ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΑΤΡΩΝ
UNIVERSITY OF PATRAS

Πολυτεχνική Σχολή
Τμήμα Μηχανικών Η/Υ & Πληροφορικής

Διπλωματική Εργασία

Δρομολόγηση Αυτονόμων Εναέριων Οχημάτων

Σάββας Στυλιανού
Α.Μ. 1069661

Επιβλέπων
Καθηγητής Ζαρολιάγκης Χρήστος

Συνεπιβλέπων
Αναπληρωτής Καθηγητής Κοντογιάννης Σπυρίδων

Μέλος Επιτροπής Αξιολόγησης
Καθηγητής Σιούτας Σπυρίδων

Πάτρα, 2024

Στην οικογένεια μου.

Περίληψη

Η διπλωματική εργασία πραγματεύεται τη μελέτη και ανάπτυξη αλγορίθμων για τον προγραμματισμό διαδρομών UAVs σε περιβάλλοντα με εμπόδια και αβεβαιότητες. Με την ταχεία ανάπτυξη των μη επανδρωμένων εναέριων οχημάτων (UAVs), η ανάγκη για αποδοτικούς αλγορίθμους που μπορούν να διαχειριστούν την πλοήγηση σε περίπλοκα περιβάλλοντα έχει αυξηθεί σημαντικά. Η αυτόνομη πλοήγηση UAVs αντιμετωπίζει προκλήσεις όπως η αποφυγή εμποδίων και η εξεύρεση της βέλτιστης διαδρομής σε πραγματικό χρόνο. Η εργασία αυτή συμβάλλει στην επίλυση αυτού του προβλήματος, επικεντρώνοντας την έρευνα και την ανάπτυξη αλγορίθμων που χρησιμοποιούνται σε προσομοιωμένα περιβάλλοντα.

Η κύρια συμβολή της εργασίας περιλαμβάνει την ανάπτυξη και εφαρμογή τριών αλγορίθμων, συμπεριλαμβανομένων των PRM και Rectangles, καθώς και του νέου αλγορίθμου APPATTrectangles, ο οποίος σχεδιάστηκε για να βελτιώσει την αποδοτικότητα και ακρίβεια στην πλοήγηση. Οι αλγόριθμοι αυτοί εφαρμόστηκαν σε διάφορα πειραματικά σενάρια και τα αποτελέσματα έδειξαν τη σημαντική βελτίωση στην αποδοτικότητα πλοήγησης UAVs.

Η διπλωματική εργασία αξιολογεί τη βελτιστοποίηση της διαδρομής και τον υπολογιστικό χρόνο, παρέχοντας μια συνολική ανάλυση της απόδοσης των αλγορίθμων σε περιβάλλοντα με διαφορετικά επίπεδα πολυπλοκότητας και αβεβαιότητας.

Abstract

This thesis addresses the study and development of algorithms for UAV path planning in environments with obstacles and uncertainties. With the rapid development of Unmanned Aerial Vehicles (UAVs), the need for efficient algorithms that can manage navigation in complex environments has significantly increased. Autonomous UAV navigation faces challenges such as obstacle avoidance and finding the optimal path in real-time. This work contributes to solving this problem by focusing on the research and development of algorithms used in simulated environments.

The main contribution of this thesis includes the development and implementation of three algorithms, including PRM and Rectangles, as well as the new APPATTrectangles algorithm, which was designed to improve efficiency and accuracy in navigation. These algorithms were applied in various experimental scenarios, and the results demonstrated significant improvements in UAV navigation efficiency.

The thesis evaluates path optimization and computational time, providing a comprehensive analysis of the algorithms' performance in environments with varying levels of complexity and uncertainty.

Περιεχόμενα

1 ..... ***11***

Εισαγωγή ***11***

1.1 Σημασία του προβλήματος **11**

1.2 Στόχοι της Εργασίας **12**

1.3 Συνεισφορά **13**

1.4 Διάρθρωση της Διπλωματικής Εργασίας **13**

2 ..... ***15***

Γνώσεις Υποβάθρου ***15***

2.1 Εισαγωγή στον Προγραμματισμό Διαδρομών UAV **15**

2.2 Ανάλυση βασικών μεθόδων και αλγορίθμων **16**

2.2.1 Μέθοδοι Βασισμένες σε Γραφήματα (Graph-based Methods) 16

2.2.2 Μέθοδοι Βασισμένες σε Δειγματοληψία (Sampling-based Methods) 18

2.2.3 Μέθοδοι Δυναμικού Πεδίου (Potential Field Methods) 19

2.2.4 Μέθοδοι Ευφυούς Βελτιστοποίησης (Intelligent Optimization Methods) 19

2.2.5 Συμπεράσματα 21

3 ..... ***22***

Αλγόριθμοι ***22***

3.1 Ο αλγόριθμος PRM με βρόχο **22**

3.1.1 Ο Αλγόριθμος 22

3.1.1 Παράδειγμα εκτέλεσης Αλγορίθμου 23

3.1.2 Πλεονεκτήματα και Περιορισμοί 24

3.1.3 Συμπεράσματα 24

3.2 Ο αλγόριθμος Rectangles **25**

3.2.1 Ο Αλγόριθμος 25

3.2.2 Παράδειγμα εκτέλεσης Αλγορίθμου 26

3.2.3 Πλεονεκτήματα και Μειονεκτήματα 27

3.2.3.1 Πλεονεκτήματα 27

3.2.3.2 Μειονεκτήματα 27

3.2.4 Συμπεράσματα 27

3.3 Ο αλγόριθμος APPATTrectangles **28**

3.3.1 Ο Αλγόριθμος 28

3.3.2 Παράδειγμα εκτέλεσης Αλγορίθμου 29

Βήμα 5 Σύνδεση σημείων ενδιαφέροντος 30
3.3.3 Πλεονεκτήματα και Μειονεκτήματα 31
3.3.4 Συμπεράσματα 31

4 ***..... 33***

Υλοποίηση αλγορίθμων 33

4.1 Περιγραφή Περιβάλλοντος Ανάπτυξης 33
4.1.1 Βιβλιοθήκες που Χρησιμοποιήθηκαν 33
4.1.1.1 Βιβλιοθήκη Numpy 33
4.1.1.2 Βιβλιοθήκη NetworkX 34
4.1.1.3 Βιβλιοθήκη Matplotlib 34
4.1.1.4 Βιβλιοθήκη Time 35
4.1.2 Περιγραφή της Υλοποίησης 35
4.1.2.1 Προετοιμασία Δεδομένων 35
Δημιουργία του Περιβάλλοντος 35
Φόρτωση Εμποδίων 36
Φόρτωση Περιβάλλοντος από Αρχείο 36
4.1.2.2 Δημιουργία Γραφημάτων 36
Δημιουργία Γραφήματος και Προσθήκη Κόμβων 36
Σύνδεση Κόμβων με Ακμές 37
4.1.2.3 Απεικόνιση Αποτελεσμάτων 38
4.1.2.4 Μέτρηση Χρόνου Εκτέλεσης 38

4.2 Βασικές συναρτήσεις 39
4.2.1 Συνάρτηση get_closest_nodes 39
4.2.2 Συνάρτηση get_obstacle_collided_id 40
4.2.3 Συνάρτηση is_valid_edge 40
4.2.4 Συνάρτηση get_obstacle 40
4.2.5 Συνάρτηση get_nodes 41
4.2.6 Συνάρτηση add_node 42
4.2.7 Συνάρτηση add_new_obstacle_found 42

4.3 Υλοποίηση του Αλγορίθμου PRM 42

4.4 Υλοποίηση του Αλγορίθμου Rectangles 43

4.5 Υλοποίηση του Αλγορίθμου APPATTrectangles 43

4.6 Συγκριτική Ανάλυση και Προκλήσεις 44
PRM (Probabilistic Roadmap) 44
Rectangles 45
APPATTrectangles 45
4.6.1 Προκλήσεις Κατά την Υλοποίηση 45

5 ***..... 47***

Πειραματική αξιολόγηση 47

5.1 Πειραματικό Περιβάλλον 47
5.1.1 Εργαλεία και Τεχνολογίες 47

5.1.2 Τεχνικά Χαρακτηριστικά του Υπολογιστή 48
5.1.3 Περιβάλλον Ανάπτυξης (IDE) 48

5.2 Πειραματικά δεδομένα 48
5.2.1 Δημιουργία Πειραματικών Δεδομένων 48
5.2.1.1 Συνάρτηση is_overlapping 48
5.2.1.2 Αρχικοποίηση μεταβλητών 49
5.2.1.3 Προσθήκη τυχαίων εμποδίων 49
5.2.2 Αποθήκευση Πειραματικών Δεδομένων 49

5.3 Πειραματικά Αποτελέσματα 50
5.3.1 Σενάριο 1. Μικρή περιοχή. Λίγα, μικρά, αραιά εμπόδια 50
5.3.1.1 Αποτελέσματα 50
5.3.2 Σενάριο 2. Μικρή περιοχή. Λίγα, μεγάλα, πυκνά εμπόδια 52
5.3.2.1 Αποτελέσματα 53
5.3.3 Σεναρίο 3. Μικρή περιοχή. Πολλά, μικρά, πυκνά εμπόδια 55
5.3.3.1 Αποτελέσματα 55
5.3.4 Σενάριο 4. Μικρή περιοχή. Πολλά, διάφορων μεγεθών, πυκνά εμπόδια 57
5.3.4.1 Αποτελέσματα 58
5.3.5 Σενάριο 5. Μεγάλη περιοχή. Λίγα, μικρά, αραιά εμπόδια 60
5.3.5.1 Αποτελέσματα 60
5.3.6 Σενάριο 6. Μεγάλη περιοχή. Λίγα, μεγάλα, πυκνά εμπόδια 62
5.3.6.1 Αποτελέσματα 63
5.3.7 Σενάριο 7. Μεγάλη περιοχή. Πολλά, μικρά, πυκνά εμπόδια 64
5.3.7.1 Αποτελέσματα 64
5.3.8 Σενάριο 8. Μεγάλη περιοχή. Πολλά, διάφορων μεγεθών, πυκνά εμπόδια 66
5.3.8.1 Αποτελέσματα 67
5.3.9 Συμπεράσματα 69
5.3.9.1 PRM 69
5.3.9.2 Rectangles 69
5.3.9.3 APPATTrectangles 69

6 ***..... 71***

Συμπεράσματα και Προοπτικές 71
6.1.1 Συμπεράσματα 71

6.2 Προοπτικές 72
6.2.1 Βελτιώσεις με Περισσότερο Χρόνο 72
6.2.2 Συνέχιση της Έρευνας από Άλλους Ερευνητές 72

Βιβλιογραφία- Αναφορές 75

Λίστα Εικόνων

Εικόνα 1 23
Εικόνα 2 23
Εικόνα 3 24
Εικόνα 4 26
Εικόνα 5 26
Εικόνα 6 27
Εικόνα 7 29
Εικόνα 8 29
Εικόνα 9 30
Εικόνα 10 30
Εικόνα 11 31
Εικόνα 12 31
Εικόνα 13 50
Εικόνα 14: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 51
Εικόνα 15 53
Εικόνα 16: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 53
Εικόνα 17 55
Εικόνα 18: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 56
Εικόνα 19 58
Εικόνα 20: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 58
Εικόνα 21 60
Εικόνα 22: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 61
Εικόνα 23 62
Εικόνα 24: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 63
Εικόνα 25 64
Εικόνα 26: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 65
Εικόνα 27 67
Εικόνα 28: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικονα APPATTrectangles 67

Λίστα Πινάκων

Πίνακας 1: Παράμετροι σεναρίου 1 51
Πίνακας 2: Αποτελέσματα 51
Πίνακας 3: Παράμετροι σεναρίου 2 54
Πίνακας 4: Αποτελέσματα 54
Πίνακας 5: Παράμετροι σεναρίου 3 56
Πίνακας 6: Αποτελέσματα 56
Πίνακας 7: Παράμετροι σεναρίου 4 59
Πίνακας 8: Αποτελέσματα 59
Πίνακας 9: Παράμετροι σεναρίου 5 61
Πίνακας 10: Αποτελέσματα 61
Πίνακας 11: Παράμετροι σεναρίου 6 63
Πίνακας 12: Αποτελέσματα 63
Πίνακας 13: Παράμετροι σεναρίου 7 65
Πίνακας 14: Αποτελέσματα 65
Πίνακας 15: Παράμετροι σεναρίου 8 68
Πίνακας 16: Αποτελέσματα 68

Συντομογραφίες

PRM	Probabilistic Roadmap Method
UAV	Unmanned Aerial Vehicle
RRT	Rapidly-exploring Random Tree
APPATT	Algorithm for Probabilistic Path and Trajectory Tuning
VFH	Vector Field Histogram
PSO	Particle Swarm Optimization

1

Εισαγωγή

1.1 Σημασία του προβλήματος

Η χρήση μη επανδρωμένων εναέριων οχημάτων (UAVs) έχει αυξηθεί ραγδαία τα τελευταία χρόνια, επηρεάζοντας σημαντικά πολλούς τομείς, όπως την επιθεώρηση της κυκλοφορίας, την παράδοση αγαθών, την επιτήρηση και τις επιχειρήσεις διάσωσης. Η δυνατότητα των UAVs να πλοηγούνται αυτόνομα και με ασφάλεια σε περιβάλλοντα με εμπόδια είναι ζωτικής σημασίας για την επιτυχή εκτέλεση των αποστολών τους. Ο σχεδιασμός της διαδρομής αποτελεί ένα από τα σημαντικότερα προβλήματα που πρέπει να αντιμετωπιστούν για την αποτελεσματική και ασφαλή χρήση των UAVs.

Ο προγραμματισμός διαδρομών (path planning) είναι ένα πεδίο έρευνας που ασχολείται με την εύρεση βέλτιστων διαδρομών για την μετακίνηση οχημάτων ή ρομπότ από ένα αρχικό σημείο σε ένα σημείο στόχο, αποφεύγοντας τα εμπόδια. Στα πλαίσια της παρούσας διπλωματικής εργασίας, το πρόβλημα αυτό εξετάζεται στον τομέα των UAVs, όπου η αποδοτική και ασφαλής πλοήγηση σε δισδιάστατα περιβάλλοντα είναι ιδιαίτερα κρίσιμη.

Η βιβλιογραφία προσφέρει 2 προσεγγίσεις για την επίλυση του προβλήματος του προγραμματισμού διαδρομών. Σύμφωνα με το άρθρο [1], οι υπάρχουσες μέθοδοι μπορούν να ταξινομηθούν σε δύο κύριες κατηγορίες: τις μεθόδους που βασίζονται στη θεωρία των γραφημάτων και τις μεθόδους που δεν βασίζονται σε αυτήν.

Οι αλγόριθμοι που βασίζονται στη θεωρία των γραφημάτων περιλαμβάνουν τους αλγορίθμους Dijkstra και A*, καθώς και διάφορες βελτιώσεις τους. Ο αλγόριθμος Dijkstra, είναι ένας κλασικός αλγόριθμος εύρεσης της συντομότερης διαδρομής σε ένα γράφημα με μη αρνητικά βάρη. Ο αλγόριθμος A*, επεκτείνει τον αλγόριθμο Dijkstra χρησιμοποιώντας μια ευρετική συνάρτηση για να καθοδηγήσει την αναζήτηση προς τον στόχο. Αυτοί οι αλγόριθμοι έχουν αποδειχθεί ιδιαίτερα αποτελεσματικοί σε πολλές περιπτώσεις, αλλά συχνά απαιτούν σημαντικούς υπολογιστικούς πόρους, καθιστώντας τους μη κατάλληλους για περιβάλλοντα με πολλές αβεβαιότητες ή με μεγάλη κλίμακα.

Οι μέθοδοι που δεν βασίζονται στη θεωρία των γραφημάτων περιλαμβάνουν αλγορίθμους όπως οι PRM (Probabilistic Roadmap Method) και RRT (Rapidly-exploring Random Tree). Ο αλγόριθμος PRM, χρησιμοποιεί δειγματοληψία για να δημιουργήσει έναν χάρτη δρόμων (roadmap) στον χώρο αναζήτησης και στη συνέχεια χρησιμοποιεί αυτόν τον χάρτη για να βρει

διαδρομές. Ο αλγόριθμος RRT, αναπτύσσει ένα δέντρο δειγματοληπτικών σημείων που εξερευνά γρήγορα τον χώρο αναζήτησης. Αυτοί οι αλγόριθμοι είναι πιο αποδοτικοί σε δυναμικά περιβάλλοντα και μπορούν να ανταπεξέλθουν καλύτερα σε πραγματικό χρόνο, αλλά δεν εξασφαλίζουν πάντα την εύρεση της βέλτιστης διαδρομής.

Η ανάπτυξη αποτελεσματικών αλγορίθμων πλοήγησης για UAVs είναι αυξημένης σημασίας για την ασφαλή και αποδοτική εκτέλεση αποστολών σε πυκνοκατοικημένες περιοχές ή σε περιβάλλοντα με πολλά εμπόδια. Οι απαιτήσεις για την πλοήγηση UAVs περιλαμβάνουν την αποφυγή συγκρούσεων, τη μείωση της κατανάλωσης ενέργειας και τη βελτίωση της ακρίβειας πλοήγησης.

Η παρούσα διπλωματική εργασία εξετάζει τις προαναφερόμενες απαιτήσεις, εστιάζοντας σε αλγορίθμους που συνδυάζουν τις προσεγγίσεις της θεωρίας των γραφημάτων και της δειγματοληψίας. Συγκεκριμένα, η εργασία εξετάζει τον αλγόριθμο APPATT (Algorithm for Probabilistic Path and Trajectory Tuning) [1] και τον αλγόριθμο Rectangles [2].

Ο αλγόριθμος APPATT χρησιμοποιεί μια συνδυαστική προσέγγιση που εκμεταλλεύεται τα πλεονεκτήματα των μεθόδων δειγματοληψίας και της ευρετικής αναζήτησης. Σύμφωνα με το άρθρο [1] αλγόριθμος APPATT μπορεί να παράγει διαδρομές υψηλής ποιότητας με ελάχιστο υπολογιστικό κόστος, καθιστώντας τον κατάλληλο για χρήση σε περιβάλλοντα με υψηλή πυκνότητα εμποδίων και αβεβαιότητες.

Ο αλγόριθμος Rectangles, από την άλλη, χρησιμοποιεί έναν πιο κλασικό προγραμματισμό βασισμένο σε γραφήματα για την αποφυγή εμποδίων. Σύμφωνα με το άρθρο [2], ο αλγόριθμος αυτός μπορεί να βελτιώσει την αποδοτικότητα των διαδρομών UAVs σε περιβάλλοντα με κανονικές ή προβλέψιμες κατανομές εμποδίων.

1.2 Στόχοι της Εργασίας

Ο κύριος στόχος αυτής της διπλωματικής εργασίας είναι η ανάπτυξη και αξιολόγηση αλγορίθμων για τον προγραμματισμό διαδρομών UAVs σε περιβάλλοντα με εμπόδια και αβεβαιότητες. Ειδικότερα, οι στόχοι της εργασίας περιλαμβάνουν:

- Τη μελέτη των υπαρχόντων αλγορίθμων προγραμματισμού διαδρομών και την κατανόηση των πλεονεκτημάτων και περιορισμών τους.
- Την ανάπτυξη ενός νέου αλγορίθμου που συνδυάζει τις μεθόδους κατασκευής γραφήματος του αλγορίθμου Rectangles, και αποφυγής εμποδίων του αλγορίθμου APPATT
- Την επέκταση και βελτίωση του αλγοριθμου PRM για την αντιμετώπιση περιβαλλόντων με υψηλή πυκνότητα εμποδίων.
- Την πειραματική αξιολόγηση των αλγορίθμων σε διάφορα σενάρια για την επιβεβαίωση της αποδοτικότητας και της αποτελεσματικότητάς τους.

1.3 Συνεισφορά

Στα πλαίσια της παρούσας διπλωματικής εργασίας, πραγματοποιήθηκαν διάφορες ενέργειες που οδήγησαν στην ανάπτυξη και αξιολόγηση αλγορίθμων για τον προγραμματισμό διαδρομών UAVs. Αρχικά, έγινε μελέτη και ανάλυση της βιβλιογραφίας, όπου αναλύθηκαν οι βασικές θεωρίες και οι σύγχρονες προσεγγίσεις στον προγραμματισμό διαδρομών UAVs, με έμφαση στις μεθόδους που βασίζονται στη θεωρία των γραφημάτων και στις μεθόδους δειγματοληψίας.

Στη συνέχεια, αναπτύχθηκε και υλοποιήθηκε ο αλγόριθμος APPATTrectangles, ο οποίος συνδυάζει τη στρατηγική οδήγησης προς τον στόχο με την κατασκευή του γραφήματος που προτείνει ο αλγόριθμος Rectangles. Ο αλγόριθμος αυτός βελτιστοποιεί τη διαδικασία εύρεσης διαδρομών, προσφέροντας υψηλή ακρίβεια και αποδοτικότητα με μειωμένο υπολογιστικό κόστος.

Παράλληλα, έγινε πλήρης υλοποίηση του αλγορίθμου Rectangles όπως τον παρουσιάζει η βιβλιογραφία, προσφέροντας τη βάση για τη συγκριτική ανάλυση με άλλους αλγορίθμους. Επιπλέον, επεκτάθηκε και βελτιώθηκε ο αλγόριθμος PRM, ώστε να μπορεί να ανταπεξέλθει σε περιβάλλοντα με υψηλή πυκνότητα εμποδίων. Οι βελτιώσεις αυτές περιλάμβαναν την προσθήκη βελτιωμένων τεχνικών δειγματοληψίας, καθιστώντας τον PRM πιο αποδοτικό σε πολύπλοκα περιβάλλοντα.

Μετά την υλοποίηση των αλγορίθμων, πραγματοποιήθηκαν πειράματα σε προσομοιωμένα περιβάλλοντα με διάφορα επίπεδα πολυπλοκότητας και αβεβαιότητας. Τα πειράματα αυτά επέτρεψαν την αξιολόγηση της απόδοσης και της αποτελεσματικότητας των αλγορίθμων σε συνθήκες που προσομοιώνουν πραγματικά περιβάλλοντα πτήσης UAVs.

Τέλος, τα αποτελέσματα των πειραμάτων συγκρίθηκαν, με στόχο να επιβεβαιωθεί η υπεροχή του APPATTrectangles τόσο σε ποιότητα διαδρομής όσο και σε χρόνο υπολογισμού. Οι συγκρίσεις αυτές έδειξαν ότι ο APPATTrectangles προσφέρει καλύτερες διαδρομές με μικρότερο υπολογιστικό κόστος σε σχέση με παραδοσιακούς αλγορίθμους όπως ο PRM και ο Rectangles.

1.4 Διάρθρωση της Διπλωματικής Εργασίας

Η διπλωματική εργασία διαρθρώνεται ως εξής:

- **Κεφάλαιο 2: Γνώσεις Υποβάθρου:** Παρουσιάζονται οι βασικές θεωρητικές έννοιες και τεχνικές που αφορούν τον προγραμματισμό διαδρομών UAVs, με αναφορά στη σχετική βιβλιογραφία. Αναλύονται οι αλγόριθμοι Dijkstra, A*, PRM και RRT, VFH καθώς και οι βασικές αρχές της θεωρίας των γραφημάτων και της δειγματοληψίας.
- **Κεφάλαιο 3: Αλγόριθμοι:** Αναλύονται οι διάφοροι αλγόριθμοι προγραμματισμού διαδρομών που μελετήθηκαν και χρησιμοποιήθηκαν στην εργασία. Παρουσιάζονται οι θεωρητικές αρχές, ένα παράδειγμα βήμα προς βήμα για την κατανόηση λειτουργίας του αλγορίθμου, τα πλεονεκτήματα και μειονεκτήματα κάθε αλγορίθμου και τα συμπεράσματα.

- **Κεφάλαιο 4: Υλοποίηση Αλγορίθμων:** Περιγράφεται η υλοποίηση των αλγορίθμων, με λεπτομέρειες για τις τεχνικές και τα εργαλεία που χρησιμοποιήθηκαν. Αναλύονται οι προκλήσεις που αντιμετωπίστηκαν κατά την υλοποίηση και οι λύσεις που δόθηκαν. Παρουσιάζονται οι βασικές συνάρτησης που έχουν υλοποιηθεί και η υλοποίηση κάθε αλγορίθμου με βάση αυτές τις συναρτήσεις.
- **Κεφάλαιο 5: Πειραματική Αξιολόγηση:** Παρουσιάζονται τα αποτελέσματα των πειραμάτων που πραγματοποιήθηκαν για την αξιολόγηση των αλγορίθμων σε διάφορα σενάρια. Τα πειράματα περιλαμβάνουν την αξιολόγηση της απόδοσης των αλγορίθμων σε περιβάλλοντα με διαφορετικά επίπεδα πολυπλοκότητας και αβεβαιότητας. Τα αποτελέσματα παρουσιάζονται με γραφήματα και πίνακες, και αναλύονται για την εξαγωγή συμπερασμάτων.
- **Κεφάλαιο 6: Συμπεράσματα και Προοπτικές:** Συνοψίζονται τα κύρια συμπεράσματα της εργασίας και προτείνονται κατευθύνσεις για μελλοντική έρευνα και ανάπτυξη. Αναλύονται τα δυνατά και αδύνατα σημεία των αλγορίθμων και προτείνονται βελτιώσεις και νέες προσεγγίσεις για την αντιμετώπιση των προκλήσεων στον προγραμματισμό διαδρομών UAVs.

2

Γνώσεις Υποβάθρου

Στο κεφάλαιο αυτό θα παρουσιαστούν οι βασικές θεωρητικές έννοιες και τεχνικές που σχετίζονται με τον προγραμματισμό διαδρομών για μη επανδρωμένα εναέρια οχήματα (UAVs). Η κατανόηση αυτών των εννοιών είναι απαραίτητη για την ανάλυση και την υλοποίηση των αλγορίθμων που θα συζητηθούν στα επόμενα κεφάλαια της διπλωματικής εργασίας.

Ο προγραμματισμός διαδρομών για UAVs είναι ένα πεδίο που συνδυάζει στοιχεία από τη ρομποτική, την τεχνητή νοημοσύνη και τα συστήματα ελέγχου. Οι προκλήσεις που αντιμετωπίζονται περιλαμβάνουν την αποφυγή εμποδίων, τη διαχείριση της ενέργειας, και την επίτευξη της βέλτιστης διαδρομής σε πραγματικό χρόνο. Η διεθνής βιβλιογραφία προσφέρει πολλές προσεγγίσεις για την επίλυση αυτών των προβλημάτων, οι οποίες μπορούν να ταξινομηθούν σε τέσσερις κύριες κατηγορίες: τις μεθόδους που βασίζονται σε γραφήματα, τις μεθόδους που βασίζονται σε δειγματοληψία, τις μεθόδους δυναμικού πεδίου, και τις μεθόδους ευφυούς βελτιστοποίησης.

Η παρούσα διπλωματική εργασία επικεντρώνεται στις μεθόδους που βασίζονται σε γραφήματα και σε δειγματοληψία, αναλύοντας τους αλγόριθμους PRM, APPATT και Rectangles. Στο κεφάλαιο αυτό, θα παρουσιαστούν οι προσεγγίσεις για τον προγραμματισμό διαδρομών UAVs. Η ανάλυση αυτή θα αποτελέσει τη βάση για την κατανόηση των μεθόδων που χρησιμοποιούνται στην υλοποίηση και την πειραματική αξιολόγηση των αλγορίθμων στον προγραμματισμό διαδρομών UAVs.

