**Εξαμηνιαία Εργασία 2 (ΜΕΡΟΣ Γ)**

**Σχεδιασμός τροχιάς με αποφυγή εμποδίου από κινούμενο ρομπότ σε γνωστό χάρτη**
(Mobile robots: Path planning and obstacle avoidance)

Ρομποτική 2 Ευφυή Ρομποτικά Συστήματα 2020-2021

8ο Εξάμηνο

Η εργασία έγινε υπό την επιμέλεια των φοιτητών **Αναστασία Κριθαρούλα** με αριθμό μητρώου **el17073** και **Διονύση Κριθαρούλα** με αριθμό μητρώου **el17875.**

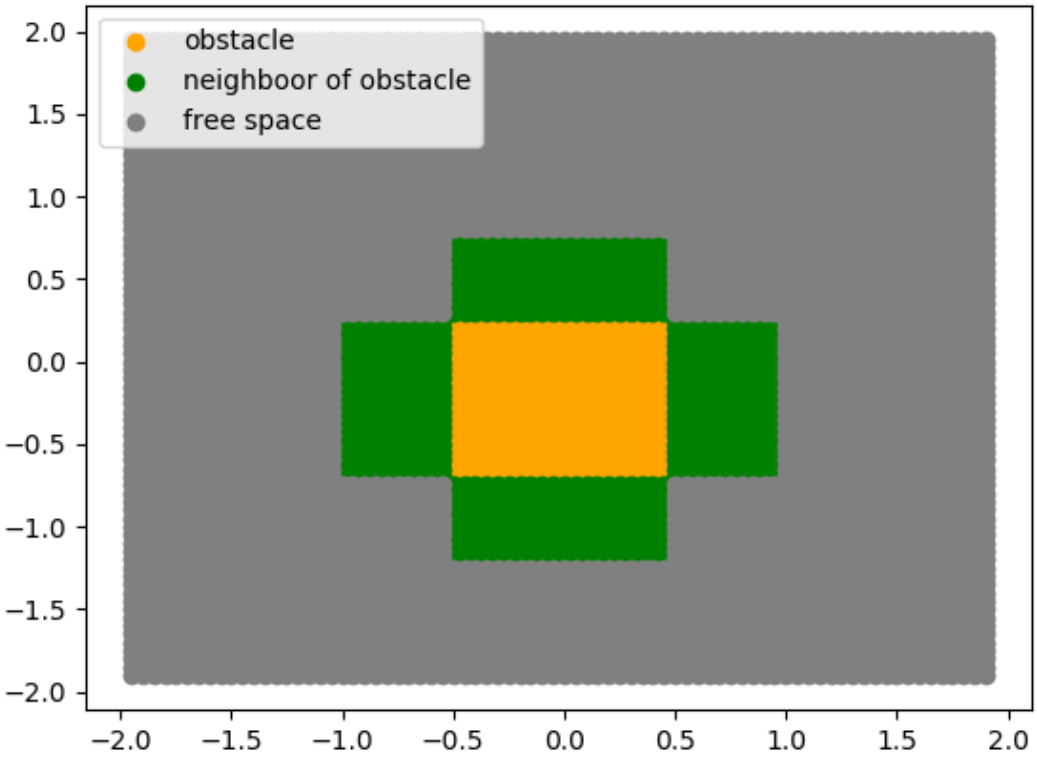
29/8/2021

**Εισαγωγή**

Στόχος της συγκεκριμένης εργασίας είναι η υλοποίηση ενός αλγορίθμου για την προσέγγιση συγκεκριμένου στόχου από το κινούμενο ρομπότ μέσα σε χώρο γνωστών διαστάσεων (4mx4m) ο οποίος δίνεται στην εκφώνηση της εργασίας. Για το σκοπό αυτό με χρήση του αλγορίθμου A* (A-star) σχεδιάστηκε βέλτιστη διαδρομή (path-planning) την οποία πρέπει να ακολουθήσει το ρομπότ προκειμένου να προσεγγίσει τον στόχο. Η διαδρομή αυτή σχεδιάστηκε λαμβάνοντας υπόψιν το τετράπλευρο εμπόδιο γνωστών διαστάσεων και γνωστής θέσης προκειμένου να αποφεύγεται η σύγκρουση του ρομπότ με αυτό και τελικά το ρομπότ να φτάνει επιτυχώς στον στόχο.

