|  |
| --- |
| Ατομική Διπλωματική Εργασία  **Σχεδίαση Bug Αλγόριθμου για Προγραμματισμό Κίνησης και Υλοποίηση Προσομοιωτή για Αλγόριθμους από την Οικογένεια Βug**    Κονναρής Ανδρέας  **ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**  UCY-new-logo  **ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**  **Μάιος 2018** |

**ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΥΠΡΟΥ**

**ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ**

**Σχεδίαση Bug Αλγόριθμου για Προγραμματισμό Κίνησης και Υλοποίηση Προσομοιωτή για Αλγόριθμους από την Οικογένεια Βug**

Κονναρής Ανδρέας

Επιβλέπουσα Καθηγήτρια

Φιλίππου Άννα

Η Ατομική Διπλωματική Εργασία υποβλήθηκε προς μερική εκπλήρωση των απαιτήσεων απόκτησης του πτυχίου Πληροφορικής του Τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου

Μάιος 2018

**Ευχαριστίες**

Η παρούσα διπλωματική εργασία εκπονήθηκε το διάστημα μεταξύ Σεπτεμβρίου 2017 και Μαΐου 2018 στα πλαίσια του προπτυχιακού προγράμματος στη Πληροφορική/ Τεχνολογία Λογισμικού του τμήματος Πληροφορικής του Πανεπιστημίου Κύπρου. Ως την ελάχιστη δυνατή ένδειξη ευχαριστιών, με την παρούσα παράγραφο οφείλω να ευχαριστήσω όλους όσους συνέβαλαν στην εκπόνησή της και ιδιαίτερα:

Την επιβλέπουσα καθηγήτριά μου, κα Φιλίππου Άννα, για την πολύτιμη υποστήριξή της, τις παραγωγικές υποδείξεις της και το πολύ καλό κλίμα συνεργασίας που διαμόρφωσε συμβάλλοντας τα μέγιστα για την κατάρτιση της διπλωματικής μου εργασίας.

Ιδιαίτερα θερµές ευχαριστίες θέλω να δώσω στην οικογένεια µου για τη συνεχή συµπαράσταση τους, για τις πολύτιµες συµβουλές τους και για όλα όσα µου έχουν προσφέρει όλα αυτά τα χρόνια των σπουδών µου.

Τέλος θα ήθελα να ευχαριστήσω όλους τους συναδέλφους μου προπτυχιακούς φοιτητές για τις ανταλλαγές απόψεων, το ειλικρινές ενδιαφέρον τους και για τη σημαντική βοήθειά τους σε όλα τα στάδια της εργασίας.

**Περίληψη**

Η πτυχιακή έρευνα αυτή, έχει κάνει μελέτη στο πρόβλημα της σχεδίασης διαδρομής ενός κινητού ρομπότ και έχει ασχοληθεί με τη μελέτη των αλγόριθμων της οικογένειας Bug, οι οποίοι ασχολούνται με δυναμικά και άγνωστα περιβάλλοντα. Μέσα από τη μελέτη αυτή των αλγόριθμων επινοήθηκε και ένας νέος αλγόριθμος, ο οποίος προτείνεται από τον συγγραφέα. Ο αλγόριθμος αυτός έχει βασικά στοιχεία από τους αλγόριθμους Bug2 και Alg1 και έχει σκοπό την βελτιστοποίησή τους. Επίσης, με την εκπόνηση της συγκεκριμένης πτυχιακής έρευνας, έχει υλοποιηθεί ένας προσομοιωτής στη γλώσσα προγραμματισμού Java που υποστηρίζει τους αλγόριθμους που μελετηθήκαν καθώς επίσης και τον αλγόριθμο που προτείνεται. Τέλος έχουν γίνει κάποια πειράματα για τη σύγκριση της αποδοτικότητας όλων των αλγόριθμων βάσει του μεγέθους της διαδρομής που σχεδιάζεται, αλλά και τη σύγκριση του αλγόριθμου που προτείνεται στην πτυχιακή εργασία, με τους υπόλοιπους. Τα αποτελέσματα αυτά μας οδηγούν στο συμπέρασμα πως κανένας αλγόριθμος δεν είναι ο βέλτιστος σε όλα τα περιβάλλοντα

**Περιεχόμενα**

Κεφάλαιο **1 Εισαγωγή**............................................................................................................ **1**

1.1 Κίνητρα......................................................................................................... 1

1.2 Σκοπός........................................................................................................... 3

1.3 Μεθοδολογία................................................................................................. 3

1.4 Συνεισφορά Πτυχιακής Εργασίας............................................................ 5

1.5 Δομή Πτυχιακής Εργασίας......................................................................... 6

Κεφάλαιο **2 Σχετικές Εργασίες**.............................................................................................. **7**

2.1 Στατικό και Γνωστό Περιβάλλον................................................................... 10

2.2 Στατικό και άγνωστο περιβάλλον ............................................................ 10

2.3 Δυναμικό και γνωστό περιβάλλον............................................................. 11

2.4 Δυναμικό και άγνωστο περιβάλλον........................................................... 11

2.5 Γλώσσα προγραμματισμού ....................................................................... 12

2.5.1 Java ............................................................................................... 12

2.5.1 Βιβλιοθήκες ................................................................................. 12

Κεφάλαιο **3 Υλοποιημένοι Αλγόριθμοι** ............................................................................. **14**

3.1 Αλγόριθμος Bug1......................................................................................... 15

3.2 Αλγόριθμος Bug2 ........................................................................................ 17

3.3 Αλγόριθμος Alg1 ......................................................................................... 20

3.4 Αλγόριθμος Αlg2 ......................................................................................... 21

3.5 Αλγόριθμος Class1 ..................................................................................... 23

3.6 Αλγόριθμος Com ......................................................................................... 25

3.7 Συμπεράσματα ............................................................................................... 27

Κεφάλαιο **4** **Αλγόριθμος MBack**............................................................................................. 28

4.1 Εισαγωγή......................................................................................................... 28

4.2 Περιγραφή....................................................................................................... 29

4.3 Ψευδοκώδικας............................................................................................... 31

4.4 Μέθοδος Μ-Γραμμής................................................................................... 33

4.5 Περιμετρική Διάσχιση Εμποδίου............................................................... 33

4.6 Κατεύθυνση προς τον στόχο....................................................................... 34

4.7Αποθήκευση σημείων και κινήσεων........................................................... 34

4.8 Απόδειξη τερματισμού................................................................................ 35

Κεφάλαιο **5 Υλοποίηση**......................................................................................................... **37**

* 1. Από τη θεωρία στην υλοποίηση............................................................... 38

5.1.1 Ανανέωση θέσης............................................................................ 38

5.1.2Αποθήκευση σημείων.................................................................. 39

5.1.3 Εμπόδια ........................................................................................ 39

5.1.4 Εξοπλισμός αισθητήρων............................................................. 40

5.2 Κατεύθυνση προς τον στόχο.......................................................................... 41

5.3 Δυναμικό Περιβάλλον.................................................................................... 43

5.4 Υλοποίηση στην Java...................................................................................... 44

5.4.1 Βιβλιοθήκες .................................................................................... 44

5.4.2 Κλάσεις και Μέθοδοι ..................................................................... 47

5.4.2.1 Κλάση Create ............................................................... 47

5.4.2.2 Κλάση Point ................................................................. 48

5.4.2.3 Κλάση Environment .................................................... 48

5.4.2.4 Κλάση Robot ................................................................ 49

5.4.2.5 Κλάση Μanagement .................................................... 51

Κεφάλαιο **6 Πειραματική Αξιολόγηση Αλγόριθμων** ........................................................ **52**

6.1 Μεθοδολογία και Διευκρινήσεις .............................................................. 52

6.2 Πειράματα........................................................................................... 53

6.3 Συμπεράσματα............................................................................................. 64

Κεφάλαιο **7 Συμπεράσματα** ............................................................................................  **66**

7.1 Περίληψη ............................................................................................ 66

7.2 Μελλοντικές εργασίες ......................................................................... 67

**Βιβιλιογραφία**.................................................................................................................. **68**

**Κεφάλαιο 1**

**Εισαγωγή**

1.1 Κίνητρα 1

1.2 Σκοπός 3

1.3 Μεθοδολογία 3

1.4 Συνεισφορά Πτυχιακής Εργασίας 5

1.5 Δομή Πτυχιακής Εργασίας 6

Η παρούσα πτυχιακή εργασία αναπτύσσει και υλοποιεί έναν προσομοιωτή πλοήγησης για ρομπότ και υποστηρίζει 7 διαφορετικούς αλγορίθμους. Σκοπός του προσομοιωτή είναι ο σχεδιασμός διαδρομής για ένα κινητό ρομπότ σε άγνωστα και δυναμικά περιβάλλοντα. Υλοποιήθηκαν 6 προϋπάρχοντες αλγόριθμοι καθώς επίσης και ένας νέος όπου προτείνεται από τον συγγραφέα. Πέρα από την υλοποίηση των αλγορίθμων, στην εργασία αυτή γίνεται σύγκριση της αποδοτικότητας και της αποτελεσματικότητας τους. Το κεφάλαιο αυτό εισάγει μερικές βασικές έννοιες και πληροφορίες για τον σχεδιασμό διαδρομών για ρομπότ, επεξηγεί τα κίνητρα του έργου, τον σκοπό της εργασίας, την μεθοδολογία και την συνεισφορά της έρευνας αυτής και τη δομή που απαρτίζεται η πτυχιακή εργασία.

* 1. **Κίνητρα**

Σήμερα, η έρευνα σχετικά με τους διάφορους αλγόριθμους για προγραμματισμό και σχεδίαση διαδρομών για κινητά ρομπότ είναι ένα πολύ καυτό θέμα. Τα κινητά ρομπότ χρησιμοποιούνται ευρέως σε πολλά πεδία και προπαντός σε επικίνδυνα, όπου μπορεί να υπάρχουν κίνδυνοι για τον άνθρωπο, όπως οι πυρηνικές βιομηχανίες.

Η εύρεση μιας ασφαλούς διαδρομής σε ένα περιβάλλον, ειδικότερα όταν αυτό είναι επικίνδυνο, είναι απαραίτητη προϋπόθεση για την επιτυχία της εκτέλεσης ενός αλγόριθμου σχεδιασμού διαδρομής για κινητά ρομπότ. Επομένως η έρευνα στους αλγόριθμους για σχεδιασμό διαδρομής , όπου θα επιτύχουν και να εγγυηθούν πως ένα ρομπότ θα καταφέρει να μετακινηθεί από την αρχική του θέση στον τελικό του προορισμό, είναι θεμελιώδης απαίτηση για την ασφάλεια των ρομπότ. Επιπρόσθετα για να μπορέσει να μειωθεί ο χρόνος επεξεργασίας των δεδομένων και η κατανάλωση ενέργειας θα πρέπει οι αλγόριθμοι να προσπαθούν για εύρεση λύσης που να είναι όσο το δυνατόν καλύτερη και βέλτιστη, δηλαδή τον σχεδιασμό ενός μονοπατιού που θα ακολουθήσει το κινητό ρομπότ, που να είναι όσο πιο μικρό σε μήκος γίνεται.

Η αλγόριθμοι σχεδίασης διαδρομής για ένα κινητό ρομπότ κατηγοριοποιούνται σε 4 διαφορετικές κατηγορίες οι οποίες είναι, γνωστό και στατικό περιβάλλον, γνωστό και δυναμικό περιβάλλον, άγνωστο και στατικό περιβάλλον και τέλος άγνωστο και δυναμικό περιβάλλον. Επειδή οι πληροφορίες ενός δυναμικού περιβάλλοντος αλλάζουν καθώς αλλάζουν και τα εμπόδια που υπάρχουν σε αυτό, η πολυπλοκότητα και η αβεβαιότητα του προβλήματος για το σχεδιασμό μιας διαδρομής αυξάνεται σημαντικά σε δυναμικά περιβάλλοντα. Έτσι για τον λόγο αυτό, παραδοσιακές μέθοδοι για σχεδιασμό διαδρομής όπως γράφοι [21], πλέγματα [24] και διαγράμματα Voronoi [22] δεν είναι κατάλληλες για σχεδιασμό διαδρομής σε δυναμικά περιβάλλοντα. Πριν από λίγα χρόνια οι Wang, Sillitoe και Davide [31] εισήγαγαν ένα γενετικό αλγόριθμο με σκοπό την αντιμετώπιση του προβλήματος για σχεδίαση διαδρομής σε περιβάλλον με κινητά εμπόδια. Ωστόσο η μέθοδος αυτή εξακολουθεί να έχει κάποια μειονεκτήματα [31] όπως, αφού σε μερικές περιπτώσεις δεν καταφέρνει να βρει το βέλτιστο μονοπάτι.

Η παρούσα εργασία ασχολείται και υλοποιεί αλγόριθμους από την οικογένεια των Bug [] αλγόριθμων όπου εφαρμόζονται σε άγνωστα και δυναμικά περιβάλλοντα. Σκοπός των αλγορίθμων αυτών είναι να ολοκληρώσουν μία διαδρομή από ένα αρχικό σημείο προς ένα τελικό, χωρίς να έρθουν σε άμεση επαφή με οποιοδήποτε εμπόδιο. Οι bug αλγόριθμοι είναι οι απλούστεροι αλγόριθμοι εύρεσης διαδρομής. Χρησιμοποιούνται εκεί όπου δεν υπάρχει πληροφορία για το περιβάλλον και οι θέσεις αλλά και τα σχήματα των όποιων εμποδίων είναι άγνωστα. Τα μόνα στοιχεία που δέχονται αυτοί οι αλγόριθμοι είναι μέσω των αισθητήρων τους. Έτσι οι αλγόριθμοι μπορούν και υποθέτουν πληροφορίες σχετικά με το περιβάλλον και τον προορισμό τους. Εάν δεν υπάρχει προσβάσιμο μονοπάτι, τότε ο αλγόριθμος μπορεί και το αναγνωρίζει και ενημερώνει ότι δεν μπορεί να φτάσει στον ζητούμενο προορισμό, αποφεύγοντας άσκοπες διαδρομές.

* 1. **Σκοπός**

Όπως προαναφέρθηκε και στο προηγούμενο υποκεφάλαιο η σχεδίαση διαδρομής για κινητά ρομπότ είναι ένα θέμα το οποίο τυγχάνει αρκετών ερευνών ενώ αντιμετωπίζονται πολλά προβλήματα όσο αφορά τους αλγόριθμους που εκτελούνται για άγνωστα και δυναμικά περιβάλλοντα. Σκοπός της παρούσας πτυχιακής εργασίας είναι η μελέτη διάφορων αλγόριθμων για πλοήγηση κινούμενων ρομπότ και μέσα από την έρευνα αυτή να υπάρξει μια υλοποίηση από μερικούς επιλεγμένους αλγόριθμους καθώς επίσης και η σύγκριση τους όσο αφορά την αποδοτικότητα και την αποτελεσματικότητα τους. Επιπρόσθετα η εργασία, μέσα από την μελέτη αυτή, έχει σκοπό την επινόηση ενός νέου αλγόριθμου ο οποίος βασίζεται στους αλγόριθμους που επιλέχτηκαν για υλοποίηση, ενώ έχει και κύριο άξονα την χρήση της μεθόδου οπισθοδρόμησης.

* 1. **Μεθοδολογία**

Με την έναρξη της παρούσας πτυχιακής εργασίας μελετήθηκε για αρχή η έννοια του σχεδιασμού διαδρομής για κινητά ρομπότ ούτως ώστε να δημιουργηθεί μια σφαιρική εικόνα γύρω από το συγκεκριμένο θέμα. Μελετήθηκαν επιστημονικά άρθρα όπου εξηγούν την έννοια αυτή, τις δυσκολίες και τα προβλήματα που υπάρχουν στην σχεδίαση διαδρομής, καθώς επίσης και τις εφαρμογές μου τυγχάνει ο συγκεκριμένος τομέας. Η κατανόηση και η εκμάθηση πληροφοριών για τον σχεδιασμό διαδρομής ήταν αναγκαία για την μετ’ έπειτα πορεία της διπλωματικής εργασίας. Έχοντας βάση αυτό, υπήρχε η ικανότητα μετά για την μελέτη διάφορων αλγορίθμων που έχουν στόχο τον σχεδιασμό διαδρομών και πλοήγησης κινητών ρομπότ.

Αφού πλέον κατανοήθηκε η έννοια του σχεδιασμού διαδρομής, επόμενο βήμα ήταν η μελέτη διάφορων αλγορίθμων όπου είχαν στόχο την πλοήγηση ενός ρομπότ. Μελετήθηκαν αλγόριθμοι από όλο το φάσμα της έννοιας της σχεδίασης διαδρομής με σκοπό την πλήρη κατανόηση των προβλημάτων που αντιμετωπίζουν στην εφαρμογή τους. Επίσης με την μελέτη των αλγορίθμων δημιουργήθηκε μια πιο πρακτική εικόνα όσο αφορά το σκοπό της σχεδίασης διαδρομής και τα προβλήματα που αντιμετωπίζει. Αφού πλέον έγινε μια μελέτη στους διάφορους αλγόριθμους πάρθηκε μια απόφαση που αφορούσε σε ποιους από αυτούς θα δοθεί επικέντρωση για ανάλυση και υλοποίηση τους.

Οι αλγόριθμοι που επιλέχτηκαν να αναλυθούν και να υλοποιηθούν είναι από την οικογένεια των Bug αλγόριθμων και είναι οι Bug1, Bug2, Alg1, Alg2, Class1 και Com. Ο λόγος που επιλέχτηκαν αυτοί οι αλγόριθμοι είναι επειδή ανήκουν στην ίδια κατηγορία και υπάρχει μια συνοχή μεταξύ τους και ως συνήθως ο ένας είναι μία εξέλιξη και βελτιστοποίηση κάποιου άλλου. Επίσης οι συγκεκριμένοι αλγόριθμοι είναι ιδανικοί για άγνωστα περιβάλλοντα και το ρομπότ χρειάζεται να γνωρίζει μόνο την αρχική του θέση και την θέση του τελικού του προορισμού. Το γεγονός ότι οι αλγόριθμοι αυτοί εκτελούνται σε άγνωστα περιβάλλοντα, έκανε πιο ενδιαφέρουσα την ενασχόλησή μου με το θέμα ενώ τέλος δεν χρειάζονται πολύ μεγάλη υπολογιστική ισχύ και μνήμη κατά την εκτέλεσή τους. Καθώς αναλύθηκαν αυτοί οι αλγόριθμοι πιο πολύ και πιο λεπτομερώς καταγράφηκαν και παρατηρήθηκαν διάφορα μειονεκτήματα που έχουν και δυσκολίες που αντιμετωπίζουν, έτσι δημιουργήθηκε η ανάγκη για την προσπάθεια επινόησης ενός νέου αλγόριθμου όπου θα μπορούσε να βελτιστοποιήσει ως προς την απόδοση τους πιο πάνω αλγόριθμους.

Με βάση τα μειονεκτήματα που αντιμετωπίζουν αλγόριθμοι που αναλύθηκαν και σε συνδυασμό τη γνώση για τις δυσκολίες που αντιμετωπίζει η σχεδίαση διαδρομών επινοήθηκε ένας νέος αλγόριθμος όπου αποφέρει καλύτερα αποτελέσματα από τους υπόλοιπους όμως αυτό δεν είναι δεδομένο και εγγυημένο για όλες τις περιπτώσεις. Όπως προαναφέρθηκε και πριν μπορεί οι αλγόριθμοι αυτοί να είναι μια βελτίωση από κάποιους άλλους αλλά αυτό δεν είναι απόλυτο. Στην μετέπειτα πορεία ο νέος αυτός αλγόριθμος αναλύθηκε, αποδείχτηκε στη θεωρία η ορθότητα του και αυτό που έμεινε είναι η υλοποίηση του μαζί με όλους τους αλγόριθμους που επιλέχθηκαν καθώς επίσης και η εξαγωγή αποτελεσμάτων για την αποδοτικότητα τους μέσα από διάφορα πειράματα.

Πλέον αφού έχει γίνει μια ολοκληρωμένη έρευνα για την σχεδίαση διαδρομών για κινητά ρομπότ και των αλγορίθμων , αφού έγινε επιλογή μερικών από αυτών και περαιτέρω ανάλυση τους και αφού προτάθηκε ένας νέος αλγόριθμος σειρά είχε η υλοποίησή τους. Αναπτύχθηκε ένας προσομοιωτής πλοήγησης για ρομπότ στη γλώσσα προγραμματισμού Java όπου υλοποιήθηκαν συνολικά 7 αλγόριθμοι. Οι 6 που επιλέχθηκαν όπως προαναφερθήκαν, μαζί με αυτόν που προτείνεται. Η επιλογή της Java σαν γλώσσα προγραμματισμού έγινε για το λόγο ότι η συγκεκριμένη γλώσσα προσφέρει βιβλιοθήκες όπου μπορούν να υποστηρίξουν μια καλή γραφική διαπροσωπία μιας και ο στόχος ήτανε η ανάπτυξη ενός προσομοιωτή. Αφού έγινε και η υλοποίηση, σαν τελευταίο βήμα της παρούσας πτυχιακής εργασίας ήταν η εξαγωγή αποτελεσμάτων μέσα από διάφορα πειράματα. Αυτό έγινε με την εκτέλεση των αλγορίθμων σε διαφορετικά περιβάλλοντα.