Οι πληροφορίες που παρουσιάζονται σε αυτό το κεφάλαιο αντλούνται από τη σχετική διεθνή βιβλιογραφία. Η κατανόηση αυτών των θεωρητικών βάσεων είναι κρίσιμη για την αποτελεσματική υλοποίηση και βελτίωση των αλγορίθμων που θα παρουσιαστούν στη συνέχεια.

2.1 Εισαγωγή στον Προγραμματισμό Διαδρομών UAV

Ο προγραμματισμός διαδρομών (path planning) για τα μη επανδρωμένα εναέρια οχήματα (UAVs) είναι ένα κρίσιμο πεδίο έρευνας και εφαρμογών, καθώς καθορίζει την ικανότητα των UAVs να πλοηγούνται αυτόνομα σε περιβάλλοντα με εμπόδια, εξασφαλίζοντας την ασφάλεια και την αποδοτικότητα της πτήσης τους. Η ανάπτυξη αποτελεσματικών αλγορίθμων για τον

προγραμματισμό διαδρομών είναι ζωτικής σημασίας για την ευρεία εφαρμογή των UAVs σε τομείς όπως η επιθεώρηση υποδομών, οι επιχειρήσεις διάσωσης, η γεωργία και η παράδοση αγαθών.

Η διεθνής βιβλιογραφία έχει προτείνει πολυάριθμες προσεγγίσεις για τον προγραμματισμό διαδρομών UAVs, οι οποίες μπορούν να ταξινομηθούν σε τέσσερις κύριες κατηγορίες:

- **Μέθοδοι βασισμένες σε γραφήματα** (graph-based methods): Αυτές περιλαμβάνουν αλγορίθμους όπως οι Dijkstra και A*, οι οποίοι βασίζονται σε θεωρία των γραφημάτων για να βρουν τις συντομότερες διαδρομές σε ένα γράφημα με κόμβους και ακμές.
- **Μέθοδοι βασισμένες σε δειγματοληψία** (sampling-based methods): Περιλαμβάνουν τους αλγορίθμους Probabilistic Roadmap Method (PRM) και Rapidly-exploring Random Tree (RRT), οι οποίοι χρησιμοποιούν τυχαία δειγματοληψία για την κατασκευή γραφημάτων που αναπαριστούν τον χώρο αναζήτησης.
- **Μέθοδοι δυναμικού πεδίου** (potential field methods): Όπως ο Vector Field Histogram (VFH), που χρησιμοποιούν δυναμικά πεδία για την αποφυγή εμποδίων.
- **Μέθοδοι ευφυούς βελτιστοποίησης** (intelligent optimization methods): Χρησιμοποιούν τεχνικές όπως οι γενετικοί αλγόριθμοι και οι αλγόριθμοι σμήνους σωματιδίων για την εύρεση βέλτιστων διαδρομών.

2.2 Ανάλυση βασικών μεθόδων και αλγορίθμων

2.2.1 Μέθοδοι Βασισμένες σε Γραφήματα (Graph-based Methods)

Οι μέθοδοι που βασίζονται σε γραφήματα χρησιμοποιούν τη θεωρία των γραφημάτων για την επίλυση προβλημάτων πλοήγησης, όπου ο χώρος αναζήτησης αναπαρίσταται ως ένα γράφημα αποτελούμενο από κόμβους (που αντιπροσωπεύουν θέσεις ή σημεία ενδιαφέροντος) και ακμές (που αντιπροσωπεύουν πιθανές διαδρομές μεταξύ αυτών των θέσεων). Δύο από τους πιο γνωστούς αλγορίθμους αυτής της κατηγορίας είναι οι **Dijkstra** και **A***.

Dijkstra

Ο αλγόριθμος του Dijkstra είναι ένας γνωστός αλγόριθμος εύρεσης της συντομότερης διαδρομής σε ένα γράφημα με μη αρνητικά βάρη στις ακμές. Ο αλγόριθμος λειτουργεί σε κατευθυνόμενα ή μη κατευθυνόμενα γραφήματα και υπολογίζει την ελάχιστη απόσταση από έναν κόμβο εκκίνησης προς όλους τους άλλους κόμβους. Ο αλγόριθμος βασίζεται στη διαρκή ενημέρωση των αποστάσεων των κόμβων από την αρχική πηγή και χρησιμοποιεί έναν μηχανισμό εξερεύνησης για να προσδιορίσει τον συντομότερο δρόμο.

Περιγραφή του Αλγορίθμου

Ο αλγόριθμος ξεκινά από έναν κόμβο-πηγή και υποθέτει ότι η αρχική απόσταση προς όλους τους άλλους κόμβους είναι άπειρη, εκτός από τον ίδιο τον κόμβο εκκίνησης που έχει απόσταση 0. Στη συνέχεια, ο αλγόριθμος διατρέχει τους κόμβους, εξετάζει τις γειτονικές ακμές και ενημερώνει τις αποστάσεις προς τους γειτονικούς κόμβους εάν βρει μικρότερες αποστάσεις.

Αυτό επαναλαμβάνεται έως ότου επισκεφθούν όλοι οι κόμβοι ή έως ότου βρεθεί η συντομότερη διαδρομή προς έναν συγκεκριμένο κόμβο.

Βήματα

- Ο αρχικός κόμβος έχει απόσταση 0 από τον εαυτό του, και όλοι οι άλλοι κόμβοι έχουν απόσταση άπειρο.
- Προσθέτουμε όλους τους κόμβους σε μια ουρά προτεραιότητας, με βάση την απόσταση από τον αρχικό κόμβο.
- Για κάθε γειτονικό κόμβο του εξεταζόμενου κόμβου, ελέγχουμε αν μπορούμε να μειώσουμε την απόσταση προς αυτόν μέσω του εξεταζόμενου κόμβου.
- Επαναλαμβάνουμε τη διαδικασία έως ότου όλοι οι κόμβοι επισκεφθούν ή βρεθεί η συντομότερη διαδρομή προς τον προορισμό.

A*

Ο αλγόριθμος A* είναι μια βελτιωμένη έκδοση του αλγορίθμου Dijkstra, σχεδιασμένος να μειώνει το υπολογιστικό κόστος χρησιμοποιώντας μια ευρετική συνάρτηση για να καθοδηγεί την αναζήτηση. Η κύρια διαφορά του από τον Dijkstra είναι ότι δεν εξετάζει μόνο την απόσταση από την πηγή αλλά λαμβάνει υπόψη και μια εκτίμηση της απόστασης από τον στόχο, προσπαθώντας να κατευθύνει την αναζήτηση πιο αποδοτικά προς τη σωστή κατεύθυνση. Αυτή η ευρετική συνάρτηση βοηθά στη μείωση των κόμβων που εξετάζονται και επιταχύνει τη διαδικασία εύρεσης της συντομότερης διαδρομής.

Περιγραφή του Αλγορίθμου

Ο αλγόριθμος A* χρησιμοποιεί δύο συναρτήσεις κόστους. Το πραγματικό κόστος της συντομότερης διαδρομής από τον αρχικό κόμβο μέχρι τον κόμβο n g(n). Μια ευρετική συνάρτηση που εκτιμά το κόστος από τον κόμβο n μέχρι τον στόχο. Συνήθως, αυτή η ευρετική είναι η ευκλείδεια ή η Manhattan απόσταση h(n).

Η συνάρτηση f(n), που καθορίζει την προτεραιότητα ενός κόμβου n, υπολογίζεται ως: f(n)=g(n)+h(n)f(n) = g(n) + h(n)f(n)=g(n)+h(n) Ο αλγόριθμος επιλέγει κάθε φορά τον κόμβο με τη μικρότερη τιμή f(n), συνδυάζοντας το γνωστό κόστος της διαδρομής με την εκτίμηση της υπόλοιπης απόστασης προς τον στόχο.

Βασικά Βήματα

- Ξεκινάμε από τον αρχικό κόμβο και ορίζουμε το κόστος g(αρχικό) = 0.
- Εισάγουμε όλους τους γειτονικούς κόμβους στην ουρά προτεραιότητας με βάση την τιμή f(n).
- Σε κάθε βήμα, επιλέγουμε τον κόμβο με τη μικρότερη τιμή f(n) και εξετάζουμε τους γειτονικούς του κόμβους.
- Υπολογίζουμε το κόστος g(n) για κάθε γειτονικό κόμβο και ενημερώνουμε αν βρεθεί μικρότερη τιμή.
- Η διαδικασία συνεχίζεται μέχρι να βρεθεί ο στόχος ή να εξαντληθούν οι κόμβοι.

2.2.2 Μέθοδοι Βασισμένες σε Δειγματοληψία (Sampling-based Methods)

Οι μέθοδοι δειγματοληψίας είναι ιδιαίτερα χρήσιμες σε πολύπλοκα και μεγάλης κλίμακας περιβάλλοντα, καθώς δεν απαιτούν πλήρη αναπαράσταση του χώρου αναζήτησης. Οι αλγόριθμοι αυτοί επιλέγουν τυχαία σημεία (δείγματα) από τον χώρο αναζήτησης και δημιουργούν διαδρομές μεταξύ αυτών των σημείων.

Probabilistic Roadmap Method (PRM)

Ο αλγόριθμος PRM [8] δημιουργεί τυχαίους κόμβους στον χώρο αναζήτησης και τους συνδέει με γειτονικούς κόμβους μέσω ακμών, δημιουργώντας έτσι ένα γράφημα. Η κύρια ιδέα είναι η δειγματοληψία του χώρου και η δημιουργία ενός δικτύου διαδρομών που αποφεύγουν τα εμπόδια. Αυτή η μέθοδος είναι εξαιρετικά αποτελεσματική για μεγάλα, άγνωστα ή πολυδιάστατα περιβάλλοντα.

Βασικά Βήματα

- Δημιουργία τυχαίων σημείων (κόμβων) στον χώρο.
- Σύνδεση των κόμβων με τις κοντινότερες γειτονικές θέσεις, εφόσον οι ακμές μεταξύ των κόμβων δεν διασταυρώνονται με εμπόδια.
- Προσθήκη του σημείου εκκίνησης και του σημείου στόχου στο γράφημα.
- Χρήση αλγορίθμου εύρεσης συντομότερης διαδρομής για να βρεθεί η διαδρομή από το σημείο εκκίνησης στον στόχο.

Rapidly-exploring Random Tree (RRT)

Ο αλγόριθμος RRT [9] είναι σχεδιασμένος για την ταχεία εξερεύνηση μεγάλων χώρων αναζήτησης, ξεκινώντας από το σημείο εκκίνησης και δημιουργώντας σταδιακά ένα δέντρο διαδρομών που επεκτείνεται προς τον στόχο. Ο RRT είναι αποτελεσματικός για την αποφυγή εμποδίων, ειδικά σε δυναμικά περιβάλλοντα ή σε περιβάλλοντα με πολλαπλά εμπόδια, καθώς επεκτείνει συνεχώς το δέντρο του προς τον στόχο.

Η βασική ιδέα του αλγορίθμου RRT είναι να δημιουργηθεί ένα δέντρο από διαδρομές, το οποίο ξεκινά από τον αρχικό κόμβο και εξερευνά τον χώρο με τυχαία επιλεγμένα σημεία. Ο αλγόριθμος επεκτείνει το δέντρο προσθέτοντας κάθε νέο τυχαίο σημείο και το συνδέει με το πλησιέστερο κόμβο που έχει ήδη προστεθεί στο δέντρο, δημιουργώντας έτσι διαδρομές προς τον στόχο.

Βασικά Βήματα

- Αρχικοποίηση με τον κόμβο εκκίνησης ως ρίζα του δέντρου.
- Σε κάθε βήμα, επιλέγεται ένα τυχαίο σημείο στον χώρο.
- Βρίσκεται ο πλησιέστερος κόμβος του δέντρου στο τυχαίο σημείο.
- Προστίθεται ένα νέο σημείο στο δέντρο, το οποίο είναι πιο κοντά στο τυχαίο σημείο αλλά εντός ενός μέγιστου επιτρεπόμενου βήματος.

- Η διαδικασία συνεχίζεται μέχρι να προσεγγιστεί ο στόχος ή μέχρι να φτάσουμε στον προκαθορισμένο αριθμό κόμβων.

2.2.3 Μέθοδοι Δυναμικού Πεδίου (Potential Field Methods)

Οι μέθοδοι δυναμικού πεδίου χρησιμοποιούν ένα είδος "δυνάμεων" για να καθοδηγήσουν το UAV μέσα από το περιβάλλον. Ο στόχος ασκεί έλξη προς το UAV, ενώ τα εμπόδια ασκούν απωστικές δυνάμεις, αποτρέποντας συγκρούσεις.

Vector Field Histogram (VFH)

Ο Vector Field Histogram (VFH) [3], [4] είναι ένας αλγόριθμος δυναμικής αποφυγής εμποδίων για ρομποτική πλοήγηση σε πραγματικό χρόνο. Η βασική αρχή του VFH είναι να υπολογίζει ένα δυναμικό πεδίο από τα δεδομένα των αισθητήρων, το οποίο καθοδηγεί το ρομπότ (ή UAV) προς τον στόχο, αποφεύγοντας ταυτόχρονα τα εμπόδια. Ο αλγόριθμος χρησιμοποιεί μια πολική αναπαράσταση των δεδομένων του περιβάλλοντος για να προσδιορίσει την κατεύθυνση με το λιγότερο ρίσκο σύγκρουσης.

Ο VFH λαμβάνει δεδομένα από αισθητήρες, όπως αισθητήρες υπερήχων ή LIDAR, τα οποία παρέχουν πληροφορίες για τη θέση των εμποδίων γύρω από το ρομπότ. Ο αλγόριθμος δημιουργεί έναν ιστόγραμμα δυναμικού πεδίου που αναπαριστά το ρίσκο σύγκρουσης σε κάθε κατεύθυνση. Στη συνέχεια, επιλέγει την κατεύθυνση με τη χαμηλότερη πιθανότητα σύγκρουσης και καθοδηγεί το ρομπότ προς αυτήν, πάντα με γνώμονα την πορεία προς τον τελικό στόχο.

Βασικά Βήματα

- **Δημιουργία Τοπικού Ιστογράμματος (Local Histogram):** Ο αλγόριθμος δημιουργεί ένα τοπικό ιστόγραμμα που αναπαριστά τα εμπόδια γύρω από το ρομπότ, χρησιμοποιώντας δεδομένα από τους αισθητήρες.
- **Φιλτράρισμα του Ιστογράμματος (Histogram Filtering)**: Ο αλγόριθμος εφαρμόζει ένα φιλτράρισμα για να εξαλείψει τον "θόρυβο" από τα δεδομένα των αισθητήρων και να διατηρήσει μόνο τις πιο σημαντικές πληροφορίες για τα εμπόδια.
- **Υπολογισμός Διόδων (Candidate Directions)**: Υπολογίζονται οι πιθανές κατευθύνσεις που είναι διαθέσιμες, δηλαδή εκείνες που δεν περιέχουν εμπόδια.
- **Επιλογή Κατεύθυνσης (Direction Selection)**: Από τις υποψήφιες κατευθύνσεις, επιλέγεται αυτή που έχει τη χαμηλότερη πιθανότητα σύγκρουσης, ενώ πλησιάζει όσο το δυνατόν περισσότερο στον στόχο.
- **Εντολή Κίνησης (Motion Command)**: Ο αλγόριθμος δίνει εντολή στο ρομπότ να κινηθεί προς την επιλεγμένη κατεύθυνση.

2.2.4 Μέθοδοι Ευφυούς Βελτιστοποίησης (Intelligent Optimization Methods)

Οι μέθοδοι ευφυούς βελτιστοποίησης χρησιμοποιούν τεχνικές βιολογικής έμπνευσης, όπως οι γενετικοί αλγόριθμοι και οι αλγόριθμοι σμήνους σωματιδίων, για την εύρεση βέλτιστων

διαδρομών. Αυτές οι μέθοδοι βασίζονται σε στοχαστικές διαδικασίες και είναι κατάλληλες για πολύπλοκα προβλήματα όπου άλλοι αλγόριθμοι δεν αποδίδουν αποτελεσματικά [5].

Γενετικοί Αλγόριθμοι (Genetic Algorithms)

Οι Γενετικοί Αλγόριθμοι (Genetic Algorithms) [6] είναι μέθοδοι ευφυούς βελτιστοποίησης εμπνευσμένες από τη διαδικασία της φυσικής εξέλιξης. Οι αλγόριθμοι αυτοί μιμούνται τη διαδικασία της φυσικής επιλογής, όπου οι καλύτερες λύσεις επιβιώνουν και αναπαράγονται, δημιουργώντας νέες και καλύτερες λύσεις σε κάθε επανάληψη. Είναι ιδιαίτερα κατάλληλοι για προβλήματα βελτιστοποίησης μεγάλης κλίμακας και πολύπλοκα περιβάλλοντα όπου άλλες μέθοδοι δεν είναι αποτελεσματικές.

Ο Γενετικός Αλγόριθμος ξεκινά με έναν αρχικό πληθυσμό πιθανών λύσεων (ονομάζονται χρωμοσώματα), όπου κάθε λύση αναπαριστάται από ένα σύνολο παραμέτρων. Οι λύσεις αυτές αξιολογούνται με βάση μια συνάρτηση καταλληλότητας (fitness function), η οποία μετράει πόσο "καλή" είναι κάθε λύση σε σχέση με το πρόβλημα που προσπαθούμε να λύσουμε.

Στη συνέχεια, οι καλύτερες λύσεις επιλέγονται για αναπαραγωγή, και μέσω διαδικασιών όπως η διασταύρωση (crossover) και η μετάλλαξη (mutation), δημιουργούνται νέες λύσεις (απόγονοι) για τον επόμενο πληθυσμό. Η διαδικασία αυτή συνεχίζεται επαναληπτικά μέχρι να επιτευχθεί η επιθυμητή λύση ή να εκπληρωθεί κάποιο κριτήριο τερματισμού.

Βασικά Βήματα

- **Αρχικοποίηση Πληθυσμού**: Δημιουργείται ένας αρχικός πληθυσμός τυχαίων λύσεων (χρωμοσωμάτων).
- **Αξιολόγηση Καταλληλότητας (Fitness Evaluation)**: Υπολογίζεται η καταλληλότητα κάθε λύσης με βάση τη συνάρτηση καταλληλότητας.
- **Επιλογή (Selection)**: Επιλέγονται οι καλύτερες λύσεις από τον πληθυσμό με βάση την καταλληλότητά τους.
- **Διασταύρωση (Crossover)**: Συνδυάζονται τα χαρακτηριστικά δύο λύσεων (γονέων) για να δημιουργήσουν μία ή περισσότερες νέες λύσεις (απογόνους).
- **Μετάλλαξη (Mutation)**: Ορισμένα τμήματα των λύσεων αλλάζουν τυχαία για να εξερευνηθούν νέες περιοχές του χώρου λύσεων.
- **Αντικατάσταση Πληθυσμού**: Οι νέοι απόγονοι αντικαθιστούν κάποιες από τις παλιές λύσεις, σχηματίζοντας τον νέο πληθυσμό.
- **Επανάληψη**: Τα βήματα από το 2 έως το 6 επαναλαμβάνονται μέχρι να ικανοποιηθεί κάποιο κριτήριο τερματισμού, όπως ο αριθμός επαναλήψεων ή η εύρεση μιας λύσης που να πληροί το επιθυμητό επίπεδο καταλληλότητας.

Αλγόριθμος Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO)

Ο Αλγόριθμος Σμήνους Σωματιδίων (Particle Swarm Optimization - PSO) [7] είναι μια στοχαστική μέθοδος βελτιστοποίησης εμπνευσμένη από τη συλλογική συμπεριφορά σμηνών,

όπως αυτά των πτηνών ή των ψαριών. Κάθε λύση του προβλήματος αναπαρίσταται από ένα σωματίδιο, το οποίο κινείται στον χώρο λύσεων, επηρεασμένο από τη δική του εμπειρία και την εμπειρία των υπόλοιπων σωματιδίων του σμήνους. Στόχος του PSO είναι να βρει το βέλτιστο σημείο σε έναν χώρο πολλών διαστάσεων, όπου κάθε σωματίδιο αναζητά τη βέλτιστη λύση, μαθαίνοντας από τον εαυτό του και το σμήνος.

Κάθε σωματίδιο έχει τα εξής χαρακτηριστικά:

- **Θέση (position):** Η τρέχουσα θέση του σωματιδίου στον χώρο αναζήτησης.
- **Ταχύτητα (velocity):** Η ταχύτητα με την οποία το σωματίδιο μετακινείται στον χώρο λύσεων.
- **Καταλληλότητα (fitness):** Η ποιότητα της λύσης που αναπαριστά η τρέχουσα θέση του σωματιδίου.
- **Καλύτερη Θέση (personal best):** Η καλύτερη θέση που έχει επιτύχει το σωματίδιο μέχρι τώρα.
- **Καλύτερη Θέση Σμήνους (global best):** Η καλύτερη θέση που έχει επιτύχει ολόκληρο το σμήνος μέχρι τώρα.

Βασικά Βήματα

- Κάθε σωματίδιο ξεκινά από μια τυχαία θέση και ταχύτητα στον χώρο λύσεων.
- Σε κάθε επανάληψη, το σωματίδιο ενημερώνει τη θέση και την ταχύτητά του με βάση δύο παράγοντες:
    - Το **personal best**, που είναι η καλύτερη θέση που έχει βρει το ίδιο το σωματίδιο.
    - Το **global best**, που είναι η καλύτερη θέση που έχει βρει οποιοδήποτε σωματίδιο στο σμήνος.
- Τα σωματίδια κινούνται στον χώρο λύσεων και βελτιώνουν τη θέση τους σε κάθε βήμα.
- Η διαδικασία συνεχίζεται μέχρι να επιτευχθεί η βέλτιστη λύση ή να εκπληρωθεί κάποιο κριτήριο τερματισμού, όπως ο αριθμός επαναλήψεων.
-

2.2.5 Συμπεράσματα

Κάθε κατηγορία αλγορίθμων έχει τα δικά της πλεονεκτήματα και περιορισμούς, ανάλογα με τις ανάγκες και τις απαιτήσεις του περιβάλλοντος πλοήγησης. Οι μέθοδοι βασισμένες σε γραφήματα είναι ιδανικές για στατικά περιβάλλοντα με γνωστά εμπόδια, ενώ οι μέθοδοι δειγματοληψίας είναι κατάλληλες για μεγάλα και άγνωστα περιβάλλοντα. Οι δυναμικές μέθοδοι και οι μέθοδοι ευφυούς βελτιστοποίησης προσφέρουν λύσεις για πιο πολύπλοκα και δυναμικά περιβάλλοντα, όπου απαιτείται γρήγορη απόκριση ή εξερεύνηση των διαδρομών σε πραγματικό χρόνο. Σε αυτή την διπλωματική εργασία θα επικεντρωθούμε στις μεθόδους που βασίζονται σε γραφήματα και στις μεθόδους που βασίζονται στην δειγματοληψία.

3

Αλγόριθμοι

Στο τρίτο κεφάλαιο της διατριβής παρουσιάζονται λεπτομερώς οι τρεις αλγόριθμοι πορείας UAV που υλοποιήθηκαν στο πλαίσιο της έρευνας. Κάθε αλγόριθμος περιγράφεται με λεπτομερή τρόπο, περιλαμβάνοντας τη θεωρητική του βάση και τα βήματα εκτέλεσής του. Επίσης, γίνεται ανάλυση των χαρακτηριστικών και των παραμέτρων κάθε αλγορίθμου, καθώς και σύγκρισή τους μεταξύ τους όσον αφορά την απόδοση και την αποτελεσματικότητά τους σε διαφορετικά πειραματικά σενάρια. Αυτό το κεφάλαιο αποτελεί τη βάση για την πειραματική αξιολόγηση που ακολουθεί, καθώς και για τα συμπεράσματα που θα προκύψουν από την εφαρμογή των αλγορίθμων στο πειραματικό περιβάλλον [1].

3.1 Ο αλγόριθμος PRM με βρόχο

Ο αλγόριθμος Probabilistic Roadmap (PRM) με βρόχο κάνει χρήση τυχαία τοποθετημένων κόμβων στον ελεύθερο χώρο ώστε να δημιουργηθεί ένα γράφημα το οποίο συνδέει το αρχικό με το τελικό σημείο [1].

3.1.1 Ο Αλγόριθμος

Ακολουθεί μια συνοπτική περιγραφή των βημάτων που ακολουθεί ο αλγόριθμος

- Εύρεση Ελεύθερου Χώρου
    - Ανάλυση του περιβάλλοντος για τον εντοπισμό του ελεύθερου χώρου, όπου ο UAV μπορεί να κινηθεί χωρίς εμπόδια.
- Προσθήκη Κόμβων
    - Προσθήκη N τυχαίων κόμβων στον ελεύθερο χώρο. Οι κόμβοι αυτοί αντιπροσωπεύουν δυνατές θέσεις στον χώρο που μπορεί να βρεθεί ο UAV.
- Σύνδεση Κόμβων
    - Κάθε νέος κόμβος, συνδέεται με τους υπόλοιπους κόμβους μέσω ακμών. Οι ακμές αυτές αναπαριστούν πιθανές διαδρομές μεταξύ των κόμβων. Εάν δεν υπάρχει σύγκρουση στη διαδρομή μεταξύ δύο κόμβων, η ακμή προστίθεται στο γράφημα.

- Εκτέλεση Αλγορίθμου Dijkstra
    - Εφαρμογή του αλγορίθμου Dijkstra στο γράφημα που δημιουργήθηκε με τους κόμβους και τις ακμές. Ο αλγόριθμος εντοπίζει την βέλτιστη διαδρομή από το αρχικό σημείο προς τον στόχο, εάν υπάρχει. Εάν δεν υπάρχει τότε ο αλγόριθμος επιστρέφει στο βήμα 2 (Προσθήκη Κόμβων).

3.1.1 Παράδειγμα εκτέλεσης Αλγορίθμου

Βήμα 1 Εύρεση Ελεύθερου Χώρου
Εξετάζουμε ένα περιβάλλον με εμπόδια και καθορίζουμε τις περιοχές που είναι ελεύθερες από εμπόδια.

Εικόνα 1

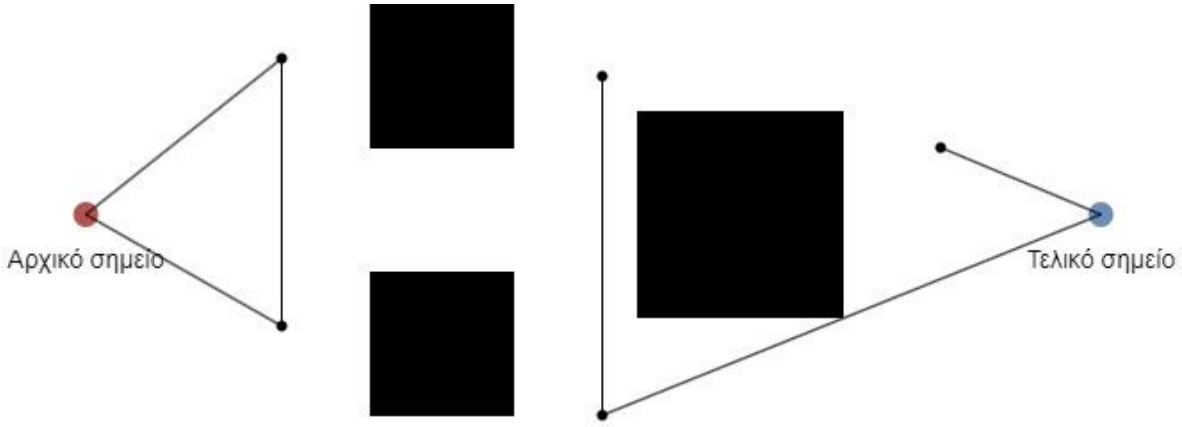
Βήμα 2 Προσθήκη Κόμβων

Προσθέτουμε N=5 τυχαίους κόμβους στον ελεύθερο χώρο. Αυτοί οι κόμβοι επιλέγονται τυχαία στον χώρο που δεν περιέχει εμπόδια.

