Ευφυή Συστήματα Ρομπότ
ΤΗΜΜΥ ΑΠΘ

Αργύριος Κοκκίνης

ΑΕΜ: 8459

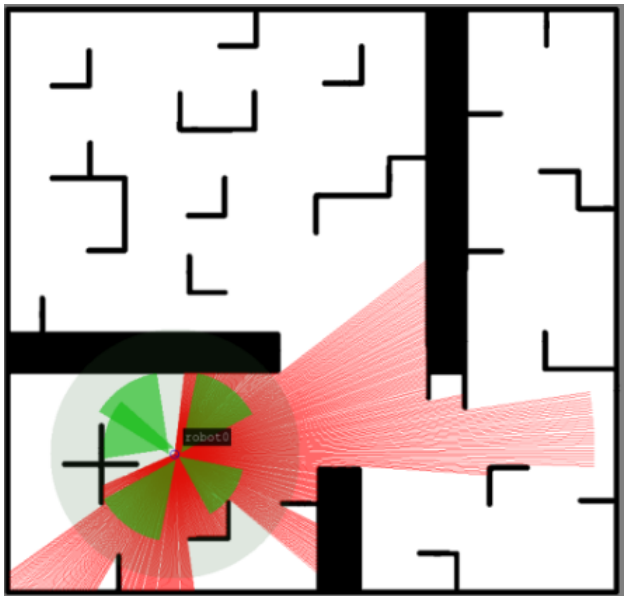
Ιούλιος 2019

1 Introduction

Το θέμα της συγκεκριμένης εργάσιας είναι η σχεδίαση ενός συστήματος αυτόματης εξευρένησης άγνωστων αρχικά χώρων (SLAM). Η εργασία υλοποιείται επεκτείνοντας τα ήδη υπάρχοντα modules του ρομποτικού πράκτορα με στόχο την πλήρη-βέλτιστη (σε χρόνο,κινήσεις) εξευρένηση χρησιμοποιόντας τεχνικές και αλγορίθμους που εφαρμόζονται σε αντίστοιχα συστήματα.

Ο κώδικας υλοποιείται σε Python στηριζόμενος σε πακέτα και βιβλιοθήκες του ROS, ενώ χρησιμοποιούνται τα εργαλεία Rviz και ο STDR Simulator για την προσομοίωση.

Ο πράκτορας που χρησιμοποιείται είναι κυκλικό σώμα χωρίς μάζα με ακτίνα $0.1m$ που κινείται στον 2D χώρο. Η κίνηση του στον χώρο μπορεί να είναι μεταφορική ή/και περιστροφική γύρω απο νοητό άξονα κοντά στον πράκτορα και όχι επιτόπια.Είναι εξοπλισμένος με αισθητήρες Lidar $[-120°, 120°]$, Sonars και RFID. Παρ'ολα αυτά στην υλοποίηση που ακολούθησα χρησιμοποήσα μόνο τις πληροφορίες απο το Lidar.



Σ 1: agent in the RTSD simulator

2 Implementation

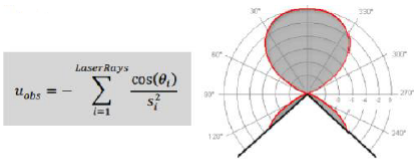
Η υλοποίηση αποτελείται απο τρία στάδια. Αρχικά είναι η αποφυγή των εμποδίων και η ακολούθηση του μονοπατιού με ταυτόχρονη αποφυγή των εμποδίων. Χρησιμοποίησα το Motor Schema προκειμένου να συνδυάσω αυτα τα δύο. Στην συνέχεια είναι η επιλογή ενός σημείου στον χάρτη στο οπόιο θα κινηθεί ο πράκτορας. Τέλος, η βελτιστοποίηση του μονοπατιού πάνω στο οποίο κινείται το ρομπότ.