* 1. **Συνεισφορά Πτυχιακής Εργασίας**

Με την εκπόνηση της πτυχιακής εργασίας οι στόχοι που ορίστηκαν επιτεύχθηκαν, και ένας προσομοιωτής για πλοήγηση κινητών ρομπότ σε άγνωστα και δυναμικά περιβάλλοντα έχει αναπτυχθεί επιτυχώς. Αλγόριθμοι που είναι δημοσιευμένοι ως επί το πλείστων σαν ψευδοκώδικες, αν και υπάρχουν υλοποιημένοι και σε περιβάλλοντα ρομπότ, πλέον μετατράπηκαν σε κώδικα και είναι εκτελέσιμοι όλοι μαζί σε ένα προσομοιωτή όπου κάποιος χρήστης μπορεί να τους τρέξει σε περιβάλλοντα δημιουργημένα από αυτόν. Επίσης έχει γίνει αξιολόγηση και σύγκριση των αλγόριθμων, ενώ τέλος ο αλγόριθμος που προτείνεται βελτιστοποιεί τους προηγούμενους κατά μεγάλο βαθμό.

* 1. **Δομή Πτυχιακής Εργασίας**

Το υπόλοιπο της πτυχιακής εργασίας οργανώνεται ως εξής: Στο Κεφάλαιο 2 υπάρχει μια περιγραφή όλων των σχετικών και προηγούμενων εργασιών. Επιπρόσθετα γίνεται μια αναφορά στα εργαλεία που χρησιμοποιήθηκαν στην εργασία, καθώς επίσης και μια πιο εκτενής αναφορά στο πρόβλημα του σχεδιασμού διαδρομής. Τέλος εξηγούνται μερικά πράγματα για τη γλώσσα προγραμματισμού που έγινε η υλοποίηση μαζί με της βιβλιοθήκες που χρησιμοποιήθηκαν. Στην συνέχεια, στο Κεφάλαιο 3 εξηγούνται αναλυτικά ένας-ένας οι αλγόριθμοι που επιλέχθηκαν για ανάλυση και υλοποίηση. Υπάρχει αρχικά μια περιγραφή για αυτούς, μετά παρουσιάζεται ο ψευδοκώδικας τους και στο τέλος κάποιο συμπέρασμα για τον καθένα. Φτάνοντας στο Κεφάλαιο 4, περιγράφεται αναλυτικά ο αλγόριθμος που προτείνεται, παρουσιάζεται ο ψευδοκώδικας του και τέλος αποδεικνύεται η ορθότητα του όσο αφορά τον τερματισμό του. Στο Κεφάλαιο 5 εξηγείται λεπτομερώς πώς έγινε η υλοποίηση, οι αποφάσεις που πάρθηκαν για αυτήν και πως αναπτύχθηκαν σημαντικά κομμάτια της. Στο Κεφάλαιο 6 παρουσιάζονται τα πειράματα που έχουν γίνει κατά την διάρκεια της εργασίας και τι αποτελέσματα και συμπεράσματα έχουν εξαχθεί. Τέλος, στο Κεφάλαιο 7 περιγράφονται γενικά τα συμπεράσματα για την πτυχιακή εργασία, ως επίσης και μερικές δυσκολίες που αντιμετωπιστήκαν κατά τη διάρκειά της.

**Κεφάλαιο 2**

**Σχετικές εργασίες**

2.1 Στατικό και Γνωστό Περιβάλλον 10

2.2 Στατικό και Άγνωστο Περιβάλλον 10

2.3 Δυναμικό και Γνωστό Περιβάλλον 11

2.4 Δυναμικό και Άγνωστο Περιβάλλον 11

2.5 Γλώσσα Προγραμματισμού 12

Σε μια πλοήγηση ενός ρομπότ σε περιβάλλον 2 διαστάσεων, το περιβάλλον περιέχει ένα σημείο όπου καθορίζει την αρχική θέση του ρομπότ και ένα σημείο όπου αναπαριστά τον τελικό του προορισμό. Επίσης ένας πεπερασμένος αριθμός από εμπόδια τοποθετείται στο περιβάλλον. Το ρομπότ ξεκινά από την αρχική του θέση και καλείται να βρει ένα συνεχόμενο μονοπάτι μέχρι τον προορισμό του αποφεύγοντας τα εμπόδια που πρόκειται να συναντήσει.

Ο στόχος των Bug αλγόριθμων είναι να καθοδηγήσουν το ρομπότ ξεκινώντας από την αρχική του θέση προς τον στόχο δεδομένου πως το ρομπότ δεν έχει καμία πληροφορία για το περιβάλλον. Το ρομπότ θα πρέπει να επιτύχει τον στόχο αυτό με όσο το δυνατό λιγότερες πληροφορίες γίνεται για το περιβάλλον. Στην πράξη αυτό σημαίνει πως το ρομπότ μπορεί να θυμάται μερικά σημεία του περιβάλλοντος αλλά σε καμία περίπτωση δεν μπορεί να δημιουργήσει χαρτογράφηση αυτού. Εάν δεν υπάρχει κάποια διαδρομή από την αρχική θέση προς τον τελικό προορισμό τότε ο αλγόριθμος τερματίζει και ειδοποιεί πως ο προορισμός είναι μη-προσβάσιμος. Αυτή η λειτουργία των Bug αλγόριθμων ονομάζεται τερματισμός [13].

Οι Bug αλγόριθμοι μπορούν να εφαρμοστούν σε οποιοδήποτε ρομπότ εξοπλισμένο με αισθητήρες εμβέλειας και ικανότητα εντοπισμού, όπως για παράδειγμα εντοπισμού ενός σημαντικού σημείου. Στην συνέχεια το ρομπότ είναι σε θέση να ταξιδέψει προς τον τελικό του προορισμό σχεδιάζοντας ένα επιθυμητό μονοπάτι για την επίτευξη του στόχου αυτού. Ο Lumelsky[16, 17, 18] έχει εφαρμόσει στην πράξη την έρευνα των Bug αλγόριθμων, σε ένα ρομποτικό βραχίονα όπου προσπαθούσε ο βραχίονας αυτός να φτάσει σε μια επιθυμητή στάση. Σε αυτήν την περίπτωση οι κινήσεις του βραχίονα είναι πολύ παρόμοιες με την πλοήγηση ενός ρομπότ σε ένα περιβάλλον άγνωστο για αυτό.

Δεδομένου ότι το περιβάλλον είναι άγνωστο, ενδέχεται συνεχώς να αλλάζει συμπεριφορά, δηλαδή πολλά αντικείμενα μέσα σε αυτό να αλλάζουν θέση ενώ παράλληλα λίγες πληροφορίες για αυτό μπορεί να γνωρίζει το ρομπότ ανά πάσα στιγμή. Έτσι η στρατηγική την οποία θα ακολουθήσει το ρομπότ πρέπει να μπορέσει να οδηγήσει το ρομπότ στον τελικό του προορισμό γνωρίζοντας το δυνατό λιγότερο πληροφορίες, που προτιμάται να είναι μόνο η αρχική του θέση και η θέση του τελικού του προορισμού. Μερικές γνωστές μέθοδοι για σχεδίαση διαδρομής για κινητά ρομπότ είναι ο A\* [8, 25, 32], ο αλγόριθμος Dijkstra [5], μετασχηματισμός απόστασης (distance transformation) [4, 7, 10], και η μέθοδος των πιθανών πεδίων (potential fields) [2, 30]. Το πρόβλημα των μεθόδων και των αλγόριθμων αυτών είναι πως απαιτούν περισσότερες πληροφορίες για το περιβάλλον, ακόμη και ολόκληρες τις πληροφορίες για αυτό.

Ο Kim, ο Russell και ο Koo σχεδίασαν τον αλγόριθμο SensBug για χωματουργικές εργασίες στον τομέα της κατασκευαστικής βιομηχανίας (construction industry) [33]. Επιπρόσθετα σημείωσαν την ανάγκη για βελτίωση της νοημοσύνης για τα ρομπότ σε επικίνδυνα περιβάλλοντα όπως σε περιοχές με ραδιενέργεια ή με χημικές ουσίες.

Οι Langer, Coelho και Oliveira [9] παρατήρησαν πως υπάρχει αυξημένη ανάγκη για σχεδίαση αλγόριθμων όπου θα επιτυγχάνουν σχεδίαση διαδρομής για ρομπότ σε άγνωστα περιβάλλοντα με χρήση την μεταφορά και αποθήκευση προϊόντων, ακόμα και για στρατιωτικέ εφαρμογές και για ηλεκτρονικά παιχνίδια [11, 26]. Προσομοίωσαν έναν Bug αλγόριθμο και τον εκτέλεσαν σε περιβάλλον όπου έμοιαζε με ένα γραφείο και μέσα από τα αποτελέσματα που πήρανε διαπίστωσαν πως οι διαδρομές που σχεδιάζονταν ήτανε πολύ ανταγωνιστικές ως προς το μήκος του με αυτές όπου παράγει o A\*.

Προηγούμενες εργασίες όπου αφορούν το πρόβλημα αυτό έγιναν από τον Lumelsky [12, 13, 14, 15, 19, 29]. Πριν από τις εργασίες που είχε κάνει ο Lumelsky, η σχεδίαση διαδρομής και η πλοήγηση ενός ρομπότ σε άγνωστο περιβάλλον αποτελούταν από αλγόριθμους Που ήταν σχεδιασμένοι για λαβύρινθους όπως για παράδειγμα ο αλγόριθμος Pledge [1] και o Tarry’s αλγόριθμος [23]. Η απόδοση των 2 τελευταίων αλγόριθμων, ως προς το μήκος της διαδρομής, μπορούν να είναι πολύ μεγάλες. Επίσης υπήρχαν και οι heuristical μέθοδοι [3], όμως προϋπόθεταν το ρομπότ να έχει γνώση του περιβάλλοντος για μιας περιορισμένης περιοχής γύρω από αυτό.

Οι Bug αλγόριθμοι ήταν οι πρώτοι μη heuristic αλγόριθμοι για την σχεδίαση διαδρομής για κινητό ρομπότ σε ένα άγνωστο περιβάλλον οι οποίοι εγγυούνται τερματισμό. Επιπλέον το ρομπότ δεν χρειάζεται να δημιουργήσει ένα χάρτη για το περιβάλλον, να το χαρτογραφήσει δηλαδή. Το γεγονός αυτό καθιστά τους Bug αλγόριθμους κατάλληλους για υλοποίηση και εκτέλεση σε έναν πραγματικό κόσμο. Στην πορεία, ο Lumelsky και ο Skewis επέκτειναν την εργασία αυτή συμπεριλαμβάνοντας και αισθητήρες απόστασης [20], με τους οποίους το ρομπότ μπορεί να ανιχνεύσει σημαντικά σημεία για να μπορεί να πάρει κατάλληλες αποφάσεις όπου θα οδηγήσουν στην δημιουργία ενός μικρότερου μονοπατιού.

**2.1 Στατικό και Γνωστό Περιβάλλον**

Σε ένα στατικό και γνωστό περιβάλλον , το ρομπότ γνωρίζει όλες τις πληροφορίες του πριν ξεκινήσει το ταξίδι του από την αρχική προς τον τελικό του προορισμό. Επομένως ένας αλγόριθμος για στατικά και γνωστά περιβάλλοντα, μπορεί να υπολογίσει και να σχεδιάσει μια βέλτιστη διαδρομή για το κινητό ρομπότ πριν ξεκινήσει αυτό την πλοήγησή του και χωρίς να χρειάζεται το ρομπότ να κάνει οπουδήποτε υπολογισμό όσο κινείται. Η κύρια μέθοδος που χρησιμοποιείται στους αλγόριθμους για σχεδίαση διαδρομών για ρομπότ σε στατικά και γνωστά περιβάλλοντα είναι η χρήση γράφων

Σε αυτήν την μέθοδο, για την σχεδίαση διαδρομών για στατικά και γνωστά περιβάλλοντα, χρησιμοποιούνται γράφοι. Η κύρια ιδέα της μεθόδου αυτής είναι ότι βασίζεται στις κορυφές των εμποδίων και με βάση αυτών δημιουργεί ένα μονοπάτι από την αρχική θέση προς τον τελικό προορισμό περνώντας από τις κορυφές των εμποδίων. Ένας γράφος αποτελείται από κόμβους και ακμές. Οι κόμβοι αναπαριστούν την αρχική θέση του ρομπότ, τον τελικό προορισμό και όλες τις κορυφές όλων των εμποδίων. Από την άλλη οι ακμές είναι ευθείες γραμμές όπου ενώνουν 2 κόμβους και δεν διαπερνάτε από κάποιο εμπόδιο.

**2.2 Στατικό και Άγνωστο Περιβάλλον**

Σε ένα στατικό και άγνωστο περιβάλλον η σχεδίαση διαδρομής είναι δυσκολότερη από αυτή για στατικά για γνωστά περιβάλλοντα. Αυτό οφείλεται για την αβεβαιότητα που υπάρχει στο ρομπότ όσο αφορά το περιβάλλον στο οποίο πρόκειται να ταξιδέψει, με αποτέλεσμα η εύρεση της βέλτιστης λύσης, ως προς το μήκος της διαδρομής, που θα ακολουθήσει το ρομπότ ναι γίνεται δυσκολότερη. Για το σκοπό αυτό το ρομπότ χρησιμοποιεί τοπικές πληροφορίες όπου παίρνει μέσα από διάφορους αισθητήρες και μέσω αυτών κάνει και διάφορους υπολογισμούς για να καταφέρει να φτάσει στο τελικό του προορισμό αποφεύγοντας τα εμπόδια.

**2.3 Δυναμικό και Γνωστό Περιβάλλον**

Οι έρευνες όσο αφορά τις μεθόδους που αφορούν την σχεδίαση μιας διαδρομής, για στατικά περιβάλλοντα έχουν αποφέρει πολύ καλά αποτελέσματα και έχουν αναπτυχθεί αρκετές μέθοδοι όπου να χρησιμοποιούν διάφοροι αλγόριθμοι όπου σχεδιάζουν την βέλτιστη διαδρομή για ένα ρομπότ.

Όμως πιο πρακτικά ένα ρομπότ αφού ταξιδεύει από μια αρχική θέση προς ένα προορισμό, συχνά ίσως συναντήσει εμπόδια που να μην είναι στατικά στο περιβάλλον, αλλά κινούνται σε αυτό. Έτσι έχουμε κάποια εμπόδια όπου συνεχώς αλλάζουν θέση μέσα στο περιβάλλον με αποτέλεσμα να αλλάζουν και οι πληροφορίες που έχει το ρομπότ για το περιβάλλον αυτό. Τα κινητά εμπόδια σε ένα δυναμικό περιβάλλον αυξάνουν κατά πολύ την σχεδίαση διαδρομής των ρομπότ.

Η πολυπλοκότητα και αβεβαιότητα αυξάνονται με τον αριθμό των δυναμικών εμποδίων έτσι παραδοσιακοί αλγόριθμοι σχεδίασης διαδρομών όπως γράφοι, μέθοδοι πλεγμάτων και διαγράμματα Voronoi δεν λειτουργούν καλά σε δυναμικά περιβάλλοντα. Επίσης χρησιμοποιώντας τις μεθόδους αυτές είναι δύσκολη εύρεση του βέλτιστου μονοπατιού σε δυναμικά περιβάλλοντα.

**2.4 Δυναμικό και Άγνωστο Περιβάλλον**

Ο σχεδιασμός διαδρομής σε δυναμικό και άγνωστο περιβάλλον είναι η πιο περίπλοκη περίπτωση στην σχεδίαση διαδρομών για κινητά ρομπότ, όμως από την άλλη είναι και η πιο συνηθισμένη στον αληθινό κόσμο για ένα κινητό ρομπότ.

Λόγω της περιπλοκότητας και της άγνοιας που έχει το ρομπότ για το περιβάλλον δεν μπορεί να δημιουργήσει για αρχή μια διαδρομή όπου θα βασιστεί σε αυτήν για την επίτευξη του στόχου του. Επομένως η εύρεση της βέλτιστης λύσης μονοπατιού είναι αρκετά δύσκολη. Το ρομπότ πρέπει να χρησιμοποιεί αισθητήρες για να λαμβάνει τις κατάλληλες πληροφορίες που αφορούν το περιβάλλον στο οποίο ταξιδεύει και η σχεδίαση της διαδρομής γίνεται ενόσω αυτό κινείται. Ο χρόνος στον οποίο το ρομπότ θα πρέπει να λαμβάνει τις κατάλληλες αποφάσεις για την σχεδίαση της διαδρομής θα πρέπει να είναι πολύ σύντομος αφού ήδη σπαταλάει αρκετό χρόνο για να προσαρμόσει την κίνηση του για την αποφυγή των εμποδίων.

**2.5 Γλώσσα Προγραμματισμού**

Στην παρούσα πτυχιακή εργασία η υλοποίηση της πλατφόρμας που να υποστηρίζει διάφορους αλγόριθμους για σχεδίαση διαδρομής για κινητά ρομπότ, είναι βασισμένη στην γλώσσα προγραμματισμού Java ενώ χρησιμοποιήθηκαν και αρκετές βιβλιοθήκες από την συγκεκριμένη γλώσσα προγραμματισμού όπως για παράδειγμα java.util, java.swing, java.awt.

**2.5.1 Java**

Ως αντικειμενοστρεφής γλώσσα, η Java εστιάζει τη δραστηριότητα του προγραμματιστή στον ορισμό αντικειμένων και λειτουργιών πάνω σε αυτά. Στη Java ιδιαίτερη σημασία έχει η έννοια της κλάσης. Μια κλάση περιγράφει µία συλλογή δεδομένων καθώς και τις λειτουργίες που αυτά επιδέχονται. Κάθε κλάση προέρχεται από κάποια άλλη ήδη ορισμένη κλάση µέσω κληρονομικότητας. Κατ’ αυτό τον τρόπο, ορίζεται µία ιεραρχία κλάσεων, η οποία έχει στην κορυφή της τη βασική κλάση Object. Μια κλάση είναι απλά µία περιγραφή της μορφής των αντικειμένων τα οποία περιέχει. Αντικείμενα μιας κλάσης μπορούν να οριστούν και να χρησιμοποιηθούν σε ένα πρόγραμμα Java. Κάθε αντικείμενο δημιουργείται κατά τη διάρκεια εκτέλεσης ενός προγράμματος Java και υπάρχει μέχρι να καταστραφεί.

**2.5.2 Βιβλιοθήκες**

Η Java προσφέρει αρκετές βιβλιοθήκες για έναν προγραμματιστή όπου του δίνει την δυνατότητα να τις χρησιμοποιεί με αποτέλεσμα να μειώνει τον φόρτο εργασίας του προγραμματιστή ενώ οι αλγόριθμοι όπου χρησιμοποιούν οι βιβλιοθήκες είναι βέλτιστοι με αποτέλεσμά οι προγραμματιστές να έχουν και καλή επίδοση ως προς τον χρόνο στα προγράμματα τους. Μερικές βιβλιοθήκες που προσφέρει η Java είναι:

* *java.lang* Προσφέρει διάφορες χρήσιμες λειτουργίες γενικής χρήσεως
* *java.util* Προσφέρει δομές δεδομένων, αλγορίθμους, κανονικές εκφράσεις, κ.ά.
* *java.io* Προσφέρει δυνατότητες αποθήκευσης και ανάκτησης αντικειμένων
* *java.sql* Διασύνδεση με σχεσιακές βάσεις δεδομένων
* *java.swing* Δημιουργία γραφικών περιβαλλόντων επικοινωνίας
* *java.awt* Σχεδίαση γραφικών και εικόνων

**Κεφάλαιο 3**

**Υλοποιημένοι Αλγόριθμοι**

3.1 Αλγόριθμος Bug1 15

3.2 Αλγόριθμος Bug2 17

3.3 Αλγόριθμος Alg1 20

3.4 Αλγόριθμος Alg2 21

3.5 Αλγόριθμος Class1 23

3.6 Αλγόριθμος Com 25

3.7 Συμπεράσματα 27

Στο κεφάλαιο αυτό παρουσιάζονται λεπτομερώς όλοι οι αλγόριθμοι που επιλέχθηκαν να υλοποιηθούν κατά τη διάρκεια της πτυχιακής μου εργασίας. Στόχος του συγκεκριμένου κεφαλαίου είναι η πλήρης κατανόηση όλων των αλγορίθμων καθώς επίσης και η κατανόηση όλων των μειονεκτημάτων που έχει ο καθένας από αυτούς. Υπάρχει μια σύντομη περιγραφή για κάθε αλγόριθμο ενώ ακολουθεί και ο αντίστοιχος ψευδοκώδικας. Στο τέλος κάθε περιγραφής αναλύονται τα θετικά και τα αρνητικά στοιχεία, και με ποιον τρόπο ο κάθε αλγόριθμος επέλυσε μειονεκτήματα προηγούμενων αλγόριθμων. Τέλος παρουσιάζεται μια εικόνα από κάθε αλγόριθμο, για καλύτερη κατανόηση, όπου δείχνει την διαδρομή που ακολουθεί το ρομπότ σε κάποιο περιβάλλον.

Μιας και οι αλγόριθμοι που επιλέχθηκαν για ανάλυση και υλοποίηση είναι από την οικογένεια των Bug αλγόριθμων αξίζει να σημειωθεί πως οι συγκεκριμένοι αλγόριθμοι είναι εμπνευσμένοι από τις ικανότητες των εντόμων και κάνουν κάποιες υποθέσεις όπου είναι αναγκαίες για την εκτέλεση των αλγόριθμων.