Εικόνα 2

Βήμα 3 Σύνδεση Κόμβων

Σχεδιάζουμε ακμές που συνδέουν κάθε κόμβο με όλους τους υπολοίπους κόμβους, ελέγχοντας για συγκρούσεις με εμπόδια. Αν δεν υπάρχει σύγκρουση, η ακμή προστίθεται στο γράφημα.



Εικόνα 3

Βήμα 4 Εκτέλεση Αλγορίθμου Dijkstra

Εφαρμόζουμε τον αλγόριθμο Dijkstra στο γράφημα για να βρούμε τη συντομότερη διαδρομή από το αρχικό σημείο προς το τελικό σημείο. Όπως μπορούμε να παρατηρήσουμε δεν υπάρχει εφικτή διαδρομή από το αρχικό σημείο στο τελικό. Επομένως πρέπει ο αλγόριθμος να ξανά εκτελεστεί από το βήμα 2 μέχρι να βρεθεί μια εφικτή διαδρομή.

3.1.2 Πλεονεκτήματα και Περιορισμοί

Ο αλγόριθμος PRM είναι ιδιαίτερα χρήσιμος σε περιβάλλοντα με υψηλή πολυπλοκότητα, καθώς μπορεί να χειριστεί μεγάλο αριθμό εμποδίων και να δημιουργήσει διαδρομές γρήγορα. Ωστόσο, δεδομένου ότι βασίζεται σε τυχαία τοποθετημένους κόμβους, δεν εγγυάται πάντα την εύρεση της βέλτιστης διαδρομής.

Ο αλγόριθμος PRM είναι ευέλικτος και μπορεί να προσαρμοστεί για χρήση σε διάφορα είδη περιβαλλόντων και εφαρμογών, καθιστώντας τον μια σημαντική μέθοδο στον τομέα του σχεδιασμού διαδρομών UAV.

3.1.3 Συμπεράσματα

Ο αλγόριθμος Probabilistic Roadmap (PRM) αποδεικνύεται ιδιαίτερα χρήσιμος για την εύρεση διαδρομών σε πολύπλοκα περιβάλλοντα με πολλά εμπόδια. Η βασική του ιδέα στηρίζεται στην τυχαία τοποθέτηση κόμβων στον ελεύθερο χώρο και τη σύνδεση αυτών με ακμές, δημιουργώντας έτσι ένα γράφημα που μπορεί να χρησιμοποιηθεί για τον εντοπισμό διαδρομών από το αρχικό σημείο προς τον στόχο. Παρότι η τυχαιότητα αυτή επιτρέπει στον αλγόριθμο να καλύπτει μεγάλους και πολύπλοκους χώρους, έχει και κάποιες περιορισμένες αδυναμίες. Ακολουθούν τα κύρια συμπεράσματα που προκύπτουν από την ανάλυση του αλγορίθμου PRM.

Ο αλγόριθμος PRM είναι ευέλικτος και μπορεί να εφαρμοστεί σε διάφορα είδη περιβαλλόντων, ανεξάρτητα από την πολυπλοκότητά τους. Η τυχαία τοποθέτηση κόμβων επιτρέπει στον αλγόριθμο να εξερευνά τον χώρο αποτελεσματικά, εντοπίζοντας πιθανές διαδρομές σε περιοχές που δεν έχουν εμφανή μονοπάτια.

Η πολυπλοκότητα του αλγορίθμου εξαρτάται άμεσα από τον αριθμό των τυχαία τοποθετημένων κόμβων. Η αύξηση του αριθμού των κόμβων βελτιώνει τις πιθανότητες εύρεσης μιας διαδρομής, αλλά ταυτόχρονα αυξάνει την υπολογιστική πολυπλοκότητα και τον χρόνο εκτέλεσης του αλγορίθμου. Συνεπώς, πρέπει να βρεθεί μια ισορροπία μεταξύ του αριθμού των κόμβων και της αποδοτικότητας.

Λόγω της τυχαίας φύσης της τοποθέτησης των κόμβων, ο PRM δεν εγγυάται την εύρεση της βέλτιστης διαδρομής. Οι διαδρομές που εντοπίζονται μπορεί να είναι εφικτές, αλλά όχι απαραίτητα οι συντομότερες ή οι πιο αποδοτικές. Αυτό μπορεί να αποτελεί περιορισμό σε εφαρμογές όπου η βέλτιστη διαδρομή είναι κρίσιμη.

Ο αλγόριθμος είναι ικανός να χειρίζεται πολύπλοκα περιβάλλοντα με πολλαπλά εμπόδια. Οι κόμβοι τοποθετούνται σε ελεύθερους χώρους και συνδέονται με ακμές που δεν διασχίζουν εμπόδια, διασφαλίζοντας έτσι την ασφάλεια των διαδρομών. Ωστόσο, η αποτελεσματικότητα του ελέγχου σύγκρουσης παίζει κρίσιμο ρόλο στην απόδοση του αλγορίθμου.

3.2 Ο αλγόριθμος Rectangles

Ο αλγόριθμος Rectangles βασίζεται στη δημιουργία κόμβων γύρω από τις γωνίες των εμποδίων και στη σύνδεση αυτών των κόμβων για τον υπολογισμό της βέλτιστης διαδρομής.

3.2.1 Ο Αλγόριθμος

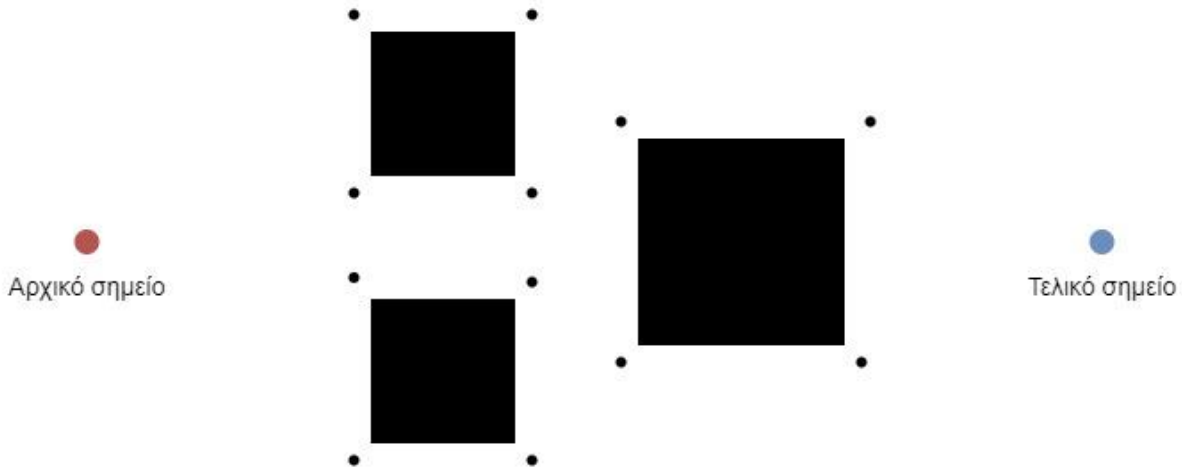
Ακολουθεί μια συνοπτική περιγραφή των βημάτων που ακολουθεί ο αλγόριθμος

- Δημιουργία Κόμβων γύρω από τις Γωνίες των Εμποδίων
    - Για κάθε εμπόδιο στο γράφημα, δημιουργούμε κόμβους στις γωνίες του. Αυτοί οι κόμβοι θα αποτελέσουν τα βασικά σημεία που θα χρησιμοποιηθούν για τον υπολογισμό των διαδρομών. Η τοποθέτηση των κόμβων στις γωνίες των εμποδίων διασφαλίζει ότι λαμβάνουμε υπόψη όλες τις πιθανές αλλαγές κατεύθυνσης που μπορεί να χρειαστεί να ακολουθήσει το UAV.
- Σύνδεση Κόμβων
    - Εξετάζουμε όλους τους κόμβους που δημιουργήθηκαν γύρω από τα εμπόδια και συνδέουμε όσους μπορούν να συνδεθούν μεταξύ τους χωρίς να υπάρχει εμπόδιο στη διαδρομή τους. Το κόστος των ακμών που συνδέουν τους κόμβους είναι η απόσταση μεταξύ τους.
    - Η διαδικασία αυτή δημιουργεί ένα γράφημα με κόμβους και ακμές, όπου οι ακμές αντιπροσωπεύουν τις πιθανές διαδρομές μεταξύ των κόμβων.
- Εκτέλεση Αλγορίθμου Dijkstra
    - Εφαρμόζουμε τον αλγόριθμο Dijkstra στο γράφημα που δημιουργήθηκε με τους κόμβους και τις ακμές. Ο αλγόριθμος Dijkstra χρησιμοποιείται για να βρει τη συντομότερη διαδρομή από το αρχικό σημείο προς το τελικό σημείο στο γράφημα, λαμβάνοντας υπόψη το κόστος των ακμών.
    - Η συντομότερη διαδρομή υπολογίζεται με βάση τις αποστάσεις μεταξύ των κόμβων, εξασφαλίζοντας ότι το UAV θα ακολουθήσει την πιο αποδοτική διαδρομή που αποφεύγει τα εμπόδια.

3.2.2 Παράδειγμα εκτέλεσης Αλγορίθμου

Βήμα 1 Δημιουργία Κόμβων γύρω από τις Γωνίες των Εμποδίων

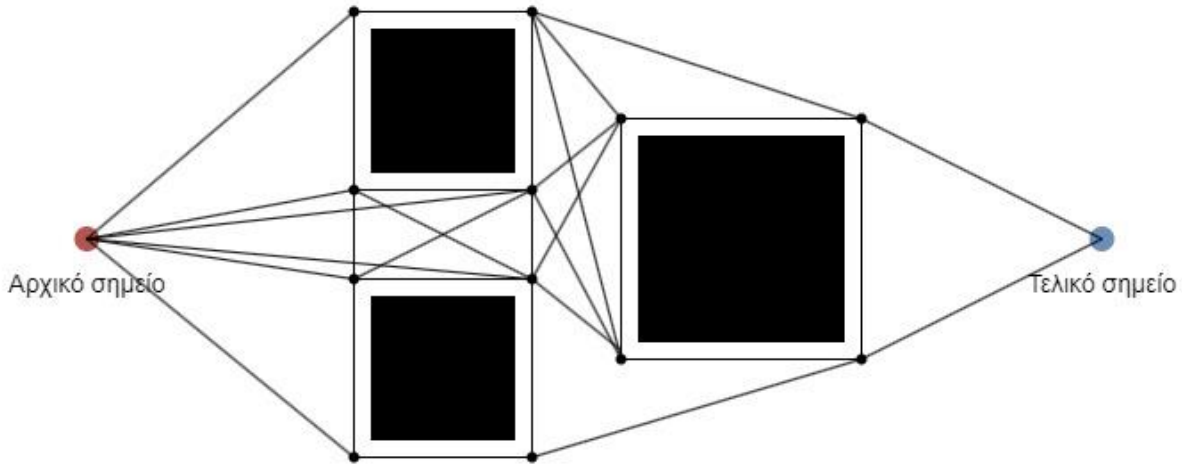
Για κάθε εμπόδιο στο περιβάλλον, δημιουργούμε κόμβους στις γωνίες του. Αυτοί οι κόμβοι αντιπροσωπεύουν τα σημεία ενδιαφέροντος όπου το UAV μπορεί να αλλάξει κατεύθυνση.



Εικόνα 4

Βήμα 2 Σύνδεση Κόμβων

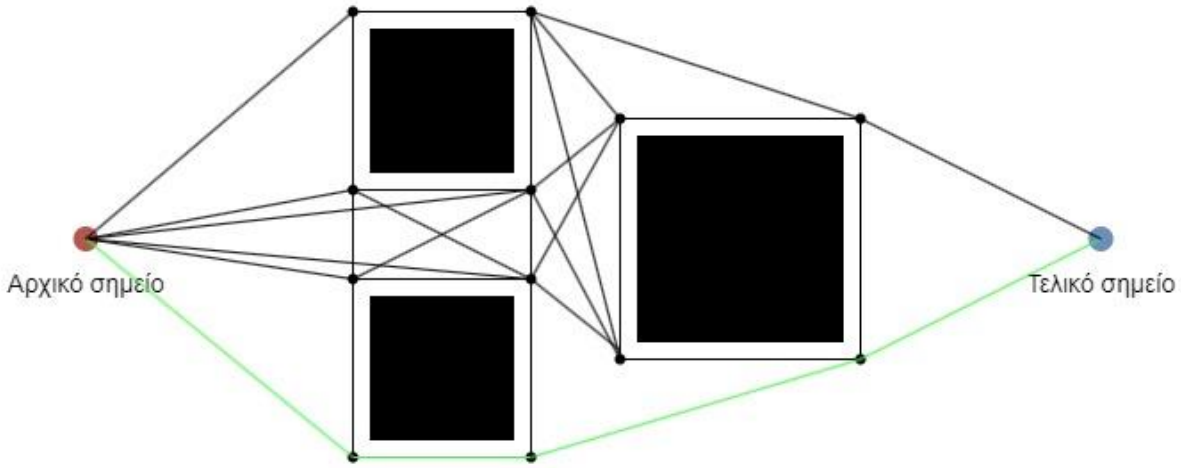
Για κάθε κόμβο στο γράφημα τον συνδέουμε μέσο μιας ακμής με όλους τους υπολοίπους κόμβους. Εάν στην ακμή αυτή δεν υπάρχει σύγκρουση τότε η ακμή αυτή προστίθεται στο γράφημα. Το κόστος των ακμών είναι η ευκλείδια απόσταση μεταξύ των κόμβων.
Το γράφημα που δημιουργείται περιλαμβάνει όλες τις ακμές που συνδέουν τους κόμβους χωρίς να διασταυρώνονται με εμπόδια. Αυτό το γράφημα αναπαριστά όλες τις δυνατές διαδρομές που μπορεί να ακολουθήσει το UAV



Εικόνα 5

Βήμα 3 Εκτέλεση Αλγορίθμου Dijkstra

Αφού δημιουργηθεί το γράφημα με τους κόμβους και τις ακμές, εφαρμόζουμε τον αλγόριθμο Dijkstra για να βρούμε τη συντομότερη διαδρομή από το αρχικό σημείο προς το τελικό σημείο.



Εικόνα 6

3.2.3 Πλεονεκτήματα και Μειονεκτήματα

3.2.3.1 Πλεονεκτήματα

Εύρεση Συντομότερης Διαδρομής: Ο αλγόριθμος Rectangles εγγυάται την εύρεση της συντομότερης διαδρομής, καθώς οι κόμβοι στις γωνίες των εμποδίων και η χρήση του αλγόριθμου Dijkstra εξασφαλίζουν ότι η διαδρομή που θα επιλεγεί θα είναι η πιο αποδοτική όσον αφορά την απόσταση.

Απλή Υλοποίηση: Η τοποθέτηση κόμβων στις γωνίες των εμποδίων και η σύνδεσή τους είναι μια σχετικά απλή διαδικασία που μπορεί να εφαρμοστεί εύκολα σε πολλά περιβάλλοντα.

3.2.3.2 Μειονεκτήματα

Υπολογιστική Πολυπλοκότητα: Ο αλγόριθμος μπορεί να γίνει εξαιρετικά αργός σε περιβάλλοντα με μεγάλο αριθμό εμποδίων, καθώς η διαδικασία δημιουργίας και σύνδεσης κόμβων απαιτεί σημαντικούς υπολογιστικούς πόρους.

3.2.4 Συμπεράσματα

Ο αλγόριθμος Rectangles είναι μια ισχυρή μέθοδος για τον σχεδιασμό διαδρομών σε περιβάλλοντα με εμπόδια, ο οποίος εγγυάται την εύρεση της συντομότερης διαδρομής. Παρά την απλή υλοποίησή του και την αποτελεσματικότητά του στην εύρεση της βέλτιστης διαδρομής, η υπολογιστική του πολυπλοκότητα μπορεί να αποτελέσει σημαντικό περιορισμό σε περιβάλλοντα με μεγάλο αριθμό εμποδίων. Συνεπώς, η χρήση του αλγορίθμου Rectangles είναι προτιμότερη σε περιβάλλοντα με μέτρια πολυπλοκότητα, όπου η εύρεση της συντομότερης διαδρομής είναι κρίσιμη, αλλά οι υπολογιστικοί πόροι επαρκούν για την εκτέλεση του αλγορίθμου.

3.3 Ο αλγόριθμος APPATTrectangles

Ο αλγόριθμος APPATTrectangles είναι μια παραλλαγή του αλγορίθμου APPATT (Approximate Path Planning Algorithm for Tangents) που χρησιμοποιείται για τον σχεδιασμό διαδρομών σε περιβάλλοντα με εμπόδια. Ο αρχικός αλγόριθμος APPATT βασίζεται στην ιδέα ότι τα εμπόδια μπορούν να μοντελοποιηθούν ως ελλείψεις, και τα σημεία ενδιαφέροντος τοποθετούνται στις εφαπτόμενες αυτών των ελλείψεων. Στον αλγόριθμο APPATTrectangles, η μοντελοποίηση των εμποδίων γίνεται με ορθογώνια, και τα σημεία ενδιαφέροντος τοποθετούνται στις γωνίες αυτών των ορθογωνίων. Αυτή η προσέγγιση επιτρέπει την απλούστευση της διαδικασίας και την καλύτερη προσαρμογή σε περιβάλλοντα όπου τα εμπόδια είναι πιο κατάλληλα να αναπαρασταθούν με ορθογώνια.

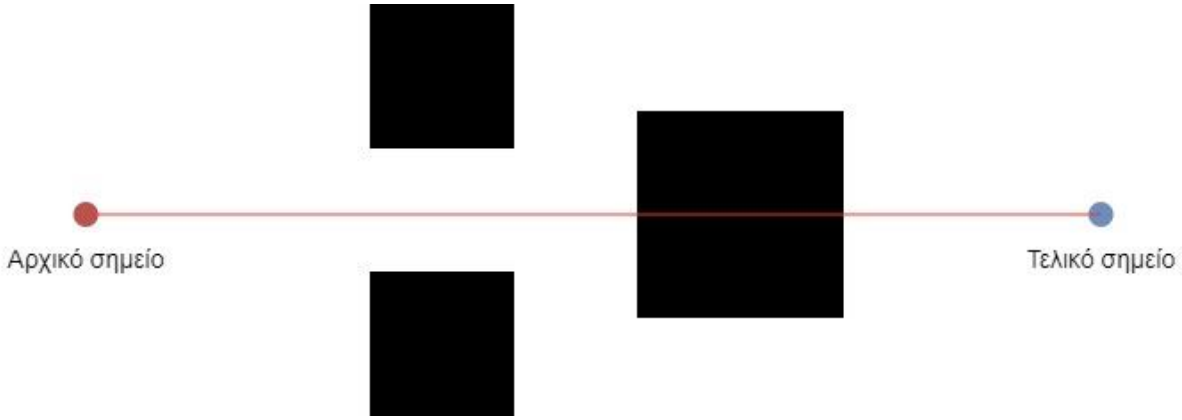
3.3.1 Ο Αλγόριθμος

- Αρχική Σύνδεση Αρχικου και Τέλικου σημειου
    - Συνδέουμε την αρχική θέση και τον στόχο μέσω μιας ακμής. Αυτή η ακμή αντιπροσωπεύει την απευθείας διαδρομή από την αρχή προς το τέλος.
- Έλεγχος Σύγκρουσης
    - Εάν η ακμή που συνδέει την αρχή με τον στόχο δεν συγκρούεται με εμπόδια, προχωράμε στο βήμα 4 (Σύνδεση Σημείων Ενδιαφέροντος).
    - Εάν υπάρχει σύγκρουση, προσθέτουμε το id του εμποδίου στην λίστα obstacles_found[] και βρίσκουμε τα σημεία ενδιαφέροντος γύρω από τις γωνίες του εμποδίου και τα προσθέτουμε στη λίστα graph_nodes[].
- Διερεύνηση Σημείων Ενδιαφέροντος
    - Για κάθε σημείο στη λίστα graph_nodes[], εκτελούμε τα παρακάτω βήματα
        - **Σύνδεση με την Αρχική Θέση:** Συνδέουμε το σημείο με την αρχική θέση και ελέγχουμε αν υπάρχει σύγκρουση. Αν υπάρχει σύγκρουση και το εμπόδιο δεν βρίσκεται στην λίστα obstacles_found[], τότε προσθέτουμε το εμπόδιο στην λίστα και βρίσκουμε νέα σημεία ενδιαφέροντος γύρω από τις γωνίες του εμποδίου και τα προσθέτουμε στη λίστα graph_nodes[].
        - **Σύνδεση με την Τελική Θέση:** Συνδέουμε το σημείο με την τελική θέση. Αν υπάρχει σύγκρουση και το εμπόδιο δεν βρίσκεται στην λίστα obstacles_found[], προσθέτουμε το εμπόδιο στην λίστα και βρίσκουμε νέα σημεία ενδιαφέροντος γύρω από τις γωνίες του εμποδίου και τα προσθέτουμε στη λίστα graph_nodes[]. Στη συνέχεια, επιστρέφουμε στο βήμα 3 (Διερεύνηση Σημείων Ενδιαφέροντος). Εάν η ακμή που συνδέει το σημειο με τον στόχο δεν συγκρούεται με εμπόδια, προχωράμε στο βήμα 4 (Σύνδεση Σημείων Ενδιαφέροντος).
- Σύνδεση Σημείων Ενδιαφέροντος
    - Για κάθε σημείο στη λίστα graph_nodes[], το συνδέουμε με όλα τα υπόλοιπα σημεία της λίστας μέσω ακμών. Εάν οι ακμές αυτές δεν έχουν σύγκρουση, προστίθενται στο γράφημα.
- Εκτέλεση Αλγορίθμου Dijkstra

- Τέλος, εφαρμόζουμε τον αλγόριθμο Dijkstra στο γράφημα που έχει δημιουργηθεί με τους κόμβους και τις ακμές, για να βρούμε τη συντομότερη διαδρομή από την αρχή προς τον στόχο.

3.3.2 Παράδειγμα εκτέλεσης Αλγορίθμου

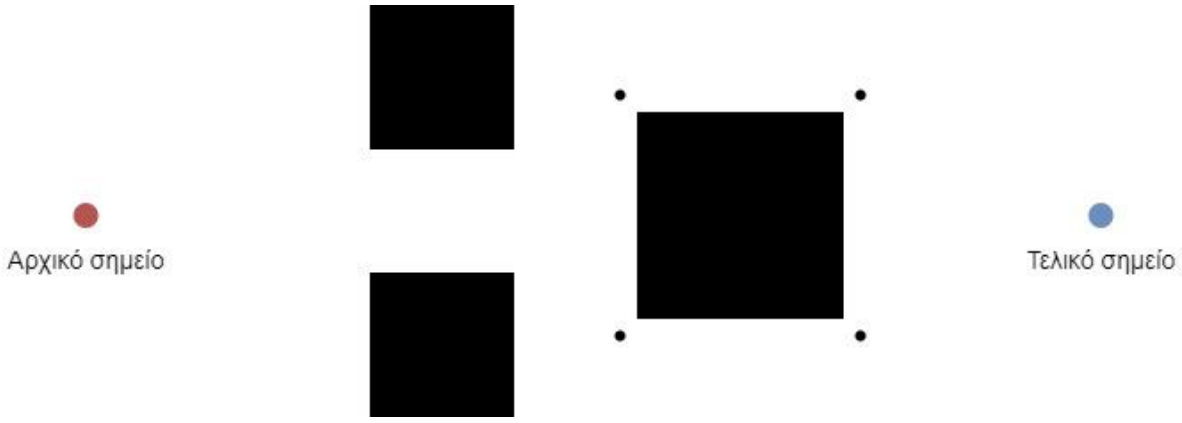
Βήμα 1 Σύνδεση Αρχικού και Τελικού σημείου



Εικόνα 7

Βήμα 2 Έλεγχος σύγκρουσης

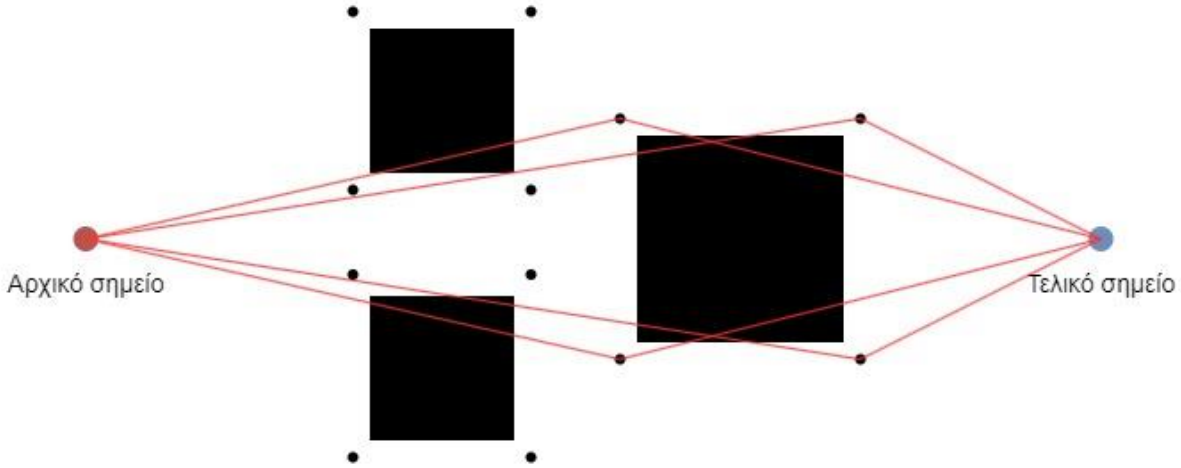
Όπως μπορούμε να διακρίνουμε η ακμή που συνδέει το αρχικό με το τελικό σημείο συγκρούετε με ένα εμπόδιο. Επομένως πρέπει να μπουν κομβόι γύρω από τις γωνίες του έτσι ώστε να γίνει αποφυγή του εμποδίου και οι κομβόι αυτοί να προστεθούν στην λίστα graph_nodes[]. Επίσης προσθέτουμε το εμπόδιο στην λίστα obstacles_found[].