2.1 Obstacle Avoidance - Path Following

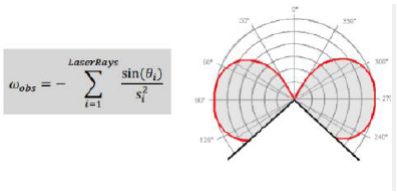
Για να βρώ τις ταχύτητες (μεταφορική,περιστροφική) που πρέπει να έχει ο πράκτορας κάθε χρονική στιγμή ώστε να διασφαλιστεί ότι δεν θα χτυπήσει σε εμπόδιο έπρεπε να χρησιμοποιήσω τα attributes `angle_min` και `angle_max` της κλάσης LaserScan ώστε να βρώ την αρχική και την τελική γωνία του Lidar scan σε κάθε στιγμή. Με αυτες τις πληροφορίες μπορώ να υπολογίσω την γωνία που σχηματίζουν όλες οι ακτίνες του laser, θεωρώντας ότι ισαπέχουν μεταξύ τους. Γνωρίζοντας ότι για τον υπολογισμό της μεταφορικής ταχύτητας ενδιαφερόμαστε κυρίως για τις μετρήσεις των μπροστιών ακτινών του Lidar ενώ για την περιστροφική των πλάγιων καταλήγουμε ότι :

$$\upsilon_{abs} = -\sum_{i}^{LaserRays} \frac{\cos(\theta_i)}{s_i^2} \qquad \omega_{abs} = -\sum_{i}^{LaserRays} \frac{\sin(\theta_i)}{s_i^2}$$

Όπου s_i η απόσταση που μετράει η i-οστή ακτίνα και θ_i η γωνία που σχηματίζει αυτή.



Σ 2: Frontal rays

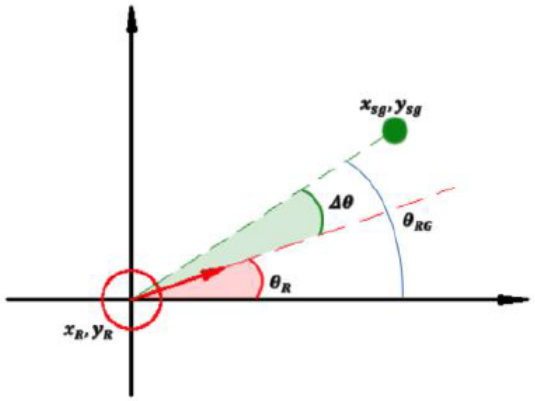


Σ 3: Side rays

Με τις τελικές ταχύτητες που πρέπει να έχει ο πράκτορας για αποφυγή εμποδίων να προκύπτουν:

$\upsilon = \upsilon_{max} + c_u * \upsilon_{abs}$, $\omega = c_\omega * \omega_{abs}$ με $\upsilon_{max} = 0.3$ και οι συντελεστές να προκύπτουν πειραματικά $c_u = c_\omega = 0.3$.

Για τον υπολογισμό των ταχυτήτων που πρέπει να έχει ο πράκτορας ώστε να προσεγγίσει τον στόχο τα δεδομένα τα οποία γνωρίζουμε είναι η θέση (συντεταγμένες) του στόχου και η θέση του ρομπότ στον χάρτη.



Σ 4: agent-target

με $\tan\theta_{rg} = \frac{y_{sg}-y_r}{x_{sg}-x_r}$ και με $\Delta\theta = \theta_{rg} - \theta_r$ Κανονικοποιώντας έτσι το $\Delta\theta$ ώστε να ανήκει στο διάστημα $[-\pi, \pi]$ προκύπτει η γωνιακή ταχύτητα $\omega = \frac{\Delta\theta}{\pi}$.

Φράζοντας την παραπάνω ταχύτητα στο διάστημα που ορίζει η μέγιστη γωνιακή ταχύτητα που μπορεί να έχει ο πράκτορας προκύπτει η τελική ταχύτητα που πρέπει να έχει για να προσεγγίσει τον στόχο.

$$\omega_{path} = 0.3 * \omega$$

Αντίστοιχα για τον υπολογισμό της μεταφορικής ταχύτητας:

$$\upsilon = \upsilon_{max} * [1 - |\omega_{path}|]^n$$

με το $v_{max} = 0.3$ και το n να προκύπτει πειραματικά ίσο με 5.
Οι ταχύτητες που προέκυψαν ότι πρέπει να ακολουθήσει το ρομπότ για την προσέγγιση του στόχου πρέπει να συνδυαστούν με τις ταχύτητες για την αποφυγή εμποδίων. Οι ταχύτητες συνδυάζονται χρησιμοποιώντας Motor Schema και προκύπτουν:

$$v = v_{goal} + c_u * v_{obs} \qquad \omega = \omega_{goal} + c_\omega * \omega_{obs}$$

Με τους συντελεστές για την μεταφορική και γωνιακή ταχύτητα να προκύπτουν 0.15 και 0.2 αντίστοιχα.