**Α. Θεωρητική Ανάλυση**

Πρώτο βήμα προκειμένου να βρεθεί η διαδρομή που θα ακολουθήσει το ρομπότ είναι η παράσταση του χώρου κίνησης. Γνωρίζοντας τις διαστάσεις του χώρου κίνησης του ρομπότ αλλά και του τετράπλευρου εμποδίου υλοποιήσαμε μία ομοιόμορφη αποσύνθεση (uniform decomposition) του χώρου κίνησης δημιουργώντας το λεγόμενο occupancy grid. Κάθε τετράγωνο που σχηματίζεται από την διαδικασία αυτή αποτελεί έναν κόμβο στον οποίο μπορεί να βρεθεί το κινούμενο ρομπότ. Το κάθε τετράγωνο επιλέχθηκε να έχει πλευρά μήκους 0.05 (resolution). Έτσι εφόσον οι διαστάσεις του χάρτη είναι 4mx4m δημιουργήθηκαν συνολικά 80x80 τετράγωνα. Επιπλέον σε κάθε τετράγωνο (κόμβο) αποδίδεται μία τιμή η οποία προσδιορίζει αν το ρομπότ μπορεί να βρεθεί στο συγκεκριμένο σημείο ή όχι. Στα σημεία που δεν μπορεί να βρεθεί το ρομπότ δόθηκε η τιμή 254. Τα σημεία αυτά είναι τα σημεία που συνιστούν το γνωστών διαστάσεων και θέσης εμπόδιο αλλά και γειτονικά σημεία στα οποία αν βρεθεί το ρομπότ ,λόγω των διαστάσεων του, κινδυνεύει και πάλι να συγκρουστεί με το εμπόδιο. Αντίθετα στα υπόλοιπα σημεία αποδόθηκε η τιμή 0 η οποία υποδεικνύει ότι το ρομπότ μπορεί ελεύθερα να βρεθεί στα σημεία αυτά. Επιπλέον στα σημεία τα οποία παριστάνουν τα όρια κίνησης του χάρτη αποδόθηκε επίσης η τιμή 254 ώστε το ρομπότ να μην μπορεί να διαφύγει και να συγκρουστεί με κάποιον από τους εξωτερικούς τοίχους. Στην συνέχεια παρατίθεται μία απεικόνιση του occupancy grid που περιεγράφηκε προηγουμένως:




Όπως παρατηρούμε στην εικόνα με πορτοκαλί και πράσινο είναι οι περιοχές οι οποίες αντιστοιχούν στο ρομπότ και στην «γειτονιά» του ρομπότ αντίστοιχα και στις οποίες δεν έχει πρόσβαση το ρομπότ. Αντίστοιχα με γκρι είναι η περιοχή στην οποία μπορεί να βρεθεί ελεύθερα το ρομπότ.

Αφού σχεδιάσαμε τον χώρο κίνησης προχωρήσαμε στην υλοποίηση του αλγορίθμου A* ο οποίος θα μας δώσει και την διαδρομή που επιθυμούμε. Αρχικά θεωρήσαμε ότι κάθε κόμβος (δηλαδή κάθε τετράγωνο του occupancy grid) έχει ακμές προς 8 γειτονικούς του κόμβους, δηλαδή τους 4 γείτονες της κάθε πλευράς του τετραγώνου αλλά και τους 4 διαγώνιους, εφόσον αυτοί είναι ελεύθεροι. Σε περίπτωση που κάποιος δεν είναι ελεύθερος τότε δεν υπάρχει ακμή που να συνδέει τον κόμβο αυτό με κάποιον άλλο κόμβο κάνοντας τον έτσι απροσπέλαστο από το ρομπότ.

Δημιουργήσαμε έτσι τελικά έναν γράφο πάνω στον οποίο τρέξαμε τελικά τον γνωστό αλγόριθμο A* για την εύρεση βέλτιστης διαδρομής.