Εικόνα 8

Βήμα 3 Διερεύνηση σημείων ενδιαφέροντος

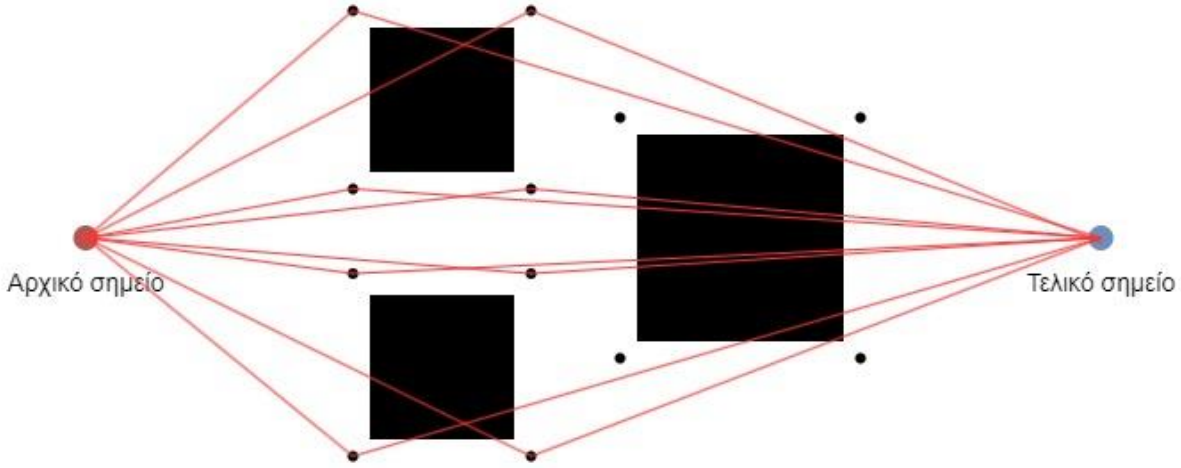
Συνδέουμε κάθε κόμβο στο αρχικό και τελικό σημείο και ελέγχουμε για συγκρούσεις. Παρατηρούμε ότι με την σύνδεση των σημείων στην αρχική θέση δυο νέα εμπόδια ανακαλύπτονται και πρέπει να γίνει αποφυγή τους. Επόμενος προσθέτουμε σημεία ενδιαφέροντος γύρω από τις γωνίες τους.



Εικόνα 9

Βήμα 4 Διερεύνηση νέων σημείων ενδιαφέροντος

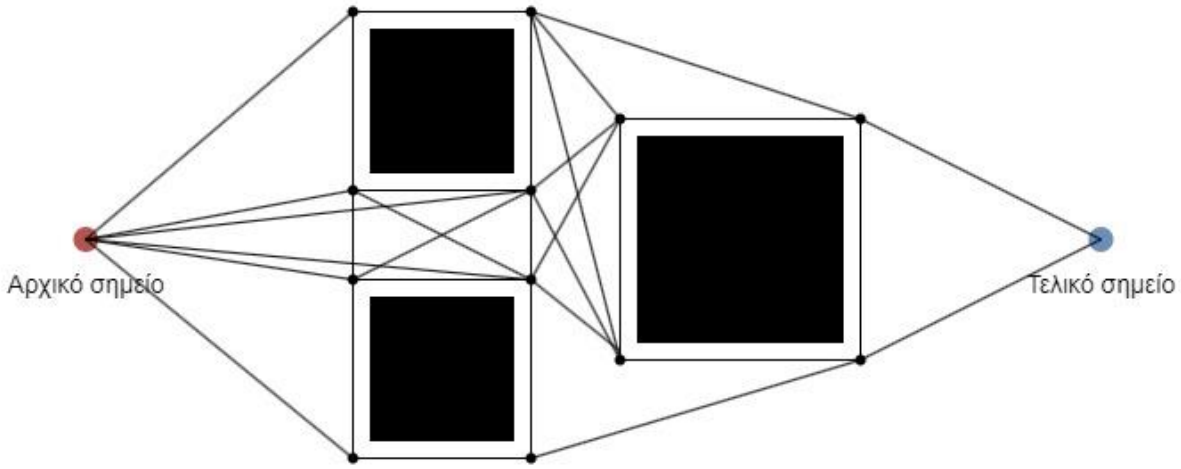
Οι διαδικασία του προηγουμένου βήματος επαναλαμβάνετε για τους νέους κόμβους που έχουν προστεθεί στο γράφημα. Όπως μπορούμε να παρατηρήσουμε δεν ανακαλύπτεται κανένα νέο εμπόδιο. Επομένως μπορεί να γίνει η σύνδεση των κόμβων και η εύρεση συντομότερης διαδρομής με την χρήση του Dijkstra.



Εικόνα 10

Βήμα 5 Σύνδεση σημείων ενδιαφέροντος

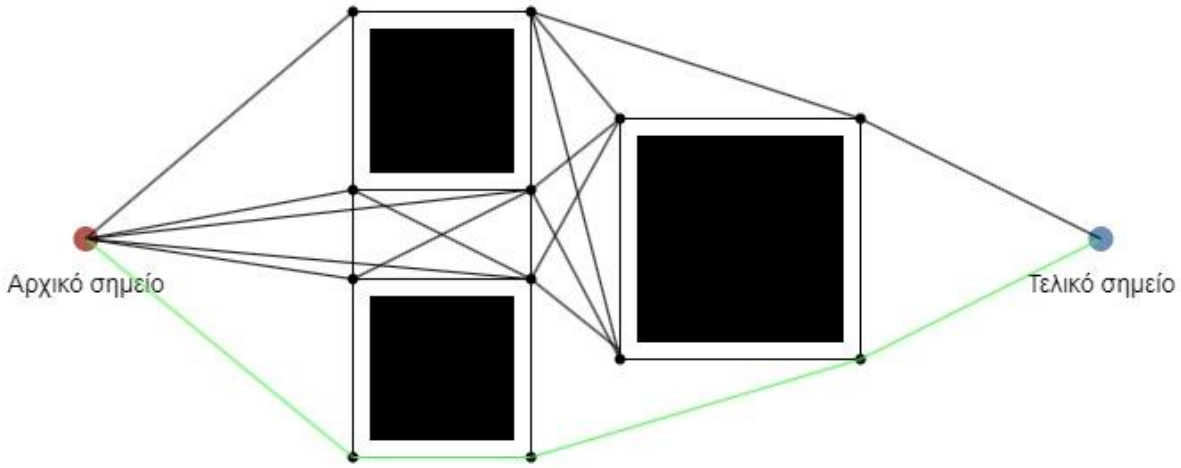
Για κάθε σημείο στη λίστα graph_nodes[], το συνδέουμε με όλα τα υπόλοιπα σημεία της λίστας μέσω ακμών. Εάν οι ακμές αυτές δεν έχουν σύγκρουση, προστίθενται στο γράφημα



Εικόνα 11

Βήμα 6 Εκτέλεση αλγορίθμου του Dijkstra.

Εφαρμόζουμε τον αλγόριθμο Dijkstra στο γράφημα που έχει δημιουργηθεί με τους κόμβους και τις ακμές, για να βρούμε τη συντομότερη διαδρομή από την αρχή προς τον στόχο.



Εικόνα 12

3.3.3 Πλεονεκτήματα και Μειονεκτήματα

Ο αλγόριθμος μπορεί να βρει κοντά στην βέλτιστη διαδρομή σε σχετικά μικρό χρονικό διάστημα συγκριτικά με τους αύλους δυο αλγορίθμους.

Σε ορισμένα περιβάλλοντα με παγίδες μπορεί να μην βρεθεί μια εφικτή διαδρομή από το αρχικό στο τελικό σημείο.

3.3.4 Συμπεράσματα

Ο αλγόριθμος APPATTrectangles αποτελεί μια αποτελεσματική μέθοδο για τον σχεδιασμό διαδρομών σε περιβάλλοντα με εμπόδια, ο οποίος εγγυάται την εύρεση εύρεσης διαδρομών κοντά στη βέλτιστη μέσω της χρήσης του αλγορίθμου Dijkstra. Η κύρια διαφορά του από τον αρχικό αλγόριθμο APPATT είναι η μοντελοποίηση των εμποδίων ως ορθογώνια, με σημεία ενδιαφέροντος στις γωνίες τους αντί για ελλείψεις με σημεία ενδιαφέροντος στις εφαπτόμενες τους. Παρόλο που η υπολογιστική του πολυπλοκότητα μπορεί να αποτελέσει περιορισμό σε

περιβάλλοντα με μεγάλο αριθμό εμποδίων, η ευελιξία και η ακρίβεια του τον καθιστούν μια αξιόπιστη επιλογή για τον σχεδιασμό διαδρομών UAV σε πολύπλοκα περιβάλλοντα. Ωστόσο, θα πρέπει να λαμβάνεται υπόψη ότι ο αλγόριθμος μπορεί να αποτύχει σε περιβάλλοντα με παγίδες, γεγονός που καθιστά απαραίτητη την εξέταση εναλλακτικών λύσεων ή συνδυασμών αλγορίθμων για την αντιμετώπιση τέτοιων περιπτώσεων.

4

Υλοποίηση αλγορίθμων

Σε αυτό το κεφάλαιο παρουσιάζονται οι λεπτομέρειες της υλοποίησης των αλγορίθμων που αναλύθηκαν στα προηγούμενα κεφάλαια. Συγκεκριμένα, γίνεται περιγραφή των διαδικασιών και των τεχνικών που χρησιμοποιήθηκαν για την ανάπτυξη των αλγορίθμων PRM, Rectangles και APPATTrectangles σε περιβάλλον Python.

Αρχικά, παρέχεται μια επισκόπηση του περιβάλλοντος ανάπτυξης, περιλαμβάνοντας τη γλώσσα προγραμματισμού, τις βιβλιοθήκες και τα εργαλεία που χρησιμοποιήθηκαν. Στη συνέχεια, παρουσιάζεται αναλυτικά ο κώδικας για κάθε αλγόριθμο, εξηγώντας τη δομή και τη λειτουργία του. Ειδική έμφαση δίνεται στις προκλήσεις που αντιμετωπίστηκαν κατά την υλοποίηση και στους τρόπους που επιλύθηκαν αυτά τα ζητήματα.

4.1 Περιγραφή Περιβάλλοντος Ανάπτυξης

Η υλοποίηση των αλγορίθμων για τον σχεδιασμό διαδρομών UAV πραγματοποιήθηκε στη γλώσσα προγραμματισμού Python. Η Python επιλέχθηκε λόγω της ευκολίας χρήσης της, της μεγάλης βιβλιοθήκης έτοιμων πακέτων και της ευρείας κοινότητας χρηστών που παρέχει υποστήριξη και πόρους. Στην ανάπτυξη του κώδικα χρησιμοποιήθηκαν διάφορες βιβλιοθήκες της Python, οι οποίες εξυπηρετούν διαφορετικές ανάγκες κατά την υλοποίηση των αλγορίθμων.

4.1.1 Βιβλιοθήκες που Χρησιμοποιήθηκαν

4.1.1.1 Βιβλιοθήκη Numpy

Η βιβλιοθήκη numpy χρησιμοποιήθηκε για τη διαχείριση και επεξεργασία πολυδιάστατων πινάκων και την εκτέλεση μαθηματικών πράξεων υψηλής απόδοσης. Ορισμένες από τις βασικές χρήσεις της numpy στην υλοποίηση περιλαμβάνουν τα εξής.

Η numpy επιτρέπει τη δημιουργία πινάκων δύο διαστάσεων που αναπαριστούν το περιβάλλον με τα εμπόδια. Στο κώδικα υλοποίησης των τριών αλγορίθμων, η μεταβλητή area είναι ένας πίνακας numpy που αποθηκεύει την κατανομή των εμποδίων στο περιβάλλον. Κάθε θέση του πίνακα είναι ίση με μηδέν αν είναι ελεύθερος χώρος ή ίση με το ‘id’ του εμποδίου που είναι στην θέση αυτή.

Ακόμη η numpy παρέχει λειτουργίες για τον υπολογισμό αποστάσεων και γεωμετρικών σχέσεων. Στο κώδικα υλοποίησης των τριών αλγορίθμων, η συνάρτηση np.linalg.norm χρησιμοποιείται για τον υπολογισμό της ευκλείδιας απόστασης μεταξύ δύο κόμβων.

Επισης επιτρέπει την εύκολη επαλήθευση συνθηκών μέσα σε πίνακες. Στη συνάρτηση is_valid_edge, η numpy χρησιμοποιείται για να ελέγξει αν μια ακμή περνά μέσα από εμπόδια, χρησιμοποιώντας τη λειτουργία np.linspace για τη δημιουργία σημείων κατά μήκος της ακμής.

4.1.1.2 Βιβλιοθήκη NetworkX

Η βιβλιοθήκη networkx είναι ένα ισχυρό εργαλείο για τη δημιουργία, διαχείριση και ανάλυση γραφημάτων. Στην υλοποίηση των αλγορίθμων, η networkx χρησιμοποιήθηκε για τη δημιουργία των γραφημάτων που αναπαριστούν τα σημεία ενδιαφέροντος και τις διαδρομές μεταξύ τους. Παρακάτω αναλύονται οι βασικές λειτουργίες και χρήσεις της networkx στον κώδικα υλοποίησης των τριών αλγορίθμων.

Η συνάρτηση nx.read_gexf χρησιμοποιείται για την ανάγνωση γραφήματος από ένα αρχείο .gexf. Αυτό επιτρέπει τη φόρτωση προϋπάρχοντων γραφημάτων που έχουν δημιουργηθεί και αποθηκευτεί προηγουμένως.

Οι κόμβοι αντιπροσωπεύουν σημεία ενδιαφέροντος, όπως οι γωνίες των εμποδίων και τα σημεία εκκίνησης και τερματισμού. Οι ακμές αντιπροσωπεύουν τις πιθανές διαδρομές μεταξύ των κόμβων, και κάθε ακμή έχει ένα βάρος που αντιπροσωπεύει την απόσταση μεταξύ των δύο κόμβων. Η μέθοδος G.add_nodes_from χρησιμοποιείται για την προσθήκη πολλαπλών κόμβων ταυτόχρονα, ενώ η μέθοδος G.add_edge προσθέτει ακμές με το αντίστοιχο βάρος.

Η συνάρτηση nx.shortest_path χρησιμοποιείται για να βρεθεί η συντομότερη διαδρομή μεταξύ δύο κόμβων στο γράφημα, λαμβάνοντας υπόψη το βάρος των ακμών. Επιπλέον, η nx.shortest_path_length υπολογίζει το συνολικό μήκος της συντομότερης διαδρομής.

4.1.1.3 Βιβλιοθήκη Matplotlib

Η βιβλιοθήκη matplotlib είναι βιβλιοθήκη που χρησιμοποιείται για τη δημιουργία γραφικών στην Python. Στην υλοποίηση των τριών αλγορίθμων, η matplotlib χρησιμοποιείται για την απεικόνιση του περιβάλλοντος με τα εμπόδια, την τοποθέτηση των κόμβων, και την οπτικοποίηση της συντομότερης διαδρομής που υπολογίζεται από τους αλγορίθμους. Παρακάτω αναλύονται οι βασικές λειτουργίες και χρήσεις της matplotlib στον κώδικα.

Η matplotlib παρέχει δυνατότητες για τη δημιουργία προσαρμοσμένων χρωματικών χαρτών. Ο χρωματικός χάρτης ListedColormap χρησιμοποιείται για να αναπαραστήσει τα εμπόδια (μαύρο) και τον ελεύθερο χώρο (λευκό).

Η συνάρτηση plt.imshow χρησιμοποιείται για την απεικόνιση του πίνακα array_2d που αναπαριστά το περιβάλλον. Οι τιμές του πίνακα καθορίζουν εάν ένα κελί είναι εμπόδιο ή ελεύθερος χώρος.

Τα σημεία εκκίνησης (μπλε) και τερματισμού (κίτρινο) αναπαρίστανται χρησιμοποιώντας τη συνάρτηση plt.plot. Αυτό διευκολύνει την οπτικοποίηση της θέσης εκκίνησης και του στόχου στον χάρτη.

Η συντομότερη διαδρομή που υπολογίζεται από τον αλγόριθμο απεικονίζεται με κόκκινη γραμμή, χρησιμοποιώντας τη plt.plot. Οι συντεταγμένες της διαδρομής αποθηκεύονται σε έναν πίνακα numpy, ο οποίος στη συνέχεια χρησιμοποιείται για την απεικόνιση της διαδρομής.

Τέλος, η συνάρτηση plt.show χρησιμοποιείται για την εμφάνιση του γραφήματος. Αυτή η εντολή εμφανίζει το περιβάλλον με τα εμπόδια, τους κόμβους και τη διαδρομή που υπολογίστηκε.

4.1.1.4 Βιβλιοθήκη Time

Η βιβλιοθήκη time της Python παρέχει διάφορες λειτουργίες για τη διαχείριση και τη μέτρηση του χρόνου. Στην υλοποίηση των αλγορίθμων, η βιβλιοθήκη time χρησιμοποιείται για τη μέτρηση του χρόνου εκτέλεσης κάθε αλγορίθμου, επιτρέποντας τη σύγκριση της αποδοτικότητάς τους. Παρακάτω αναλύονται οι βασικές λειτουργίες και χρήσεις της time στον κώδικα.

Η συνάρτηση time.time() χρησιμοποιείται για την καταγραφή του τρέχοντος χρόνου σε δευτερόλεπτα. Αυτή η μέτρηση χρησιμοποιείται ως χρόνος έναρξης της διαδικασίας.

Μεταξύ των καταγραφών του χρόνου έναρξης και λήξης, εκτελείται ο αλγόριθμος ή η διαδικασία που θέλουμε να μετρήσουμε. Στον κώδικα υλοποίησης των τριών αλγορίθμων, εκτελούνται διάφορες λειτουργίες όπως η δημιουργία του γραφήματος, η προσθήκη κόμβων και ακμών, και η εύρεση της συντομότερης διαδρομής με τον αλγόριθμο Dijkstra.

Μετά την ολοκλήρωση της διαδικασίας, η συνάρτηση time.time() χρησιμοποιείται ξανά για την καταγραφή του τρέχοντος χρόνου, που αντιπροσωπεύει το χρόνο λήξης της διαδικασίας. Ακολούθως ο συνολικός χρόνος εκτέλεσης υπολογίζεται ως η διαφορά μεταξύ του χρόνου λήξης και του χρόνου έναρξης. Αυτός ο χρόνος εκφράζεται σε δευτερόλεπτα και μπορεί να χρησιμοποιηθεί για την ανάλυση της αποδοτικότητας των αλγορίθμων.

4.1.2 Περιγραφή της Υλοποίησης

Η υλοποίηση των αλγορίθμων πραγματοποιήθηκε σε ένα περιβάλλον ανάπτυξης που περιλάμβανε τα εξής βήματα

4.1.2.1 Προετοιμασία Δεδομένων

Η προετοιμασία δεδομένων περιλαμβάνει τη δημιουργία του περιβάλλοντος, την εισαγωγή των εμποδίων και τον καθορισμό των σημείων εκκίνησης και τερματισμού. Τα δεδομένα των εμποδίων αποθηκεύονται σε αρχεία και φορτώνονται στη μνήμη για επεξεργασία.

Δημιουργία του Περιβάλλοντος

Αρχικά, δημιουργούμε τον πίνακα που αναπαριστά το περιβάλλον, ορίζοντας το μέγεθός του με βάση τα δεδομένα από ένα αρχείο. Ο πίνακας αυτός περιέχει τιμές που υποδεικνύουν αν ένα κελί είναι εμπόδιο ή ελεύθερος χώρος.

```
import numpy as np

# Διαβάζουμε το μέγεθος του περιβάλλοντος από το αρχείο
file_path = 'Graphs/Graph1/area_size.txt'
with open(file_path, 'r') as f:
    for line in f:
        area_size = int(line.strip())

# Δημιουργούμε τον πίνακα για το περιβάλλον
area = np.zeros((area_size, area_size), dtype=int)
```

Φόρτωση Εμποδίων

Τα δεδομένα των εμποδίων φορτώνονται από ένα αρχείο κειμένου και αποθηκεύονται σε μια λίστα από λεξικά. Κάθε εμπόδιο ορίζεται από τις συντεταγμένες της θέσης του και τις διαστάσεις του.

```
# Φόρτωση δεδομένων εμποδίων από αρχείο
file_path = 'Graphs/Graph1/obstacles.txt'
listOfObstacles = []
with open(file_path, 'r') as f:
    for line in f:
        parts = line.strip().split(', ')
        obstacle = {}
        for part in parts:
            key, value = part.split(": ")
            obstacle[key] = int(value)
        listOfObstacles.append(obstacle)

# Η λίστα listOfObstacles περιέχει τώρα τα δεδομένα των εμποδίων
print(listOfObstacles)
```

Φόρτωση Περιβάλλοντος από Αρχείο

Ο πίνακας του περιβάλλοντος φορτώνεται από ένα αρχείο numpy, το οποίο περιέχει την αναπαράσταση του περιβάλλοντος με τα εμπόδια.

```
# Φόρτωση του πίνακα του περιβάλλοντος από αρχείο
area = np.load('Graphs/Graph1/area.npy')
print(area_size)
```

4.1.2.2 Δημιουργία Γραφημάτων

Η δημιουργία γραφημάτων περιλαμβάνει, στην περίπτωση των αλγορίθμων Rectangles και APPATTrectangles τη τοποθέτηση κόμβων στα σημεία ενδιαφέροντος (γωνίες ορθογώνιων εμποδίων), και στην περίπτωση του αλγορίθμου PRM την τοποθέτηση κόμβων σε τυχαία σημεία. Επίσης περιλαμβάνει τη σύνδεσή τους με ακμές που αναπαριστούν τις πιθανές διαδρομές. Χρησιμοποιείται η βιβλιοθήκη networkx για την κατασκευή του γραφήματος και τον υπολογισμό της συντομότερης διαδρομής.

Δημιουργία Γραφήματος και Προσθήκη Κόμβων

Στην περιπτωση των αλγοριθμων APPATTrectangles και Rectangles οι κόμβοι τοποθετούνται στις γωνίες των εμποδίων και στα σημεία εκκίνησης και τερματισμού. Στη συνέχεια, προστίθενται στο γράφημα.

```
import networkx as nx

# Δημιουργία γραφήματος
G = nx.Graph()

# Προσθήκη κόμβων στις γωνίες των εμποδίων
for obstacle in listOfObstacles:
```

```
    x, y = obstacle['x'], obstacle['y']
    width, height = obstacle['width'], obstacle['height']
    if area[x-1][y-1] == 0:
        graph_nodes.append((x-1, y-1))
    if area[x-1][y+height] == 0:
        graph_nodes.append((x-1, y+height))
    if area[x+width][y-1] == 0:
        graph_nodes.append((x+width, y-1))
    if area[x+width][y+height] == 0:
        graph_nodes.append((x+width, y+height))

# Προσθήκη σημείων εκκίνησης και τερματισμού
start_point = (area_size//2, 1)
end_point = (area_size//2, area_size - 2)
graph_nodes.extend([start_point, end_point])
G.add_nodes_from(graph_nodes)
```

Στην περίπτωση του αλγορίθμου PRM οι κομβόι τοποθετούνται σε τυχαία σημεία.

```
# Βρες διαθέσιμο ελεύθερο χώρο (έξω από εμπόδια)
available_space_nodes = [(i, j) for i in range(1, area_size - 1) for
j in range(1, area_size - 1) if area[i, j] == 0]

# δημιουργία κόμβων σε τυχαία σημεία
for i in range(num_nodes):
    random_index = np.random.choice(len(available_space_nodes))
    random_point = available_space_nodes[random_index]
    graph_nodes.append(random_point)
    temp_nodes.append(random_point)
```

Σύνδεση Κόμβων με Ακμές

Οι κόμβοι συνδέονται μεταξύ τους με ακμές, εφόσον η απόσταση μεταξύ τους είναι μικρότερη από ένα προκαθορισμένο όριο και η ακμή δεν τέμνει κάποιο εμπόδιο. Στο παράδειγμα κώδικα που ακολουθεί το όριο αυτό είναι ίσο με το διπλάσιο του πλάτους της περιοχής. Έχει τεθεί έτσι ώστε να είμαστε βέβαιοι ότι όλοι οι κομβόι συνδέονται μεταξύ τους.

```
# Συνάρτηση για τον έλεγχο αν μια ακμή είναι έγκυρη (δεν τέμνει
εμπόδια)
def is_valid_edge(edge, area):
    for point in np.linspace(edge[0], edge[1], num=100):
        x, y = map(int, point)
        if area[x, y] != 0:
            return False
    return True

# Σύνδεση κόμβων με ακμές και υπολογισμός βάρους (απόστασης)
threshold = 2 * area_size
for i, node in enumerate(graph_nodes):
    for j in range(i + 1, len(graph_nodes)):
        other_node = graph_nodes[j]
        distance = np.linalg.norm(np.array(node) -
np.array(other_node))
```

```
        if distance < threshold and is_valid_edge([node,
other_node], area):
            G.add_edge(tuple(node), tuple(other_node),
weight=distance)
```

4.1.2.3 Απεικόνιση Αποτελεσμάτων

Η Matplotlib χρησιμοποιήθηκε για την απεικόνιση των εμποδίων, των κόμβων και των διαδρομών. Μέσω γραφημάτων, μπορούσαμε να οπτικοποιήσουμε το περιβάλλον και τις διαδρομές που υπολογίστηκαν από τους αλγορίθμους, διευκολύνοντας την κατανόηση και την ανάλυση των αποτελεσμάτων.

```
import matplotlib.pyplot as plt

# Συνάρτηση απεικόνισης γραφήματος και διαδρομής
def display_array_with_graph_and_path(array_2d, start_point,
end_point, path):
    cmap = plt.cm.colors.ListedColormap(['white', 'black'])
    bounds = [0, 1]
    norm = plt.cm.colors.BoundaryNorm(bounds, cmap.N)

    plt.imshow(array_2d, cmap=cmap, norm=norm, interpolation='none')

    plt.plot(start_point[1], start_point[0], 'bo', markersize=8,
label='Start (A)')
    plt.plot(end_point[1], end_point[0], 'yo', markersize=8,
label='End (B)')

    if path:
        path_nodes = np.array(path)
        plt.plot(path_nodes[:, 1], path_nodes[:, 0], 'r-',
linewidth=2, label='Shortest Path (A to B)')

    cbar = plt.colorbar(ticks=[0, 1, 2])
    cbar.set_label('Color', rotation=270, labelpad=15)

    plt.show()

# Απεικόνιση του περιβάλλοντος, των κόμβων και της διαδρομής
display_array_with_graph_and_path(area, start_point, end_point,
shortest_path)
```

4.1.2.4 Μέτρηση Χρόνου Εκτέλεσης

Η μέτρηση του χρόνου εκτέλεσης γίνεται με τη βιβλιοθήκη time, καταγράφοντας τον χρόνο έναρξης και λήξης κάθε διαδικασίας. Αυτό επιτρέπει τη σύγκριση της αποδοτικότητας των διαφορετικών αλγορίθμων.

Η συνάρτηση time.time() χρησιμοποιείται για την καταγραφή του χρόνου έναρξης και λήξης, και ο συνολικός χρόνος εκτέλεσης υπολογίζεται ως η διαφορά μεταξύ αυτών των δύο τιμών.

```
import time
```

```
# Προετοιμασία δεδομένων

# Καταγραφή χρόνου έναρξης
start_time = time.time()

# Εκτέλεση αλγορίθμου

# Καταγραφή χρόνου λήξης
end_time = time.time()

# Υπολογισμός χρόνου εκτέλεσης αλγορίθμου
total_time = end_time - start_time
print(f"Total execution time: {total_time} seconds")
```

4.2 Βασικές συναρτήσεις

Σε αυτή την ενότητα παρουσιάζονται και εξηγούνται οι βασικές συναρτήσεις που χρησιμοποιούνται από τους αλγόριθμους PRM, Rectangles και APPATTrectangles για την υλοποίησή τους. Κάθε συνάρτηση αναλύεται ως προς τη λειτουργία της και τον ρόλο της στη σωστή εκτέλεση των αλγορίθμων.