1. Το ρομπότ γνωρίζει την αρχική του θέση και την θέση του τελικού του προορισμού
2. Το ρομπότ είναι εξοπλισμένο με αισθητήρες μικρής εμβέλειας που μπορεί να ανιχνεύσει ένα εμπόδιο από πολύ μικρή απόσταση.
3. Τα εμπόδια είναι άγνωστα
4. Το ρομπότ έχει έναν αισθητήρα που του επιτρέπει να γνωρίζει πάντα την κατεύθυνση και την ευκλείδεια απόσταση από το στόχο.
5. Υπάρχει πεπερασμένος αριθμών εμποδίων

Επίσης οι αλγόριθμοι Bug όταν συναντήσουν κάποιο εμπόδιο κινούνται γύρω από αυτό έχοντας το εμπόδιο στα δεξιά του, όπου αυτή η μέθοδος ονομάζεται μέθοδος της δεξιάς παλάμης.

**3.1 Αλγόριθμος Bug1**

Ο αλγόριθμος Bug1 είναι ο πρώτος αλγόριθμος στην οικογένεια των bug algorithms με δημιουργούς τους Lumelsky και Stepanov [13, 14, 15]. Είναι γενικότερα ο πρώτος στην κατηγορία των αλγορίθμων αποφυγής εμποδίων. Είναι εύχρηστος και εύκολος στη χρήση, αλλά κάποιες φορές κατευθύνει το ρομπότ πολύ μακριά από τον στόχο. Σε αυτόν τον αλγόριθμο, το ρομπότ αφού εντοπίσει ένα εμπόδιο αρχίζει και κινείται γύρω από αυτό μέχρι να φτάσει στο σημείο που το εντόπισε για πρώτη φορά. Ταυτόχρονα υπολογίζει την απόσταση από την τρέχουσα θέση μέχρι τον τελικό προορισμό και εν τέλει αποθηκεύει το σημείο το οποίο βρίσκεται στην μικρότερη απόσταση από τον στόχο. Αφού ολοκληρώσει έναν κύκλο το ρομπότ, το σημείο αυτό ορίζεται ως το σημείο από το οποίο το ρομπότ θα ξεκινήσει την διαδρομή του. Οπότε το ρομπότ ξεκινά πάλι τον κύκλο γύρω από το εμπόδιο μέχρι να φτάσει στο σημείο αυτό. Αφού αφήσει πίσω του το εμπόδιο, το ρομπότ υπολογίζει τη νέα διαδρομή από το σημείο αυτό προς τον τελικό προορισμό.

Παρακάτω παρουσιάζεται ο αλγόριθμος Bug1 σε ψευδοκώδικα.

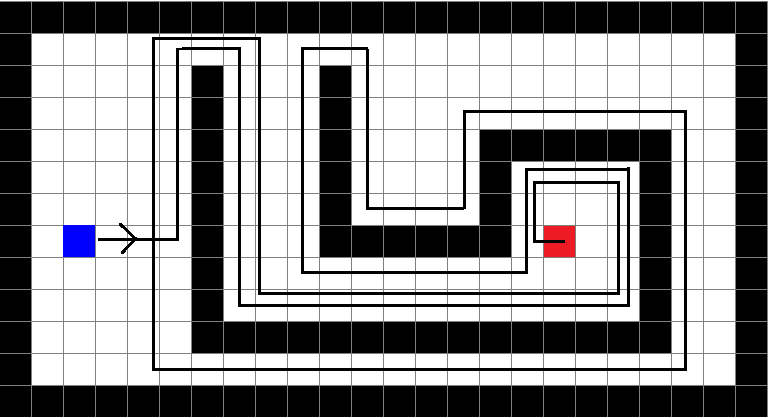
1. Θέσε i=0
2. Κατευθύνσου προς τον τελικό προορισμό, μέχρις ότου:

* Φτάσεις στον προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Όρισε το σημείο επαφής Hi . Πήδα στο βήμα 3.

1. Κινήσου κυκλικά του σημείου επαφής σύμφωνα με την φορά των δεικτών του ρολογιού και όσο κινείσαι ανανέωνε συνεχώς το Li με την πιο κοντινή απόσταση στον προορισμό μέχρι εκείνη τη στιγμή, μέχρις ότου:
   * Φτάσεις στον προορισμό. Ο αλγόριθμος σταματάει.
   * Έρθεις σε επαφή με το σημείο επαφής Hi ξανά. Έλεγξε εάν από το σημείο Li ο τελικός στόχος είναι προσβάσιμος:

* Εάν δεν ισχύει, τότε ο στόχος δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.
* Εάν ισχύει, τότε γύρισε στην τελευταία πιο κοντινή στον προορισμό θέση. Αύξησε το i κατά 1. Πήδα στο βήμα 2.

Συνοπτικά λοιπόν, ο αλγόριθμος Bug1 σαρώνει το κάθε εμπόδιο που εντοπίζει για το σημείο που βρίσκεται πιο κοντά στον προορισμό. Όταν αυτή η θέση οριστεί, το ρομπότ ελέγχει αν μπορεί να συνεχίσει προς τον προορισμό ή όχι. Εάν δεν μπορεί, τότε ο προορισμός δεν είναι προσβάσιμος. Εάν μπορεί, τότε το ρομπότ γνωρίζει πως όταν φύγει από αυτό το συγκεκριμένο σημείο δεν θα ξαναέρθει σε επαφή με το συγκεκριμένο εμπόδιο. Ο αλγόριθμος Bug1 όμως έχει σαν μειονέκτημα τις μεγάλες διαδρομές που σχεδιάζει μέχρι να φτάσει στον προορισμό. Ο λόγος είναι επειδή σε κάθε εμπόδιο που συναντά το ρομπότ, κάνει μια πλήρη περιστροφή από αυτό, πράγμα που το κάνει πολύ χρονοβόρο ενώ παράλληλα κάνει τις ίδιες διαδρομές πάνω από μια φορά για κάθε αποφυγή εμποδίου. Στο παρακάτω σχήμα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.

Σχήμα 3.1: Παράδειγμα αλγορίθμου Bug1

**3.2 Αλγόριθμος Bug2**

Όπως ο αλγόριθμος Bug1 έτσι και αυτός είναι δημιούργημα των Lumelsky και Stepanov [13, 14, 15]. Είναι λίγο πιο εξελιγμένος (συντηρητικός) από τον Bug1 επειδή το ρομπότ λόγω της Μ- γραμμής μπορεί και αποφεύγει την άσκοπη περιστροφή γύρω από το εμπόδιο όταν βρει το πιο κοντινό σημείο στον προορισμό. Ο Bug2 αλγόριθμος δημιουργεί την αρχική διαδρομή από την εκκίνηση έως τον προορισμό και αποθηκεύει τον συντελεστή κατεύθυνσης κάθε φορά καθώς κινείται. Η συμπεριφορά του αλγορίθμου αλλάζει όταν βρίσκει εμπόδιο. Πραγματοποιεί τον γύρο του εμποδίου και υπολογίζει την κλίση της ευθείας από την τρέχουσα θέση μέχρι τον στόχο. Όταν η κλίση γίνει ίδια με την κλίση του αρχικού υπολογισμού, το ρομπότ αλλάζει συμπεριφορά και μετακινείται προς στον στόχο. Έτσι το ρομπότ καταφέρνει και ακολουθεί μη επαναλαμβανόμενη διαδρομή κατά την διάρκεια της κίνησης του. Ο Bug2 αλγόριθμος είναι πιο αποτελεσματικός από τον Bug1, επειδή επιτρέπει στο ρομπότ να φτάσει στον στόχο σε λιγότερο χρόνο.

Παρακάτω παρουσιάζεται ο αλγόριθμος Bug2 σε ψευδοκώδικα.

Όρισε με T με την θέση του τελικού προορισμού.

1. Σχεδίασε μια νοητή γραμμή Μ, από την αρχική θέση μέχρι τον τελικό προορισμό και θέσε το i=0.

2) Αύξησε το i κατά 1 και ακολούθησε την γραμμή μέχρι τον τελικό προορισμό, μέχρις ότου:

• Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.

• Βρεις εμπόδιο. Όρισε το σημείο επαφής Hi . Πήδα στο βήμα 3.

3) Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του, μέχρις ότου:

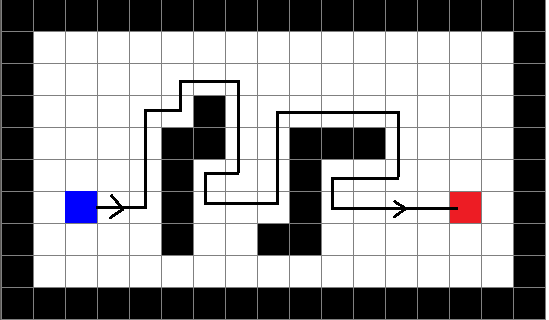
• Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.

• Βρεις σημείο κατά μήκος της Μ-γραμμής, όπου ισχύει d(x,T) < d (Hi,T), όπου d(X1,X2) ορίζεται η ευκλείδεια απόσταση μεταξύ των σημείων X1 και X2. Εάν το ρομπότ μπορεί να κινηθεί προς τον προορισμό, όρισε αυτό το σημείο ως Li. Πήδα στο βήμα 2. Αλλιώς, αντικατέστησε το d(Hi,T) με το d(x, T).

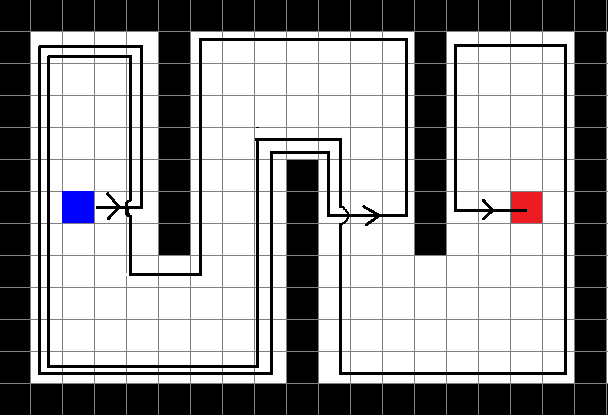
• Επιστρέψεις στο Hi. Ο προορισμός δεν είναι προσβάσιμος. Ο

αλγόριθμος σταματάει.

Εν κατακλείδι, ο αλγόριθμος Bug2 ξεκινώντας την πλοήγηση του δημιουργεί μία ευθεία γραμμή (Μ-Γραμμή) από την αρχική του θέση προς τον τελικό προορισμό όπου και το ρομπότ κινείται πάνω σε αυτήν την γραμμή. Μόλις συναντήσει ένα εμπόδιο τότε κινείται αριστερόστροφα από αυτό και μόλις συναντήσει ξανά την Μ-Γραμμή τότε μπορεί να φύγει προς τον τελικό προορισμό ξανά. Αυτό είναι μια πολύ σημαντική βελτίωση σε σχέση με τον Bug1 όμως και πάλι ο αλγόριθμος Bug2 έχει κάποια μειονεκτήματα αφού υπάρχουνε σενάρια όπου το ρομπότ επαναλαμβάνει διαδρομές μέχρι να φτάσει στον προορισμό του. Στα παρακάτω σχήματα βλέπουμε κάποια απλά παραδείγματα του αλγορίθμου.



Σχήμα 3.2: Παράδειγμα αλγορίθμου Bug2

Π f

Σχήμα 3.3: Παράδειγμα αλγορίθμου Bug2 με επαναλαμβανόμενη διαδρομή

**3.3 Αλγόριθμος Alg1**

Ο αλγόριθμος Alg1 είναι μία επέκταση του αλγόριθμου Bug2 και δημιουργοί του είναι ο Sankaranarayanan και Vidyasagar [27]. Η αδυναμία του Bug2 είναι ότι μπορεί να διασχίσει το ίδιο μονοπάτι περισσότερες από μία φορά και έτσι δημιουργεί μεγαλύτερες διαδρομές. Ο Alg1 διορθώνει αυτήν την αδυναμία καθώς θυμάται τις προηγούμενες θέσεις σύγκρουσης και διαφυγής και τις χρησιμοποιεί για να δημιουργήσει μικρότερα μονοπάτια.

Παρακάτω παρουσιάζεται ο αλγόριθμος Alg1 σε ψευδοκώδικα.

1. Σχεδίασε μια νοητή γραμμή Μ, από την αρχική θέση μέχρι τον τελικό προορισμό και θέσε το i=0.
2. Αύξησε το i κατά 1 και ακολούθησε τη γραμμή μέχρι τον τελικό προορισμό, μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Όρισε το σημείο επαφής Hi. Πήδα στο βήμα 3.

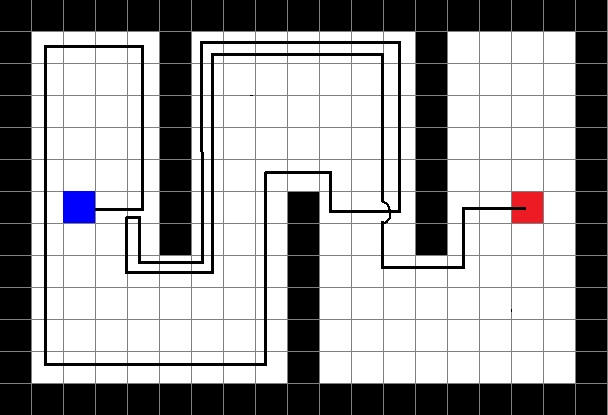
1. Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του, μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις σημείο y το οποίο:
* να βρίσκεται πάνω στην Μ-γραμμή.
* να ισχύει ότι d(y,T)<d(x,T) για όλα τα x που ανήκουν στην Μ- γραμμή και έχει ήδη επισκεφτεί το ρομπότ .
* το ρομπότ να είναι σε κατάλληλη θέση, ώστε να μπορεί να κινηθεί προς τον προορισμό.

Όρισε αυτό το σημείο ως Li. Πήδα στο βήμα 2.

* Φτάσεις σε μία προηγούμενη θέση Hj ή Lj, όπου ισχύει j<i. Γύρνα πίσω στο Hi. Όταν φτάσεις στο Hi, ακολούθησε αντίθετα τα όρια του εμποδίου. Αυτός ο κανόνας δεν μπορεί να ισχύσει ξανά άμα δεν οριστεί το Li.
* Επιστρέψεις στο Hi. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

Συνοπτικά ο αλγόριθμος Alg1 κάνει και αυτός χρήση της Μ-Γραμμής και με το να θυμάται και να αποθηκεύει τις προηγούμενες θέσεις σύγκρουσης και φυγής αποφεύγει τις επαναλαμβανόμενες διαδρομές και έτσι βελτιστοποιεί την απόδοση του σε σχέση με τον αλγόριθμο Bug2. Όμως ο περιορισμός που έχει να χρησιμοποιεί την Μ-Γραμμή κάνει τον αλγόριθμο πιο ευάλωτο ως προς το μέγεθος του μονοπατιού που διασχίζει. Στο παρακάτω σχήμα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Σχήμα 3.4: Παράδειγμα αλγορίθμου Alg1

**3.4 Αλγόριθμος Alg2**

Ο αλγόριθμος Alg2 είναι μία βελτιωμένη έκδοση του Alg1 με δημιουργούς και πάλι τους Sankaranarayanan και Vidyasagar [28]. Το ρομπότ παύει να λειτουργεί βάση της Μ-γραμμής και πλέον υιοθετείται νέα συνθήκη.

Παρακάτω παρουσιάζεται ένα απλό παράδειγμα ψευδοκώδικα του αλγορίθμου Alg2.

1. Θέσε i=0.
2. Αύξησε το i κατά ένα και προχώρησε προς τον προορισμό ανανεώνοντας συνεχώς το Q με d(x,T) εάν ισχύει ότι Q > d(x,T). Άρα τώρα το Q συμβολίζει την κοντινότερη απόσταση που έχει βρεθεί το ρομπότ από τον προορισμό του.

Επανάλαβε αυτό το βήμα μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Όρισε το σημείο επαφής Hi. Πήδα στο βήμα 3.

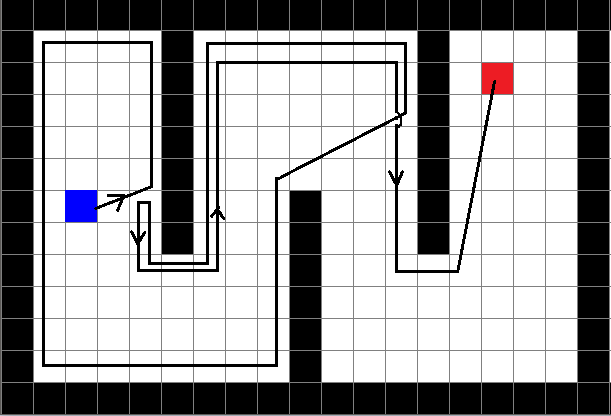
1. Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του και ανανέωνε συνεχώς το Q με d(x,T) εάν ισχύει ότι Q > d(x,T), μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις σημείο y:
  + - Να ισχύει, ότι d(y,T) < Q και
    - το ρομπότ να είναι σε κατάλληλη θέση, ώστε να μπορεί να κινηθεί προς τον προορισμό.

Όρισε αυτό το σημείο ως Li. Πήδα στο βήμα 2.

* Φτάσεις σε μία προηγούμενη θέση Hj ή Lj, όπου ισχύει j<i. Γύρνα πίσω στο Hi. Όταν φτάσεις στο Hi, ακολούθησε τα όρια του εμποδίου. Αυτός ο κανόνας δεν μπορεί να ισχύσει ξανά αν δεν οριστεί το Li.
* Επιστρέψεις στο Hi. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.

Η νέα συνθήκη αυτού του αλγορίθμου είναι μία ιδανική βελτίωση, καθώς το ρομπότ δεν χρειάζεται να βρίσκεται στην Μ-γραμμή για να αποφύγει το εμπόδιο. Το γεγονός αυτό βελτιώνει την επίδοση του ρομπότ ως προς την διαδρομή που θα ακολουθήσει με αποτέλεσμα να την κάνει πιο αποτελεσματική. Παράλληλα ο αλγόριθμος Alg2 εκτός από μια βελτίωση του Alg1 είναι και βελτίωση του Class1 αφού πλέον τα σημεία σύγκρουσης και φυγής αποθηκεύονται και έτσι δεν επαναλαμβάνονται ίδια μονοπάτια. Στην παρακάτω εικόνα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Σχήμα 3.6: Παράδειγμα αλγορίθμου Alg2

**3.5 Αλγόριθμος Class1**

O αλγόριθμος Class1 [6] δεν εγγυάται τερματισμό. Χρησιμοποιείται για να μας δείχνει τι συμβαίνει όταν το ρομπότ έχει την άδεια να φύγει αν βρίσκεται ήδη στο κοντινότερο σημείο από τον προορισμό και μπορεί να κατευθυνθεί προς αυτόν. Ο Class1 γενικότερα χρησιμοποιείται για την ανάπτυξη των υπόλοιπων Bug αλγόριθμων και για να σημάνει την χρησιμότητα των ειδικών κανόνων που επιτρέπουν στο ρομπότ να φύγει από την εκάστοτε θέση που βρίσκεται.

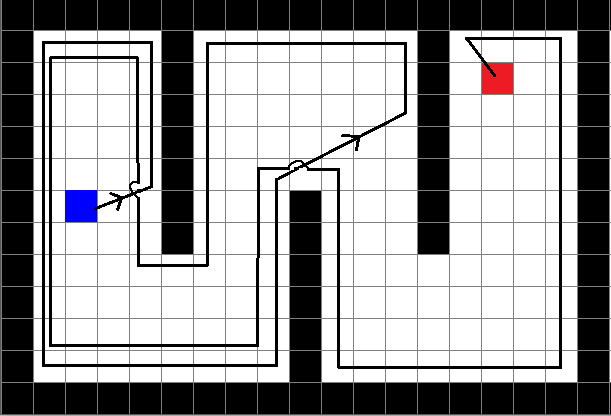
Παρακάτω παρουσιάζεται ο αλγόριθμος Class1 σε ψευδοκώδικα.

1. Κινήσου προς τον προορισμό, μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Ακολούθησε αριστερόστροφα τα όρια του εμποδίου. Πήδα στο βήμα 2.

1. Φύγε εάν το ρομπότ μπορεί να κατευθυνθεί προς τον προορισμό και είναι πιο κοντά σε αυτόν από τα τις προηγούμενες θέσεις που πέρασε. Πήδα στο βήμα 1.

Ο αλγόριθμος Class1 εκτός από το μεγάλο μειονέκτημα που έχει, να μην εγγυάται τερματισμό, σε συγκεκριμένα περιβάλλοντα επαναλαμβάνει διαδρομές, όπως φαίνεται και στο σχήμα 3.5. Να αναφερθεί πως ο αλγόριθμος Class1 δεν χρησιμοποιεί την Μ-Γραμμή, πράγμα που δεν περιορίζει το σημείο διαφυγής του από τα εμπόδια με αποτέλεσμα να αποφέρει συντομότερα μονοπάτια για απλά περιβάλλοντα. Τέλος ο Class1 βελτιώνει τον Com μιας και πλέον για να οριστεί το σημείο διαφυγής μπαίνει ένας νέος όρος που λέει το ρομπότ πρέπει να είναι και πιο κοντά στον προορισμό από όλες τις προηγούμενες θέσεις που έχει περάσει, γεγονός που. Στο παρακάτω σχήμα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.