**Περιληπτική εξήγηση της λειτουργίας του αλγορίθμου A***

Ο αλγόριθμος σε κάθε του βήμα επιλέγει τον κόμβο με την μικρότερη συνολική απόσταση. Η συνολική απόσταση ορίζεται ως η απόσταση από τον αρχικό κόμβο συν την απόσταση από τον κόμβο στόχο. Η απόσταση από τον κόμβο στόχο βρίσκεται με την βοήθεια μίας ευρηστικής συνάρτησης. Στην δικιά μας περίπτωση επιλέχθηκε η Ευκλείδεια απόσταση η οποία υπολογίζεται ως εξής:

$$d = \sqrt{(xgoal - x)^2 + (ygoal - y)^2}$$

όπου [xgoal,ygoal] οι συντεταγμένες του κόμβου στόχου και [x,y] οι συντεταγμένες του επιλεγμένου κόμβου.

Εφόσον επιλεχθεί ένας κόμβος στην συνέχεια βρίσκουμε τους γειτονικούς του κόμβους . Για τον κάθε γειτονικό κόμβο έχουμε τρεις περιπτώσεις:

- Αν ο κόμβος αυτός έχει επιλεχθεί στο παρελθόν ως ο κόμβος με την μικρότερη συνολική απόσταση τότε αγνοείται.
- Αν ο κόμβος αυτός δεν έχει εξερευνηθεί στο παρελθόν τότε ανανεώνεται η απόσταση του με βάση την απόσταση από τον αρχικό κόμβο συν την ευκλείδεια απόσταση του η οποία υπολογίζεται όπως εξηγήθηκε προηγουμένως. Η απόσταση από τον αρχικό κόμβο υπολογίζεται ως εξής:

  d = d[πατέρα] + w

  όπου d[πατέρα] η απόσταση του πατέρα από τον αρχικό κόμβο και w το κόστος μετάβασης από τον κόμβο-πατέρα στον συγκεκριμένο γειτονικό κόμβο. Επιπλέον ως πατέρας του κόμβου αυτού ορίζεται ο επιλεγμένος κόμβος. Τέλος ο κόμβος αυτός προστίθεται σε μία λίστα που ονομάζεται open_list και η οποία περιέχει όλους τους κόμβους που έχουν εξερευνηθεί τουλάχιστον μία φορά αλλά δεν έχουν επιλεγεί ως κόμβοι με την μικρότερη συνολική απόσταση.

- Αν έχει εξερευνηθεί στο παρελθόν από άλλον κόμβο «πατέρα» αλλά δεν έχει επιλεγεί ως ο κόμβος με την μικρότερη συνολική απόσταση σε κάποιο προηγούμενο βήμα του αλγορίθμου τότε ελέγχουμε μήπως ανανεώσουμε την συνολική απόσταση του ως εξής:

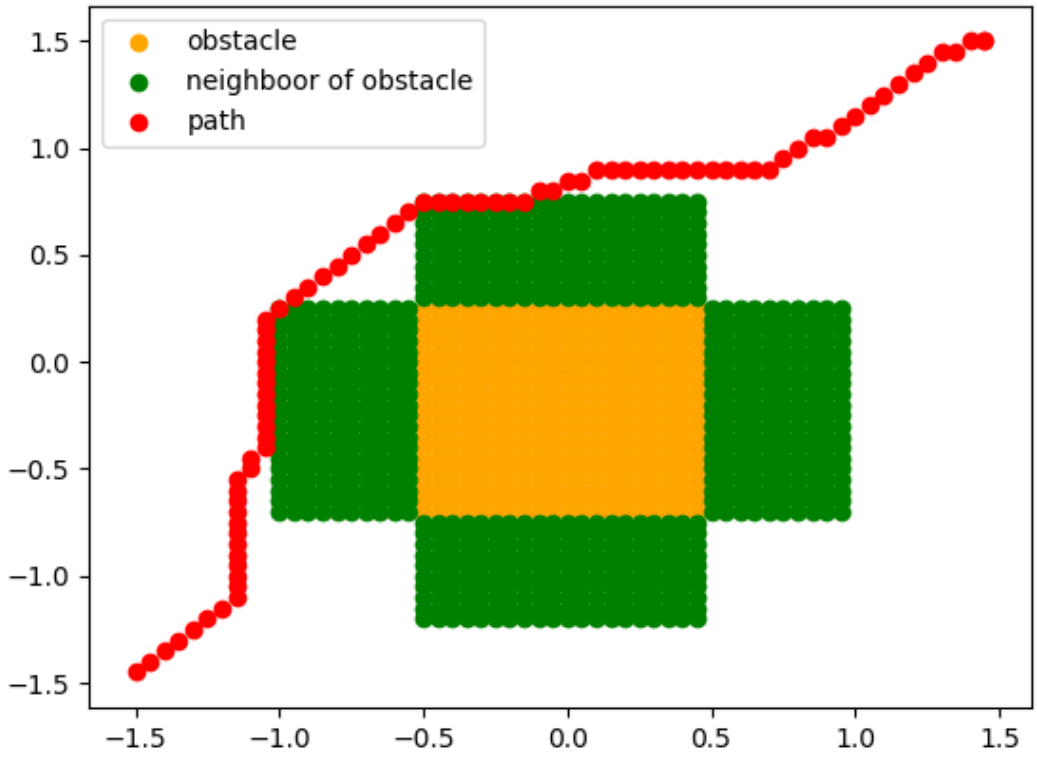
  Αν d[πατέρα] + w < d τότε d = d[πατέρα] + w,