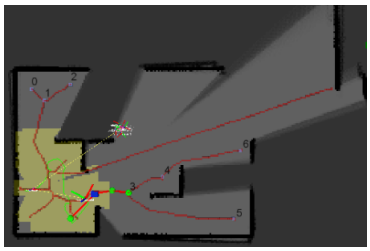
2.2 Target Selection

Η επιλογή του στόχου (σημείου του χάρτη) προς το οποίο θα κινηθεί ο πράκτορας πρέπει να είναι τέτοιος ώστε να διασφαλίζει την πλήρη εξευρεύνηση του χάρτη με το μικρότερο δυνατό κόστος (κινήσεις,χρόνος).

Εφόσον δεν έχει υλοποιηθεί συγκεκριμένη μέθοδος για target selection επιλέγεται ένα τυχαίο σημείο του χάρτη το όποιο δεν έχει εξερευνηθεί ακόμη.Αυτη η μέθοδος παρ'ότι θα οδήγησει κάποια στιγμή σε full coverage οδηγεί το ρομπότ να κάνει πολλές ανούσιες κινήσεις και διαδρομές.

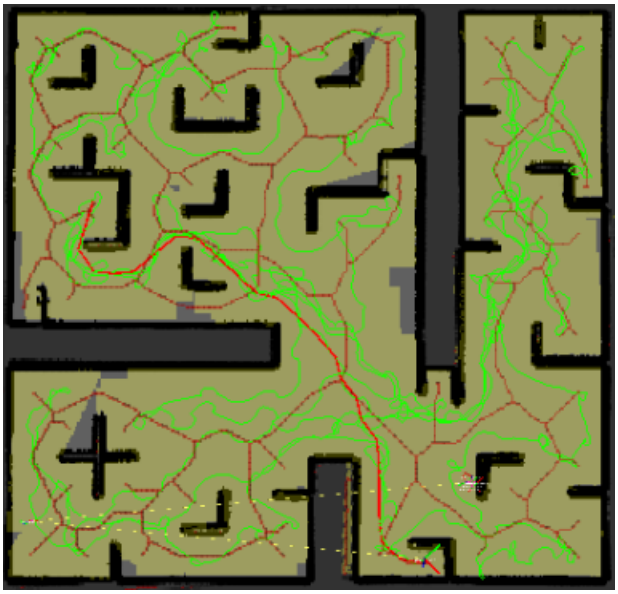
Η μέθοδος που εφάρμοσα στην επιλογή στόχου είναι μια απλουστευμένη μέθοδος του Cost based selection όπου επιλέγεται ο κόμβος του τοπολογικού γράφου που απέχει την ελάχιστη ευκλείδεια απόσταση απο το ρομπότ.Επιλέγοντας κόμβους που ανήκουν στον τοπολογικό γράφο είμαστε βέβαιοι ότι εξερευνόντας τους όλους θα έχουμε πετύχει full coverage. Ουσιαστικά είναι μια εφαρμογή του cost based selection όπου λαμβάνεται υπ'οψιν μόνο η ευκλείδεια απόσταση.

Η συγκεκριμένη μέθοδος είναι πολύ απλή και υλοποιείται ώς εξής. Σε κάθε επανάληψη βρίσκεται ο τοπολογικός γράφος και οι κόμβοι που δέν έχουν ακόμη εξευρευνηθεί συλλέγονται σε μια λίστα. Γνωρίζοντας τις συντεταγμένες του πράκτορα υπολογίζεται η ευκλείδεια απόσταση μεταξύ του ρομπότ και όλων των σημείων της λίστας.Σαν στόχος επιλέγεται ο κόμβος με την ελάχιστη απόσταση.Σε περίπτωση όπου το path planning προς τον συγκεκριμένο στόχο αποοτύχει επιλέγεται ένα τυχαίο σημείο του χάρτη που δεν έχει ακόμη εξερευνηθεί.