Σχήμα 3.5: Παράδειγμα αλγορίθμου Class1

**3.6 Αλγόριθμος Com**

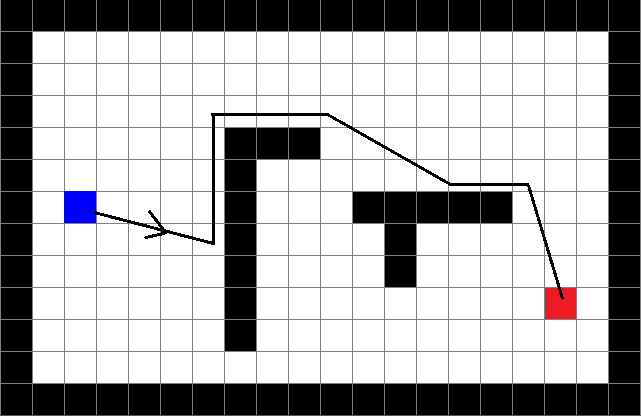
Ο αλγόριθμος Com [28] δεν είναι επίσημος αλγόριθμος Bug και δεν εγγυάται τερματισμό. Αντ’ αυτού, χρησιμοποιείται για να απεικονίσει τι συμβαίνει όταν το ρομπότ μπορεί να φύγει για το στόχο όποτε είναι σε θέση να το κάνει. Ο Αλγόριθμος Com χρησιμοποιείται για την ανάπτυξη των Bug αλγόριθμων και να αιτιολογήσει γιατί πρέπει να υπάρχουν κανόνες φυγής.

1) Κινήσου προς τον προορισμό, μέχρις ότου:

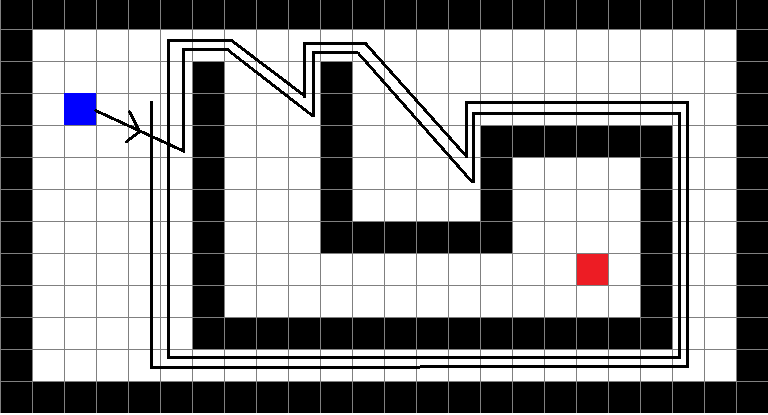
* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Ακολούθησε αριστερόστροφα τα όρια του εμποδίου. Πήδα στο βήμα 2.

2) Φύγε εάν το ρομπότ μπορεί να κατευθυνθεί προς τον προορισμό. Πήδα στο βήμα 1.

Συνοπτικά, ο αλγόριθμος Com είναι ένας αλγόριθμος όπου στην τελική αναπτύχθηκε για να μας δείξει πόσο πολύ έμφαση πρέπει να δώσουμε στους κανόνες όπου χαρακτηρίζουν κάποιο σημείο ως σημείο φυγής ούτως ώστε να αποφύγουμε τον μη τερματισμό του αλγόριθμου καθώς επίσης και τις επαναλαμβανόμενες διαδρομές. Τα μειονεκτήματα αυτά επιλύθηκαν όπως είδαμε προηγουμένως στους αλγόριθμους Class1 και Alg2. Πιο κάτω βλέπουμε 2 απλά παραδείγματα του αλγορίθμου.



Σχήμα 3.7: Παράδειγμα αλγορίθμου Com



Σχήμα 3.8: Παράδειγμα αλγορίθμου Com που δεν τερματίζει

**3.7 Συμπεράσματα**

Μέσα από μια λεπτομερή περιγραφή των παραπάνω αλγόριθμων μπορούμε να πούμε κάποια συγκριτικά σχόλια για αυτούς. Καταρχάς ο αλγόριθμος Com και ο αλγόριθμος Class1 δεν εγγυόνται τερματισμό άρα το γεγονός αυτό τους καθιστά ως τους χειρότερους. Επίσης βλέπουμε πως ο αλγόριθμος Bug1 δεν είναι πολύ αποδοτικός ως προς το μέγεθος του μονοπατιού που ακολουθεί το ρομπότ, αφού χρειάζεται πρώτα να κάνει μια πλήρη περιστροφή από κάθε εμπόδιο, γεγονός που επιφέρει στην σχεδίαση πολύ μεγάλων αποστάσεων μέχρι το ρομπότ να καταφέρει να φτάσει στο τελικό του προορισμό. Αν και μη αποδοτικός ο αλγόριθμος Bu1, εγγυάται όμως σίγουρο τερματισμό. Στην συνέχεια ο αλγόριθμος Bug2, βελτιώνει αυτό το μειονέκτημα του Bug1 αφού με την προσθήκη της Μ-γραμμής το ρομπότ δεν χρειάζεται να κάνει μια πλήρη στροφή από κάθε εμπόδιο. Ο Bug2 όμως παρουσιάζει σαν μειονέκτημα την δημιουργία ίδιων μονοπατιών σε μερικές περιπτώσεις. Έτσι έρχεται ο αλγόριθμος Alg1 όπου διορθώνει αυτό το μειονέκτημα του Bug2 με το να θυμάται τα σημεία συνάντησης με κάποιο εμπόδιο και τα σημεία διαφυγής. Η χρήση όμως της Μ-γραμμής περιορίζει όμως την κίνηση του ρομπότ σε αυτή και έτσι δημιουργούνται μεγαλύτερα μονοπάτια ως προς το μήκος, πράγμα που έρχεται ο Alg2 και το βελτιώνει αφού πλέον γη χρήση της Μ-Γραμμής δεν γίνεται και το ρομπότ συνεχώς κατευθύνεται προς τον στόχο.

Όλα αυτά όμως δεν είναι εντελώς απόλυτα, αφού υπάρχουν περιπτώσεις όπου κάποιος αλγόριθμος όπου δημιουργήθηκε για βελτίωση κάποιου αλλού, δεν αποδίδει καλύτερα από αυτόν. Αυτά θα αναλυθούν περαιτέρω στο Κεφάλαιο 6.

Τέλος με την υπόθεση πως οι Bug αλγόριθμοι έχουν αισθητήρες μικρής εμβέλειας, σε περίπτωση που χρησιμοποιούσαν πιο ικανούς αισθητήρες αυτό θα μείωνε την απόσταση της διαδρομής που θα εκτελούσαν τα ρομπότ, αφού θα εντόπιζαν τα εμπόδια από μεγαλύτερη απόσταση και επίσης ίσως γνώριζαν και σε μερικές περιπτώσεις τα όρια ενός εμποδίου και θα μπορούσαν να αποφασίσουν πια κατεύθυνση να ακολουθήσουν γύρω από κάθε εμπόδιο αν και πάλι αυτό δεν απόλυτο.

**Κεφάλαιο 4**

**Αλγόριθμος ΜΒack**

4.1 Εισαγωγή 28

4.2 Περιγραφή 29

4.3 Ψευδοκώδικας 31

4.4 Μέθοδος Μ-Γραμμής 33

4.5 Περιμετρική Διάσχιση Εμποδίου 33

4.6 Κατεύθυνση προς τον στόχο 34

4.7 Αποθήκευση σημείων και κινήσεων 34

4.8 Απόδειξη τερματισμού 35

**4.1 Εισαγωγή**

Στο κεφάλαιο αυτό προτείνεται ένας νέος αλγόριθμος πλοήγησης για ρομπότ, o MBack, για άγνωστα περιβάλλοντα ο οποίος αναπτύχθηκε και προτάθηκε από τον συγγραφέα της πτυχιακής εργασίας αυτής. Ο αλγόριθμος αυτός κτίστηκε μέσα από την μελέτη και την υλοποίηση των αλγορίθμων που αναλύθηκαν στο Κεφάλαιο 3. Το όνομα του προέρχεται μέσα από την ιδέα του αλγόριθμου όπου όταν το ρομπότ φτάσει πάνω στην Μ-Γραμμή και είναι πιο μακριά από το στόχο τότε εκεί γίνεται μια εκτέλεση κάποιων λειτουργιών. Η περιγραφή του MBack αλγόριθμου θα εξηγηθεί αναλυτικότερα στην συνέχεια. Για αυτόν τον λόγο και ο συγκεκριμένος αλγόριθμος είναι παρόμοιος με τους αλγορίθμους της οικογένειας Bug. Έτσι μπορούμε να πούμε πως ο αλγόριθμος αυτός είναι μια εξέλιξη των προηγούμενων αλγορίθμων και είναι κυρίως βασισμένος στον Bug2 και τον Alg1 αφού περιέχει κύρια στοιχεία από τους 2 προαναφέροντες αλγόριθμους. Μέσα από την υλοποίηση των αλγορίθμων που αναπτύχθηκαν στο κεφάλαιο 3, διεξάγει μια σειρά από πειράματα, όπου αναλύονται στο επόμενο κεφάλαιο, σε διάφορα περιβάλλοντα με την εμπλοκή όλων των αλγορίθμων με σκοπό να σημειωθούν και να καταγραφούν τα μειονεκτήματα που έχουν οι αλγόριθμοι και να κτιστεί ένας νέος αλγόριθμος όπως αυτός που παρουσιάζεται στο συγκεκριμένο κεφάλαιο. Σκοπός αυτού του νέου αλγόριθμου είναι η εύρεση ενός συντομότερου μονοπατιού, για ρομπότ που κινείται σε ένα άγνωστο περιβάλλον, σε σχέση με τους προηγούμενους αλγορίθμους.

Κύριος άξονας για τη δημιουργία αυτού του αλγόριθμου ήταν η ανάστροφη κίνηση στο ρομπότ ούτως ώστε να γλιτώσει άσκοπες κινήσεις όπου θα επιφέρουν μεγαλύτερη διαδρομή προς τον προορισμό του. Δηλαδή όλη η ιδέα ήταν με ποιο τρόπο ένα ρομπότ ενόσω κινείται να υπάρξει σε κάποιο σημείο ένας κανόνας όπου θα του επιτρέπει με την εκτέλεση κάποιων αντίστροφων κινήσεων που έκανε μέχρι σ’ ένα σημείο να καταφέρει να διανύσει μικρότερο μέγεθος μονοπάτι προς τον προορισμό του.

Τέλος μετά την μελέτη και την εισήγηση του συγκεκριμένου αλγόριθμου έγινε και η υλοποίηση του όπου αυτή να αναλυθεί λεπτομερώς στα επόμενα κεφάλαια.

**4.2 Περιγραφή**

Ο αλγόριθμος που προτείνεται από τον συγγραφέα έχει στοιχεία από τους αλγόριθμους της οικογένειας Bug και πιο συγκεκριμένα από τον Bug2 και Alg1. Ο αλγόριθμος αυτός είναι για άγνωστα περιβάλλοντα αφού το ρομπότ δεν γνωρίζει καθόλου για το τι συμβαίνει στο περιβάλλον όπου πρόκειται να κινηθεί. Επίσης ο αλγόριθμος είναι σχεδιασμένος για δυναμικά περιβάλλοντα, δηλαδή ανά πάσα στιγμή ένα νέο εμπόδιο μπορεί να εμφανιστεί στην πορεία του ρομπότ κατά την διάρκεια της πλοήγησης του.

To μόνο που γνωρίζει το ρομπότ κατά την εκκίνηση του, είναι το σημείο στο οποίο βρίσκεται και το σημείο του τελικού προορισμού. Ο αλγόριθμος κάνει χρήση της Μ-γραμμής κατά την εκτέλεσή του. Ξέροντας το σημείο της αρχικής του θέσης και το σημείο του προορισμού υπολογίζει τον συντελεστή κατεύθυνσης, δημιουργεί μια ευθεία γραμμή από την αρχική θέση προς τον στόχο και τότε ξεκινά η πλοήγησή του στο άγνωστο για αυτό περιβάλλον πάνω στην γραμμή αυτή.

Όταν συναντήσει κάποιο εμπόδιο τότε το ρομπότ αναγκάζεται να εκτελέσει κινήσεις μακριά από την Μ-Γραμμή. Το σημείο αυτό, όπου το ρομπότ βρίσκει κάποιο εμπόδιο, αποθηκεύεται για τυχόν μετέπειτα χρήση του ενώ παράλληλα όλες οι κινήσεις του από εδώ και στο εξής αποθηκεύονται και αυτές. Το ρομπότ ξεκινά να εξερευνά το εμπόδιο που βρήκε διασχίζοντας το περιμετρικά και με τη φορά του ρολογιού. Πλέον ο συντελεστής κατεύθυνσης του, είναι διαφορετικός από τον αρχικό συντελεστή αφού δεν κινείται στην Μ-Γραμμή. Για κάθε κίνηση που κάνει το ρομπότ υπολογίζεται ο νέος συντελεστής με βάση την τρέχουσα θέση του και το τελικό προορισμό. Όταν φτάσει το ρομπότ σε κάποιο σημείο όπου ο συντελεστής κατεύθυνσης είναι ο ίδιος με τον αρχικό συντελεστή τότε το σημείο αυτό ορίζεται σαν σημείο φυγής και το ρομπότ πλέον έχει ξεπεράσει το εμπόδιο όπου βρήκε και εγγυάται πως δεν θα το ξανασυναντήσει μέχρι το τέλος της πλοήγησης του. Επίσης όταν καθοριστεί το σημείο φυγής όλες οι κινήσεις που αποθηκεύονταν από την στιγμή της συνάντησης του με το εμπόδιο μέχρι τώρα διαγράφονται επειδή το ρομπότ ξεπέρασε το συγκεκριμένο εμπόδιο και σημαίνει δεν θα χρειαστεί να τις τρέξει αντίστροφα αυτές τις κινήσεις στο μέλλον. Φτάνοντας τώρα στο σημείο φυγής το ρομπότ, στην Μ-Γραμμή δηλαδή, και αφού μπορεί να προχωρήσει πάνω σε αυτήν, συνεχίζει το ταξίδι προς τον προορισμό έχοντας υπόψη όλα όσα γράφτηκαν προηγουμένως.

Κατά τη διάρκεια όπου το ρομπότ κινείται γύρω από κάποιο εμπόδιο, εάν φτάσει σε κάποιο σημείο όπου η κλίση της τρέχουσας θέσης είναι η ίδια με την αρχική κλίση αλλά η απόσταση από τον προορισμό είναι μεγαλύτερη από την απόσταση του σημείου συνάντησης-σύγκρουσης με το εμπόδιο, και τον τελικό προορισμό εδώ είναι που μπαίνει ο κανόνας για έναρξη αντίστροφης εκτέλεσης των κινήσεων που αποθηκεύονταν από την ώρα της σύγκρουσης μέχρι την παρούσα φάση. Το ρομπότ ξεκινά να εκτελεί αντίθετα όλα τα βήματα μέχρι να φτάσει στο σημείο σύγκρουσης. Όταν φτάσει εκεί ξεκινά να εξερευνά το εμπόδιο από την αντίθετη πλευρά αυτή την φορά, και πάλι υπολογίζοντας τον συντελεστή κατεύθυνσης για κάθε του κίνηση. Όταν πλέον ο τρέχων συντελεστής είναι ίδιος με τον αρχικό, το σημείο αυτό ορίζεται σαν σημείο φυγής και πλέον το ρομπότ έχει ξεπεράσει το εμπόδιο και συνεχίζει να κινείται προς τον τελικό προορισμό του. Εάν όμως φτάσει στο σημείο και πάλι όπου ξεκίνησε η αντίστροφη εκτέλεση των κινήσεων τότε ο προορισμός δεν είναι προσβάσιμος και έτσι ο αλγόριθμος σταματά.

* 1. **Ψευδοκώδικας**

Παρακάτω παρουσιάζεται ο αλγόριθμος σε ψευδοκώδικα

1. Σχεδίασε μια νοητή γραμμή Μ, από την αρχική θέση μέχρι τον τελικό προορισμό και θέσε το i=0.
2. Αύξησε το i κατά 1 και ακολούθησε την γραμμή μέχρι τον τελικό προορισμό, μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Βρεις εμπόδιο. Όρισε το σημείο επαφής Hi. Πήδα στο βήμα 3.

1. Κινήσου αριστερόστροφα γύρω από το εμπόδιο, ακολουθώντας τα όρια του, μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Επιστρέψεις στο Hi. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.
* Βρεις σημείο y το οποίο:
* να βρίσκεται πάνω στην Μ-γραμμή.
* να ισχύει ότι d(x,T) < d(Hi,T)
* το ρομπότ να είναι σε κατάλληλη θέση, ώστε να μπορεί να κινηθεί προς τον προορισμό.

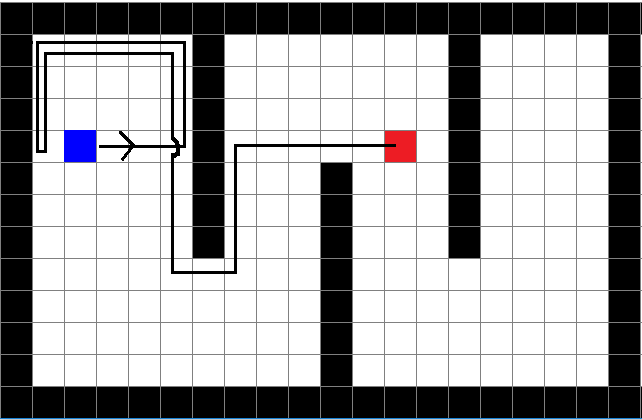
Όρισε αυτό το σημείο ως Li. Πήδα στο βήμα 2.

* Βρεις σημείο y το οποίο:
* να βρίσκεται πάνω στην Μ-γραμμή
* να ισχύει ότι d(x,T) > d(Hi,T)

Όρισε το σημείο αυτό ως R. Επέστρεψε στο Hi και κινήσου από την άλλη πλευρά αυτήν την φορά μέχρις ότου:

* Φτάσεις στο προορισμό. Ο αλγόριθμος σταματάει.
* Επιστρέψεις στο R. Ο προορισμός δεν είναι προσβάσιμος. Ο αλγόριθμος σταματάει.
* να βρίσκεται πάνω στην Μ-γραμμή.
* να ισχύει ότι d(x,T) < d(Hi,T)

Όρισε αυτό το σημείο ως Li. Πήδα στο βήμα 2.

Στο παρακάτω σχήμα βλέπουμε ένα απλό παράδειγμα του αλγορίθμου.

Σχήμα 4.1: Παράδειγμα αλγορίθμου

**4.4 Μέθοδος Μ-Γραμμής**

Ο αλγόριθμος που προτείνεται κάνει χρήση της Μ-Γραμμής. Δημιουργεί μια ευθεία γραμμή από την αρχική θέση S προς τον τελικό προορισμό T. Το ρομπότ κινείται πάνω σε αυτήν την γραμμή καθώς επίσης φεύγει από κάποιο εμπόδιο από την Μ-Γραμμή αντίστοιχα. Η σύγκρουση της ευθείας αυτής με κάποιο εμπόδιο δημιουργεί ένα μοναδικό σημείο. Με την υπόθεση πώς μόνο ένας πεπερασμένος αριθμών συγκρούσεων με την Μ-γραμμή μπορούν να υπάρξουν , τότε αντίστοιχα μόνο ένας πεπερασμένος αριθμών σημείων φυγής μπορούν να υπάρξουν σε όλο το ταξίδι. Έτσι η μέθοδος αυτή εγγυάται πως μετά από κάποιο σημείο συνάντησης με ένα εμπόδιο, υπάρχει και σημείο φυγής από αυτό.

**4.5 Περιμετρική Διάσχιση Εμποδίου**

Για να εφαρμοστεί η μέθοδος της Μ-Γραμμής , ο αλγόριθμος πρέπει να διασφαλίσει πως το ρομπότ πρέπει να αφήσει το εμπόδιο μόνο σε σημεία της Μ-Γραμμής. Εάν αυτό δεν μπορεί να συμβεί τότε σημαίνει πως ο αριθμός των σημείων φυγής από κάποιο εμπόδιο είναι μη πεπερασμένος, γεγονός που δεν κάνει τον τερματισμό του αλγόριθμου εγγυημένο. Αυτό μπορεί να επιτευχθεί με την απαίτηση το ρομπότ να ταξιδέψει τουλάχιστο από μία κορυφή του εμποδίου πριν φύγει από αυτό.

Όταν το ρομπότ φτάσει σε ένα εμπόδιο τότε η κίνηση του προς τον τελικό προορισμό αλλάζει και ο αλγόριθμος μπαίνει σε μια νέα εκτέλεση εντολών. Το ρομπότ κινείται με την φορά των δεικτών του ρολογιού γύρω από το εμπόδιο και πλέον υπολογίζει συνεχώς τον νέο τρέχον συντελεστή κατεύθυνσης. Το εμπόδιο συνεχώς πρέπει να είναι στα δεξιά του. Όταν πλέον έχει περάσει από μία τουλάχιστο κορυφή και η κλίση της τρέχουσας θέσης του και τον προορισμό είναι η ίδια με αυτή της Μ-Γραμμής τότε είναι σε θέση να φύγει από το εμπόδιο.