4.2.1 Συνάρτηση get_closest_nodes

Αυτή η συνάρτηση χρησιμοποιείται στον αλγόριθμο PRM για να εντοπίσει τους πλησιέστερους κόμβους από έναν δεδομένο κόμβο. Ο αλγόριθμος PRM μπορεί να τρέξει και χωρίς αυτή την συνάρτηση (δηλαδή να συνδέονται όλοι οι δυνατοί κομβόι μεταξύ τους και όχι οι πλησιέστεροι) παρόλα αυτά έχει γίνει χρήση της για περισσότερη ευελιξία του αλγορίθμου.

```
def get_closest_nodes(current_node, all_nodes, n):
    distances = [np.linalg.norm(np.array(current_node) -
np.array(node)) for node in all_nodes]
    sorted_indices = np.argsort(distances)
    return [all_nodes[i] for i in sorted_indices[:n]]
```

- Είσοδοι:
    - **current_node:** Ο κόμβος από τον οποίο θέλουμε να βρούμε τους πλησιέστερους κόμβους.
    - **all_nodes:** Λίστα με όλους τους κόμβους στο γράφημα.
    - **n:** Ο αριθμός των πλησιέστερων κόμβων που θέλουμε να βρούμε.
- Έξοδος:
    - Επιστρέφει μια λίστα με τους n πλησιέστερους κόμβους από τον current_node.
- Λειτουργία:
    - Υπολογίζει την απόσταση κάθε κόμβου από τον current_node.
    - Ταξινομεί τις αποστάσεις σε αύξουσα σειρά.
    - Επιστρέφει τους n πλησιέστερους κόμβους.

4.2.2 Συνάρτηση get_obstacle_collided_id

Αυτή η συνάρτηση χρησιμοποιείται στον αλγόριθμο APPATTrectangles για να ελέγξει αν μια ακμή τέμνει κάποιο εμπόδιο και να επιστρέψει το ID του εμποδίου.

```
def get_obstacle_collided_id(edge, area):
    for point in np.linspace(edge[0], edge[1], num=100):
        x, y = map(int, point)
        if area[x, y] != 0:
            return area[x,y]
    return -1
```

- Είσοδοι:
    - **edge:** Η ακμή που θέλουμε να ελέγξουμε.
    - **area:** Ο πίνακας που αναπαριστά το περιβάλλον με τα εμπόδια.
- Έξοδος:
    - Επιστρέφει το ID του εμποδίου που τέμνει την ακμή ή -1 αν δεν υπάρχει σύγκρουση.
- Λειτουργία:
    - Δημιουργεί σημεία κατά μήκος της ακμής.
    - Ελέγχει αν κάποιο από αυτά τα σημεία βρίσκεται πάνω σε εμπόδιο.
    - Επιστρέφει το ID του εμποδίου αν υπάρχει σύγκρουση, αλλιώς επιστρέφει -1.

4.2.3 Συνάρτηση is_valid_edge

Αυτή η συνάρτηση χρησιμοποιείται για να ελέγξει αν μια ακμή είναι έγκυρη, δηλαδή αν δεν τέμνει κάποιο εμπόδιο.

```
def is_valid_edge(edge, area):
    for point in np.linspace(edge[0], edge[1], num=100):
        x, y = map(int, point)
        if area[x, y] != 0:
            return False
    return True
```

- Είσοδοι:
    - **edge:** Η ακμή που θέλουμε να ελέγξουμε.
    - **area:** Ο πίνακας που αναπαριστά το περιβάλλον με τα εμπόδια.
- Έξοδος:
    - Επιστρέφει True αν η ακμή δεν τέμνει κάποιο εμπόδιο, αλλιώς επιστρέφει False.
- Λειτουργία:
    - Δημιουργεί σημεία κατά μήκος της ακμής.
    - Ελέγχει αν κάποιο από αυτά τα σημεία βρίσκεται πάνω σε εμπόδιο.
    - Επιστρέφει False αν υπάρχει σύγκρουση, αλλιώς επιστρέφει True.
    - Στην περίπτωση του αλγορίθμου APPATTrectangles η συνάρτηση επιστρέφει την τιμή area[x,y] ώστε να γνωρίζουμε το id του εμποδίου με το οποίο συγκρούεται η συγκεκριμένη ακμή.

4.2.4 Συνάρτηση get_obstacle

Αυτή η συνάρτηση επιστρέφει τις πληροφορίες ενός εμποδίου με βάση το ID του.

```
def get_obstacle(listOfObstacles, id):
    for obstacle in listOfObstacles:
        if obstacle['id'] == id:
            return obstacle
```

- Είσοδοι:
    - **listOfObstacles**: Λίστα με όλα τα εμπόδια.
    - **id:** Το ID του εμποδίου που θέλουμε να βρούμε.
- Έξοδος:
    - Επιστρέφει το dictionary με τις πληροφορίες του εμποδίου που έχει το συγκεκριμένο ID.
- Λειτουργία:
    - Διατρέχει τη λίστα των εμποδίων.
    - Επιστρέφει το εμπόδιο με το αντίστοιχο ID.

4.2.5 Συνάρτηση get_nodes

Η συνάρτηση αυτή προσθέτει κόμβους στης γωνίες του εμποδίου που έχει βρει η συνάρτηση get_obstacle.

```
def get_temp_nodes(obstacle, area):
    counter = 0
    graph_nodes = []

    x = obstacle['x']
    y = obstacle['y']
    obstacle_width_for = obstacle['width']
    obstacle_height_for = obstacle['height']
    counter = counter + 1
    if area[x - 1][y - 1] == 0:
        graph_nodes.append((x - 1, y - 1))
    if area[x - 1][y + obstacle_height_for] == 0:
        graph_nodes.append((x - 1, y + obstacle_height_for))
    if area[x + obstacle_width_for][y - 1] == 0:
        graph_nodes.append((x + obstacle_width_for, y - 1))
    if area[x + obstacle_width_for][y + obstacle_height_for] == 0:
        graph_nodes.append((x + obstacle_width_for, y +
obstacle_height_for))
    return graph_nodes
```

- Είσοδοι:
    - **obstacle:** Το εμπόδιο γύρω από το οποίο θα δημιουργηθούν κόμβοι.
    - **area:** Ο πίνακας που αναπαριστά το περιβάλλον με τα εμπόδια.
- Έξοδος:
    - Επιστρέφει μια λίστα με τους προσωρινούς κόμβους γύρω από το εμπόδιο.
- Λειτουργία:
    - Δημιουργεί κόμβους στις γωνίες του εμποδίου.
    - Ελέγχει αν οι κόμβοι αυτοί βρίσκονται σε ελεύθερο χώρο και τους προσθέτει στη λίστα αν είναι έγκυροι.

4.2.6 Συνάρτηση add_node

Ελέγχει αν ο κόμβος δεν είναι στο σύνολο μοναδικών κόμβων. Αν δεν είναι τότε τον προσθέτει στην λίστα με τους κόμβους τους γραφήματος και στο σύνολο τον μοναδικών κόμβων.

4.2.7 Συνάρτηση add_new_obstacle_found

Ελέγχει αν το id του δοθέντος εμποδίου βρίσκεται στο σύνολο με μοναδικά id εμποδίων. Αν δεν βρίσκεται τότε προσθέτει το id στο σύνολο αυτό και επίσης προσθέτει το εμπόδιο στην λίστα με τα εμπόδια που βρήκε ο αλγόριθμος μέχρι στιγμής.

4.3 Υλοποίηση του Αλγορίθμου PRM

Σε αυτή την ενότητα παρουσιάζεται η υλοποίηση του αλγορίθμου PRM σε ψευδοκώδικα με βάση τις συναρτήσεις που έχουμε ορίσει ενότητα 4.2 Βασικές Συναρτήσεις. Ακολουθούν τα βήματα του αλγορίθμου:

- Αρχικά τα δεδομένα φορτώνονται όπως ορίζει η υπο-ενότητα 4.1.2.1 Προετοιμασία Δεδομένων.
- Ορισμός αρχικού και τελικού σημείου.
- Ορισμός αριθμού κόμβων που θα προσθέτει ο αλγόριθμος σε κάθε επανάληψη num_nodes
- Ορισμός αριθμού κοντινότερων κόμβων n_connections
- Εύρεση ελευθέρου χώρου όπως ορίζει η υπο-παράγραφος 4.1.2.2.1 για τον αλγόριθμο PRM
- While path_created == False
    - Δημιουργία τυχαίων κόμβων των οποίων το πλήθος είναι ίσο με num_nodes όπως ορίζει η υπο-παράγραφος 4.1.2.2.1
    - Προσθήκη των κόμβων στις λίστες temp_nodes και graph_nodes.
    - For temp_node in temp_nodes
        - For target_node in graph_nodes
            - Edge = (temp_node, target_node)
            - If is_valid_edge(Edge, area)
                - Πρόσθεσε την ακμή στο γράφημα όπως ορίζει η υπο-παράγραφος 4.1.1.2.2
    - Closest_start_nodes = get_closest_nodes(start_point, temp_nodes, n_connections)
    - For target_node in Closest_start_nodes
        - Edge = (start_point, target_node)
        - If is_valid_edge(Edge, area)
            - Πρόσθεσε την ακμή στο γράφημα όπως ορίζει η υπο-παράγραφος 4.1.1.2.2
    - Closest_end_nodes = get_closest_nodes(end_point, temp_nodes, n_connections)
    - For target_node in Closest_end_nodes
        - Edge = (end_point, target_node)

        - If is_valid_edge(Edge, area)
            - Πρόσθεσε την ακμή στο γράφημα όπως ορίζει η υπο-παράγραφος 4.1.1.2.2
    - temp_nodes.clear()
    - Τρέξε αλγόριθμο Dijkstra στο γράφημα που έχει δημιουργηθεί όπως ορίζει η υπο-παράγραφος 4.1.1.2.3 και βρες συντομότερη διαδρομή από το αρχικό στο τελικό σημείο.
    - Αν η διαδρομή βρεθεί
        - Path_created = True
- Απεικόνιση αποτελεσμάτων όπως εξηγείται στην παράγραφο 4.1.2.3

4.4 Υλοποίηση του Αλγορίθμου Rectangles

Σε αυτή την ενότητα παρουσιάζεται η υλοποίηση του αλγορίθμου Rectangles σε ψευδοκώδικα με βάση τις συναρτήσεις που έχουμε ορίσει ενότητα 4.2 Βασικές Συναρτήσεις. Ακολουθούν τα βήματα του αλγορίθμου:

- Αρχικά τα δεδομένα φορτώνονται όπως ορίζει η υπο-ενότητα 4.1.2.1 Προετοιμασία Δεδομένων.
- Ορισμός αρχικού και τελικού σημείου και προσθήκη τους στην λίστα graph_nodes.
- For obstacle in listOfObstacles
    - nodes = get_nodes(obstacle, area)
    - graph_nodes = graph_nodes + nodes
    - Καθαρισμός λίστας nodes
- Δημιουργία γραφήματος όπως εξηγεί η παράγραφος 4.1.2.2
- Τρέξε αλγόριθμο Dijkstra στο γράφημα που έχει δημιουργηθεί όπως ορίζει η υπο-παράγραφος 4.1.1.2.3 και βρες συντομότερη διαδρομή από το αρχικό στο τελικό σημείο.
- Απεικόνιση αποτελεσμάτων όπως εξηγείτε στην παράγραφο 4.1.2.3

4.5 Υλοποίηση του Αλγορίθμου APPATTrectangles

Σε αυτή την ενότητα παρουσιάζεται η υλοποίηση του αλγορίθμου APPATTrectangles σε ψευδοκώδικα με βάση τις συναρτήσεις που έχουμε ορίσει ενότητα 4.2 Βασικές Συναρτήσεις. Ακολουθούν τα βήματα του αλγορίθμου:

- Αρχικά τα δεδομένα φορτώνονται όπως ορίζει η υπο-ενότητα 4.1.2.1 Προετοιμασία Δεδομένων.
- Ορισμός αρχικού και τελικού σημείου και προσθήκη τους στην λίστα graph_nodes.
- edge = (start_point, end_point)
- if get_obstacle_collided_id(edge, area) != -1
    - id = get_obstacle_collided_id(edge, area)
    - obstacle = get_obstacle(listOfObstacles, id)
    - nodes = get_nodes(obstacle, area)
    - for node in nodes
        - add_node(node)

- καθαρισμός λίστας nodes
- add_new_obstacle_found(obstacle)
- for node in graph_nodes
  - edge_front = (node, end_point)
  - edge_back = (node, start_point)
  - if get_obstacle_collided_id(edge front, area) == -1
    - τερματισμος βρογχου
  - else
    - if get_obstacle_collided_id(edge_back, area) != -1
      - id_back = get_obstacle_collided_id(edge_back, area)
      - obstacle_back = get_obstacle(listOfObstacles, id_back)
      - if obstacle_back['id'] not in unique_obstacles_found
        - nodes = get_nodes(obstacle_back, area)
        - for node in nodes
          - add_node(node)
          - καθαρισμός λίστας nodes
          - add_new_obstacle_found(obstacle_back)
    - if get_obstacle_collided_id(edge_front, area) != -1
      - id_front = get_obstacle_collided_id(edge_front, area)
      - obstacle_front = get_obstacle(listOfObstacles, id_front)
      - if obstacle_front['id'] not in unique_obstacles_found
        - nodes = get_nodes(obstacle_front, area)
        - for node in nodes
          - add_node(node)
          - καθαρισμός λίστας nodes
          - add_new_obstacle_found(obstacle_front)

- Δημιουργία γραφήματος όπως εξηγεί η παράγραφος 4.1.2.2
- Τρέξε αλγόριθμο Dijkstra στο γράφημα που έχει δημιουργηθεί όπως ορίζει η υπο-παράγραφος 4.1.1.2.3 και βρες συντομότερη διαδρομή από το αρχικό στο τελικό σημείο.
- Απεικόνιση αποτελεσμάτων όπως εξηγείτε στην παράγραφο 4.1.2.3

4.6 Συγκριτική Ανάλυση και Προκλήσεις

Οι τρεις αλγόριθμοι που αναπτύχθηκαν ακολουθούν διαφορετικές προσεγγίσεις για την επίλυση του προβλήματος σχεδιασμού διαδρομής UAV σε περιβάλλοντα με εμπόδια. Καθένας από αυτούς προσφέρει πλεονεκτήματα και μειονεκτήματα, τα οποία εξετάζονται αναλυτικά.

PRM (Probabilistic Roadmap)

Ο PRM βασίζεται στην τυχαία τοποθέτηση κόμβων σε ένα ελεύθερο περιβάλλον. Η τυχαία φύση του τον καθιστά έναν εξαιρετικά ευέλικτο αλγόριθμο που μπορεί να εφαρμοστεί σε περιβάλλοντα με διαφορετικά χαρακτηριστικά, αλλά ταυτόχρονα περιορίζει την ακρίβειά του στην εύρεση της βέλτιστης διαδρομής. Ένα από τα μεγαλύτερα προβλήματα που αντιμετωπίστηκαν ήταν η πολυπλοκότητα της σύνδεσης των κόμβων με άλλους κόμβους, εξαιτίας των συνεχών συγκρούσεων με εμπόδια. Κατά τη διάρκεια της υλοποίησης του PRM, προκλήσεις προέκυψαν κατά τον καθορισμό του κατάλληλου αριθμού κόμβων και την επιλογή της απόστασης σύνδεσης των κόμβων.

Rectangles

Ο αλγόριθμος Rectangles προσφέρει εγγυημένη εύρεση της βέλτιστης διαδρομής, καθώς βασίζεται σε πλήρη ανάλυση των γεωμετρικών σχέσεων μεταξύ των εμποδίων και των διαδρομών. Κατά την υλοποίηση του αλγορίθμου Rectangles, η μεγαλύτερη πρόκληση ήταν η αυξημένη υπολογιστική πολυπλοκότητα σε περιβάλλοντα με πολλά εμπόδια. Ο αλγόριθμος εξετάζει όλες τις πιθανές συνδέσεις μεταξύ των κόμβων που δημιουργούνται γύρω από τα εμπόδια, γεγονός που οδηγεί σε σημαντικές καθυστερήσεις όταν αυξάνεται ο αριθμός των εμποδίων. Ωστόσο, η αξιοπιστία του ως προς την εύρεση της βέλτιστης διαδρομής καθιστά τον αλγόριθμο ιδανικό για περιβάλλοντα όπου η ακρίβεια είναι πιο σημαντική από την ταχύτητα.

APPATTrectangles

Ο αλγόριθμος APPATTrectangles συνδυάζει τα πλεονεκτήματα των δύο προηγούμενων αλγορίθμων. Ενώ ο αλγόριθμος Rectangles απαιτεί τον έλεγχο όλων των πιθανών συνδέσεων, ο APPATTrectangles επικεντρώνεται μόνο σε κρίσιμες περιοχές του περιβάλλοντος, δηλαδή γύρω από τα εμπόδια, γεγονός που μειώνει σημαντικά τον χρόνο υπολογισμού. Η εφαρμογή του έδειξε ότι μπορεί να βρει διαδρομές με πολύ μικρό ποσοστό σφάλματος σε σχέση με τη βέλτιστη διαδρομή. Ο συνδυασμός ταχύτητας και ακρίβειας τον καθιστά ιδανικό για περιβάλλοντα όπου απαιτείται γρήγορη απόκριση.

4.6.1 Προκλήσεις Κατά την Υλοποίηση

Η υλοποίηση των αλγορίθμων παρουσίασε σημαντικές προκλήσεις, οι οποίες αφορούσαν κυρίως την αποτελεσματική διαχείριση της πολυπλοκότητας των γραφημάτων και την αποδοτική ανίχνευση συγκρούσεων με τα εμπόδια.

- **Διαχείριση πολυπλοκότητας**: Ένας από τους βασικούς περιορισμούς κατά την υλοποίηση των αλγορίθμων ήταν η διαχείριση των κόμβων και των ακμών στα γραφήματα, ειδικά σε περιβάλλοντα με μεγάλο αριθμό εμποδίων. Ο αλγόριθμος Rectangles αντιμετώπισε τις μεγαλύτερες δυσκολίες σε αυτόν τον τομέα, καθώς κάθε επιπλέον εμπόδιο αυξάνει εκθετικά τον αριθμό των κόμβων και ακμών που πρέπει να εξεταστούν.
- **Ανίχνευση συγκρούσεων**: Η ανίχνευση συγκρούσεων μεταξύ των ακμών του γράφου και των εμποδίων αποδείχθηκε ένας από τους πιο απαιτητικούς υπολογιστικούς στόχους. Κατά την εφαρμογή των αλγορίθμων, ήταν απαραίτητο να διασφαλιστεί ότι οι ακμές που προστίθενται στο γράφημα δεν τέμνονται με εμπόδια. Αυτό προκάλεσε καθυστερήσεις στην απόδοση των αλγορίθμων σε πολύπλοκα περιβάλλοντα, όπου ο αριθμός των ελέγχων αυξανόταν δραματικά.
- **Βελτιστοποίηση για μεγάλα περιβάλλοντα**: Ένα από τα κύρια ζητήματα ήταν η εφαρμογή των αλγορίθμων σε μεγάλα περιβάλλοντα με υψηλή πυκνότητα εμποδίων. Ο APPATTrectangles αποδείχθηκε ο πιο αποδοτικός από τους τρεις, καθώς ήταν ικανός να επικεντρώνεται στα κρίσιμα σημεία του περιβάλλοντος, μειώνοντας την ανάγκη για υπερβολικούς υπολογισμούς.

Η σύγκριση των αλγορίθμων έδειξε ότι, παρόλο που ο PRM προσφέρει μεγαλύτερη ευελιξία και ο Rectangles εγγυάται τη βέλτιστη λύση, ο APPATTrectangles καταφέρνει να ισορροπήσει μεταξύ ταχύτητας και ακρίβειας, αποτελώντας τη βέλτιστη λύση για γρήγορες και αποδοτικές πτήσεις UAV σε περιβάλλοντα με εμπόδια .

5

Πειραματική αξιολόγηση

Στο κεφάλαιο αυτό, θα εξετάσουμε τον τρόπο με τον οποίο οι αλγόριθμοι PRM, Rectangles και APPATTrectangles αποδίδουν σε διάφορα σενάρια και περιβάλλοντα. Μέσω αυτής της αξιολόγησης, θα αναδειχθούν τα πλεονεκτήματα και τα μειονεκτήματα κάθε αλγορίθμου, καθώς και οι συνθήκες υπό τις οποίες αποδίδουν καλύτερα.

Αρχικά, θα περιγράψουμε το πειραματικό περιβάλλον στο οποίο εκτελέστηκαν οι αλγόριθμοι, συμπεριλαμβανομένων των εργαλείων και των τεχνολογιών που χρησιμοποιήθηκαν. Στη συνέχεια, θα αναλύσουμε τα πειραματικά δεδομένα πως αυτά δημιουργήθηκαν και πως χρησιμοποιήθηκαν για την εκτέλεση των αλγορίθμων. Τέλος, θα παρουσιάσουμε τα πειραματικά αποτελέσματα, συγκρίνοντας τους αλγορίθμους και εξάγοντας συμπεράσματα για την αποδοτικότητά τους.

Η διεξαγωγή αυτών των πειραμάτων θα μας επιτρέψει να αποκτήσουμε μια σαφή εικόνα για την απόδοση των αλγορίθμων σε διαφορετικά περιβάλλοντα, ενισχύοντας έτσι την αξιοπιστία των συμπερασμάτων μας και παρέχοντας χρήσιμες πληροφορίες για μελλοντικές βελτιώσεις και εφαρμογές τους.

5.1 Πειραματικό Περιβάλλον

Σε αυτή την ενότητα θα περιγράψουμε το πειραματικό περιβάλλον στο οποίο υλοποιήθηκαν και εκτελέστηκαν οι αλγόριθμοι. Τα τεχνικά χαρακτηριστικά του υπολογιστή και το IDE που χρησιμοποιήθηκε εξασφάλισαν ότι τα πειράματα εκτελέστηκαν σε ένα σταθερό και αποδοτικό περιβάλλον, παρέχοντας αξιόπιστα αποτελέσματα για την αξιολόγηση των αλγορίθμων.

5.1.1 Εργαλεία και Τεχνολογίες

Για την υλοποίηση και εκτέλεση των πειραμάτων χρησιμοποιήθηκαν τα εξής εργαλεία και τεχνολογίες:

- Γλώσσα Προγραμματισμού**:** Python
- Βιβλιοθήκες:
    - **numpy:** Για τη διαχείριση και την επεξεργασία των δεδομένων.
    - **networkx:** Για τη δημιουργία και την ανάλυση των γραφημάτων.

- **matplotlib:** Για την οπτικοποίηση των αποτελεσμάτων.
- **time**: Για τη μέτρηση του χρόνου εκτέλεσης των αλγορίθμων.

5.1.2 Τεχνικά Χαρακτηριστικά του Υπολογιστή

Τα πειράματα εκτελέστηκαν σε υπολογιστή με τα εξής τεχνικά χαρακτηριστικά: ο επεξεργαστής (CPU) που χρησιμοποιήθηκε ήταν ο Intel Core i5-7500 με συχνότητα 3.40GHz και 4 πυρήνες, ενώ η μνήμη (RAM) του συστήματος ήταν 8 GB τύπου DDR4. Η κάρτα γραφικών (GPU) που υποστήριζε τη διαδικασία ήταν η NVIDIA GeForce GTX 1060, και το λειτουργικό σύστημα που χρησιμοποιήθηκε ήταν τα Windows 10 σε έκδοση 64-bit. Ο αποθηκευτικός χώρος που ήταν διαθέσιμος στον υπολογιστή ήταν 256 GB SSD.

5.1.3 Περιβάλλον Ανάπτυξης (IDE)

Η ανάπτυξη των αλγορίθμων πραγματοποιήθηκε στο περιβάλλον ανάπτυξης PyCharm Professional 2021.3. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε ήταν η Python, στην έκδοση 3.8.10. Όλα τα εξωτερικά πακέτα που χρειάστηκαν για την υλοποίηση των αλγορίθμων εγκαταστάθηκαν και διαχειρίστηκαν μέσω του διαχειριστή πακέτων pip.

5.2 Πειραματικά δεδομένα

Σε αυτή την ενότητα θα εξηγήσουμε πώς δημιουργούνται και αποθηκεύονται τα πειραματικά δεδομένα. Η διαδικασία περιλαμβάνει τη δημιουργία ενός τυχαίου περιβάλλοντος με εμπόδια και την αποθήκευση των δεδομένων που θα χρησιμοποιηθούν για την εκτέλεση των αλγορίθμων.

5.2.1 Δημιουργία Πειραματικών Δεδομένων

Η δημιουργία των πειραματικών δεδομένων περιλαμβάνει τη δημιουργία ενός δισδιάστατου πλέγματος που αναπαριστά το περιβάλλον και την τοποθέτηση τυχαίων ορθογώνιων εμποδίων σε αυτό. Παρακάτω παρουσιάζεται η ανάλυση του κώδικα που χρησιμοποιείται για τη δημιουργία των πειραματικών δεδομένων.

5.2.1.1 Συνάρτηση is_overlapping

Η διαδικασία ελέγχει αν ένα νέο εμπόδιο επικαλύπτεται με κάποιο ήδη υπάρχον εμπόδιο. Αρχικά, διατρέχει τη λίστα των εμποδίων και εξετάζει αν τα όρια του νέου εμποδίου τέμνονται με τα όρια των υπαρχόντων εμποδίων. Εάν διαπιστωθεί επικάλυψη, η διαδικασία επιστρέφει True, διαφορετικά επιστρέφει False.

```
# Συνάρτηση για τον έλεγχο αν ένα νέο εμπόδιο επικαλύπτεται με υπάρχοντα
εμπόδια
def is_overlapping(new_obstacle, obstacles):
    for obstacle in obstacles:
        if not (new_obstacle['x'] + new_obstacle['width'] < obstacle['x']
or
                new_obstacle['x'] > obstacle['x'] + obstacle['width'] or
                new_obstacle['y'] + new_obstacle['height'] < obstacle['y']
or
```

```
                new_obstacle['y'] > obstacle['y'] + obstacle['height']):
            return True
    return False
```

5.2.1.2 Αρχικοποίηση μεταβλητών

Ορίζονται αρχικά διάφορες παράμετροι για τη διαδικασία τοποθέτησης των εμποδίων, όπως ο μέγιστος αριθμός προσπαθειών για την τοποθέτηση κάθε εμποδίου, το μέγεθος του πλέγματος που αναπαριστά το περιβάλλον (area_size), το πλήθος των εμποδίων (num_obstacles), το ελάχιστο μέγεθος των εμποδίων (min_obstacle_size), το μέγιστο μέγεθος των εμποδίων (max_obstacle_size) και το μέγεθος της απόστασης ασφάλειας από τα εμπόδια. Στη συνέχεια, δημιουργείται ένας πίνακας area διαστάσεων area_size x area_size, ο οποίος αρχικοποιείται με μηδενικές τιμές, όπου το 0 αναπαριστά τον ελεύθερο χώρο.

