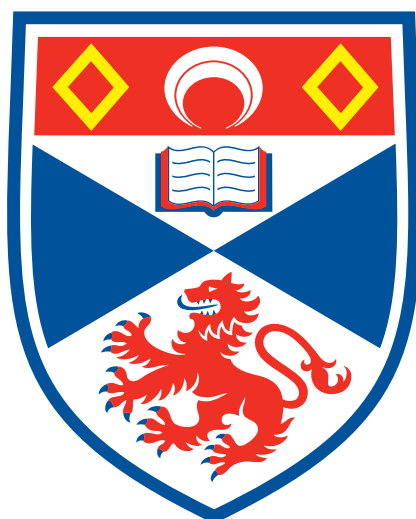


---

# 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 21, 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

<b>1</b>	<b>Design</b>	<b>3</b>
<b>2</b>	<b>Experimental Setup</b>	<b>6</b>
<b>3</b>	<b>Discussion</b>	<b>7</b>

# Chapter 1

## Design

$$P_a(x, y, t_0, t_m) = \int_{t_0}^{t_m} \mathcal{N}(\zeta_a(t), \alpha \cdot (t - t_0)^2 + \beta, x, y) \cdot (t_m - t)^\gamma dt \quad (1.1)$$

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

$$P_A(x, y, t_0, t_m) = \frac{\sum_{a \in A} P_a(x, y, t_0, t_m)}{|A|} \quad (1.2)$$

$$C_A(i, j) = \int_0^1 \exp\left(P_A(x(\lambda), y(\lambda), i_t, j_t)\right) \cdot \|i - j\|_2 d\lambda \quad (1.3)$$

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  $j$ .

$$\tilde{\zeta}_a(t) = \begin{cases} \tilde{\zeta}_a(t - \delta t) + \zeta'_a(t) \cdot \delta t + \rho & \text{if } t > 0 \\ \zeta_a^{(0)} & \text{if } t = 0 \end{cases} \quad (1.4)$$

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

---

**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:    $q \leftarrow \text{RANDOMPOINT2D}(w, h)$ 
3:   if  $\bigwedge_{o \in O} \neg \text{COLLISION}(o, q)$  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:  $(V, E) \leftarrow \text{ROADMAP}(n, d, w, h, O)$ 
2:  $\Pi \leftarrow \text{SET}()$ 
3:  $q \leftarrow p$ 
4: while  $\|\text{BACK}(\Pi) - g\|_2 > R$  do
5:    $\pi \leftarrow \text{SEARCHGRAPH}(V, E, R, A, q, g)$ 
6:   for all  $i \in \pi$  do
7:      $\Pi \leftarrow \Pi \cup \{i\}$ 
8:     for all  $a \in A$  do
9:        $\text{STEP}(a)$ 
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:         $\text{UPDATE}(\zeta_a, \tilde{\zeta}_a)$ 
13:       $q \leftarrow i$ 
14:    break
15: return  $\Pi$ 

```

---

---

**Algorithm 3** SEARCHGRAPH( $V, E, R, A, p, g$ )

---

```

1:  $Q \leftarrow \text{PRIORITYQUEUE}()$ 
2:  $D \leftarrow \text{DICTIONARY}()$ 
3:  $\Pi \leftarrow \text{DICTIONARY}()$ 
4:  $\text{INSERT}(Q, p, 0)$ 
5: while  $\neg \text{EMPTY}(Q)$  do
6:    $q, w \leftarrow \text{POP}(Q)$ 
7:   if  $\|q - g\|_2 \leq R$  then
8:     return BACKTRACKPATH( $p, g, \Pi$ )
9:    $N \leftarrow \text{GETTEMPORALNEIGHBOURS}(V, E, q)$ 
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:    if  $w > c$  then
15:       $c \leftarrow w$ 
16:     $Q \leftarrow \text{INSERT}(Q, n, q)$ 

```

---



---

**Algorithm 4** GETTEMPORALNEIGHBOURS( $V, E, q$ )

---

```

1:  $S \leftarrow \text{Set}()$ 
2:  $N \leftarrow \text{NEIGHBOURS}(V, E, q)$ 
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( $p, g, \Pi$ )

---

```

1:  $q \leftarrow g$ 
2:  $S \leftarrow \text{STACK}()$ 
3: while  $\Pi_q \neq p$  do
4:    $S \leftarrow \text{PUSH}(S, q)$ 
5:    $q \leftarrow \Pi_q$ 
6:  $S \leftarrow \text{PUSH}(S, p)$ 
7: return  $S$ 

```

---

## Chapter 2

# Experimental Setup

---

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

---

```
1:  $q_{min} \leftarrow q$ 
2:  $p_{min} \leftarrow \infty$ 
3:  $\theta \leftarrow 0$ 
4: while  $\theta \leq 2\pi$  do
5:    $q' \leftarrow q + \delta t \cdot s \cdot \text{ROT}(\theta)$ 
6:    $p \leftarrow U_{\text{rep}}(q', O \cup A) + U_{\text{att}}(q', g)$ 
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  $\{q_{min}\} \cup \text{PF}(q_{min}, g, O, R)$ 
```

---



## Chapter 3

## Discussion