Καθώς το ρομπότ ταξιδεύει περιμετρικά από το εμπόδιο και φτάσει σε σημείο που πρέπει να εκτελέσει αντίστροφα κάποιες του κινήσεις, όπως αναλύθηκε στα προηγούμενα υποκεφάλαια, και αφού έφτασε στο σημείο σύγκρουσης, τότε ο αλγόριθμος ξεκινά να εκτελεί και πάλι διαφορετικές εντολές. Το ρομπότ ταξιδεύει γύρω από το εμπόδιο, αυτή την φορά αντίθετα με την φορά του ρολογιού και πρέπει το εμπόδιο να βρίσκεται στα αριστερά του. Όταν βρεθεί στην Μ-Γραμμή τότε είναι σε θέση να φύγει.

Επίσης κατά το ταξίδι του γύρω από το εμπόδιο εάν ο προορισμός δεν είναι προσβάσιμος τότε ο αλγόριθμος το καταλαβαίνει και τερματίζει.

**4.6 Κατεύθυνση προς τον στόχο**

Μετά την δημιουργία της Μ-Γραμμής και την περιμετρική διάσχιση ενός εμποδίου, εάν το ρομπότ φτάσει και πάλι στην Μ-Γραμμή και είναι σε θέση να κατευθυνθεί ξανά προς τον στόχο του, εκεί υπάρχει το ερώτημα αν όντως τώρα μπορεί ξανά το ρομπότ να κινηθεί πάνω στην Μ-Γραμμή. Ο αλγόριθμος σε αυτήν την φάση ελέγχει κατά πόσο μπορεί το ρομπότ να κινηθεί προς τον προορισμό του και πάνω στην Μ-Γραμμή, σε περίπτωση που υπάρχει κάποιο άλλο εμπόδιο μπροστά του και δεν υπάρξει έλεγχος τότε το ρομπότ θα συγκρουστεί με το εμπόδιο αυτό. Έτσι με τον έλεγχο αυτόν που κάνει ο αλγόριθμος εξασφαλίζει την ασφαλή κίνηση του ρομπότ. Εάν όμως το περιβάλλον είναι ελεύθερο τότε ο αλγόριθμος επιτρέπει στο ρομπότ να κατευθυνθεί προς το στόχο.

**4.7 Αποθήκευση σημείων και κινήσεων**

Κύριος στόχος του αλγορίθμου είναι με την αντίστροφη εκτέλεση κάποιων βημάτων του να επιτευχθεί μια καλύτερη λύση ως προς το μονοπάτι που θα ακολουθήσει το ρομπότ κατά το ταξίδι του. Έτσι ο αλγόριθμος από την στιγμή που θα βρει κάποιο σημείο συνάντησης με εμπόδιο ξεκινά να αποθηκεύει όλες του τις κινήσεις και όταν φτάσει στο σημείο, όπου ο κανονισμός που μπήκε στον αλγόριθμο, επιτρέπει στην αντίστροφη εκτέλεση των εντολών αυτών, ξεκινά αυτήν την αντίστροφη εκτέλεση.

Εάν το ρομπότ κατάφερε και βρήκε σημείο φυγής χωρίς να χρησιμοποιήσει αυτές τις κινήσεις που αποθηκεύτηκαν τότε ο αλγόριθμος τις διαγράφει και περιμένει ένα νέο σημείο συνάντησης-σύγκρουσης όπου και θα ξανά ξεκινήσει να αποθηκεύει κινήσεις.

**4.8 Απόδειξη τερματισμού**

Στο υποκεφάλαιο αυτό τίθενται κάποια ερωτήματα κατά πόσο ο αλγόριθμος που προτείνεται εγγυάται τερματισμό ενώ συνεχίζει με την απόδειξή του τερματισμού του.

Κάποιοι κίνδυνοι που μπορεί να οδηγήσουν τον αλγόριθμο να μην τερματίσει είναι το μέγεθος του μονοπατιού που θα ακολουθήσει το ρομπότ να μην είναι πεπερασμένο, το ρομπότ να κάνει κύκλους συνέχεια γύρω από κάποια διαδρομή και τέλος ο προορισμός να μην είναι προσβάσιμος.

**Λήμμα 4.1:** *Ο αριθμός των εμποδίων στο περιβάλλον είναι πεπερασμένος.*

**Απόδειξη:** Κατά την κατεύθυνση του το ρομπότ προς τον στόχο η απόσταση της θέσης του με τον προορισμό μειώνεται αφού κινείται πάνω στην Μ-Γραμμή.

d(x,T) < d(xprev,T)

όπου x είναι η θέση του ρομπότ, xprev η προηγούμενη του θέση και T η θέση του προορισμού.

Επίσης κατά την διάρκεια της περιμετρικής διάσχισης του ρομπότ γύρω από κάποιο εμπόδιο, σε κάποια στιγμή φτάνει σε ένα σημείο όπου ονομάζεται σημείο φυγής, L. Το σημείο φυγής (L) είναι πιο κοντά στο στόχο από το τελευταίο σημείο σύγκρουσης (Η).

d(Li,T)< d(Hi,T)

όπου Li είναι σημείο φυγής και Hi είναι σημείο συνάντησης με εμπόδιο.

Άρα συνεχώς η απόσταση του ρομπότ καθώς κινείται μειώνεται σε σχέση με τον προορισμό του ,και με την οριοθέτηση που έχουμε στο περιβάλλον, αυτό σημαίνει πως ο αλγόριθμος θα τερματίσει όταν ο προορισμός είναι προσβάσιμος.

**Λήμμα 4.2:** *Ένα σημείο δεν μπορεί να χρησιμοποιηθεί σαν σημείο φυγής πάνω από 1 φορά.*

Ένα σημείο ορίζεται σαν σημείο φυγής εάν και μόνο εάν το ρομπότ βρίσκεται πάνω στην Μ-Γραμμή και ισχύει d(Li,T) < d(Hi,T).

**Απόδειξη:** Αυτό σημαίνει πως όταν ένα σημείο οριστεί σαν σημείο φυγής, η απόσταση από το σημείο αυτό με τον προορισμό είναι μικρότερο από την απόσταση του αντίστοιχου σημείου σύγκρουσης με τον προορισμό. Επομένως αφού για το επόμενο σημείο σύγκρουσης ισχύει d(Hi+1,T) < d(Hi,T). Κατά συνέπεια για να οριστεί το Li+1 πρέπει d(Li+1,T) < d(Hi+1,T). Έτσι με αυτόν τον τρόπο αποδεικνύεται πως ποτέ ένα σημείο δεν μπορεί να οριστεί σαν σημείο φυγής για πάνω από 1 φορά.

Συνεπώς αφού αποδείχθηκε πως το ρομπότ δεν πρόκειται να κάνει κύκλους και αφού η απόσταση του ρομπότ με τον στόχο συνεχώς μειώνεται μέχρι να μηδενιστεί τότε εάν ο προορισμός είναι προσβάσιμος τότε ο αλγόριθμος θα καταφέρει να οδηγήσει το ρομπότ στον τελικό προορισμό και να τερματίσει.

**Λήμμα 4.3:** *Όταν ο προορισμός δεν είναι προσβάσιμος το ρομπότ δεν φτάνει σε αυτόν, από Λήμμα 4.1 και Λήμμα 4.2.*

**Απόδειξη:** Εάν ο προορισμός δεν είναι προσβάσιμος , αυτό επακολουθεί πως το ρομπότ κατά το ταξίδι του θα συναντήσει κάποιο εμπόδιο. Εάν το ρομπότ σαν κινείται περιμετρικά από το εμπόδιο φτάσει ξανά στο σημείο σύγκρουσης τότε σημαίνει ο προορισμός περικλείεται από κάποιο εμπόδιο και τότε ο αλγόριθμος τερματίζει. Εάν όμως κατά την περιμετρική διάσχιση του εμπόδιού φτάσει στο σημείο όπου πρέπει να επιστρέψει πίσω, και μετά ξανά φτάσει στο ίδιο σημείο τότε σημαίνει πως ο προορισμός βρίσκεται πίσω από έναν ‘τοίχο’ και έτσι ο αλγόριθμος τερματίζει.

**Κεφάλαιο 5**

**Υλοποίηση**

5.1 Από τη θεωρία στην υλοποίηση 38

5.2 Κατεύθυνση προς τον στόχο 41

5.3 Δυναμικό Περιβάλλον 43

5.4 Υλοποίηση στην Java 44

Στο κεφάλαιο αυτό αναλύεται λεπτομερώς όλη η διαδικασία της υλοποίησης που έγινε στα πλαίσια της πτυχιακής εργασίας. Θα συζητηθεί πώς μέσα από τη θεωρία πίσω από τους αλγόριθμους κτίστηκε η υλοποίηση, η οποία είναι γραμμένη στη γλώσσα προγραμματισμού Java. Η υλοποίηση περιλαμβάνει ένα προσομοιωτή πλοήγησης για ρομπότ με υλοποιημένους τους 6 αλγόριθμους που αναλύθηκαν στο Κεφάλαιο 3 καθώς επίσης και τον αλγόριθμο που προτείνει ο συγγραφέας που αναλύθηκε στο Κεφάλαιο 4. Επιπρόσθετα στο κεφάλαιο αυτό θα εξηγηθεί πώς λειτουργούν κύριες λειτουργίες της υλοποίησης.

Σκοπός της υλοποίησης είναι να δημιουργηθεί ένας προσομοιωτής όπου να εφαρμόζονται οι αλγόριθμοι πλοήγησης που επιλέχτηκαν μαζί με αυτόν που έχει προταθεί. Το ρομπότ πρέπει να καταφέρνει να φτάσει στο τελικό του προορισμό και αν αυτός δεν είναι προσβάσιμος τότε ο προσομοιωτής να τερματίζει την διαδικασία της πλοήγησης του ρομπότ. Επίσης πέρα από αυτά στόχος του προσομοιωτή πλοήγησης είναι η εκτέλεση πειραμάτων και η σύγκριση των αλγορίθμων για διάφορα περιβάλλοντα και η εξαγωγή συμπερασμάτων για την αποδοτικότητα του καθενός από αυτούς.

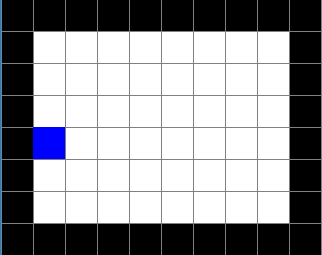
**5.1 Από τη θεωρία στην υλοποίηση**

Οι αλγόριθμοι που επιλέχτηκαν να υλοποιηθούν ανήκουν στην οικογένεια των Bug αλγορίθμων και συνήθως είναι δημοσιευμένοι μόνο σαν ψευδοκώδικες με αποτέλεσμα να υπάρχει δυσκολία κατά την ερμηνεία τους. Ως εκ τούτου είναι σημαντικό να προσδιοριστούν όλες οι προσαρμογές που χρειάζονται για την μετατροπή των ψευδοκώδικων αυτών σε εκτελέσιμους αλγορίθμους. Επίσης μιας και παρουσιάζεται ένας προσομοιωτής πλοήγησης για ρομπότ θα πρέπει να εξηγηθεί πώς κάποια στοιχεία που έχει ένα πραγματικό ρομπότ, όπως για παράδειγμα αισθητήρες, προσαρμόστηκαν στην υλοποίηση. Ακολουθεί μια συζήτηση για ορισμένα από τα θέματα κατά τη φάση της υλοποίησης.

**5.1.1 Ανανέωση θέσης**

Στηθεωρία οι Bug αλγόριθμοι ανανεώνουν συνεχώς τα δεδομένα της θέσης τους και μπορούν αυτόματα να ανιχνεύσουν αυτόματα ότι πληρούνται οι κανόνες πλοήγησης. Για παράδειγμα στον αλγόριθμο Bug2 μόλις το ρομπότ φτάσει στην Μ-Γραμμή, ο αλγόριθμος το ανιχνεύει αυτόματα αυτό και δρα αναλόγως. Στην πράξη αυτό σημαίνει πως τα δεδομένα της θέσης του ρομπότ απαιτούν συνεχώς ανανέωση και έλεγχο των κανόνων πλοήγησης. Το γεγονός αυτό απαιτεί συνεχείς υπολογισμούς με αποτέλεσμα να σπαταλούνται πόροι, είτε μνήμη, είτε χρόνος, και έτσι η ανανέωση της θέσης του ρομπότ δεν γίνεται συνέχεια.

Για το λόγο αυτό μέσα στην υλοποίηση η θέση του ρομπότ ανανεώνεται όταν έχει ταξιδέψει ένα τετραγωνάκι. Αυτό θα γίνει αντιληπτό στην συνέχεια όταν δείτε μερικές εικόνες από τον προσομοιωτή. Συνοπτικά μέσα στον προσομοιωτή που υλοποιήθηκε, το ρόλο του σημείου παίζει κάθε τετράγωνο, και όχι κάθε πραγματικό σημείο. Στην πιο κάτω φωτογραφία μπορείτε να δείτε το περιβάλλον του προσομοιωτή με τα τετράγωνα που αναπαριστούν κάθε σημείο.



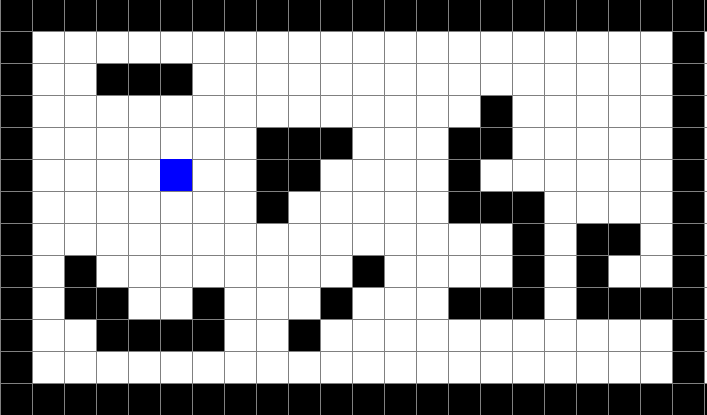
Σχήμα 5.1: Παράδειγμα σημείων

**5.1.2 Αποθήκευση σημείων**

Στη θεωρία οι Bug αλγόριθμοι χρησιμοποιούν πολλά μικρά σημεία για να αναπαραστήσουν την αρχική θέση του ρομπότ, την τελική θέση του ρομπότ, σημεία συνάντησης με εμπόδιο, σημεία φυγής από το εμπόδιο και άλλα σημαντικά σημεία που χρειάζονται να αποθηκεύουν και να θυμούνται. Στην πράξη αυτό δεν μπορεί να εφαρμοστεί λόγω του περιορισμού στη ανανέωση των δεδομένων της θέσης του, όπως εξηγήθηκε πιο πάνω. Ως εκ τούτου στην υλοποίηση ένα σημαντικό σημείο όπως αρχική θέση, τελικός προορισμός, σημείο συνάντησης με εμπόδιο και άλλα αναπαρίστανται με ένα τετράγωνο. Να σημειωθεί πως το μέγεθος του ρομπότ επιλέχτηκε να είναι το ίδιο μέγεθος με ένα τετράγωνο ούτως ώστε κάθε κύκλος ανανέωσης να γίνεται σωστά.

**5.1.3 Εμπόδια**

Ένα σχήμα μπορεί να έχει οποιοδήποτε σχήμα, μπορεί να έχει γωνίες ή ακόμα αυτές να είναι κυρτές. Επίσης κάποιο εμπόδιο ίσως είναι κυκλικό και γενικότερα να έχει οτιδήποτε ακανόνιστο σχήμα και μέγεθος. Για σκοπούς υλοποίησης η αναπαράσταση των εμποδίων και το σχήμα τους αποτελείται από πολλά τετράγωνα χρώματος μαύρου όπως θα δούμε και στη συνέχεια. Ο λόγος είναι επειδή επιλέχτηκε ένα σημείο να αναπαρίσταται από ένα τετράγωνο, τότε τα σημεία που θα αναπαριστούν κάποιο εμπόδιο πρέπει και αυτά να είναι τα ίδια τετράγωνα. Έτσι με αυτόν τον τρόπο θέσαμε με το ίδιο μέτρο όλα τα σημεία είτε αυτά αναπαριστούν κάποια θέση, είτε αναπαριστούν κάποιο σημαντικό σημείο είτε κάποιο εμπόδιο. Στην πιο κάτω εικόνα βλέπουμε την αναπαράσταση εμποδίων στον προσομοιωτή πλοήγησης που υλοποιήθηκε. Με μαύρο χρώμα είναι τα σημεία όπου αναπαριστούν εμπόδια, με άσπρο τον χώρο που είναι ελεύθερος για πλοήγηση και με μπλε το ρομπότ.



Σχήμα 5.2: Παράδειγμα αναπαράστασης εμποδίων

**5.1.4 Εξοπλισμός αισθητήρων**

Από την στιγμή που έχει υλοποιηθεί ένας προσομοιωτής πλοήγησης για ρομπότ, όλων των ειδών αισθητήρων απουσιάζουν. Κατά την πλοήγηση τους προς τον τελικό προορισμό τα ρομπότ έχουν αισθητήρες ούτως ώστε να ανιχνεύουν τυχόν εμπόδια. Στην υλοποίηση που παρουσιάζεται η ανίχνευση εμποδίου γίνεται ως εξής. Κάθε σημείο (τετράγωνο) του περιβάλλοντος, στο κώδικα αναπαρίσταται με ένα σημείο στον άξονα x, με ένα σημείο στον άξονα y, και μια μεταβλητή που δείχνει κατά πόσο είναι προσβάσιμο. Το ρομπότ δεν γνωρίζει καθόλου για το τι συμβαίνει στο περιβάλλον όπου πρόκειται να κινηθεί, αλλά έχει την δυνατότητα να δει την προσβασιμότητα του επόμενου του τετραγώνου μόνο. Έτσι με αυτό τον τρόπο επιτυγχάνεται η λειτουργία του αισθητήρα για εντοπισμό εμποδίων.

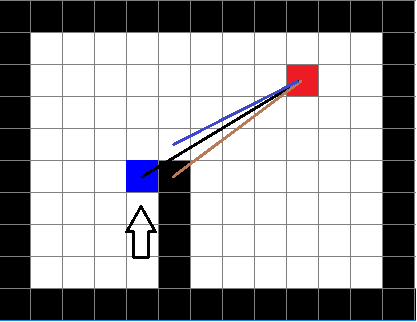
**5.2 Κατεύθυνση προς τον στόχο**

Σε όλους του αλγόριθμους το ρομπότ για να μπορέσει να φύγει από κάποιο εμπόδιο υπάρχει ο περιορισμός που λέει πως το ρομπότ πρέπει να μπορεί να φύγει προς τον στόχο. Στην υποενότητα αυτή θα αναλυθεί ο τρόπος με τον οποίο ελέγχεται στην υλοποίηση εάν το ρομπότ μπορεί να κατευθυνθεί προς τον στόχο.

Οι αλγόριθμοι όπως ο Bug2, Alg1 και αυτός που προτείνεται από τον συγγραφέα χρησιμοποιούν τη μέθοδο της Μ-Γραμμής, δηλαδή για να φύγουν από κάποιο εμπόδιο θα πρέπει να φτάσουν στην Μ-Γραμμή και εκεί να ελέγχει αν το ρομπότ μπορεί να κινηθεί προς το στόχο. Οι άλλοι αλγόριθμοι δεν έχουν αυτόν τον περιορισμό της Μ-Γραμμής αλλά μόλις μπορούν να φύγουν, το κάνουν. Και στις 2 περιπτώσεις όμως γίνεται ένας έλεγχος προς το επόμενο σημείο (τετράγωνο) αν είναι προσβάσιμο, με βάση τον συντελεστή κατεύθυνσης που υπολογίζεται από την εκάστοτε θέση και τον τελικό προορισμό. Έτσι ο τρόπος με τον οποίο γίνεται ο έλεγχος αν το ρομπότ μπορεί να κινηθεί προς τον στόχο είναι ο ίδιος και για τις 2 περιπτώσεις.

Για τους αλγόριθμους που χρησιμοποιούν τη μέθοδο της Μ-Γραμμής τα πράγματα είναι πολύ πιο απλά. Η υλοποίηση για τους αλγόριθμους αυτούς δεν επιτρέπει στον χρήστη να ορίσει σαν τελικό προορισμό οποιοδήποτε σημείο, παρά μόνο σημεία που βρίσκονται στο ίδιο σημείο στον άξονα x με την αρχική θέση του ρομπότ, σημεία που βρίσκονται στο ίδιο σημείο στον άξονα y με την αρχική του θέση και σε σημεία όπου η κλίση τους με την αρχική θέση είναι 45 μοίρες. Ως αποτέλεσμα έχουμε η κλίση να είναι είτε 0, είτε 1/2 , είτε ∞. Ο λόγος όπου έγινε αυτό είναι για να δοθεί έμφαση και να αναδειχθεί η μέθοδος της Μ-Γραμμής. Έτσι το ρομπότ όταν φτάσει σε σημείο όπου πρέπει να εκλεχθεί κατά πόσο μπορεί να κατευθυνθεί προς τον στόχο, γίνεται με τον μηχανισμό του αισθητήρα όπου αναλύθηκε πιο πάνω και ελέγχεται το κατάλληλο τετράγωνο αναλόγως με την κατεύθυνση που έχει το ρομπότ.