5.2.1.3 Προσθήκη τυχαίων εμποδίων

Κατά την προσθήκη τυχαίων εμποδίων, κάθε εμπόδιο αποθηκεύεται σε μια λίστα με dictionaries, τη listOfObstacles. Για κάθε εμπόδιο αποθηκεύονται χαρακτηριστικά όπως το μήκος, το πλάτος, οι συντεταγμένες x και y της πάνω αριστερής γωνίας, καθώς και το μοναδικό του ID. Για τη δημιουργία κάθε εμποδίου, επιλέγονται τυχαία το μήκος και το πλάτος του, τα οποία τηρούν τα προκαθορισμένα όρια του ελάχιστου και μέγιστου μεγέθους. Στη συνέχεια, επιλέγεται ένα τυχαίο σημείο μέσα στο πλέγμα, το οποίο θα αποτελέσει την πάνω αριστερή γωνία του εμποδίου, και αυτό το σημείο πρέπει να επιλεγεί με τέτοιο τρόπο ώστε το εμπόδιο να χωράει ολόκληρο μέσα στο πλέγμα.

Έπειτα, γίνεται έλεγχος αν το νέο εμπόδιο επικαλύπτεται με κάποιο ήδη υπάρχον εμπόδιο χρησιμοποιώντας τη συνάρτηση **is_overlapping**. Εάν δεν εντοπιστεί επικάλυψη, το εμπόδιο προστίθεται στη λίστα **listOfObstacles** και ενημερώνεται ο πίνακας **area** με το ID του νέου εμποδίου. Εάν, όμως, μετά από τον προκαθορισμένο αριθμό προσπαθειών (**max_attempts_per_obstacle**) δεν είναι δυνατόν να τοποθετηθεί το εμπόδιο, εμφανίζεται μήνυμα αποτυχίας τοποθέτησης.

5.2.2 Αποθήκευση Πειραματικών Δεδομένων

Αφού ολοκληρωθεί η δημιουργία του πειραματικού περιβάλλοντος, τα δεδομένα αποθηκεύονται σε αρχεία για μελλοντική χρήση. Αρχικά, δημιουργείται ένα γράφημα **G** χρησιμοποιώντας τη βιβλιοθήκη **networkx** και ένας φάκελος με όνομα **Graphs/Graph1**, ο οποίος θα χρησιμεύσει για την αποθήκευση των δεδομένων που παράγονται από τα πειράματα. Το γράφημα, αν και στην παρούσα φάση είναι κενό, αποθηκεύεται για λόγους πληρότητας σε αρχείο με το όνομα **graph.gexf**.

Επιπλέον, οι πληροφορίες που αφορούν τα εμπόδια, όπως οι συντεταγμένες, το ID, το πλάτος και το ύψος τους, αποθηκεύονται σε αρχείο κειμένου με όνομα **obstacles.txt**. Για τη διατήρηση πληροφοριών σχετικά με το μέγεθος της περιοχής του πειραματικού περιβάλλοντος, το μέγεθος αυτό αποθηκεύεται σε αρχείο με όνομα **area_size.txt**. Τέλος, ο πίνακας που αναπαριστά την περιοχή αποθηκεύεται σε αρχείο **area.npy** χρησιμοποιώντας τη συνάρτηση **np.save** της βιβλιοθήκης **numpy**,

Αυτή η διαδικασία διασφαλίζει ότι όλα τα πειραματικά δεδομένα είναι σωστά οργανωμένα και έτοιμα για χρήση στις εκτελέσεις των αλγορίθμων. Η αποθήκευση των δεδομένων σε αρχεία επιτρέπει την επαναληπτική εκτέλεση των πειραμάτων με σταθερά δεδομένα, διευκολύνοντας την ανάλυση και τη σύγκριση των αποτελεσμάτων.

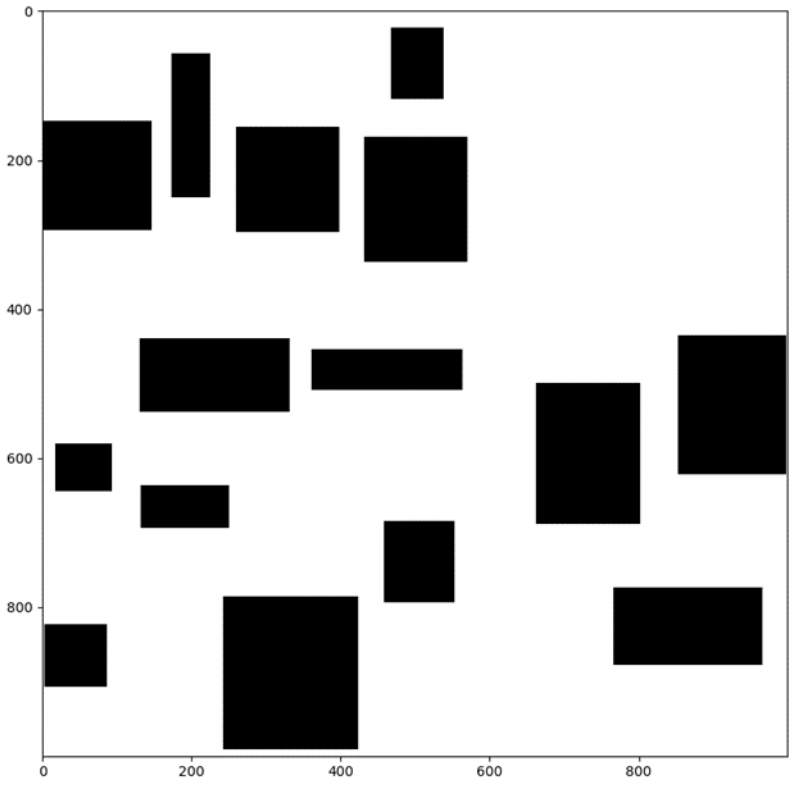
5.3 Πειραματικά Αποτελέσματα

Σε αυτή την ενότητα θα γίνει η παρουσίαση των πειραματικών αποτελεσμάτων. Μέσα από τη διεξαγωγή πειραμάτων σε διαφορετικά σενάρια, μπορούμε να αναδείξουμε τα δυνατά και αδύνατα σημεία κάθε αλγορίθμου, να συγκρίνουμε τις επιδόσεις τους και να εξάγουμε χρήσιμα συμπεράσματα. Τα αποτελέσματα αυτά θα μας βοηθήσουν να κατανοήσουμε καλύτερα την πρακτική εφαρμογή των αλγορίθμων και να προτείνουμε βελτιώσεις για μελλοντικές εργασίες.

5.3.1 Σενάριο 1. Μικρή περιοχή. Λίγα, μικρά, αραιά εμπόδια

Στο σενάριο 1 επιδιώκουμε να δοκιμάσουμε τους αλγορίθμους σε ένα απλό περιβάλλον ώστε να φανούν τα διαφορά μονοπάτια τα οποία θα διαλέξουν και να συγκριθεί η ποιότητα των μονοπατιών αυτών.

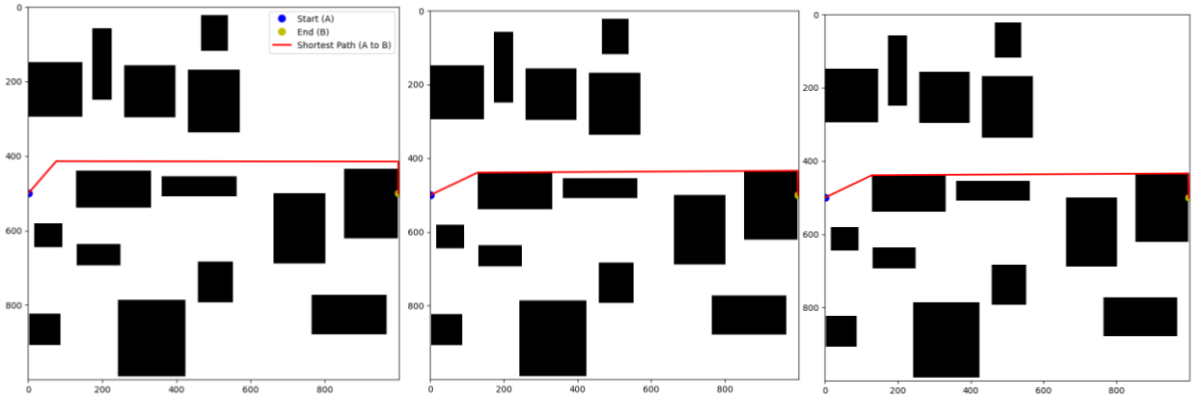
Στην εικόνα 13 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 1. Το περιβάλλον έχει διαστάσεις 1000 x 1000. Ο αριθμός εμποδίων είναι 15 με ελάχιστο μέγεθος 40 και μέγιστο 200.



Εικόνα 13

5.3.1.1 Αποτελέσματα

Στην εικόνα 14 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 13. Επίσης στους πίνακες 1 και 2 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 1 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 14: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ1.1	500 x 500	10	20	100
Σ1.2	500 x 500	10	15	250
Σ1.3	1000 x 1000	30	50	300
Σ1.4	1000 x 1000	40	10	60

***Πίνακας 1**: Παράμετροι σεναρίου 1*

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ1.1	Rectangles	497.0	0.13	0
Σ1.1	PRM	507.58	0.66	1.06
Σ1.1	APPATTrectangles	497.0	0.00	0
Σ1.2	Rectangles	536.74	0.06	0
Σ1.2	PRM	755.18	0.30	40.69
Σ1.2	APPATTrectangles	536.74	0.03	0
Σ1.3	Rectangles	1083.45	0.46	0
Σ1.3	PRM	2068.47	0.63	90.91
Σ1.3	APPATTrectangles	1083.45	0.13	0
Σ1.4	Rectangles	998.54	1.26	0
Σ1.4	PRM	1001.32	0.57	0.27
Σ1.4	APPATTrectangles	998.54	0.03	0

Πίνακας 2: *Αποτελέσματα*

Συμπεράσματα

Ο **αλγόριθμος Rectangles** εμφανίζει σταθερά καλή απόδοση όσον αφορά το μήκος της διαδρομής και την ακρίβεια. Ωστόσο, σε μεγαλύτερα περιβάλλοντα με αυξημένο αριθμό εμποδίων (όπως στο Σενάριο 1.4), παρατηρείται αύξηση του χρόνου εκτέλεσης, φτάνοντας το 1.26 δευτερόλεπτα. Αυτό δείχνει ότι, ενώ ο Rectangles εξασφαλίζει υψηλή ακρίβεια, παρουσιάζει μείωση της ταχύτητάς του όταν αυξάνεται η πολυπλοκότητα του περιβάλλοντος.

Ο **αλγόριθμος PRM** παρουσιάζει σημαντικές αποκλίσεις όσον αφορά την ακρίβεια, ιδιαίτερα σε περιβάλλοντα με μεγαλύτερο αριθμό εμποδίων ή μεγαλύτερο εύρος διαστάσεων, όπως φαίνεται στα Σενάρια 1.2 και 1.3. Στο Σενάριο 1.2, το σχετικό σφάλμα φτάνει το 40.69%, ενώ στο Σενάριο 1.3 η απόκλιση αυξάνεται δραματικά στο 90.91%. Ο PRM έχει τη δυνατότητα να υπολογίσει διαδρομές σε σχετικά μικρό χρονικό διάστημα, όπως παρατηρείται στο Σενάριο 1.4, όπου ο χρόνος εκτέλεσης ήταν 0.57 δευτερόλεπτα. Παρ' όλα αυτά, το υψηλό σχετικό σφάλμα τον καθιστά λιγότερο αξιόπιστο για εφαρμογές όπου απαιτείται μεγάλη ακρίβεια στη διαδρομή.

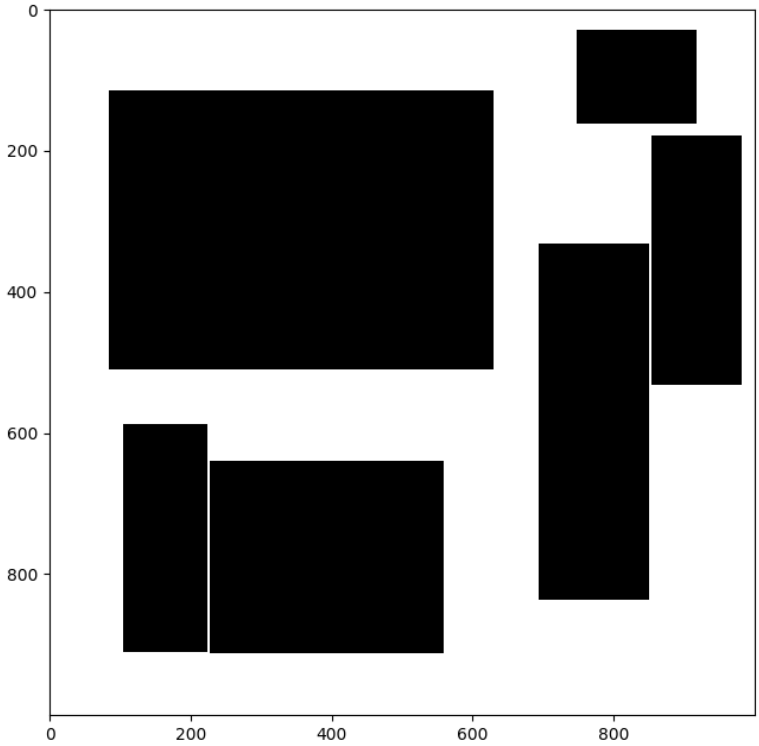
Ο **αλγόριθμος APPATTrectangles** εμφανίζει εξαιρετική απόδοση σε όλα τα σενάρια, με το μήκος της διαδρομής να ταυτίζεται με εκείνο του Rectangles και με μηδενικό σχετικό σφάλμα σε κάθε περίπτωση. Ταυτόχρονα, ο APPATTrectangles παρουσιάζει τον μικρότερο χρόνο εκτέλεσης σε όλα τα σενάρια, με τον χρόνο εκτέλεσης να είναι ιδιαίτερα χαμηλός, όπως στο Σενάριο 1.2 (0.03 δευτερόλεπτα) και στο Σενάριο 1.4 (0.03 δευτερόλεπτα). Αυτό δείχνει ότι ο APPATTrectangles επιτυγχάνει υψηλή ακρίβεια και απόδοση, ακόμα και σε περιβάλλοντα με μεγαλύτερο αριθμό εμποδίων.

Συνοψίζοντας, ο αλγόριθμος **Rectangles** προσφέρει σταθερή και ακριβή πλοήγηση, αλλά με μεγαλύτερο χρόνο εκτέλεσης σε πιο πολύπλοκα περιβάλλοντα. Ο **PRM** εμφανίζει γρηγορότερη απόδοση, αλλά με μεγάλες αποκλίσεις στο μήκος της διαδρομής, γεγονός που τον καθιστά λιγότερο κατάλληλο για εφαρμογές που απαιτούν ακρίβεια. Ο **APPATTrectangles** αποδεικνύεται ως η καλύτερη επιλογή, συνδυάζοντας ταχύτητα και ακρίβεια, γεγονός που τον καθιστά κατάλληλο για περιβάλλοντα με αυξημένη πολυπλοκότητα και εμπόδια.

5.3.2 Σενάριο 2. Μικρή περιοχή. Λίγα, μεγάλα, πυκνά εμπόδια

Στο σενάριο 2 επιδιώκουμε να δοκιμάσουμε τους αλγορίθμους σε ένα απλό περιβάλλον ώστε να φανεί η διαφορά στην συμπεριφορά των αλγορίθμων όταν αντιμετωπίζουν πυκνά εμπόδια.

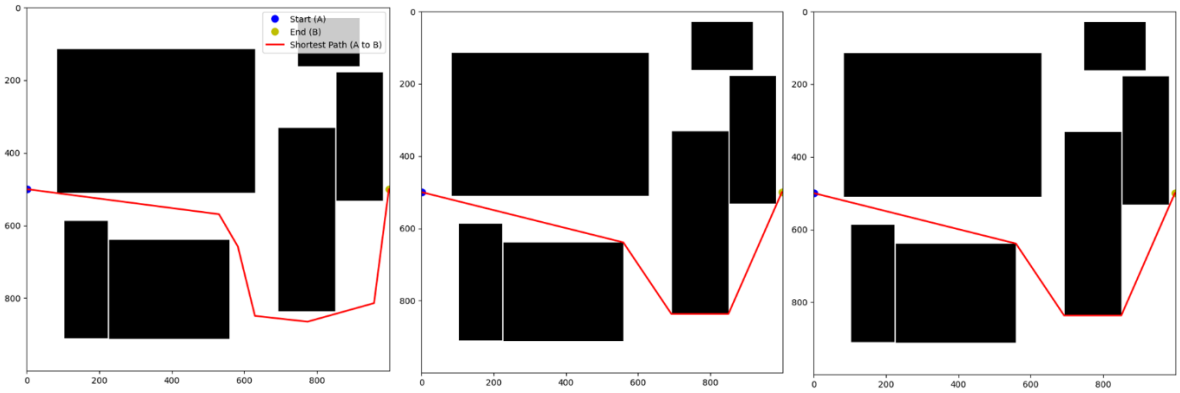
Στην εικόνα 15 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 2. Το περιβάλλον έχει διαστάσεις 1000 x 1000. Ο αριθμός εμποδίων είναι 6 με ελάχιστο μέγεθος 100 και μέγιστο 200.



Εικόνα 15

5.3.2.1 Αποτελέσματα

Στην εικόνα 16 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 15. Επίσης στους πίνακες 3 και 4 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 2 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 16: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ2.1	500 x 500	6	100	300

Σ2.2	500 x 500	8	50	300
Σ2.3	1000 x 1000	5	200	700

Πίνακας 3: Παράμετροι σεναρίου 2

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ2.1	Rectangles	572.30	0.02	0
Σ2.1	PRM	846.86	0.28	47.97
Σ2.1	APPATTrectangles	572.30	0.02	0
Σ2.2	Rectangles	656.96	0.04	0
Σ2.2	PRM	747.016	0.88	13.70
Σ2.2	APPATTrectangles	656.96	0.07	0
Σ2.3	Rectangles	999.02	0.02	0
Σ2.3	PRM	1002.87	0.45	0.38
Σ2.3	APPATTrectangles	999.02	0.005	0

Πίνακας 4: Αποτελέσματα

Συμπεράσματα

Ο **αλγόριθμος Rectangles** συνεχίζει να εμφανίζει εξαιρετική απόδοση, προσφέροντας ακριβείς διαδρομές με την συντομότερη διαδρομή σε όλα τα σενάρια. Ο χρόνος εκτέλεσης παραμένει εξαιρετικά χαμηλός, όπως στο Σενάριο 2.1 με μόλις 0.02 δευτερόλεπτα και στο Σενάριο 2.3 με τον ίδιο χρόνο. Αυτό δείχνει ότι ο Rectangles μπορεί να διατηρήσει την απόδοσή του ακόμα και όταν αυξάνεται η πολυπλοκότητα του περιβάλλοντος και το πλήθος των εμποδίων. Ο αλγόριθμος παραμένει εξαιρετικά αποδοτικός σε όρους ταχύτητας και ακρίβειας, ανεξάρτητα από το πλήθος των εμποδίων.

Ο **αλγόριθμος PRM** παρουσιάζει σημαντικές αποκλίσεις όσον αφορά την ακρίβεια, ιδιαίτερα όταν ο αριθμός των εμποδίων αυξάνεται. Στο Σενάριο 2.1, το σχετικό σφάλμα φτάνει το 47.97%, δείχνοντας μεγάλη απόκλιση στο μήκος της διαδρομής σε σχέση με τον στόχο. Στο Σενάριο 2.2, το σφάλμα μειώνεται στο 13.70%, αλλά εξακολουθεί να είναι αρκετά υψηλό. Ωστόσο, στο Σενάριο 2.3, ο αλγόριθμος PRM πλησιάζει πολύ την απόδοση του Rectangles, με μόλις 0.38% σφάλμα και διαδρομή μήκους 1002.87 μονάδων σε σχέση με τη διαδρομή των 999.02 μονάδων που βρήκε ο Rectangles. Παρ' όλα αυτά, ο χρόνος εκτέλεσης του PRM είναι αισθητά μεγαλύτερος, όπως φαίνεται στο Σενάριο 2.2, όπου φτάνει τα 0.88 δευτερόλεπτα, συγκριτικά με τον Rectangles που χρειάζεται μόλις 0.04 δευτερόλεπτα.

Ο **αλγόριθμος APPATTrectangles** αποδεικνύεται για άλλη μια φορά ως ο πιο αποδοτικός και ακριβής αλγόριθμος. Όπως και ο Rectangles, εμφανίζει μηδενικό σχετικό σφάλμα σε όλα τα σενάρια και βρίσκει διαδρομές ακριβώς ίδιου μήκους με αυτές του Rectangles. Το σημαντικότερο πλεονέκτημά του, όμως, είναι ο χρόνος εκτέλεσης, ο οποίος παραμένει εξαιρετικά χαμηλός. Στο Σενάριο 2.3, ο χρόνος εκτέλεσης του APPATTrectangles είναι μόλις 0.005 δευτερόλεπτα, ο μικρότερος από όλους τους αλγορίθμους. Αυτό δείχνει ότι ο

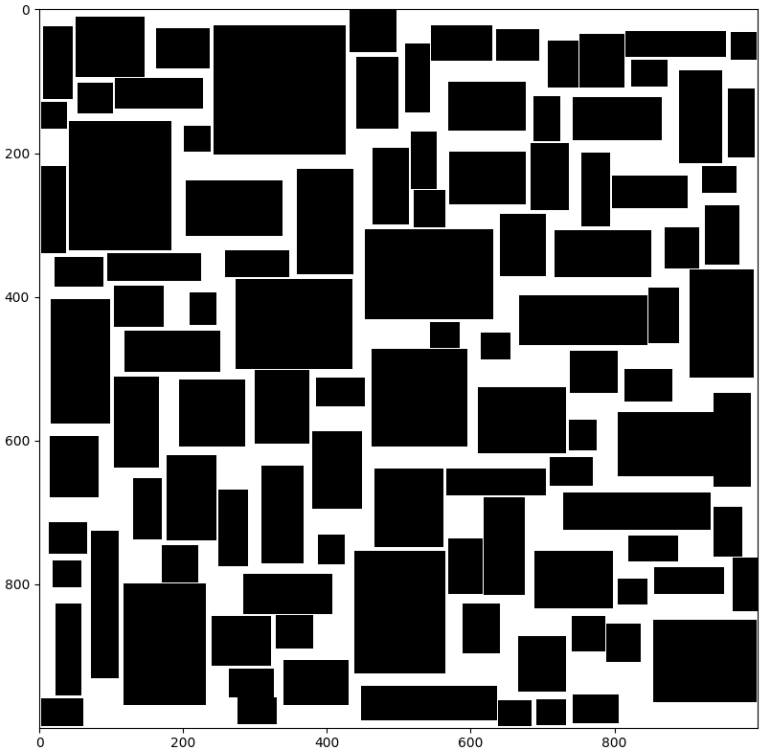
APPATTrectangles δεν είναι μόνο ακριβής, αλλά και ιδιαίτερα γρήγορος ακόμα και σε περιβάλλοντα με πυκνά εμπόδια.

Συνοψίζοντας, ο **Rectangles** προσφέρει εξαιρετική ακρίβεια και ταχύτητα, αλλά ο **APPATTrectangles** υπερτερεί ελαφρώς στον χρόνο εκτέλεσης, καθιστώντας τον την καλύτερη επιλογή σε περιβάλλοντα με πυκνά εμπόδια. Ο **PRM** παραμένει αξιόπιστος σε λιγότερο πυκνά περιβάλλοντα, αλλά παρουσιάζει μεγαλύτερη απόκλιση στην ακρίβεια και είναι πιο αργός, ιδιαίτερα όταν αυξάνεται η πολυπλοκότητα και η πυκνότητα των εμποδίων.

5.3.3 Σεναρίο 3. Μικρή περιοχή. Πολλά, μικρά, πυκνά εμπόδια

Στο σενάριο 3 επιδιώκουμε να δοκιμάσουμε τους αλγορίθμους σε ένα περιβάλλον με πολλά εμπόδια και στενούς χώρους. Έτσι θα φανεί η διαφορά των αλγορίθμων τόσο στο χρόνο εκτέλεσης, αφού καλούνται να δημιουργήσουν πιο πολύπλοκα μονοπάτια, όσο και στην ποιότητα του μονοπατιού που θα δημιουργήσουν.

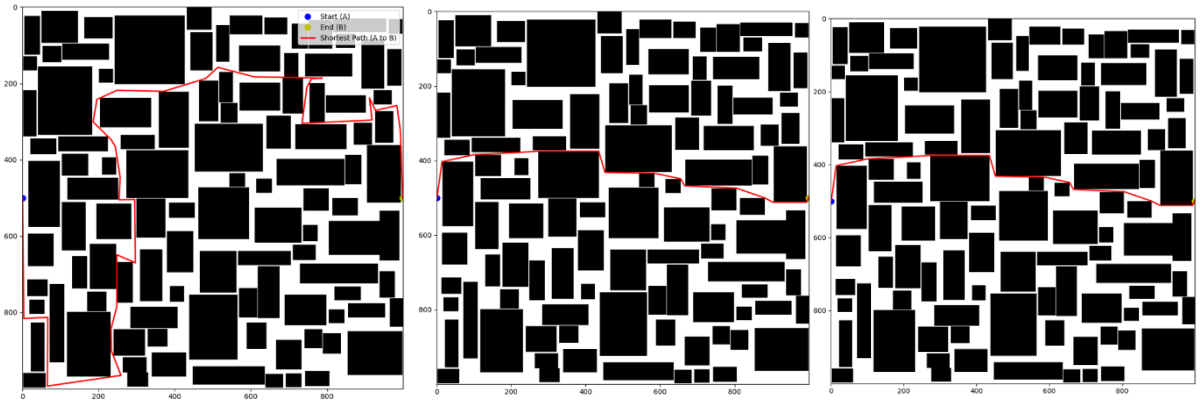
Στην εικόνα 17 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 3. Το περιβάλλον έχει διαστάσεις 1000 x 1000. Ο αριθμός εμποδίων είναι 106 με ελάχιστο μέγεθος 30 και μέγιστο 200.



Εικόνα 17

5.3.3.1 Αποτελέσματα

Στην εικόνα 18 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 17. Επίσης στους πίνακες 5 και 6 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 3 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 18: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ3.1	500 x 500	100	10	50
Σ3.2	500 x 500	152	10	50
Σ3.3	1000 x 1000	272	10	100
Σ3.4	1000 x 1000	1000	10	20

***Πίνακας 5**: Παράμετροι σεναρίου 3*

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ3.1	Rectangles	500.36	4.71	0
Σ3.1	PRM	680.90	0.67	36.08
Σ3.1	APPATTrectangles	500.36	0.32	0
Σ3.2	Rectangles	534.44	9.17	0
Σ3.2	PRM	1045.59	0.93	95.64
Σ3.2	APPATTrectangles	536.70	1.84	0.42
Σ3.3	Rectangles	1045.27	28.24	0
Σ3.3	PRM	1678.57	2.77	60.58
Σ3.3	APPATTrectangles	1045.27	2.26	0
Σ3.4	Rectangles	1022.59	356.13	0
Σ3.4	PRM	2225.44	3.37	117.62
Σ3.4	APPATTrectangles	1022.59	22.94	0

***Πίνακας 6:** Αποτελέσματα*

Συμπεράσματα

Ο **αλγόριθμος Rectangles** συνεχίζει να εμφανίζει εξαιρετική ακρίβεια. Ωστόσο, παρατηρούμε σημαντική αύξηση στον χρόνο εκτέλεσης, ιδιαίτερα στα πιο πολύπλοκα περιβάλλοντα με πολλά εμπόδια. Στο Σενάριο 3.4, όπου υπάρχουν 1000 εμπόδια, ο χρόνος εκτέλεσης φτάνει τα 356.13 δευτερόλεπτα, κάτι που τον καθιστά εξαιρετικά αργό για πρακτικές εφαρμογές σε τέτοιες συνθήκες. Αυτή η αύξηση δείχνει ότι, ενώ ο Rectangles προσφέρει υψηλή ακρίβεια, το υπολογιστικό του κόστος αυξάνεται δραματικά σε περιβάλλοντα με στενούς χώρους και πολλά εμπόδια.

