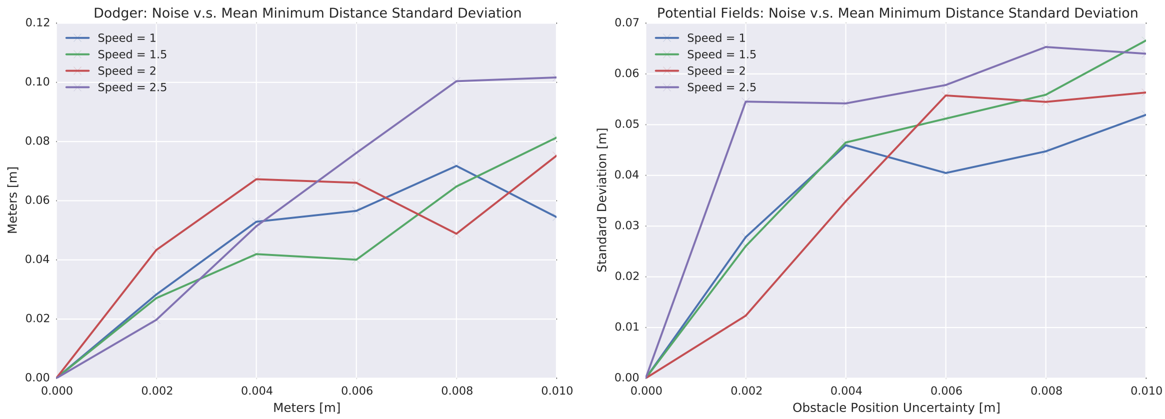

Figure 4.4:

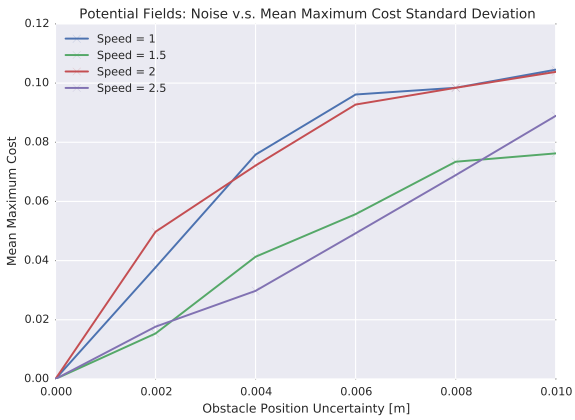

Figure 4.5:

4.1.1 Variance

4.2 Computational Time

4.2.1 Variance

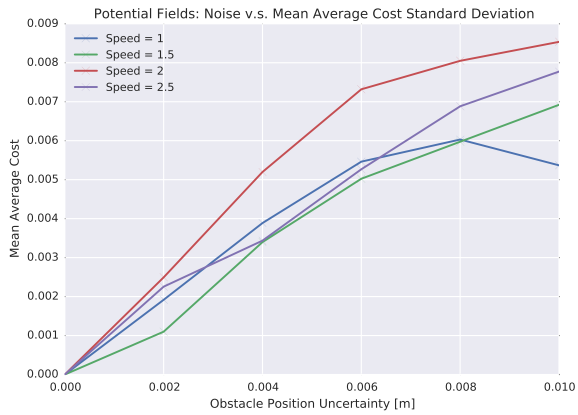

Figure 4.6:

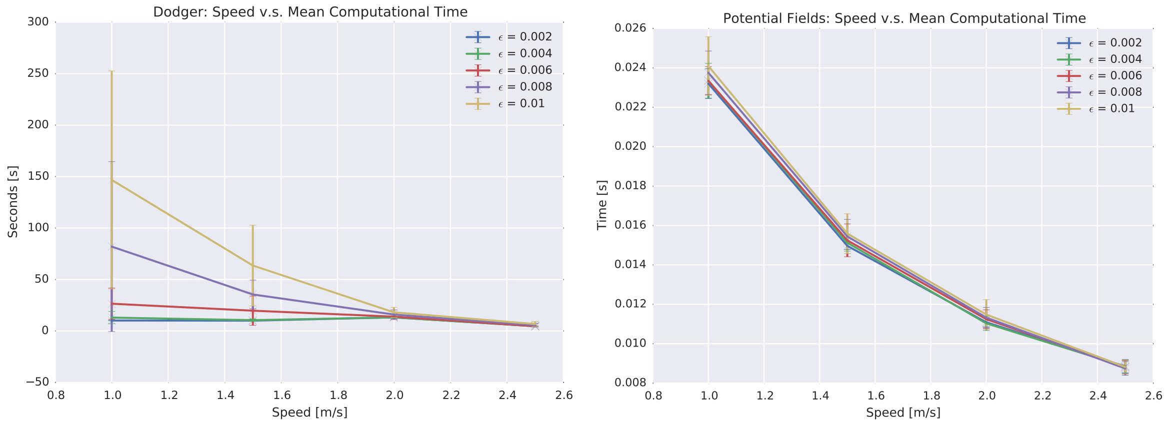

Figure 4.7: Plots showing how the computational time changes as the speed increases for various amounts of obstacle position uncertainties

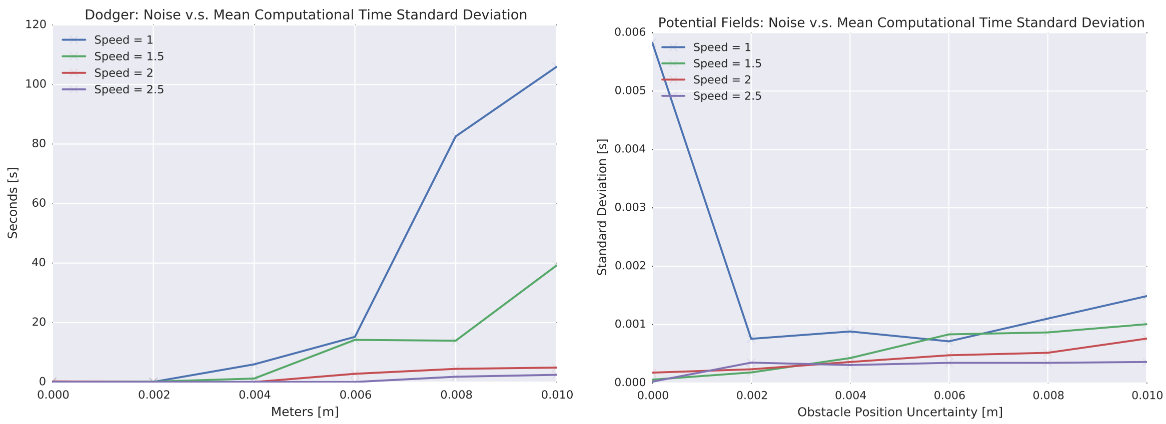

Figure 4.8:

Chapter 5

Discussion