Για τους υπόλοιπους αλγορίθμους τα πράγματα είναι λίγο πιο περίπλοκα. Όταν το ρομπότ φτάσει σε σημείο όπου πρέπει να ελεγχθεί εάν μπορεί να φύγει προς τον στόχο τότε έχοντας σαν δεδομένα την τρέχουσα θέση του και τη θέση του τελικού προορισμού υπολογίζει την κλίση των 2 αυτών σημείων. Στην συνέχεια παίρνει τα 2 πιο κοντινά σημεία προς το στόχο που βρίσκονται γύρω του και υπολογίζει και για αυτά τις κλίσεις του με το σημείο του τελικού προορισμού. Το σημείο όπου έχει πλησιέστερη κλίση με αυτήν της τρέχουσας θέσης είναι και αυτό που θα ελεγχθεί κατά πόσο είναι προσβάσιμο ή όχι. Εάν είναι τότε το ρομπότ μπορεί να κινηθεί προς τον στόχο ενώ σε αντίθεση περίπτωση δεν μπορεί. Πιο κάτω παρουσιάζεται μια εικόνα για καλύτερη κατανόηση πώς ελέγχεται εάν μπορεί το φύγει να κινηθεί προς τον στόχο.



Σχήμα 5.3: Παράδειγμα ελέγχου κατεύθυνσης προς τον στόχο

**Εξήγηση εικόνας:**

Με μπλε φαίνεται η τρέχουσα θέση του ρομπότ όπου έχει κατεύθυνση προς τα πάνω όπως δείχνει και το βέλος. Με κόκκινο χρώμα παρουσιάζεται η θέση του τελικού προορισμού. Η μαύρη ευθεία είναι η ευθεία που σχηματίζεται μεταξύ τρέχουσας θέσης και προορισμού και η κλίση της ισούται με 0.60. Αντίστοιχα βλέπουμε με μπλε και καφέ χρώμα τις ευθείες που σχηματίζονται μεταξύ των 2 κοντινότερων σημείων προς τον στόχο γύρω από την τρέχουσα θέση και τον τελικό προορισμό. Η κλίση της καφέ είναι 0.50 και της μπλε 0.75. Η κλίση όπου είναι κοντινότερη αυτής που σχηματίζεται μεταξύ τρέχουσας θέσης και τελικού προορισμού είναι αυτή της μπλε ευθείας. Άρα θα κοιτάξουμε εάν το σημείο αυτό είναι προσβάσιμο. Στο παράδειγμα που φαίνεται, ναι είναι προσβάσιμο άρα το ρομπότ μπορεί να φύγει.

**5.3 Δυναμικό περιβάλλον**

Όπως προαναφέρθηκε η υλοποίηση περιλαμβάνει και δυναμικό περιβάλλον. Κατά την εκκίνηση της διαδρομής του το ρομπότ στο ήδη διαμορφωμένο περιβάλλον, ο χρήστης μπορεί να προσθέσει ανά πάσα στιγμή επιπρόσθετα εμπόδια. Και πάλι σε αυτήν την περίπτωση αυτά τα εμπόδια αναπαριστάνονται με την έννοια του σημείου όπως εξηγήθηκε προηγουμένως, δηλαδή με τετράγωνο.

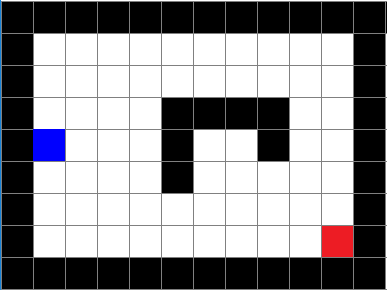
Με τον ορισμό του προορισμού τότε του ρομπότ ξεκινά την πλοήγησή του προς τον στόχο. Στην υλοποίηση αυτό γίνεται δημιουργώντας ένα νήμα όπου αυτό είναι υπεύθυνο για να κατευθύνει το ρομπότ προς τον τελικό του προορισμό και από εδώ και πέρα τα κλικς που ακούει το πρόγραμμα, σε οποιοδήποτε σημείο, τα αντιλαμβάνεται σαν προσθήκη νέου εμποδίου στο περιβάλλον και πλέον το σημείο όπου επιλέχτηκε από προσβάσιμο σημείο μετατρέπεται σε κάποιο εμπόδιο. Όταν το νήμα εκτελέσει όλες του τις λειτουργίες και τερματιστεί ο αλγόριθμος, τότε στο επόμενο κλικ που θα ακούσει το πρόγραμμα σημαίνει πως είναι ορισμός νέου σημείου προορισμού για το ρομπότ, και όχι για προσθήκη νέου εμποδίου όπως συνέβαινε προηγουμένως. Αυτό επιτυγχάνεται με χρήση μια μεταβλητής τύπου boolean όπου μετατρέπεται σε true ή false αναλόγως εάν το ρομπότ βρίσκεται σε πλοήγηση ή όχι.

**5.4 Υλοποίηση στην Java**

Ο προσομοιωτής για κινητά ρομπότ έχει αναπτυχθεί στην γλώσσα προγραμματισμού Java ενώ έχουν χρησιμοποιηθεί και μερικές βιβλιοθήκες που παρέχει η γλώσσα. Σε αυτό το υποκεφάλαιο θα ζητηθούν ποιες βιβλιοθήκες χρησιμοποιήθηκαν και θα εξηγηθούν οι κλάσεις που χρησιμοποιήθηκαν και γενικότερα οι τεχνικές που χρησιμοποιήθηκαν για την υλοποίηση του προσομοιωτή.

**5.4.1 Βιβλιοθήκες**

Για την γραφική αναπαράσταση του προσομοιωτή χρησιμοποιήθηκε η βιβλιοθήκη java.awt. Η συγκεκριμένη βιβλιοθήκη παρέχει σε ένα προγραμματιστή την δυνατότητα σχεδίασης γραφικών εικόνων. Για την υλοποίηση του προσομοιωτή, χρειάστηκε να σχεδιαστεί το περιβάλλον που πρόκειται να κινηθεί το ρομπότ, ενώ χρησιμοποιήθηκαν και μερικές μέθοδοι για τον κατάλληλο σχεδιασμό του χρώματος της τελικής θέσης του ρομπότ, του χρώματος του ρομπότ, του χρώματος των εμποδίων και τον διαχωρισμό του περιβάλλοντος σε σημεία (τετράγωνα). Μέσα από την βιβλιοθήκη java.awt χρησιμοποιήθηκε η κλάση Graphics για την δημιουργία του πλαισίου που θα απαρτίζεται το περιβάλλον. Για τον διαχωρισμό του πλαισίου σε τετράγωνα χρησιμοποιήθηκε η μέθοδος drawLine (int x1, int y1, int x2, int y2), όπου παίρνει σαν ορίσματα τις συντεταγμένες των σημείων τα οποία πρέπει να ενωθούν με μια ευθεία γραμμή. Για το σχεδιασμό ενός τετραγώνου με κάποιο χρώμα χρησιμοποιήθηκε η μέθοδος fillRect (int x, int y, int width, int height) όπου παίρνει σαν παραμέτρους τις συντεταγμένες του σημείου από το οποίο θα ξεκινήσει, το μήκος και το ύψος για το τετράγωνο όπου θέλουμε να ζωγραφίσουμε. Τέλος για την επιλογή του χρώματος όπου θα έπρεπε να δημιουργηθούν οι ευθείες γραμμές η ο σχεδιασμός των τετραγώνων έγινε με την χρήση της μεθόδου setColor(Color c) όπου δέχεται σαν παράμετρο ένα αντικείμενο της κλάσης Color που προσφέρεται από την ίδια βιβλιοθήκη. Στο παρακάτω σχήμα παρουσιάζεται ένα στιγμιότυπο από τον προσομοιωτή όπου παρουσιάζεται το αποτέλεσμα της χρήσης της κλάσης Graphics και μεθόδων της.

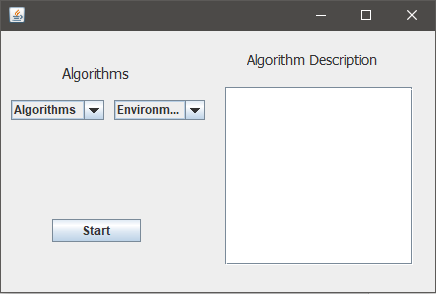


Σχήμα 5.4: Στιγμιότυπο προσομοιωτή

Με την μέθοδο setColor() έγινε η επιλογή των χρωμάτων που διαπιστώνονται, με την μέθοδο fillRect() έγινε η ο σχεδιασμός των σημείων και με την drawLine() έγινε ο διαχωρισμός του πλαισίου σε τετράγωνα.

Επιπρόσθετα από την βιβλιοθήκη java.awt χρησιμοποιήθηκαν οι κλάσεις MouseEvent, KeyEvent και MouseMotionListener. Οι συγκεκριμένες κλάσεις χρησιμοποιήθηκαν για να μπορεί ο προσομοιωτής να ακούει σε συμβάντα που προκαλούνται από τον χρήστη. Πιο συγκεκριμένα όταν ο χρήστης κάνει κλικ με το ποντίκι είτε όταν θέλει να δημιουργήσει το περιβάλλον του είτε όταν θέλει να ορίσει ένα σημείο σαν τελικό προορισμό. Επίσης με τον πάτημα ενός κουμπιού από το πληκτρολόγιο, το πρόγραμμα ακούει σε αυτό το γεγονός. Αυτό χρησιμοποιείται όταν ο χρήστης θέλει να αλλάξει την λειτουργία του προσομοιωτή από πλοήγηση σε διαμόρφωση του περιβάλλοντος. Αυτά θα εξηγηθούν αναλυτικότερα στην πορεία.

Επίσης για την υλοποίηση του προσομοιωτή έγινε η χρήση της βιβλιοθήκης java,util για χρήση δομών δεδομένων. Από την βιβλιοθήκη αυτή χρησιμοποιήθηκε η κλάση List για την αποθήκευση σημείων και κινήσεων του ρομπότ όσο ταξιδεύει όπου πιθανώς να χρησιμοποιήσει για την επίτευξη του στόχου του. Μέθοδοι που χρησιμοποιήθηκαν αυτής της κλάσης είναι η add (Object o) η οποία δέχεται σαν παράμετρο ένα στοιχείο όπου το αποθηκεύει στην λίστα, remove (int index) η οποία διαγράφει από την λίστα το στοιχείο όπου είναι στη θέση που περνιέται σαν παράμετρος, contain (Object o) η οποία ελέγχει κατά πόσο ένα αντικείμενο που περνιέται σαν παράμετρος υπάρχει στη λίστα. Η χρήση της μεθόδου αυτής γίνεται για τον έλεγχο εάν το ρομπότ έχει φτάσει σε προηγούμενο σημείο σύγκρουσης ή διαφυγής τα οποία αποθηκευτήκαν με την μέθοδο add().

Η τελευταία βιβλιοθήκη που χρησιμοποιήθηκε είναι η java.swing. Η συγκεκριμένη βιβλιοθήκη χρησιμοποιήθηκε για την υλοποίηση μιας γραφικής διεπαφής με τον χρήστη κατά την εκκίνηση του προσομοιωτή για την επιλογή αλγόριθμου που πρόκειται να χρησιμοποιήσει. Στο πιο κάτω σχήμα παρουσιάζεται η γραφική διεπαφή που αναπτύχθηκε μέσω της βιβλιοθήκης java.swing

Σχήμα 5.5: Παράδειγμα από την γραφική διαπροσωπία

**5.4.2 Κλάσεις και Μέθοδοι**

Στο υποκεφάλαιο αυτό αναλύονται οι κλάσεις του λογισμικού του προσομοιωτή μαζί με βασικές μεθόδους και λειτουργίες. Ο προσομοιωτής απαρτίζεται από μερικές κλάσεις οι οποίες περιλαμβάνουν κάποιες σημαντικές λειτουργίες όπου θα συζητηθούν.

**5.4.2.1 Κλάση Create**

Η συγκεκριμένη κλάση είναι αυτή όπου δημιουργεί όλα τα κομμάτια του προσομοιωτή, όπως την κλάση που αντιπροσωπεύει το κάθε σημείο, την κλάση που αντιπροσωπεύει το περιβάλλον και την κλάση όπου διαχειρίζεται όλα τα γεγονότα που δημιουργούνται από τον χρήστη και τέλος την κλάση όπου είναι υπεύθυνη για την κίνηση του ρομπότ ούτως ώστε να φαίνεται στον χρήστη η κίνηση που κάνει από την αρχή της διαδρομής του μέχρι το τέλος. Επίσης η κλάση Create δημιουργεί το πλαίσιο στο οποίο θα παρουσιάζεται το περιβάλλον και η κίνηση του ρομπότ.

Η κλάση αυτή περιλαμβάνει μια σημαντική λειτουργία όπου περιλαμβάνεται στην μέθοδο run(). Η μέθοδος αυτή εκτελείται συνεχώς μέχρι ο χρήστης να τερματίσει τον προσομοιωτή. Σε αυτήν την μέθοδο συνεχώς καλείται μια μέθοδος η οποία ονομάζεται render() η οποία ο ρόλος της είναι συνεχώς να σχεδιάζει το περιβάλλον ούτως ώστε να φαίνονται στον χρήστη καθ’ όλη την διάρκεια την κίνηση του ρομπότ. Επίσης συνεχώς καλείται η μέθοδος update() η οποία είναι υπεύθυνη για την ενημέρωση της θέσης του ρομπότ. Οι συναρτήσεις αυτές θα επεξηγηθούν στην συνέχεια. Άρα σε συντομία αυτή κλάση είναι υπεύθυνη για την δημιουργία όλων των επιμέρους κλάσεων του λογισμικού και επίσης να τρέχει συνέχεια καλώντας την μέθοδο update() και render(), οι οποίες είναι υλοποιημένες σε άλλες κλάσεις για την συνεχή ενημέρωση της κατάστασης της γραφικής εικόνας που βλέπει ο χρήστης.

**5.4.2.2 Κλάση Point**

Η κλάση Point αναπαριστά κάθε σημείο στο περιβάλλον. Έχει σαν πεδίο ορισμού την συντεταγμένη στον άξονα Χ, την συντεταγμένη στον άξονα Y και μια μεταβλητή τύπου Boolean όπου καθορίζει κατά πόσο το σημείο αυτό είναι προσβάσιμο για το ρομπότ η αν αντιπροσωπεύει κάποιο εμπόδιο. Επίσης περιλαμβάνει και κάποιες μεθόδους για ανάκτηση αυτών των πληροφοριών αλλά και για επεξεργασία τους. Δεν έχει κάποια σημαντική λειτουργικότητα η κλάση αυτή αφού απλά αναπαριστά ένα σημείο στο περιβάλλον.

**5.4.2.3 Κλάση Environment**

Η συγκεκριμένη κλάση αναπαριστά το περιβάλλον στο οποίο το ρομπότ κινείται. Αποτελείται από ένα δισδιάστατο πίνακα από Points. Η κλάση Environment περιέχει μια μέθοδο η οποία ονομάζεται drawMap() και εκτελεί μια βασική λειτουργία, την σχεδίαση του περιβάλλοντος. Δέχεται σαν παράμετρο το πλαίσιο και γνωρίζοντας τα στοιχεία όλων των σημείων του περιβάλλοντος σχεδιάζει το περιβάλλον. Να σημειωθεί πως τα σημεία στο περιβάλλον έχουν σαν μήκος και ύψος 32. Πιο κάτω παρουσιάζεται ένας ψευδοκώδικας της συνάρτησης αυτής.

Θέσε i=0 και j=0

Όσο το i είναι μικρότερο από το μήκος του πίνακα

Όσο το j είναι μικρότερο από το ύψος του πίνακα

Αν το σημείο [ i ] [ j ] είναι εμπόδιο

Θέσε χρώμα = μαύρο

Αλλιώς αν το σημείο [ i ] [ j ] είναι ελεύθερο

Θέσε το χρώμα = άσπρο

Αλλιώς αν το σημείο [ i ] [ j ] είναι ο τελικός προορισμός

Θέσε το χρώμα = κόκκινο

Ζωγράφισε το αντίστοιχο τετράγωνο με το χρώμα

Αύξησε το i και το j κατά 1

Επίσης η συγκεκριμένη κλάση περιέχει μέθοδο η οποία είναι υπεύθυνη για την απόφαση του επόμενου βήματος που πρέπει να ακολουθήσει το ρομπότ. Η συνάρτηση αυτή παίρνει σαν παράμετρο τις συντεταγμένες της τρέχουσας θέσης, τον χάρτη που περιγράφει το περιβάλλον, δηλαδή αντικείμενο της κλάσης Εnvironment και την κατεύθυνση που έχει το ρομπότ. Η συνάρτηση αυτή, ανάλογα με την κατεύθυνση που έχει το ρομπότ ελέγχει μόνο για τα γύρω του σημεία τι συμβαίνει και λαμβάνοντας όλους τους περιορισμούς που περιέχει ο αλγόριθμος σχεδίασης διαδρομής αποφασίζει πιο σημείο είναι το επόμενο στο οποίο πρέπει να πάει το ρομπότ. Τέλος η συνάρτηση επιστρέφει πίσω την απόφαση αυτή.

**5.4.2.4 Κλάση Robot**

Η κλάση αυτή αναπαριστά το ρομπότ. Στην ουσία συνεχώς έχει αποθηκευμένη την τρέχουσα θέση του ρομπότ, την κατεύθυνση στην οποία κινείται, την ταχύτητα με την οποία κινείται και την θέση του σημείου του τελικού προορισμού. Επίσης όπως προαναφέρθηκε κάθε σημείο αναπαρίσταται με ένα τετράγωνο ύψους 32 και μήκους 32. Για την πιο ρεαλιστική κίνηση του ρομπότ, αυτό έχει και 2 τιμές οι οποίες αναπαριστούν την μετατόπισή του από το σημείο που βρίσκεται. Για παράδειγμα αν το ρομπότ βρίσκεται μεταξύ 2 τετραγώνων αυτό σημαίνει πως η τρέχουσα του θέση είναι αυτή του αριστερού σημείου με μετατόπιση x. Η θέση του θα ανανεωθεί όταν πλέον το ρομπότ μπει ολόκληρο στο επόμενο τετράγωνο. Έτσι στο πεδίο ορισμού της κλάσης ορίζονται 2 τιμές που αντιπροσωπεύουν την μετατόπιση του στο άξονα x και την μετατόπιση του στον άξονα y.

Η κλάση αυτή είναι υπεύθυνη για την κίνηση του ρομπότ. Περιέχει μια συνάρτηση όπου ονομάζεται walk(). Όπως προαναφέρθηκε στην κλάση Create καλείτε συνεχώς καλείται συνεχώς η μέθοδος update(). H μέθοδος αυτή βρίσκεται μέσα στην κλάση Robot. Αυτή με την σειρά της ελέγχει εάν το ρομπότ είναι σε αδράνεια ή αν κινείται. Εάν κινείται τότε αυτή καλεί την walk(). Η walk() αφού έχει γίνει επιστροφή του επόμενου βήματος από την findNextStep() που περιεγράφηκε προηγούμενος, ελέγχει που βρίσκεται το σημείο αυτό σε σχέση με την τρέχουσα του θέση και το μόνο που κάνει είναι να αλλάζει την μετατόπιση της θέσης του με βάση την ταχύτητά του. Άρα στην ουσία η ταχύτητα που προαναφέρθηκε είναι ο ρυθμός μεταβολής της μετατόπισης του ρομπότ. Όταν μεταβάλει την μετατόπιση του ρομπότ τότε ελέγχει εάν το υπόλοιπο της μετατόπισης όταν διαιρεθεί με το 32 είναι 0. Εάν ισχύει τότε σημαίνει πως το ρομπότ διέσχισε ένα τετράγωνο και πλέον αλλάζει την θέση του. Παρακάτω παρουσιάζεται ένα μικρό κομμάτι της μεθόδου αυτής σε ψευδοκώδικα. Ο ψευδοκώδικας αυτός αναφέρεται σε περίπτωση που το ρομπότ κινείται από αριστερά προς δεξιά και παράλληλα με τον άξονα x.

Θέσε σαν next το σημείο όπου αποτελεί το επόμενο βήμα του ρομπότ

If ( next.getΥ() == robot.getΥ() ){

If (next.getΧ() < robot.getΧ()

Μετατόπιση =μετατόπιση – ταχύτητα

Else

Μετατόπιση =μετατόπιση + ταχύτητα

If ( μετατόπιση % 32 == 0)

Αύξησε την θέση του x του ρομπότ κατά 1

Όρισε την μετατόπιση ίσων με 0

}

Αυτό αποτελεί ένα δείγμα της λειτουργίας αυτής της συνάρτησης. Η ιδέα και για τις άλλες περιπτώσεις όπως για παράδειγμα το ρομπότ να κινηθεί παράλληλα με τον άξονα y είτε διαγώνια είναι ακριβώς η ίδια.