Ο **αλγόριθμος PRM** εμφανίζει μεγάλη απόκλιση στις διαδρομές που υπολογίζει, με το σχετικό σφάλμα να είναι ιδιαίτερα υψηλό. Στο Σενάριο 3.1, το σφάλμα φτάνει το 36.08%, ενώ στο Σενάριο 3.4 αυξάνεται ακόμα περισσότερο, φτάνοντας το 117.62%, πράγμα που υποδηλώνει ότι ο PRM δεν καταφέρνει να υπολογίσει βέλτιστες διαδρομές σε πυκνά και πολύπλοκα περιβάλλοντα. Ωστόσο, ο PRM εμφανίζει χαμηλότερους χρόνους εκτέλεσης σε σχέση με τον Rectangles, ειδικά στα πιο σύνθετα σενάρια. Για παράδειγμα, στο Σενάριο 3.4, ο χρόνος εκτέλεσης ήταν 3.37 δευτερόλεπτα, σημαντικά χαμηλότερος από τον Rectangles, αλλά αυτό έρχεται με το τίμημα της χαμηλής ποιότητας διαδρομών.

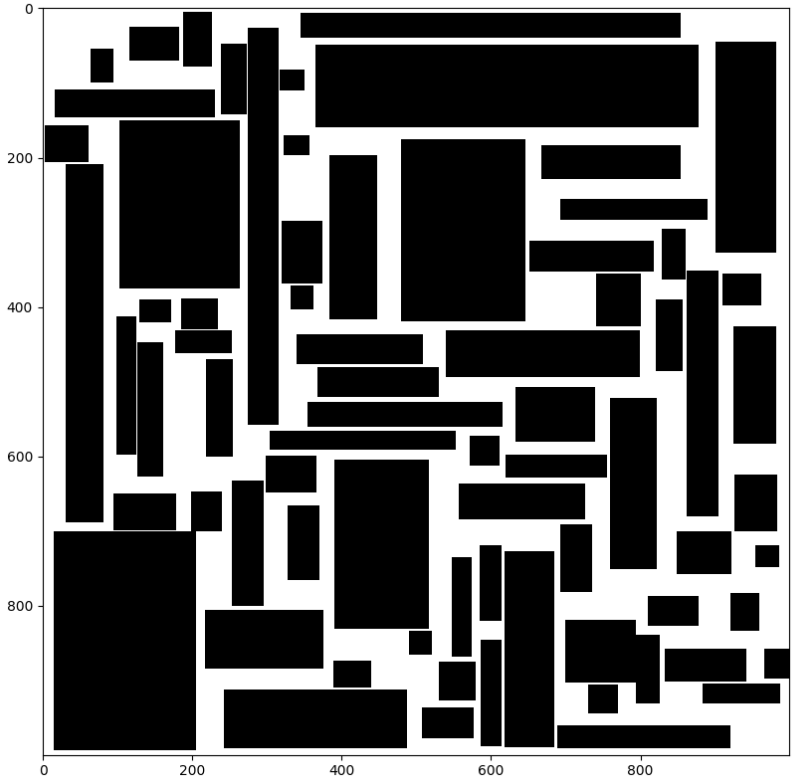
Ο **αλγόριθμος APPATTrectangles** αποδεικνύεται ξανά ως ο πιο αποδοτικός, συνδυάζοντας ακρίβεια και ταχύτητα. Με μηδενικό σχετικό σφάλμα σε όλα τα σενάρια, εξασφαλίζει τη βέλτιστη διαδρομή, ακόμα και σε περιβάλλοντα με πολλά εμπόδια. Σημαντικό πλεονέκτημα του APPATTrectangles είναι ο χαμηλός χρόνος εκτέλεσης, ο οποίος παραμένει εξαιρετικά μικρός, ακόμα και στα πιο πολύπλοκα σενάρια. Στο Σενάριο 3.4, ο χρόνος εκτέλεσης ήταν 22.94 δευτερόλεπτα, πολύ μικρότερος από τον αντίστοιχο του Rectangles (356.13 δευτερόλεπτα), καθιστώντας τον APPATTrectangles ιδιαίτερα αποτελεσματικό σε πολύπλοκα περιβάλλοντα.

Συνοψίζοντας, ο **Rectangles** παρέχει υψηλή ακρίβεια αλλά με σημαντικό κόστος στον χρόνο εκτέλεσης σε περιβάλλοντα με πολλά εμπόδια και στενούς χώρους. Ο **PRM** έχει μικρότερους χρόνους εκτέλεσης, αλλά οι διαδρομές που υπολογίζει είναι συχνά λιγότερο βέλτιστες, με υψηλό σχετικό σφάλμα. Ο **APPATTrectangles** υπερέχει και στις δύο κατηγορίες, προσφέροντας βέλτιστες διαδρομές με χαμηλό χρόνο εκτέλεσης, καθιστώντας τον τον καταλληλότερο αλγόριθμο για σύνθετα και πυκνά περιβάλλοντα.

5.3.4 Σενάριο 4. Μικρή περιοχή. Πολλά, διάφορων μεγεθών, πυκνά εμπόδια

Στο σενάριο 4 επιδιώκουμε να δοκιμάσουμε τους αλγορίθμους σε ένα περιβάλλον με πολλά, διάφορων μεγεθών εμπόδια. Αυτό θα αναγκάσει τους αλγορίθμους να κάνουν μεγάλες λοξοδρομήσεις στην πορεία του UAV και έτσι θα δημιουργηθούν πιο πολύπλοκα μονοπάτια.

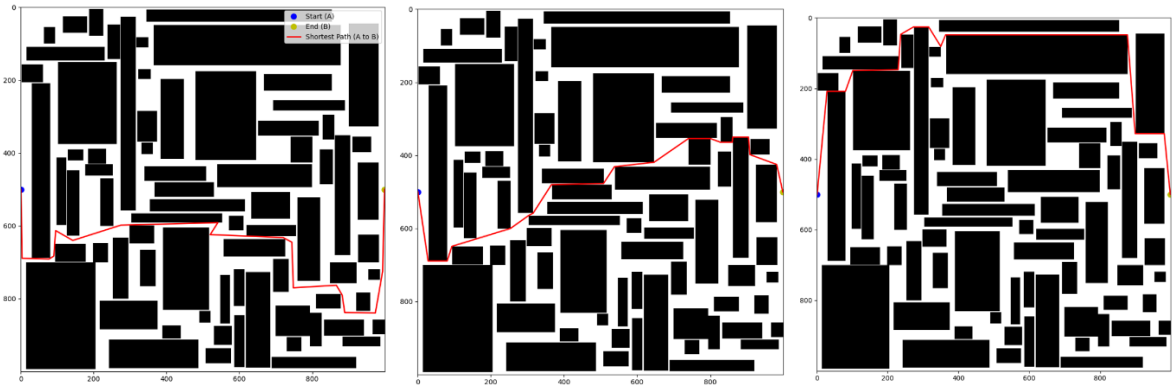
Στην εικόνα 19 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 4. Το περιβάλλον έχει διαστάσεις 1000 x 1000. Ο αριθμός εμποδίων είναι 73 με ελάχιστο μέγεθος 20 και μέγιστο 600.



Εικόνα 19

5.3.4.1 Αποτελέσματα

Στην εικόνα 20 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 19. Επίσης στους πίνακες 7 και 8 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 4 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 20: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ4.1	500 x 500	27	10	400
Σ4.2	1000 x 1000	47	10	700

***Πίνακας 7**: Παράμετροι σεναρίου 4*

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ4.1	Rectangles	850.57	0.36	0
Σ4.1	PRM	1055.32	0.41	24.07
Σ4.1	APPATTrectangles	850.57	0.18	0
Σ4.2	Rectangles	1197.45	1656.07	0.59
Σ4.2	PRM	1656.07	0.59	38.29
Σ4.2	APPATTrectangles	1205.08	0.30	0.63

***Πίνακας 8**: Αποτελέσματα*

Συμπεράσματα

Ο **αλγόριθμος Rectangles** συνεχίζει να προσφέρει υψηλή ακρίβεια, καθώς στο Σενάριο 4.1 το μήκος της διαδρομής που υπολογίζει είναι 850.57 μονάδες, με μηδενικό σχετικό σφάλμα. Ωστόσο, στο Σενάριο 4.2, όπου αυξάνεται το μέγεθος της περιοχής και ο αριθμός των εμποδίων, παρατηρείται μια ελαφριά αύξηση του μήκους της διαδρομής, η οποία ανέρχεται σε 1197.45 μονάδες. Ο χρόνος εκτέλεσης, όμως, αυξάνεται δραματικά στο Σενάριο 4.2, φτάνοντας τα 1656.07 δευτερόλεπτα. Αυτό καταδεικνύει την αδυναμία του Rectangles να ανταπεξέλθει αποτελεσματικά σε μεγάλα περιβάλλοντα με περισσότερα και μεγαλύτερα εμπόδια, παρά το γεγονός ότι εξακολουθεί να εξασφαλίζει ακρίβεια.

Ο **αλγόριθμος PRM**, από την άλλη, συνεχίζει να δείχνει μεγάλες αποκλίσεις όσον αφορά την ακρίβεια των διαδρομών που υπολογίζει. Στο Σενάριο 4.1, η απόκλιση από το βέλτιστο μονοπάτι είναι 24.07%, ενώ στο Σενάριο 4.2 το σχετικό σφάλμα αυξάνεται στο 38.29%. Αυτό δείχνει ότι ο PRM δυσκολεύεται να βρει βέλτιστες διαδρομές σε περιβάλλοντα με μεγάλα και διάσπαρτα εμπόδια. Ωστόσο, ο PRM καταφέρνει να διατηρήσει έναν σχετικά χαμηλό χρόνο εκτέλεσης σε σύγκριση με τον Rectangles, όπως παρατηρείται στο Σενάριο 4.2, όπου ο χρόνος εκτέλεσης ήταν 0.59 δευτερόλεπτα.

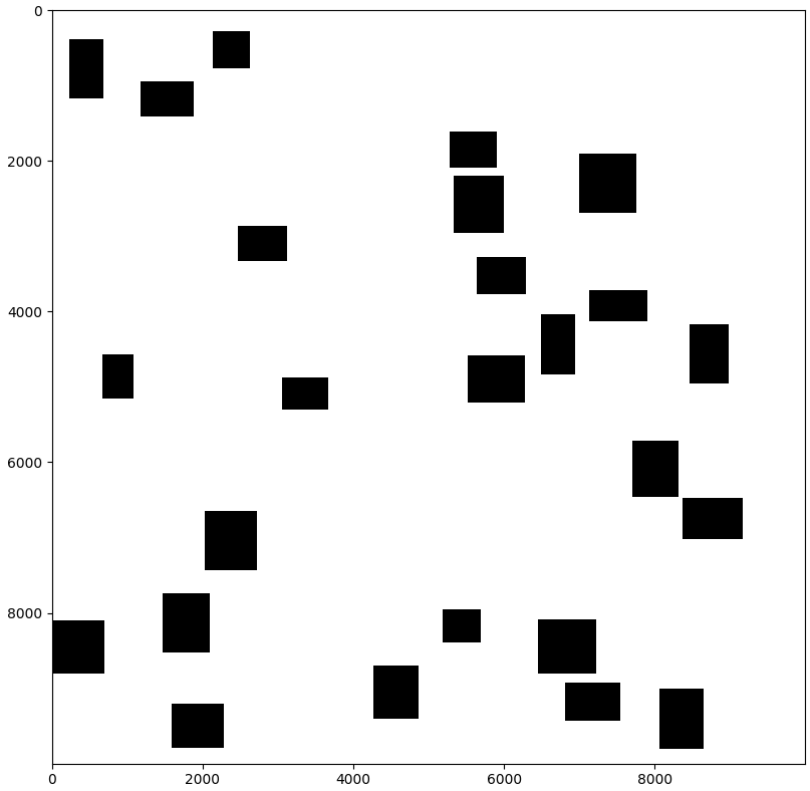
Ο **αλγόριθμος APPATTrectangles** επιδεικνύει και πάλι την καλύτερη συνολική απόδοση. Στο Σενάριο 4.1, υπολογίζει το ίδιο μήκος διαδρομής με τον Rectangles (850.57 μονάδες) με μηδενικό σφάλμα, αλλά με πολύ μικρότερο χρόνο εκτέλεσης (0.18 δευτερόλεπτα). Στο Σενάριο 4.2, ενώ η διαδρομή που υπολογίζει είναι ελαφρώς μεγαλύτερη από αυτήν του Rectangles (1205.08 μονάδες), το σχετικό σφάλμα παραμένει εξαιρετικά χαμηλό (0.63%). Ο χρόνος εκτέλεσης του APPATTrectangles παραμένει επίσης πολύ μικρός, μόλις 0.30 δευτερόλεπτα, γεγονός που τον καθιστά την πιο αποδοτική επιλογή για εφαρμογές όπου η ταχύτητα είναι κρίσιμη.

Συνοψίζοντας, ο **Rectangles** παραμένει ακριβής αλλά παρουσιάζει σημαντική αύξηση του χρόνου εκτέλεσης σε μεγαλύτερα και πιο πολύπλοκα περιβάλλοντα. Ο **PRM** προσφέρει χαμηλότερους χρόνους εκτέλεσης, αλλά με σημαντικές αποκλίσεις στην ακρίβεια των διαδρομών. Ο **APPATTrectangles** συνδυάζει την ακρίβεια με πολύ χαμηλό χρόνο εκτέλεσης, καθιστώντας τον τον πιο αποτελεσματικό αλγόριθμο για περιβάλλοντα με μεγάλα και διάσπαρτα εμπόδια.

5.3.5 Σενάριο 5. Μεγάλη περιοχή. Λίγα, μικρά, αραιά εμπόδια

Σε αυτό το σενάριο εξετάζουμε πως η αύξηση του μεγέθους της περιοχής επηρεάζει την αποδοτικότητα των αλγορίθμων σε ένα απλό περιβάλλον με λιγοστά εμπόδια.

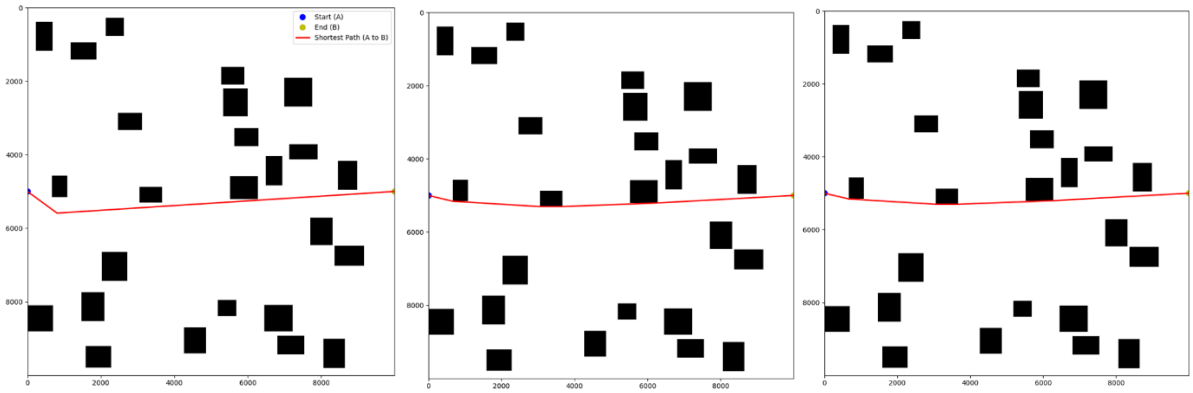
Στην εικόνα 21 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 5. Το περιβάλλον έχει διαστάσεις 10000 x 10000. Ο αριθμός εμποδίων είναι 25 με ελάχιστο μέγεθος 400 και μέγιστο 800.



Εικόνα 21

5.3.5.1 **Αποτελέσματα**

Στην εικόνα 21 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 22. Επίσης στους πίνακες 9 και 10 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 5 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 22: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ5.1	10000 x 10000	15	200	500
Σ5.2	10000 x 10000	50	500	700

Πίνακας 9: *Παράμετροι σεναρίου 5*

Σενάριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ5.1	Rectangles	9997.0	0.23	0
Σ5.1	PRM	10138.50	35.82	1.41
Σ5.1	APPATTrectangles	9997.0	0.00	0
Σ5.2	Rectangles	10019.99	1.75	0
Σ5.2	PRM	13410.641	30.44	33.83
Σ5.2	APPATTrectangles	10019.99	0.11	0

Πίνακας 10: *Αποτελέσματα*

Συμπεράσματα

Ο αλγόριθμος Rectangles παραμένει εξαιρετικά ακριβής και αποδοτικός όσον αφορά το μήκος της διαδρομής και τον χρόνο εκτέλεσης. Στο Σενάριο 5.1, με λιγότερα εμπόδια και μεγάλη περιοχή, το μήκος της διαδρομής που υπολογίζεται είναι 9997.0 μονάδες, το οποίο είναι η βέλτιστη διαδρομή, και ο χρόνος εκτέλεσης είναι εξαιρετικά χαμηλός στα 0.23 δευτερόλεπτα. Ωστόσο, καθώς αυξάνεται ο αριθμός των εμποδίων (Σενάριο 5.2), παρατηρείται αύξηση του χρόνου εκτέλεσης στα 1.75 δευτερόλεπτα, αλλά η ακρίβεια παραμένει άριστη, με μήκος διαδρομής 10019.99 μονάδες. Συνεπώς το μέγεθος της περιοχής δεν επηρεάζει τον χρόνο εκτέλεσης.

Ο **αλγόριθμος PRM** συνεχίζει να εμφανίζει αποκλίσεις στην ακρίβεια, ειδικά στο Σενάριο 5.2, όπου το σχετικό σφάλμα ανέρχεται στο 33.83%, κάτι που υποδεικνύει ότι η διαδρομή που υπολογίζει είναι σημαντικά μεγαλύτερη από την βέλτιστη. Επίσης η αύξηση του μεγέθους

περιοχής επηρεάζει δραματικά την αποδοτικότητα του αλγορίθμου όσο αφορά τον χρόνο εκτέλεσης επειδή χρειάζεται περισσότερο χρόνο για την αποθήκευση της ελεύθερης περιοχής.

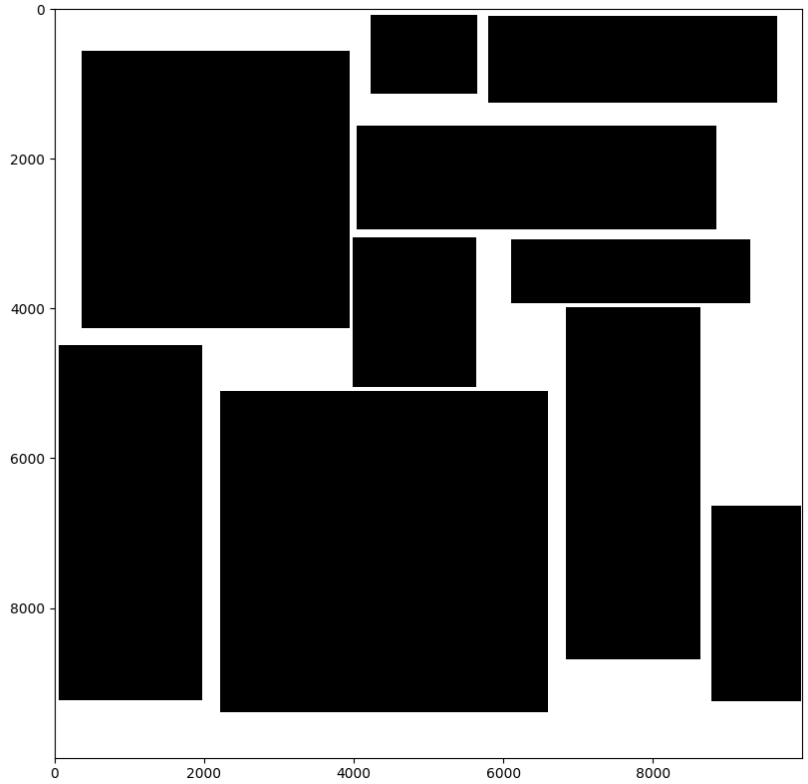
Ο **αλγόριθμος APPATTrectangles** για άλλη μια φορά αποδεικνύεται η πιο αποτελεσματική επιλογή. Στο Σενάριο 5.1, ο χρόνος εκτέλεσης είναι 0.00 δευτερόλεπτα, ενώ το μήκος της διαδρομής ταυτίζεται με αυτό του Rectangles (9997.0 μονάδες), επιδεικνύοντας μηδενικό σφάλμα. Ακόμα και όταν αυξάνεται ο αριθμός των εμποδίων στο Σενάριο 5.2, ο APPATTrectangles διατηρεί την ακρίβεια του με μήκος διαδρομής 10019.99 μονάδες και 0 σφάλμα, ενώ ο χρόνος εκτέλεσης παραμένει εξαιρετικά χαμηλός, μόλις 0.11 δευτερόλεπτα.

Συνοψίζοντας, ο **Rectangles** προσφέρει ακριβή αποτελέσματα με σχετικά χαμηλό χρόνο εκτέλεσης. Ο **PRM** παρουσιάζει μεγάλες αποκλίσεις στην ακρίβεια των διαδρομών, ιδιαίτερα σε σενάρια με περισσότερα εμπόδια, και χρειάζεται πολύ περισσότερο χρόνο για να υπολογίσει μια διαδρομή. Ο **APPATTrectangles** είναι ο πιο αποδοτικός αλγόριθμος, προσφέροντας βέλτιστες διαδρομές με ελάχιστο χρόνο εκτέλεσης, ακόμα και σε μεγάλα περιβάλλοντα με αυξημένη πολυπλοκότητα.

5.3.6 Σενάριο 6. Μεγάλη περιοχή. Λίγα, μεγάλα, πυκνά εμπόδια

Σε αυτό το σενάριο εξετάζουμε πως η αύξηση του μεγέθους της περιοχής επηρεάζει την αποδοτικότητα των αλγορίθμων σε μια μεγάλη περιοχή με πυκνά εμπόδια.

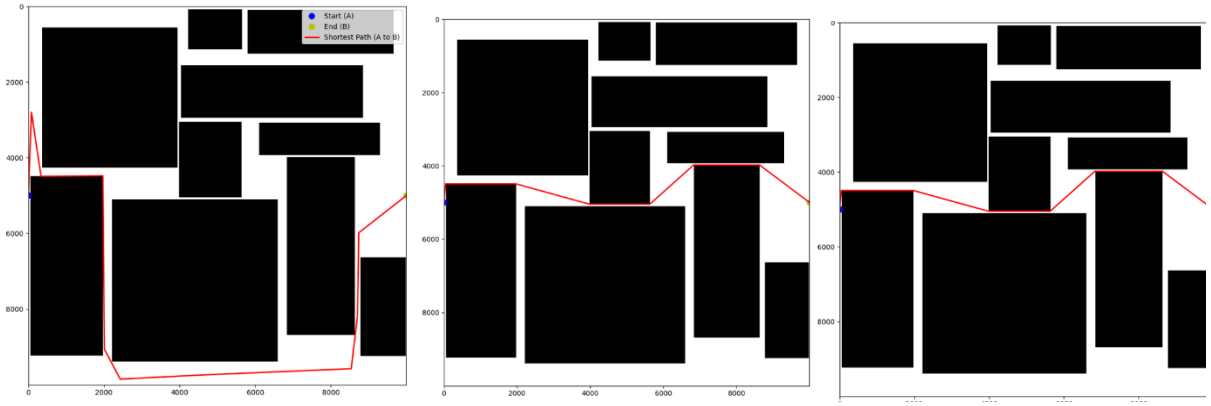
Παρακάτω φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 6. Το περιβάλλον έχει διαστάσεις 10000 x 10000. Ο αριθμός εμποδίων είναι 10 με ελάχιστο μέγεθος 800 και μέγιστο 5000.



Εικόνα 23

5.3.6.1 Αποτελέσματα

Στην εικόνα 24 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 23. Επίσης στους πίνακες 11 και 12 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 6 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.



Εικόνα 24: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ6.1	10000 x 10000	13	1000	5000
Σ6.2	10000 x 10000	15	800	5000

***Πίνακας 11**: Παράμετροι σεναρίου 6*

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ6.1	Rectangles	11939.78	0.10	0
Σ6.1	PRM	18722.25	24.13	56.80
Σ6.1	APPATTrectangles	11939.78	0.08	0
Σ6.2	Rectangles	11219.41	0.13	0
Σ6.2	PRM	27530.33	24.50	145.38
Σ6.2	APPATTrectangles	11219.41	0.04	0

***Πίνακας 12**: Αποτελέσματα*

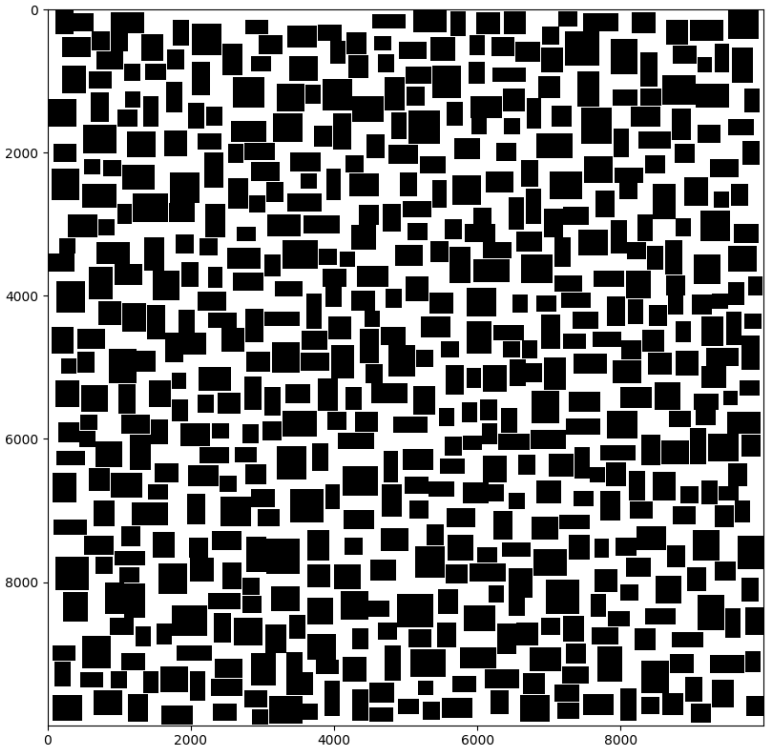
Συμπεράσματα

Ο **Rectangles** παραμένει ακριβής και αποδοτικός, με σχετικά χαμηλούς χρόνους εκτέλεσης, ακόμα και σε περιβάλλοντα με πυκνά εμπόδια. Ο **PRM**, αν και παρουσιάζει χαμηλότερους χρόνους εκτέλεσης από τον Rectangles σε προηγούμενα σενάρια, εδώ εμφανίζει μεγάλες αποκλίσεις στην ακρίβεια, γεγονός που τον καθιστά λιγότερο αξιόπιστο. Ο **APPATTrectangles** προσφέρει την καλύτερη συνολική απόδοση, συνδυάζοντας υψηλή

ακρίβεια με εξαιρετικά χαμηλό χρόνο εκτέλεσης, καθιστώντας τον ιδανικό για μεγάλες και πολύπλοκες περιοχές με πολλά εμπόδια.