Σ 5: starting exploration

Σ 6: exploring the map



Σ 7: achieving full coverage

Υπάρχουν φορές που είτε το path planning αποτυγχάνει επειδή μπορέι ο στόχος να βρίσκεται σε εμπόδιο ή κοντά σε ανεξερεύνητη περίοχη είτε μπορεί λόγω κακού skeletonization να έχουν εμφανιστεί λανθασμένοι κόμβοι ή να μην έχουν εμφανιστεί κόμβοι σε σημεία όπου έπρεπε να υπάρχουν. Αυτές οι καταστάσεις οδηγούν τον πράκτορα να μην επιλέγει πάντα τους κοντινότερους στόχους ή να επιλέγει στόχους με τυχαίο τρόπο.

Αυτό μπορεί να φανεί και στο Σχήμα 7 με κάποιες διαδρομές που επαναλαμβάνονται.

2.3 Path Alteration

Το path planning γίνεται μέσω του αλγορίθμου A^*, όπου για κάθε στόχο σχηματίζεται το μονοπάτι που αποτελείται απο υπο-στόχους οι οποίοι πρέπει να προσεγγίστουν απο τον πράκτορα και τον οδηγούν στον τελικό στόχο.Για την προσέγγιση του κάθε υπο-στόχου ορίζεται μια χρονική περίοδος 23 δευτερολέπτων, σε περίπτωση όπου αυτος ο στόχος δεν έχει προσεγγιστεί μέσα σε αυτήν την περίοδο δημιουργείται time out και η διαδικασία επαναλαμβάνεται απο την αρχή.

Ο πράκτορας κάθε φόρα βλέπει μόνο τον επόμενο υπο-στόχο και όχι τον τελικό προορίσμο. Αυτό σημαίνει ότι σε περίπτωση όπου προσεγγιστεί ενός επόμενος στόχος αυτός δεν λαμβάνεται υπ'όψιν. Για αυτόν τον λόγο κάθε φορά γνωρίζοντας της θέση του ρομπότ βρίσκω τον στόχο που απέχει την ελάχιστη ευκλείδεια απόσταση απο τον πράκτορα και ελέγχω άμα αυτός ο στόχος έχει προσεγγιστεί επαρκώς.

Το μονοπάτι το οποίο δημιουργείται απο τον A^* μπορεί να έχει γωνίες ή απότομα σημεία τα οποία να δυσκολεύουν την κίνηση του πράκτορα ή να την κάνουν αφύσικη. Για αυτον τον λόγο εφάρμοσα path smoothing χρησιμοποιώντας ελαχιστοποίηση με gradient descent.

Θεωρώντας ότι το αρχικό μονοπάτι είναι το x_i και το μονοπάτι που θέλουμε να έχουμε είναι το y_i στόχος είναι να ελαχιστοποιήσουμε την απόσταση τους ελαχιστοποιώντας την συνάρτηση:

$$f = \big(x_i - y_i\big)^2$$

Αρχικοποιώντας το μονοπάτι y_i με το x_i προσπαθούμε να ελαχιστοποιήσουμε τις αποστάσεις ανά επανάληψη μεταξύ δύο διαδοχικών σημείων του μονοπατιού.

$$min\big(y_i - y_{i-1}\big)^2 \qquad min\big(y_i - y_{i+1}\big)^2$$

Η ελαχιστοποίηση γίνεται με την gradient descent όπου

$$y_i = y_i + \alpha\big(x_i - y_i\big) + \beta\big(y_{i+1} + y_{i-1} - 2y_i\big)$$

Όπου οι συντελεστές α, β καθορίζουν το πόσο ομαλό θέλουμε να γίνει το μονοπάτι και προέκυψαν πειραματικά 0.5 και 0.1 αντίστοιχα.

Η εφαρμογή του path smoothing γίνεται όταν έχουμε τουλάχιστον 4 υπο-στόχους (δηλαδή αρκετά μεγάλο μονοπάτι) ώστε να έχει νόημα η διαδικασία.

Η χρήση του motor schema για τον συνδυασμό των ταχυτήτων μπορεί να οδηγήσει σε τοπικά ελάχιστα που εγκλωβίζουν τον πράκτορα. Αυτό συνήθως συμβαίνει όταν ο πράκτορας βρίσκεται κοντά σε εμπόδια και το μονοπάτι που δημιουργείται περνάει μέσα απο κάποιο εμπόδιο.