**5.4.2.5 Κλάση Μanagement**

Στην συγκεκριμένη κλάση είναι υπεύθυνη για την διαχείριση όλων των συμβάντων που δημιουργούνται από τον χρήστη. Εάν δημιουργηθεί ένα συμβάν (event) από ένα κλικ του ποντικιού η κλάση αυτή το ακούει και δρα αναλόγως. Εάν αυτό σηματοδοτεί ένα ορισμό σημείου προορισμού τότε καλεί τον κατάλληλο αλγόριθμο ο όποιος τρέχει σε ένα νέο νήμα. Όταν πλέον το ρομπότ είναι σε κατάσταση πλοήγησης τότε η κλάση αυτή όταν ακούσει σε ένα νέο συμβάν πατήματος κουμπιού τότε θέτει το συγκεκριμένο σημείο σαν εμπόδιο και εδώ είναι που μπαίνει και η έννοια του δυναμικού περιβάλλοντος στην υλοποίηση. Επίσης το η κλάση αυτή ακούει σε συμβάντα πατήματος ενός κουμπιού από το πληκτρολόγιο. Η χρησιμότητα της λειτουργίας αυτής είναι για την μετατροπή της κατάστασης του προσομοιωτή από κατάσταση πλοήγησης σε κατάσταση επεξεργασίας του περιβάλλοντος. Όταν δηλαδή με την εκτέλεση κάποιου αριθμού από πλοηγήσεις το ρομπότ, πατηθεί οποιοδήποτε κουμπί από το πληκτρολόγιο τότε η συγκεκριμένη κλάση βάζει τον προσομοιωτή σε κατάσταση επεξεργασίας περιβάλλοντος και πλέον τα συμβάντα των κλικ από το ποντίκι τα διαχειρίζεται σαν διαμόρφωση του περιβάλλοντος. Εάν ένα σημείο είναι εμπόδιο και επιλεχθεί με το ποντίκι σε αυτήν την κατάσταση, η κλάση αυτόματα το συγκεκριμένο σημείο το κάνει σαν προσβάσιμο σημείο και το αντίθετο.

Επίσης μέσα στην συγκεκριμένη κλάση είναι υλοποιημένη η μέθοδος render(), η οποία καλείται συνεχώς από την κλάση Create, η οποία είναι υπεύθυνη για την γραφική αναπαράσταση του ρομπότ. Δέχεται σαν είσοδο το πλαίσιο που αναπαριστά την προσομοίωση τα πλοήγησης του ρομπότ και ζωγραφίζει την θέση του ρομπότ με βάση της θέσης του και των μετατοπίσεων του που υπολογίζονται από την μέθοδο walk() της κλάσης Robot.

**Κεφάλαιο 6**

**Πειραματική Αξιολόγηση Αλγόριθμων**

6.1 Μεθοδολογία και Διευκρινήσεις 52

6.2 Πειράματα 53

6.3 Συμπεράσματα 64

Στο κεφάλαιο αυτό παρουσιάζονται τα πειράματα που έχουν γίνει για τους αλγόριθμους και αναλύονται όλα τα αποτελέσματα και τα συμπεράσματα που εξάχθηκαν. Κατά την διάρκεια των πειραμάτων που έγιναν, εκτελέστηκαν όλοι οι αλγόριθμοι που έχουν υλοποιηθεί στον προσομοιωτή πλοήγησης που αναπτύχθηκε στα πλαίσια της παρούσας πτυχιακής εργασίας. Για κάθε πείραμα παρουσιάζεται το περιβάλλον που θα πρέπει να εξερευνήσει το ρομπότ καθώς επίσης και η διαδρομή που ακολούθησε από την έναρξη της πλοήγησής του μέχρι να φτάσει στον τελικό του προορισμό. Στην συνέχεια τα αποτελέσματα που αφορούν το μέγεθος της διαδρομής που ακολούθησαν μέχρι την επίτευξη του στόχου παρουσιάζονται σε γραφική παράσταση. Τέλος συζητούνται τα αποτελέσματα όπου πάρθηκαν.

**6.1 Μεθοδολογία και Διευκρινήσεις**

Μέσα από την έρευνα και την υλοποίηση των αλγόριθμων της οικογένειας Bug, και την μελέτη για την επινόηση ενός νέου αλγόριθμου με σκοπό να βελτιστοποιεί κυρίως τους αλγόριθμους που είναι βασισμένος, δηλαδή τους Bug2 και Alg1, καταγράφηκαν κάποιες περιπτώσεις όπου οι αλγόριθμοι αντιμετωπίζουν κάποια μειονεκτήματα. Στόχος της πειραματικής αξιολόγησης είναι να φανεί κατά πόσο οι αλγόριθμοι που δημιουργήθηκαν βελτιστοποιούν τους αλγόριθμους που είναι βασισμένοι, και επίσης να παρατηρηθεί εάν ο αλγόριθμος που προτείνεται φέρει καλύτερα αποτελέσματα από τους προηγούμενους.

Η επιλογή των περιβαλλόντων έγινε με γνώμονα τις περιπτώσεις όπου οι αλγόριθμοι έχουν κάποια μειονεκτήματα για να δούμε εάν αυτά τα μειονεκτήματα λύθηκαν στην συνέχεια με άλλους αλγόριθμους.

Σε κάθε πείραμα παρουσιάζεται το περιβάλλον όπου εκτελούνται οι αλγόριθμοι. Με μαύρο χρώμα παρουσιάζονται τα εμπόδια, με άσπρο ο ελεύθερος χώρος που μπορεί να κινηθεί το ρομπότ, με μπλε η αρχική θέση του ρομπότ και τέλος με κόκκινο η θέση του τελικού προορισμού. Ο υπολογισμός του μήκους της διαδρομής υπολογίζεται μέσω της απόστασης 2 σημείων όπου δίνεται από τον πιο κάτω τύπο:

d = √(X2−X1)2+(Y2−Y1)2 ,

όπου d είναι η απόσταση των 2 σημείων,

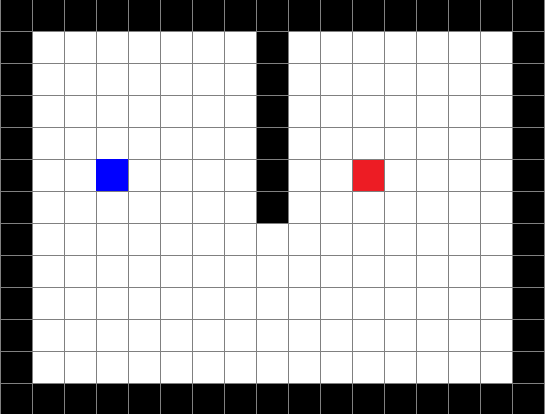
(X1,Y1)= Συντεταγμένες του 1ου σημείου,

(X2,Y2)= Συντεταγμένες του 2ου σημείου

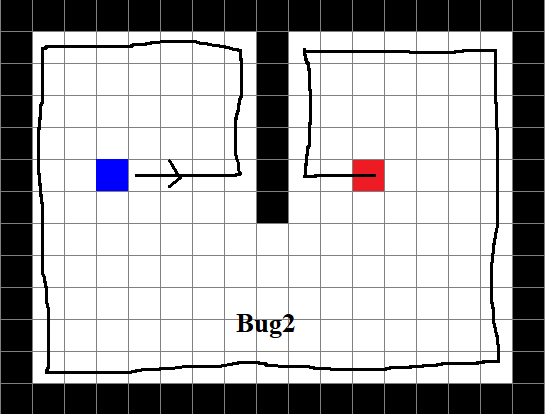
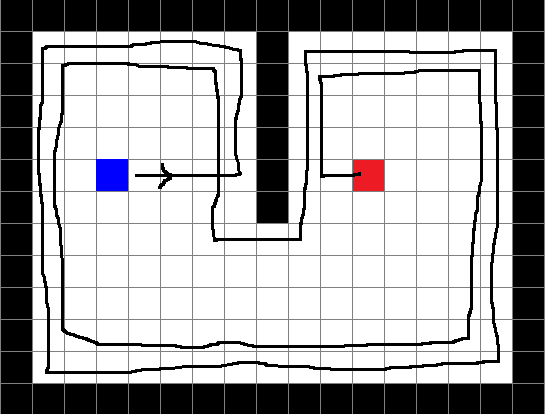
**6.2 Πειράματα**

**Πείραμα 1:**

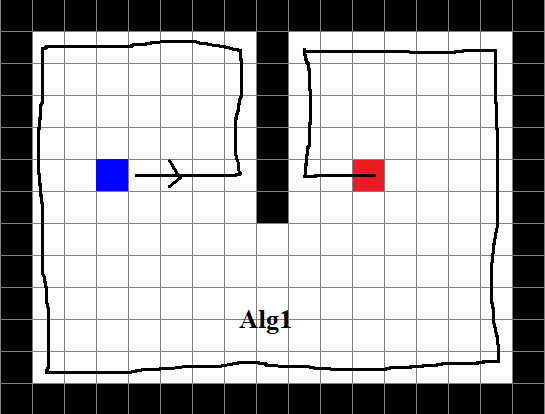
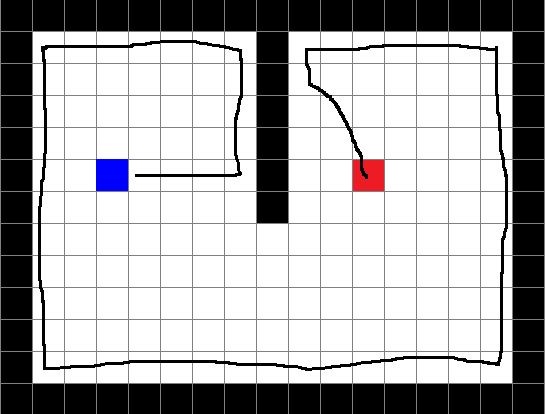
Στην πιο κάτω εικόνα παρουσιάζεται το περιβάλλον για το πρώτο πείραμα. Με μπλε αναπαρίσταται το ρομπότ και με κόκκινο το σημείο της τελικής του θέσης.



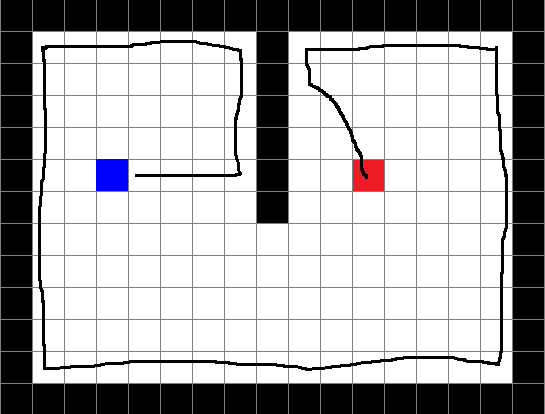
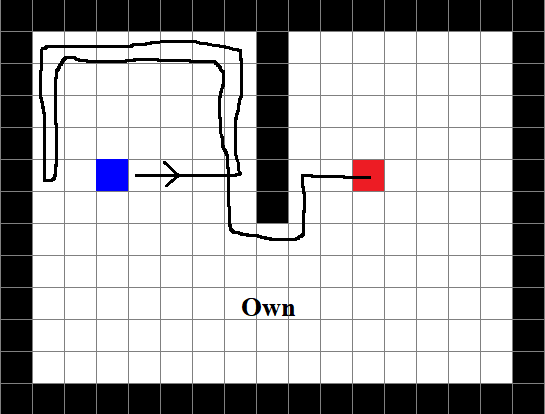
Σχήμα 6.1: Περιβάλλον Α



Bug1 Bug2



Alg1 Alg2



Class1 Own

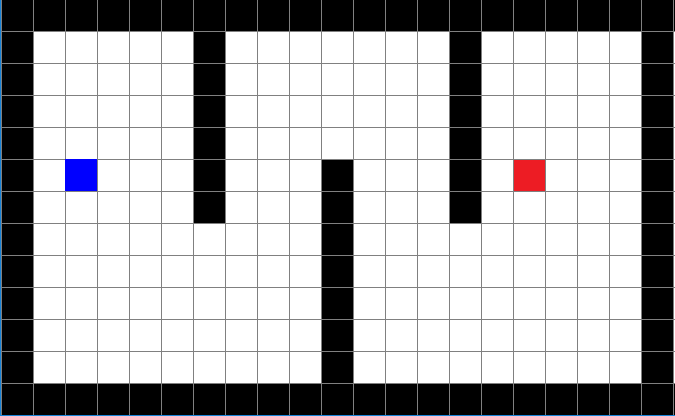
Σχήμα 6.2: Διαδρομές των Ρομπότ στο περιβάλλον Α

Σχήμα 6.3: Αποτελέσματα για το Περιβάλλον Α

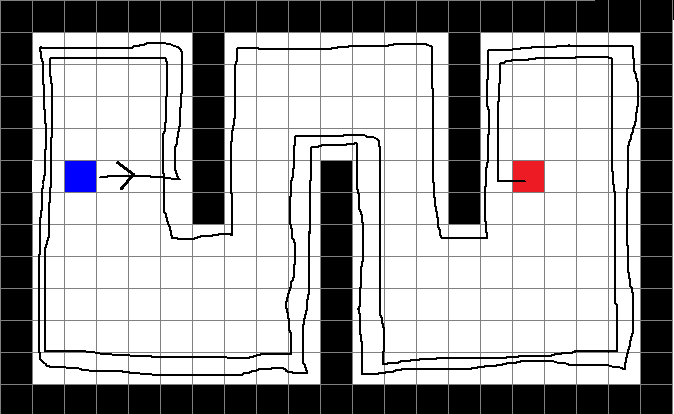
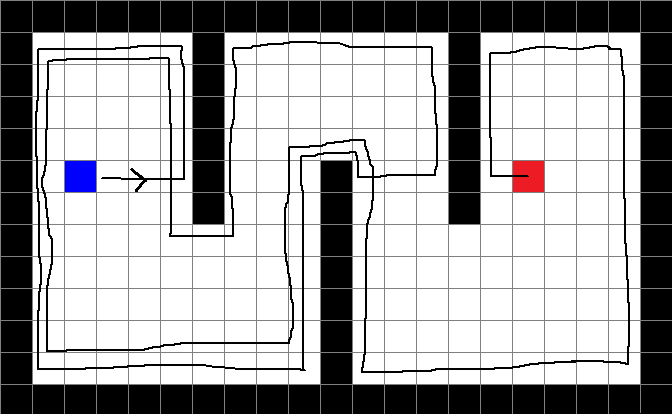
Στο περιβάλλον Α παρατηρούμε πως ο αλγόριθμος με την καλύτερη επίδοση, όσο αφορά το μέγεθος του μονοπατιού που ακολούθησε το κινητό ρομπότ μέχρι τον προορισμό του, είναι ο αλγόριθμος που προτείνεται. Επίσης παρατηρείται πως αν και ο Alg1 είναι μια βελτιστοποίηση του Bug2 το μέγεθος του μονοπατιού που διέσχισαν είναι το ίδιο. Το ίδιο συμβαίνει και με τους Class1 και Alg2 οι οποίοι και αυτοί διέσχισαν το ίδιο μέγεθος μονοπατιού. Ο πιο απογοητευτικός αλγόριθμος με διαφορά, όπως και ήταν αναμενόμενο, είναι ο Bug1. Ο Bug1 διένυσε την διπλάσια απόσταση από όλους τους υπόλοιπους αλγόριθμους ενώ από τον αλγόριθμο που προτείνεται, την τριπλάσια του. Αξιοσημείωτο η μεγάλη βελτίωση που αποφέρει ως προς το μέγεθος του μονοπατιού ο αλγόριθμος που προτείνεται.

**Πείραμα 2:**

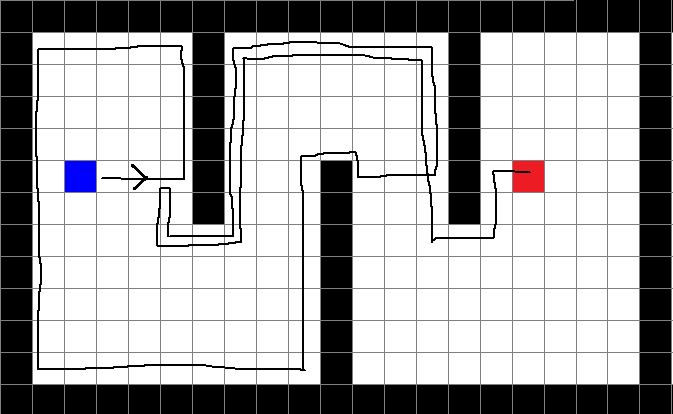
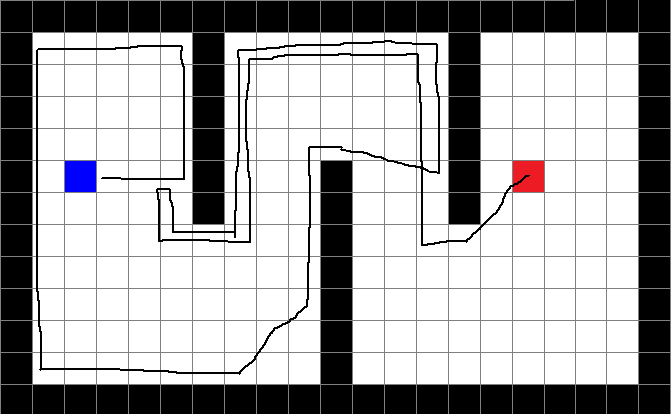
Για το δεύτερο πείραμα χρησιμοποιήθηκε περιβάλλον με αυτό του πειράματος 1, αλλά αυτήν την φορά προστέθηκαν επιπρόσθετα εμπόδια αλλά με την ίδια λογική. Σκοπός είναι να επισημανθεί πως λειτουργούν οι αλγόριθμοι σε τέτοιου είδους περιβάλλοντα . Στην πιο κάτω εικόνα φαίνεται το περιβάλλον για την το πείραμα 2



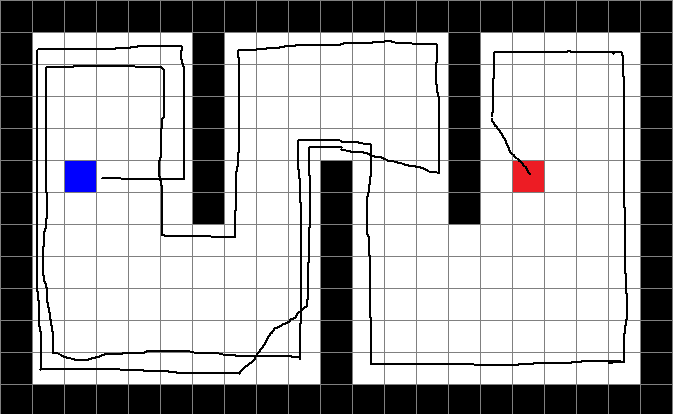
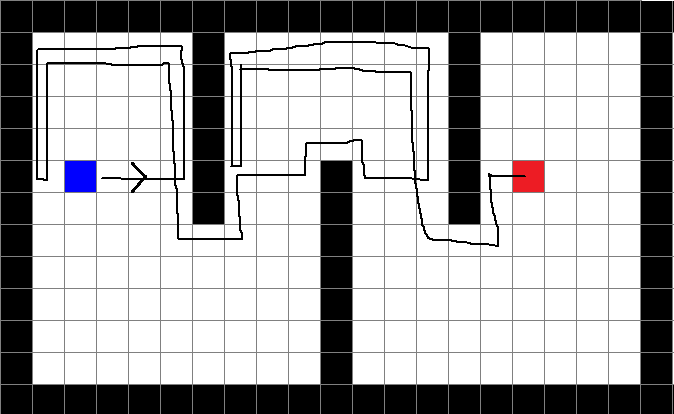
Σχήμα 6.4: Περιβάλλον Α



Bug1 Bug2



Alg1 Alg2



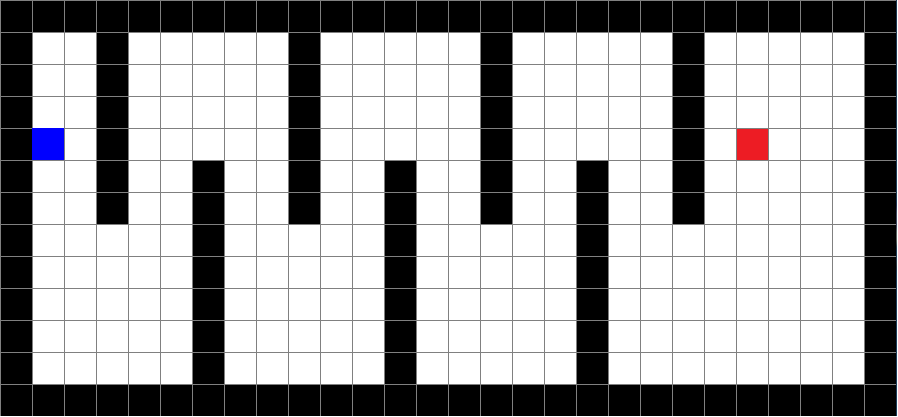
Class1 Own

Σχήμα 6.5: Διαδρομές των Ρομπότ στο Περιβάλλον Β

Σχήμα 6.6: Αποτελέσματα για το Περιβάλλον Β

Τρέχοντας τους αλγόριθμους για το περιβάλλον Β παρατηρούμε πως και πάλι ο αλγόριθμος Bug1 διασχίζει το μεγαλύτερο μονοπάτι από την αρχική του θέση μέχρι τον προορισμό. Δεύτερη χειρότερη απόδοση έχει ο Class1. Επίσης παρατηρείται πως σε τέτοιου είδους περιβάλλοντα ο Alg1 βελτιστοποιεί κατά πολύ τον Bug2 όπως επίσης και ο Alg2 τον Alg 1. Και σε αυτήν την περίπτωση ο αλγόριθμος που προτείνεται έχει την καλύτερη απόδοση σε σύγκριση με τους άλλους. Να σημειωθεί πως η μικρή διαφορά που έχει ο αλγόριθμος Alg1 με τον Alg2 οφείλεται στην Μ-Γραμμή αφού ο Alg2 δεν είναι περιορισμένος να κινείτε σε αυτήν.