5.3.7 Σενάριο 7. Μεγάλη περιοχή. Πολλά, μικρά, πυκνά εμπόδια

Σε αυτό το σενάριο επιδιώκουμε να ανακαλύψουμε τις αδυναμίες των αλγορίθμων σε ένα περιβάλλον με στενούς χώρους και ένα πάρα πολύ μεγάλο αριθμό εμποδίων. Έτσι θα φανεί η διαφορά τον αλγορίθμων ως προς τον χρόνο εκτέλεσης ανάλογα με τον αριθμό των εμποδίων. Στην εικόνα 25 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 7. Το περιβάλλον έχει διαστάσεις 10000 x 10000. Ο αριθμός εμποδίων είναι 549 με ελάχιστο μέγεθος 200 και μέγιστο 500.



Εικόνα 25

5.3.7.1 Αποτελέσματα

Στην εικόνα 26 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 25. Επίσης στους πίνακες 13 και 14 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 7 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.

Εικόνα 26: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ7.1	10000 x 10000	466	200	500
Σ1.2	10000 x 10000	1986	100	200

***Πίνακας 13**: Παράμετροι σεναρίου 7*

Σενάριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ7.1	Rectangles	10341.27	77.56	0
Σ7.1	PRM	21774.40	29.76	110.55
Σ7.1	APPATTrectangles	10341.27	3.85	0
Σ7.2	Rectangles	10127.30	1369.00	0
Σ7.2	PRM	23114.55	58.06	128.24
Σ7.2	APPATTrectangles	10127.30	60.46	0

***Πίνακας 14:** Αποτελέσματα*

Συμπεράσματα

Ο **αλγόριθμος Rectangles** συνεχίζει να παρέχει ακριβείς διαδρομές, με μηδενικό σχετικό σφάλμα και στα δύο σενάρια. Ωστόσο, παρατηρούμε μια δραματική αύξηση στον χρόνο εκτέλεσης καθώς αυξάνεται ο αριθμός των εμποδίων. Στο Σενάριο 7.1, με 466 εμπόδια, ο χρόνος εκτέλεσης ανέρχεται στα 77.56 δευτερόλεπτα, ενώ στο Σενάριο 7.2, με τα 1986 εμπόδια, ο χρόνος αυξάνεται κατακόρυφα στα 1369 δευτερόλεπτα. Αυτό υποδεικνύει την κύρια αδυναμία του Rectangles, δηλαδή τη δραματική επιβράδυνση σε περιβάλλοντα με πολύπλοκη γεωμετρία και πολλαπλά εμπόδια.

Ο **αλγόριθμος PRM** παρουσιάζει, όπως και στα προηγούμενα σενάρια, σημαντικές αποκλίσεις στην ακρίβεια των διαδρομών. Στο Σενάριο 7.1, το σχετικό σφάλμα ανέρχεται στο 110.55%, ενώ στο Σενάριο 7.2 αυξάνεται στο 128.24%, πράγμα που σημαίνει ότι οι διαδρομές που υπολογίζει είναι κατά πολύ μεγαλύτερες από τις βέλτιστες. Ωστόσο, ο χρόνος εκτέλεσης του

PRM είναι συγκριτικά μικρότερος από τον Rectangles, με 29.76 δευτερόλεπτα στο Σενάριο 7.1 και 58.06 δευτερόλεπτα στο Σενάριο 7.2. Αυτό δείχνει ότι ο PRM παραμένει γρήγορος στην εκτέλεση, αλλά εις βάρος της ακρίβειας.

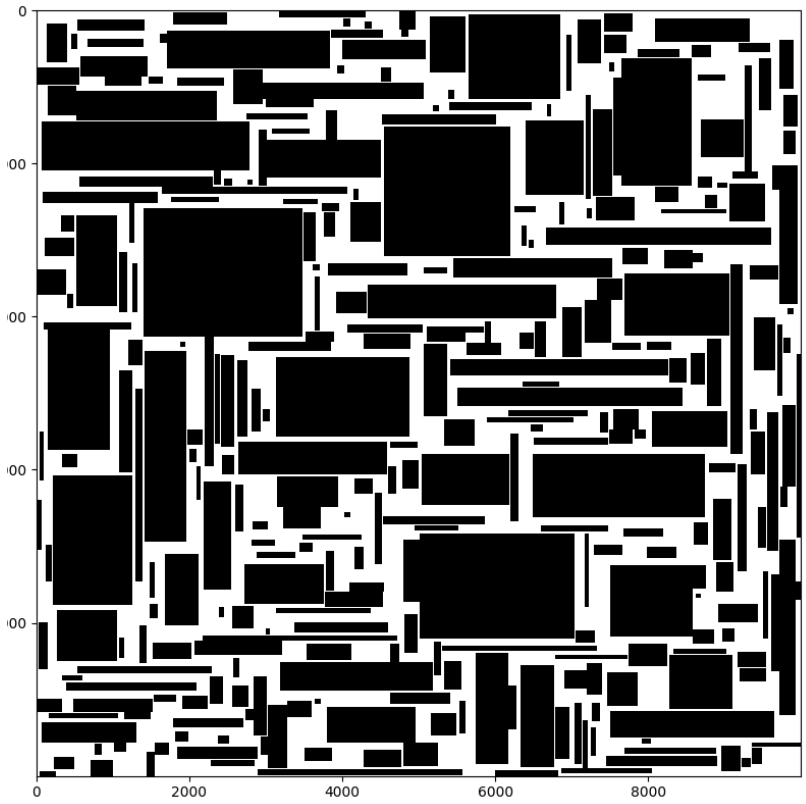
Ο **αλγόριθμος APPATTrectangles** αποδεικνύεται ξανά ο πιο αποδοτικός αλγόριθμος. Στο Σενάριο 7.1, υπολογίζει τη βέλτιστη διαδρομή μήκους 10341.27 μονάδες με μηδενικό σφάλμα σε μόλις 3.85 δευτερόλεπτα, προσφέροντας τον ταχύτερο χρόνο εκτέλεσης. Ακόμη και στο Σενάριο 7.2, με τον αυξημένο αριθμό εμποδίων, ο χρόνος εκτέλεσης του APPATTrectangles παραμένει εξαιρετικά χαμηλός στα 60.46 δευτερόλεπτα, ενώ υπολογίζει και πάλι τη βέλτιστη διαδρομή, αποδεικνύοντας ότι μπορεί να ανταποκριθεί σε πολύπλοκα περιβάλλοντα με πολλαπλά εμπόδια.

Συμπερασματικά, ο **Rectangles** προσφέρει ακρίβεια στις διαδρομές, αλλά υστερεί σημαντικά στον χρόνο εκτέλεσης σε πολύπλοκα περιβάλλοντα με πολλά εμπόδια. Ο **PRM** προσφέρει γρήγορη εκτέλεση, αλλά η ακρίβεια των διαδρομών είναι εξαιρετικά χαμηλή, ιδίως σε περιβάλλοντα με στενούς χώρους και πολλαπλά εμπόδια. Ο **APPATTrectangles** συνδυάζει την υψηλή ακρίβεια με εξαιρετικά χαμηλό χρόνο εκτέλεσης, καθιστώντας τον ιδανικό για σύνθετα σενάρια με στενούς χώρους και μεγάλο αριθμό εμποδίων.

5.3.8 Σενάριο 8. Μεγάλη περιοχή. Πολλά, διάφορων μεγεθών, πυκνά εμπόδια

Το σενάριο αυτό έχει σχεδιαστεί ώστε να δοκιμάσει τους αλγορίθμου σε περιβάλλοντα μέγιστης πολυπλοκότητας. Η μεγάλη περιοχή, ο μεγάλος αριθμός εμποδίων και η ποικιλία στα μεγέθη των εμποδίων θα εξετάσει τόσο την ποιότητα του μονοπατιού που σχεδιάζει ο κάθε αλγόριθμος όσο και τον χρόνο εκτέλεσης.

Στην εικονα 27 φαίνεται ένα παράδειγμα περιβάλλοντος το οποίο ανήκει στο σενάριο 8. Το περιβάλλον έχει διαστάσεις 10000 x 10000. Ο αριθμός εμποδίων είναι 315 με ελάχιστο μέγεθος 50 και μέγιστο 3000.



Εικόνα 27

5.3.8.1 Αποτελέσματα

Στην εικόνα 28 φαίνονται παραδείγματα εκτέλεσης των τριών αλγορίθμων στο περιβάλλον που φαίνεται στην εικόνα 27. Επίσης στους πίνακες 15 και 16 παρουσιάζονται οι παράμετροι διάφορων περιοχών που ανήκουν στο σενάριο 8 και τα αποτελέσματα των εκτελέσεων των τριών αλγορίθμων στις περιοχές αυτές.

Εικόνα 28: Αριστερή εικόνα PRM, μεσαία εικόνα Rectangles, δεξιά εικόνα APPATTrectangles

Σενάριο	Μέγεθος περιοχής	Αριθμός εμποδίων	Ελάχιστο μέγεθος εμποδίων	Μέγιστο μέγεθος εμποδίων
Σ8.1	10000 x 1000	204	100	2000
Σ8.2	10000 x 10000	336	10	2000

Σ8.3	15000 x 15000	376	50	4000
Σ8.4	15000 x 15000	275	10	6000

Πίνακας 15: *Παράμετροι σεναρίου 8*

Σεναριο	Αλγόριθμος	Μήκος Διαδρομής	Χρόνος Εκτέλεσης (seconds)	Σχετικό Σφάλμα (%)
Σ8.1	Rectangles	11324.34	14.77	0
Σ8.1	PRM	15966.10	44.63	40.98
Σ8.1	APPATTrectangles	11431.77	4.35	0.94
Σ8.2	Rectangles	10995.91	41.68	0
Σ8.2	PRM	30006.53	33.95	172.88
Σ8.2	APPATTrectangles	10995.91	8.64	0
Σ8.3	Rectangles	19840.05	51.65	0
Σ8.3	PRM	47861.53	79.24	141.23
Σ8.3	APPATTrectangles	19840.05	23.40	0
Σ8.4	Rectangles	21784.96	28.15	0
Σ8.4	PRM	42239.06	63.30	93.89
Σ8.4	APPATTrectangles	22113.25	8.21	1.50

Πίνακας 16: *Αποτελέσματα*

Συμπεράσματα

Ο **αλγόριθμος Rectangles** συνεχίζει να είναι εξαιρετικά ακριβής. Ειδικότερα, στο Σενάριο 8.1, το μήκος της διαδρομής που υπολογίζει είναι 11324.34 μονάδες με χρόνο εκτέλεσης 14.77 δευτερόλεπτα, ενώ στο Σενάριο 8.2 η διαδρομή είναι 10995.91 μονάδες με 41.68 δευτερόλεπτα. Στα πιο σύνθετα περιβάλλοντα, όπως στο Σενάριο 8.3, η διαδρομή αυξάνεται στις 19840.05 μονάδες, ενώ στο Σενάριο 8.4 η διαδρομή φτάνει τις 21784.96 μονάδες. Ωστόσο, αν και ο Rectangles διατηρεί την ακρίβειά του, ο χρόνος εκτέλεσης αυξάνεται σημαντικά στα πιο περίπλοκα περιβάλλοντα, με το μέγιστο χρόνο να φτάνει τα 51.65 δευτερόλεπτα στο Σενάριο 8.3.

Ο **αλγόριθμος PRM** εμφανίζει σημαντικά προβλήματα ακρίβειας. Σε όλα τα σενάρια, το σχετικό σφάλμα είναι υψηλό, δείχνοντας ότι οι διαδρομές που υπολογίζει είναι πολύ μεγαλύτερες από τις βέλτιστες. Στο Σενάριο 8.1, το σφάλμα ανέρχεται στο 40.98%, ενώ στο Σενάριο 8.2 το σφάλμα αυξάνεται δραματικά στο 172.88%, καθιστώντας τη διαδρομή που υπολογίζει κατά πολύ μεγαλύτερη από την βέλτιστη. Στα πιο περίπλοκα περιβάλλοντα, το σφάλμα παραμένει υψηλό, με 141.23% στο Σενάριο 8.3 και 93.89% στο Σενάριο 8.4. Παρότι ο PRM έχει σχετικά γρήγορο χρόνο εκτέλεσης, όπως στο Σενάριο 8.2 με 33.95 δευτερόλεπτα και στο Σενάριο 8.4 με 63.30 δευτερόλεπτα, η χαμηλή ακρίβεια των διαδρομών τον καθιστά λιγότερο αξιόπιστο σε περιβάλλοντα μεγάλης πολυπλοκότητας.

Ο **αλγόριθμος APPATTrectangles** δείχνει για άλλη μια φορά την υψηλή του απόδοση σε περιβάλλοντα με μεγάλη πολυπλοκότητα. Στο Σενάριο 8.1, υπολογίζει μια διαδρομή μήκους 11431.77 μονάδων με σχετικό σφάλμα 0.94% και χρόνο εκτέλεσης 4.35 δευτερόλεπτα, πολύ ταχύτερα από τον Rectangles. Στο Σενάριο 8.2, η διαδρομή που υπολογίζει ταυτίζεται με αυτήν του Rectangles (10995.91 μονάδες) με μηδενικό σφάλμα και χρόνο εκτέλεσης μόλις 8.64 δευτερόλεπτα. Ακόμα και στα πιο περίπλοκα περιβάλλοντα του Σενάριου 8.3 και Σενάριου 8.4, ο APPATTrectangles διατηρεί εξαιρετική ακρίβεια με μηδενικό σφάλμα στο Σενάριο 8.3 και σφάλμα 1.50% στο Σενάριο 8.4. Ο χρόνος εκτέλεσης παραμένει επίσης πολύ χαμηλός, με 23.40 δευτερόλεπτα στο Σενάριο 8.3 και 8.21 δευτερόλεπτα στο Σενάριο 8.4.

Συμπερασματικά, ο **Rectangles** προσφέρει ακριβείς διαδρομές, αλλά ο χρόνος εκτέλεσής του αυξάνεται σημαντικά σε πιο περίπλοκα περιβάλλοντα. Ο **PRM** παρέχει γρήγορη εκτέλεση, αλλά η ακρίβεια των διαδρομών του είναι ιδιαίτερα χαμηλή, καθιστώντας τον αναποτελεσματικό σε πολύπλοκα σενάρια. Ο **APPATTrectangles** καταφέρνει να διατηρεί υψηλή ακρίβεια και χαμηλό χρόνο εκτέλεσης, καθιστώντας τον τον πιο αποδοτικό αλγόριθμο για μεγάλα και πολύπλοκα περιβάλλοντα.

5.3.9 Συμπεράσματα

5.3.9.1 PRM

Ο αλγόριθμος PRM λόγο τις τυχαίας τοποθέτησης των κόμβων δεν εγγυάται μια βέλτιστη και ποιοτική διαδρομή για αυτό και παρατηρούμε μεγάλο σχετικό σφάλμα.

Όσο αφορά το χρόνο εκτέλεσης, εξαρτάται από το μέγεθος της περιοχής λόγο της αποθήκευσής του ελευθέρου χώρου για τοποθέτηση κόμβων μέσα σε αυτό. Ακόμη ο χρόνος δεν εξαρτάται από τον αριθμό των εμποδίων αλλά από το ποσό κοντά βρίσκονται μεταξύ τους. Αυτό συμβαίνει διότι πρέπει να δημιουργηθούν πιο πολύπλοκα μονοπάτια και έτσι ο αλγόριθμος αναγκάζεται να τρέξει περισσότερες φορές.

5.3.9.2 Rectangles

Ο αλγόριθμος Rectangles παρόλο που είναι αργός, όπως μπορούμε να παρατηρήσουμε, βρίσκει πάντα την συντομότερη διαδρομή. Αυτό συμβαίνει λόγω της στρατηγικής τοποθέτησης κόμβων γύρω από τις γωνίες των εμποδίων και την αποτελεσματική αποφυγή τους.

Όσο αφορά τον χρόνο εκτέλεσης, εξαρτάται άμεσα από τον αριθμό εμποδίων λόγω της μεγάλης πολυπλοκότητας κατασκευής του γραφήματος.

5.3.9.3 APPATTrectangles

Ο αλγόριθμος APPATTrectangles είναι ο ποιο αποδοτικός από τους τρεις αλγορίθμους. Η διαδρομή που βρίσκει ο αλγόριθμος έχει πολύ χαμηλό έως και μηδενικό σχετικό σφάλμα σε σχέση με την βέλτιστη διαδρομή που βρίσκει ο αλγόριθμος Rectangles. Ο χρόνος εκτέλεσης είναι στις πλείστες περιπτώσεις πιο σύντομος από τους υπολοίπους αλγορίθμους, και αυτό οφείλεται στο ότι ο αλγόριθμος δεν ελέγχει ολόκληρο το γράφημα αλλά μόνο όσα εμπόδια μας ενδιαφέρουν.

6

Συμπεράσματα και Προοπτικές

Σε αυτό το κεφάλαιο παρουσιάζονται τα συμπεράσματα της διπλωματικής εργασίας καθώς και οι προοπτικές για μελλοντική έρευνα και ανάπτυξη. Η ανάλυση αυτή θα δώσει μια συνολική εικόνα για τα αποτελέσματα που επιτεύχθηκαν και θα προτείνει τρόπους με τους οποίους μπορεί να εξελιχθεί περαιτέρω η έρευνα στο μέλλον.

6.1.1 Συμπεράσματα

Τα συμπεράσματα της διπλωματικής εργασίας αποτελούν μια ανασκόπηση των στόχων, των επιτευγμάτων και των ευρημάτων που προέκυψαν από την εφαρμογή και την πειραματική αξιολόγηση των αλγορίθμων PRM, Rectangles και APPATTrectangles για τον σχεδιασμό διαδρομών UAV σε περιβάλλοντα με εμπόδια.

Αρχικά, από την υλοποίηση των τριών αλγορίθμων προέκυψαν ενδιαφέροντα συμπεράσματα αναφορικά με την απόδοσή τους σε διάφορα περιβάλλοντα. Ο αλγόριθμος PRM, ο οποίος βασίζεται στην τυχαία τοποθέτηση κόμβων, επέδειξε μεγαλύτερη ευελιξία σε πολύπλοκα περιβάλλοντα, αλλά ταυτόχρονα αντιμετώπισε προβλήματα ακρίβειας. Ο χρόνος εκτέλεσης του PRM επηρεάζεται από το πόσο πυκνά είναι τα εμπόδια στο περιβάλλον, καθώς απαιτούνται περισσότερες δοκιμές και περισσότερη επεξεργασία για την εύρεση κατάλληλης διαδρομής.

Ο αλγόριθμος Rectangles, ο οποίος εγγυάται την εύρεση της συντομότερης διαδρομής, παρουσίασε πολύ υψηλή ακρίβεια, ιδιαίτερα σε περιβάλλοντα με μέτρια πολυπλοκότητα. Ωστόσο, αντιμετώπισε προβλήματα επιδόσεων σε περιβάλλοντα με μεγάλο αριθμό εμποδίων, καθώς η υπολογιστική του πολυπλοκότητα αυξάνεται σημαντικά όσο μεγαλώνει ο αριθμός των εμποδίων. Παρά τη δυνατότητά του να βρίσκει πάντα την βέλτιστη διαδρομή, ο αλγόριθμος δεν είναι πάντα πρακτικός για περιβάλλοντα με μεγάλη πυκνότητα εμποδίων.

Τέλος, ο αλγόριθμος APPATTrectangles παρουσίασε εντυπωσιακά αποτελέσματα συνδυάζοντας ακρίβεια και απόδοση. Η κύρια διαφορά του από τους άλλους δύο αλγορίθμους έγκειται στο γεγονός ότι δεν εξετάζει όλο το περιβάλλον, αλλά επικεντρώνεται μόνο στα κρίσιμα σημεία γύρω από τα εμπόδια. Έτσι, η διαδικασία γίνεται ταχύτερη χωρίς μεγάλη θυσία στην ακρίβεια της διαδρομής. Σε πολλές περιπτώσεις, η διαδρομή που εντοπίζεται είναι σχεδόν βέλτιστη, ενώ ο χρόνος εκτέλεσης είναι σημαντικά μικρότερος από αυτόν του Rectangles.

Συνολικά, η εργασία κατέληξε ότι ο αλγόριθμος APPATTrectangles αποτελεί μια αξιόπιστη και αποδοτική λύση για τον σχεδιασμό διαδρομών σε περιβάλλοντα με εμπόδια. Παρόλο που ο PRM είναι κατάλληλος για πιο ευέλικτες εφαρμογές και ο Rectangles για περιπτώσεις όπου

απαιτείται η βέλτιστη λύση, ο APPATTrectangles επιτυγχάνει τον καλύτερο συνδυασμό απόδοσης και ακρίβειας σε μεγάλο φάσμα σεναρίων.

6.2 Προοπτικές

Η παρούσα ενότητα εξετάζει τις μελλοντικές προοπτικές της εργασίας, παρουσιάζοντας τρόπους βελτίωσης που θα μπορούσαν να υλοποιηθούν αν υπήρχε περισσότερος χρόνος και προτείνοντας τρόπους συνέχισης της έρευνας από άλλους ερευνητές.

6.2.1 Βελτιώσεις με Περισσότερο Χρόνο

Αν υπήρχε περισσότερος χρόνος για την έρευνα και την ανάπτυξη της παρούσας εργασίας, θα μπορούσαν να πραγματοποιηθούν ορισμένες βελτιώσεις που θα ενίσχυαν τη λειτουργικότητα και την ακρίβεια των αλγορίθμων και των συστημάτων που αναπτύχθηκαν.

Η ενσωμάτωση δεδομένων από δορυφορικές εικόνες ή από το Global Navigation Satellite System (GNSS) θα μπορούσε να παρέχει ακριβέστερη και πιο ρεαλιστική απεικόνιση του περιβάλλοντος πτήσης των UAVs. Αυτό θα επιτρέψει τη δημιουργία πιο ακριβών και λεπτομερών πειραματικών σεναρίων, που θα προσομοιάζουν καλύτερα τις πραγματικές συνθήκες πτήσης. Χρησιμοποιώντας δορυφορικά δεδομένα, τα UAVs θα μπορούν να εντοπίζουν με μεγαλύτερη ακρίβεια τα γεωγραφικά χαρακτηριστικά της περιοχής πτήσης, επιτρέποντας έτσι την καλύτερη απόδοση των αλγορίθμων σχεδιασμού διαδρομών σε πραγματικά περιβάλλοντα.

Μια σημαντική βελτίωση θα ήταν η πλήρης δυναμική προσαρμογή του αλγορίθμου APPATTrectangles. Σήμερα, ο αλγόριθμος λαμβάνει υπόψη την αρχική θέση και τα εμπόδια στο περιβάλλον για να καθορίσει τη διαδρομή του UAV. Αν υπήρχε η δυνατότητα να λειτουργεί πλήρως δυναμικά, ο αλγόριθμος θα μπορούσε να λαμβάνει υπόψη το προηγούμενο εμπόδιο σε κάθε βήμα της πτήσης αντί της αρχικής θέσης, επιτρέποντας στο UAV να αποφασίζει την πορεία του σε πραγματικό χρόνο. Αυτή η προσέγγιση θα βελτίωνε σημαντικά την απόκριση του UAV σε απρόβλεπτα εμπόδια, αυξάνοντας την ασφάλεια και την αποδοτικότητα της πτήσης.

6.2.2 Συνέχιση της Έρευνας από Άλλους Ερευνητές

Η παρούσα εργασία παρέχει μια ισχυρή βάση για μελλοντική έρευνα και μπορεί να επεκταθεί περαιτέρω από άλλους ερευνητές. Οι ακόλουθες προτάσεις παρέχουν οδηγίες για τη συνέχιση της έρευνας.

Ένα σημαντικό βήμα για τη συνέχιση της έρευνας θα ήταν η ενσωμάτωση αισθητήρων και καμερών στα UAVs, επιτρέποντάς τους να πραγματοποιούν τοπική αναζήτηση και αποφυγή εμποδίων σε πραγματικό χρόνο. Με τη χρήση αισθητήρων, το UAV θα μπορούσε να ανιχνεύει κινούμενα εμπόδια, όπως πουλιά, ή εμπόδια που δεν είναι εμφανή στους χάρτες, όπως δέντρα. Η χρήση δεδομένων από αισθητήρες θα βελτίωνε σημαντικά την αντίληψη του UAV για το περιβάλλον και θα αύξανε την ασφάλεια των πτήσεων, ιδιαίτερα σε δυναμικά ή άγνωστα περιβάλλοντα.

Τέλος, ένα απαραίτητο βήμα για τη συνέχιση της έρευνας είναι η δοκιμή των αλγορίθμων σε πραγματικά UAV. Οι πειραματικές δοκιμές σε πραγματικά περιβάλλοντα θα επιτρέψουν την αξιολόγηση της απόδοσης των αλγορίθμων υπό ρεαλιστικές συνθήκες πτήσης. Επιπλέον, οι δοκιμές αυτές θα βοηθήσουν στην προσαρμογή των αλγορίθμων έτσι ώστε να λειτουργούν με πραγματικά δεδομένα που συλλέγονται κατά τη διάρκεια της πτήσης, όπως δεδομένα από αισθητήρες ή κάμερες. Αυτό θα μπορούσε να οδηγήσει σε βελτίωση των αλγορίθμων για

πρακτική εφαρμογή και στην αντιμετώπιση προβλημάτων που δεν είναι ορατά σε προσομοιωμένα περιβάλλοντα.

Βιβλιογραφία- Αναφορές

[1]. Huan Liu, Xiamiao Li , Mingfeng Fan , Guohua Wu , Witold Pedrycz , An Autonomous Path Planning Method for Unmanned Aerial Vehicle Based on a Tangent Intersection and Target Guidance Strategy

[2]. Nouman Bashir, Saadi Boudjit, Gabriel Dauphin, Sherali Zeadally , An obstacle avoidance approach for UAV path planning

[3]. Wenqiang Chen, Ning Wang, Xiaofeng Liu, Chenguang Yang, VFH* Based Local Path Planning For Mobile Robot

[4]. J. Borenstein, Member and Y. Koren, THE VECTOR FIELD HISTOGRAM - FAST OBSTACLE AVOIDANCE FOR MOBILE ROBOTS

[5]. Vincent Roberge, Mohammed Tarbouchi, and Gilles Labonté. Comparison of Parallel Genetic Algorithm and Particle Swarm Optimization for Real-Time UAV Path Planning

[6]. Abdurrahim Sonmez, Emre Kocyigit and Emin Kugu. Optimal Path Planning for UAVs Using Genetic Algorithm

[7]. Dongshu Wang · Dapei Tan · Lei Liu, Particle swarm optimization algorithm: an overview

[8]. Iram Noreen , Amna Khan, Zulfiqar Habib, A Comparison of RRT, RRT* and RRT*-Smart Path Planning Algorithms

[9]. Gang Chen a,* , Ning Luo a , Dan Liu a , Zhihui Zhao b,c , Changchun Liang b, Path planning for manipulators based on an improved probabilistic road