Προκειμένου να αντιμετωπίσω αυτο το πρόβλημα στηρίχτηκα στο γεγονός ότι όποτε συμβαίνει αυτό τότε θα υπάρχει time out επειδή το ρομπότ δεν θα καταφέρει να φτάσει στον στόχο του. Έτσι σε περίπτωση όπου συμβεί time out την επόμενη φορά ο πράκτορας θα αγνοήσει το γεγονός ότι πρέπει να προσεγγίσει κάποιον στόχο αλλά θα κινηθεί με αποκλειστικό σκοπό να φυγεί μακριά απο τα εμποδία. Ουσιαστικά θα μπεί σε obstacle avoidance mode κινούμενος όμως με μέγιστη μεταφορική ταχύτητα $0.2m/s$ αντί για 0.3. Επιπλέον θα βρίσκεται σε αυτο το mode για τα πρώτα 13 δευτερόλεπτα του συνολικού χρόνου που έχει στην διάθεση του ενώ στα υπόλοιπα 10 θα κινηθεί κανονίκα προσεγγίζοντας τον επόμενο στόχο που έχει επιλεγεί.

Σε περίπτωση όπου φύγει αρκετά μακριά και δεν καταφέρει να φτάσει στον στόχο προτού συμβεί νέο time out, την επόμενη φορά θα κινηθεί κανονικά αγνοώντας το time out που μόλις συνέβει.

Αυτήν την λειτουργία να επιτυγχάνω χρησιμοποιώντας δύο flags.Ένα το όποιο υποδυκνύει πότε έχευ συμβέι time out και ένα που σηματοδοτεί την επιστροφή του πράκτορα στο path following mode.

3 Discussion

Η επιλογή του closest topological node ως μέθοδος επιλογής στόχου είναι αλήθεια ότι είναι πολύ απλή και δεν λαμβάνει υπ'όψιν παραμέτρους όπως άμα ο στόχος αυτός βρίσκεται στην "πλάτη" του ρομπότ και απαιτεί μεγάλη περιστροφή, κάτι που θα έκανε την όλη κίνηση αρκετά δύσκολη για το ρομπότ. Αυτό είναι κάτι που θα μπορούσε να προστεθεί και να ληφθεί υπ'όψιν χρησιμοποιώντας αντίστοιχα βάρη δημιουργούντας έτσι μια σωστότερη cost based επιλογή στόχου.

Παρ'ολα αυτά μια τέτοια μέθοδος μπορεί να ήταν αρκέτα πιο χρονοβόρα απο αυτην που ακολουθώ μιας και θα έπρεπε πιθανόν να σχηματιστούν αρκετά διαφορετικά μονοπάτια μέσω του A^* αλγορίθμου και να συγκριθούν μεταξύ τους.

Επιπλέον η μέθοδος που χρησιμοποίησα για την απομάκρυνση του ρομπότ απο τα εμπόδια σε περίπτωση όπου κολλήσει είναι αρκετά απλή και όχι σίγουρη απο χάρτη σε χάρτη.Για παράδειγμα σε έναν χάρτη τα 13 δευτερόλεπτα που έχει το ρομπότ στην διάθεση του για να απομακρυνθεί μπορεί να μην είναι αρκετά.Παρ'ολα αυτά με tuning κατέληξα στους συγκεκριμένους χρόνους για τους χάρτες που είχαμε στην διάθεση μας. Το συγκεκριμένο πρόβλημα αποτελεί πρακτικά κομμάτι του path planning και το σωστότερο θα ήταν να αντιμετωπιστεί σαν τέτοιο. Δηλαδή, να γίνεται έλεγχος άμα το μονοπάτι που δημιουργηθηκε απο τον A^* περνάει μέσα απο κάποιο εμπόδιο, και σε περίπτωση όπου αυτό συμβαίνει το μονοπάτι να διαφοροποιείται.

Ακόμη, δεδομένου ότι τα Lidars έχουν περιορισμένη κάλυψη $240°$ σημαίνει ότι υπάρχουν αρκετά τυφλά σημεία πίσω απο το ρομπότ. Καθώς η μεταφορική ταχύτητα μπορεί να πάρει αρνητικές τιμές σημαίνει οτί υπάρχει η πιθανότητα το ρομπότ να προσκρούσει σε εμπόδιο εξαιτίας αυτού. Για την αντιμετώπιση αυτού του προβλήματος θα μπορούσαν να αξιοποιηθούν οι μετρήσεις των sonars που αν και με μικρότερο εύρος καλύπτουν τα τυφλά σημεία των lidars.