  όπου d[πατέρα] η απόσταση του πατέρα από τον αρχικό κόμβο ,w το κόστος μετάβασης από τον κόμβο-πατέρα στον συγκεκριμένο γειτονικό κόμβο και d η μέχρι τώρα απόσταση από τον αρχικό κόμβο του γειτονικού κόμβου που εξετάζουμε. Στην περίπτωση ανανέωσης της απόστασης αυτής ανανεώνεται και ο κόμβος-πατέρας του γειτονικού αυτού κόμβου και τίθεται πλέον ως ο κόμβος που έχει επιλεχθεί σε αυτό το βήμα.

Στην συνέχεια επιλέγουμε ξανά τον κόμβο με την μικρότερη συνολική απόσταση από το σύνολο τον κόμβων που ανήκουν στην λίστα open_list που αναφέραμε προηγουμένως (δηλαδή που δεν ανήκουν στην λίστα των κόμβων που έχουν ήδη επιλεγεί σε προηγούμενο βήμα ή δεν έχουν εξερευνηθεί ακόμα) και επαναλαμβάνουμε την διαδικασία (κάθε κόμβος που επιλέγεται σε ένα βήμα μπαίνει σε μία λίστα που ονομάζεται closed_list έτσι ώστε να μην επιλεγεί ξανά σε επόμενο βήμα αλλά και να αγνοηθεί σε περίπτωση που κάποιος γειτονικός του κόμβος επιλεγεί μετέπειτα.)

Το βάρος w μετακίνησης από έναν κόμβο σε έναν άλλο μέσω μίας ακμής που τους συνδέει έχει επιλεγεί να είναι 1 για κάθε ακμή του γράφου.

Η διαδρομή που τελικά βρίσκει ο αλγόριθμος A* που υλοποιήσαμε για την μετάβαση του ρομπότ από την θέση (x,y) = (-1.5,-1.5) στην θέση (x,y) = (1.5,1.5) υπολογίστηκε ως ακολουθία σημείων πάνω στο occupancy grid που δημιουργήσαμε προηγουμένως. Η διαδρομή αυτή φαίνεται στην συνέχεια:



Στην συνέχεια υλοποιήθηκε ένας ελεγκτής προκειμένου τελικά το ρομπότ να ακολουθήσει προσεγγιστικά την συγκεκριμένη ακολουθία σημείων (διαδρομή) που υπολογίστηκε προηγουμένως. Ο ελεγκτής αυτός υλοποιήθηκε ως εξής:

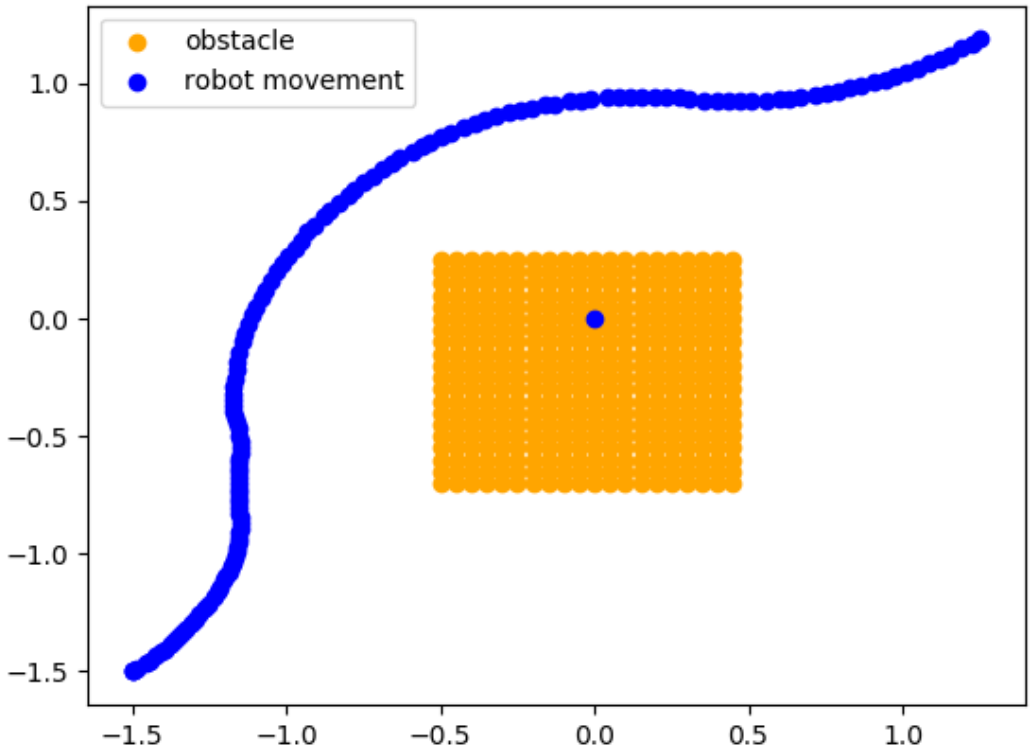
- Ξεκινώντας από το πρώτο σημείο της υπολογισμένης διαδρομής και για κάθε ένα βρίσκεται η διαφορά της απόστασης του από την τωρινή θέση του ρομπότ σε κάθε συντεταγμένη ως εξής:
    - x_error = x(σημείου διαδρομής) – x(τωρινή θέση ρομπότ)
    - y_error = y(σημείου διαδρομής) – y(τωρινή θέση ρομπότ)

- Όσο η απόσταση x_error είναι μεγαλύτερη από 0.1 ή η απόσταση y_error είναι μεγαλύτερη από 0.1 τότε:
    - Δίνουμε ως γραμμική ταχύτητα στο ρομπότ v = k1*(error_$x^2$ + error_$y^2$)
    - Δίνουμε ως γωνιακή ταχύτητα στο ρομπότ w = k2*(atan2(-error_y,-error_x)-theta) όπου theta η τωρινή γωνία του ρομπότ ως προς τον κατακόρυφο άξονα z.

- Όταν τόσο η απόσταση error_x όσο και η error_y γίνουν μικρότερες από 0.1 τότε επαναλαμβάνουμε την διαδικασία για το επόμενο σημείο της διαδρομής.

Τα κέρδη k1 , k2 υπολογίστηκαν πειραματικά.

Η διαδρομή που τελικά ακολουθεί το ρομπότ φαίνεται στην συνέχεια:




**Οδηγίες προκειμένου να τρέξει η προσομοίωση**

Το πακέτο που δημιουργήθηκε για τις ανάγκες της άσκησης ονομάζεται my_pkg. Το launch αρχείο όπου ξεκινάει όλη την διαδικασία που περιγράψαμε προηγουμένως ονομάζεται my_pkg.launch και βρίσκεται μέσα στον φάκελο launch. Ο κώδικας σε python που εκτελεί την άσκηση ονομάζεται my_code.py και βρίσκεται μέσα στο φάκελο scripts. Προκειμένου να εκτελέσουμε την προσομοίωση εκτελούμε την εντολή **roslaunch my_pkg my_pkg.launch αφού πρώτα έχουμε εκτελέσει την εντολή η οποία δημιουργεί την προσομοίωση (roslaunch mymobibot_gazebo mymobibot_world_pp.launch).** Εκτελώντας την εντολή roslaunch my_pkg my_pkg.launch θα εμφανισθεί αρχικά μία γραφική παράσταση η οποία απεικονίζει την βέλτιστη διαδρομή που υπολογίζει ο αλγόριθμος που αναλύσαμε προηγουμένως. Πατώντας την επιλογή «x» το ρομπότ αρχίζει και εκτελεί την τροχιά προκειμένου να προσεγγίσει τον κόμβο στόχο.