**Πείραμα 3:**

Στο πείραμα 3 το περιβάλλον που χρησιμοποιήθηκε είναι παρόμοιο με τα προηγούμενα περιβάλλοντα αλλά αυτή την φορά ο αριθμός των εμποδίων είναι πολύ μεγαλύτερος. Σκοπός είναι να δούμε την συμπεριφορά όλων των αλγόριθμων σε τέτοιου είδους περιβάλλοντα και πώς συνδέεται η αποτελεσματικότητα τους ως προς το μέγεθος του μονοπατιού που θα ακολουθήσουν με τον αριθμό των εμποδίων. Στο πιο κάτω σχήμα παρουσιάζεται το περιβάλλον για το Πείραμα 3.

Σχήμα 6.7: Περιβάλλον Γ

**Αποτελέσματα Πειράματος 3:**

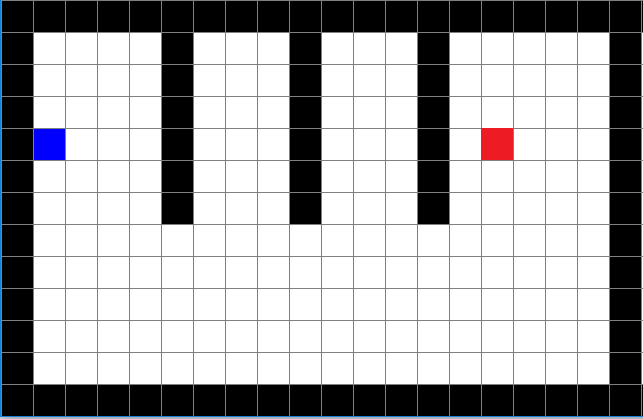
Σχήμα 6.8: Αποτελέσματα για το Περιβάλλον Γ

Στο πιο πάνω σχήμα βλέπουμε τα αποτελέσματα του περιβάλλοντος Γ για το πείραμα 3.

Σε αυτήν την περίπτωση ο χειρότερος αλγόριθμος είναι ο Bug2 και δεύτερος χειρότερος είναι ο Class1. Αξιοσημείωτο είναι πως αν και ο αλγόριθμος Bug2 δημιουργήθηκε για να βελτιώσει τον Bug1 στο συγκεκριμένο περιβάλλον αποδίδει καλύτερα ως προς το μήκος της διαδρομής που ακολουθεί το ρομπότ. Και πάλι το μικρότερο μονοπάτι δημιουργείται από τον αλγόριθμο που προτείνεται. Ο αλγόριθμος Alg1 βελτιώνει τον αλγόριθμο Bug 2 και ο Alg2 βελτιώνει τον Alg1. Το περιβάλλον Γ όπου, όπου χρησιμοποιήθηκε για να αναλυθεί πώς αντιδρούν οι αλγόριθμοι σε τέτοιου είδος περιβάλλοντα μας δείχνει πως ο αλγόριθμος που προτείνεται σε σχέση με τον Bug2 και τον Alg1, που παίρνει βασικά στοιχεία από αυτούς, η διάφορά στο μήκος του μονοπατιού που σχεδιάζεται είναι αυξημένη με μεγαλύτερο αριθμό εμποδίων.

**Πείραμα 4:**

Για το πείραμα 5 χρησιμοποιήθηκε ένα περιβάλλον λίγο παρόμοιο με τα προηγούμενα περιβάλλοντα, αλλά αυτή τη φορά απουσιάζουν τα εμπόδια στην κάτω μεριά του χάρτη. Η επιλογή αυτού του περιβάλλοντος έγινε με σκοπό να παρατηρηθεί πώς αντιδρούν οι αλγόριθμοι όταν ναι μεν χρειάζεται να επιστρέψουν πίσω λόγω κάποιου εμποδίου μόνο μια φορά σε ένα περιβάλλον με ένα σύνολο από εμπόδια. Στο παρακάτω σχήμα παρουσιάζεται το περιβάλλον Δ.



Σχήμα 6.9: Περιβάλλον Δ

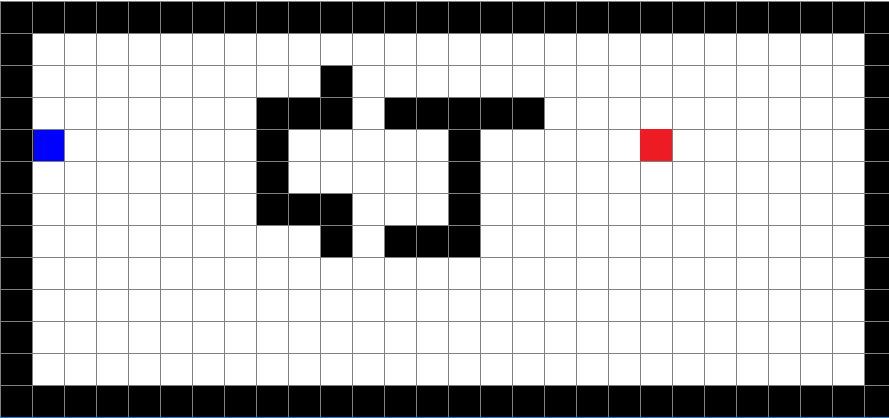
**Αποτελέσματα Πειράματος 4:**

Σχήμα 6.10:Αποτελέσματα για το Περιβάλλον Δ

Παρατηρείται πως και πάλι ο χειρότερος αλγόριθμος ως προς το μέγεθος του μονοπατιού που δημιουργείται είναι ο αλγόριθμος Bug1 ενώ δεύτερος χειρότερος είναι ο αλγόριθμος Class1. Συντομότερη διαδρομή σχεδιάζουν οι αλγόριθμοι Bug2 και Alg1 ενώ ο αλγόριθμος Alg2 που δημιουργήθηκε για βελτίωση του Alg1 σε αυτό το περιβάλλον δεν αποδίδει καλύτερα, αφού το μονοπάτι που σχεδιάστηκε είναι μεγαλύτερο σε μήκος από τον Alg1. Τέλος, ο αλγόριθμος που προτείνεται αν και σε όλα τα προηγούμενα πειράματα έβρισκε την βέλτιστη λύση σε σχέση με τους υπολοίπους, σε αυτό το περιβάλλον δεν βελτιστοποιεί τους υπόλοιπος και το μήκος του μονοπατιού που ακολούθησε είναι σχεδόν το ίδιο με αυτό του Alg2. Ο λόγος είναι επειδή στο συγκεκριμένο περιβάλλον οι υπόλοιποι αλγόριθμοι καθώς εκτελούν την περιμετρική τους κίνηση γύρω από το εμπόδιο δεν βρίσκουν ξανά κάποιο εμπόδιο το οποίο να τους φέρει ξανά στην Μ-Γραμμή πριν φτάσει στον προορισμό, σε αντίθεση με τον αλγόριθμο που προτείνεται επειδή συνεχώς επιστρέφει πίσω στο σημείο συνάντησης με το εμπόδιο και αναγκάζεται να βρίσκει συνεχώς εμπόδια.

**Πείραμα 5:**

Για το πείραμα 5, το περιβάλλον που χρησιμοποιήθηκε αποτελείται από απλά εμπόδια που δεν αναγκάζουν το ρομπότ να πάρει κατεύθυνση αντίθετη με την αρχική του. Σκοπός του συγκεκριμένου πειράματος είναι να παρατηρηθεί η συμπεριφορά των αλγόριθμων σε ένα περιβάλλον απλό, χωρίς κάποια εμπόδια που να θέτουν την λειτουργία των αλγόριθμων σε μια πολύπλοκη εξερεύνηση των εμποδίων. Στο πιο κάτω σχήμα παρουσιάζεται το περιβάλλον Ε.



Σχήμα 6.11: Περιβάλλον Ε

**Αποτελέσματα Πειράματος 5**

Σχήμα 6.12:Αποτελέσματα για το Περιβάλλον Ε

Στο πιο πάνω σχήμα παρουσιάζονται τα αποτελέσματα του Πειράματος 5. Σε αυτό το περιβάλλον χρησιμοποιήθηκαν απλά εμπόδια και παρατηρείτε πως για ακόμα μια φορά ο χειρότερος σε απόδοση αλγόριθμος, είναι o Bug1. Ο αλγόριθμος Bug 2 βελτιώνει τον Bug1, αλλά ο Alg1 δεν βελτιώνει τον Bug2, ενώ επίσης ούτε ο αλγόριθμος που προτείνεται δίνει κάποια βελτιστοποίηση. Το συντομότερο μονοπάτι σε αυτά τα είδους περιβάλλοντα, το σχεδιάζουν ο αλγόριθμος Class1 και Alg2. Ο λόγος είναι επειδή οι 2 τελευταίοι αλγόριθμοι δεν περιορίζουν την κίνησή τους στην M-Γραμμή.

**6.3 Συμπεράσματα**

Μέσα από την εκτέλεση των πιο πάνω πειραμάτων διαπιστώνουμε πως κανένας αλγόριθμος δεν έχει την μεγαλύτερη αποτελεσματικότητα ως προς το μέγεθος του μονοπατιού που θα σχεδιάσει για οποιοδήποτε περιβάλλον. Ακόμα και ο Bug1 που θεωρείται ως ο πιο μη αποδοτικός αλγόριθμος υπάρχουν περιπτώσεις όπου αποδίδει καλύτερα από άλλους, όπως για παράδειγμα στο περιβάλλον Γ. Στο συγκεκριμένο περιβάλλον παρατηρείται πως ο αλγόριθμος Bug1 σχεδιάζει μια συντομότερη διαδρομή από τους Bug2 και Class1. Επιπρόσθετα αν και ο Alg1 είναι μια βελτιστοποίηση του Bug2 υπάρχουν περιπτώσεις όπου δεν τον βελτιώνει, αλλά παράλληλα ούτε τον κάνει χειρότερο. Ο αλγόριθμος Alg2 σαν ένας αλγόριθμος που δημιουργήθηκε για να βελτιώσει τον Alg2 στο Περιβάλλον Δ αυτό δεν γίνεται. Έτσι σαν γενικό συμπέρασμα μπορούμε να πούμε πως κανένας αλγόριθμος δεν είναι ο καταλληλότερος για όλες τις περιπτώσεις.

Όσο αφορά τον αλγόριθμο ο οποίος προτείνεται, παρατηρείται πως στα περιβάλλοντα Α, Β και Γ είναι ο αποδοτικότερος από όλους όσο αφορά το μέγεθος του μονοπατιού που θα ακολουθήσει το ρομπότ. Όμως και πάλι υπάρχουνε περιβάλλοντα όπου ο αλγόριθμος δεν βελτιστοποιεί τους Bug2 και Alg1 αλλά ούτε έχει χειρότερη απόδοση, όπως για παράδειγμα το περιβάλλον Ε, το οποίο αποτελείται από απλά εμπόδια. Τέλος σε περιβάλλοντα που έχουν την μορφή αυτού του περιβάλλοντος Δ, ο αλγόριθμος αυτός έχει χειρότερη απόδοση από τους υπόλοιπους.

Σε γενικέ γραμμές, μπορούμε να πούμε πως, για περιβάλλοντα όπου αποτελούνται απλά εμπόδια, χωρίς το ρομπότ να χρειάζεται να πάρει κατεύθυνση αντίθετη με την αρχική του, οι καταλληλότεροι αλγόριθμοι είναι οι Alg2 και Class1, οι οποίοι έχουν προβάδισμα λόγω της μη χρήσης της M-Γραμμής. Εάν όμως το ρομπότ χρειαστεί να κατευθυνθεί αντίθετα από την αρχική του πορεία τότε ο αλγόριθμος Class1 ενδέχεται να δημιουργήσει ίδια μονοπάτια με αποτέλεσμα η αποδοτικότητα του να μειώνεται κατακόρυφα. Για περιβάλλοντα παρόμοια με αυτό του Περιβάλλοντος Δ ο καλύτερος αλγόριθμος είναι ο Bug 2 και ο Alg1. Τέλος ο αλγόριθμος που προτείνεται υπερισχύει τους υπόλοιπους σε όλα τα περιβάλλοντα όπου μοιάζουν με αυτά των Α, Β και Γ.

**Κεφάλαιο 7**

**Συμπεράσματα**

7.1 Περίληψη 66

7.2 Μελλοντικές εργασίες 67

**7.1 Περίληψη**

Η πτυχιακή έρευνα αυτή, έχει κάνει μελέτη στο πρόβλημα της σχεδίασης διαδρομής ενός κινητού ρομπότ και έχει ασχοληθεί με την μελέτη των αλγόριθμων της οικογένειας Bug, οι οποίοι ασχολούνται με δυναμικά και άγνωστα περιβάλλοντα. Μέσα από την μελέτη αυτή των αλγόριθμων επινοήθηκε και ένας νέος αλγόριθμος, ο οποίος προτείνεται από τον συγγραφέα. Ο αλγόριθμος αυτός έχει βασικά στοιχεία από τους αλγόριθμους Bug2 και Alg1 και έχει σκοπό την βελτιστοποίησή τους. Επίσης, με την εκπόνηση της συγκεκριμένης πτυχιακής έρευνας, έχει υλοποιηθεί ένας προσομοιωτής στην γλώσσα προγραμματισμού Java που υποστηρίζει τους αλγόριθμους που μελετηθήκαν καθώς επίσης και τον αλγόριθμο που προτείνεται. Τέλος έχουν γίνει κάποια πειράματα για την σύγκριση της αποδοτικότητας όλων των αλγόριθμων βάσει του μεγέθους της διαδρομής που σχεδιάζεται, αλλά και την σύγκριση του αλγόριθμου που προτείνεται στην πτυχιακή εργασία, με τους υπόλοιπους. Τα αποτελέσματα αυτά μας οδηγούν στο συμπέρασμα πως κανένας αλγόριθμος δεν είναι ο καταλληλότερος για όλα τα περιβάλλοντα, ενώ ο αλγόριθμος που προτείνεται βελτιστοποιεί τους υπόλοιπους αλγόριθμους σε μερικές περιπτώσεις όπως αυτές αναλύθηκαν στο Κεφάλαιο 6.

**7.2 Μελλοντικές εργασίες**

Αν και με την εκπόνηση της πτυχιακής εργασίας έχουν γίνει συγκρίσεις μεταξύ διάφορων αλγορίθμων και έχει αναπτυχθεί και προσομοιωτής που να εκτελεί τους αλγορίθμους που μελετήθηκαν και παρουσιάστηκαν μελλοντικές εργασίες μπορούν να ασχοληθούν με τα εξής:

1. Υλοποίηση των αλγορίθμων σε πιο ρεαλιστικά περιβάλλοντα όπου κάθε εμπόδιο να μπορεί να έχει οπουδήποτε σχήμα, ακόμα και με καμπύλες.
2. Υλοποίηση των αλγόριθμων με αισθητήρες μεγαλύτερης εμβέλειας, που θα δίνουν την δυνατότητα στο ρομπότ να αναγνωρίζει καλύτερα το περιβάλλον στο οποίο ταξιδεύει με αποτέλεσμα να μπορεί να πάρει αποφάσεις όπου να επιφέρουν καλύτερα αποτελέσματα ως προς την σχεδίαση μιας διαδρομής για ένα κινητό ρομπότ.

**Βιβλιογραφία**

[1] H. Abelson and E. DiSessa, “The Turtle Geometry”, MIT Press, Cambridge, Pages 176-199, 1980.

[2] T. Bräunl, “Embedded Robotics”, 2nd Edition, Springer, Berlin, 2006

[3] B. Bullock, D. Keirsey, J. Mitchell, T. Nussmeier. and D. Tseng, “Autonomous vehicle control: An overview of the Hughes project”, Proceedings of the IEEE Computing Society Conference on Trends and Applications, 1983.

[4] H. Choset, K. Lynch, K. Hutchinson, G. Kantor, W. Burgard, L. Kavarki and S. Thrun, “Principles of Robot Motion: Theory, Algorithms and Implementations”, MIT, Cambridge, MA, 2005

[5] G. Foux, M. Heymann, A. Bruckstein, “Two-dimensional robot navigation among unknown stationary polygonal obstacles”, IEEE Transactions on Robotics and Automation 9, 96–102 (1993)

[6] Y. Horiuchi and H. Noborio, “Evaluation of Path Length Made in Sensor-Based Path-Planning with the Alternative Following”, Proc. of the 2001 IEEE International Conference on Robotics and Automation, Pages 1728-1735, May 2001.

[7] R. Jarvis, “Collision-Free Trajectory Planning Using the Distance Transforms”, Mechanical Engineering Trans. of the Institution of Engineers Australia, September 1985

[8] W. Ko, L. Senevieatne, S. Earles, “Space representation and map building-A triangulation model to pathplanning with obstacle avoidance”, Proceedings of the 1993 IEEE/RSJ International Conference on Intelligent Robots and Systems, vol. 3, pp. 2222–2227 (1993)

[9] R. Langer, L. Coelho and G. Oliveira, “K-Bug, a new bug approach for mobile robot's path planning”, IEEE International Conference on Control Applications, pp. 403-408, October 2007.

[10] S. LaValle, “Planning Algorithms”, University of Illinois Press, 2003

[11] M. Littman, T. Dean, and L. Kaelbling, “Markov games as a framework for multi-agent reinforcement learning”, Eleventh International Conference on Machine Learning. San Francisco, pp. 157-163, 1994.

[12] C. Lucas, V. Lumelsky and A. Stepanov. Comments on “Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment”. IEEE. Transactions on Robotics and Automation, pp. 511-512, 1988.

[13] V. Lumelsky and A. Stepanov, “Path planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape,” Algorithmica, vol. 2, pp. 403-430, 1987.

[14] V. Lumelsky and P. Stepanov, “Effect of Uncertainty on Continuous Path Planning for an Autonomous Vehicle”, Proceedings of the 23rd Conference on Decision and Control, pp. 1616-1621, December 1984.

[15] V. Lumelsky and A. Stepanov, “Dynamic Path Planning For A Mobile Automaton With Limited Information On The Environment”, IEEE Transactions On Automatic Control, Vol. AC.31, No. 11, Nov. 1986.

[16] V. Lumelsky, “Continuous Motion Planning in Unknown Environment for a 3D Cartesian Robot Arm”, IEEE International Conference on Robotics and Automation, San Francisco, April 1986

[17] V. Lumelsky, “Dynamic Path Planning for a Planar Articulated Robot Arm Moving Amidst Unknown Obstacles”, Automatica, Vol.23 No.5, pp. 551-570, September 1987.

[18] V. Lumelsky, “Effect of Kinematics on Dynamic Path Planning for Planar Robot Arms Moving Amidst Unknown Obstacles”, IEEE Journal of Robotics and Automation. June 1987.

[19] V. Lumelsky and T. Skewis, “Incorporating range sensing in the robot navigation function”, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, Pages 1058-1068, 1990

[20] V. J. Lumelsky and T. Skewis, “Incorporating range sensing in the robot navigation function”, IEEE Transactions on Systems, Man and Cybernetics, Vol. 20, Pages 1058-1068, 1990

[21] H. Mitchell, “An algorithmic approach to some problems in terrain navigation”, Artificial Intelligence, vol. 37, pages 171–201, Dec 1988

[22] C. O’Dunlaing, and C. Yap, “A retraction method for planning the motion of a disc”, Journal of Algorithms, vol. 6, pages 104–111, 1982 [6] D. Payton, J. Rosenblatt and D. Keirsey, “Grid-based mapping for autonomous mobile robot”, Robotics and Autonomous Systems, vol. 11, pages 13–21, 1993

[23] O. Ore, “Theory of Graphs”, Providence, RI: American Mathematics Society, 1962.

[24] D. Payton, J. Rosenblatt and D. Keirsey, “Grid-based mapping for autonomous mobile robot”, Robotics and Autonomous Systems, vol. 11, pages 13–21, 1993

[25] J. Pearl, “Heuristics”, Addison-Wesley, 1984.

[26] L. Sciavicco and B. Siciliano, “Modelling and Control of Robot Manipulators”, Springer, 2000.

[27] A. Sankaranarayanan and M. Vidyasagar, “A New Path Planning Algorithm For Moving A Point Object Amidst Unknown Obstacles In A Plane”, IEEE Conference on Robotics and Automation, Pages 1930-1936, 1990

[28] [4] A. Sankaranarayanan and M. Vidyasagar, “Path Planning for Moving a Point Object Amidst Unknown Obstacles in a Plane: A New Algorithm and a General Theory for Algorithm Development”, Proc. of the IEEE Int. Conf. on Decision and Control, Pages 1111-1119, 1991.

[29] T. Skewis and V. Lumelsky, “Experiments with a Mobile Robot Operating in a Cluttered Unknown Environment”, International Conference on Robotics and Automation, pp. 1482-1487, 1992.

[30] M. Trevisan, M. A. P. Idiart, E. Prestes and P. M. Engel, “Exploratory Navigation Based on Dynamical Boundary Value Problems”, Journal of Intelligent and Robotic Systems, vol. 45, no. 2, pp. 101-114, Feb. 2006.

[31] Y. Wang, P. Sillitoe, J. Mulvaney, “Mobile Robot Path Planning in Dynamic Environments”, IEEE International Conference on Robotics and Automation, pages 71-76, May 2007

[32] R. Zhou and E. Hansen, “Multiple sequence alignment using A\*”, Proc. of the National Conference on Artificial Intelligence, 2002

[33] S. Kim, J. Russell and K. Koo, “Construction Robot Path-Planning for Earthwork Operations”, Journal of Computing in Civil Engineering, pp. 97-104, 2003.