Ευφυή Συστήματα Ρομπότ 2020

Ονοματεπώνυμο: Παπαδάμ Στέφανος
ΑΕΜ: 8885

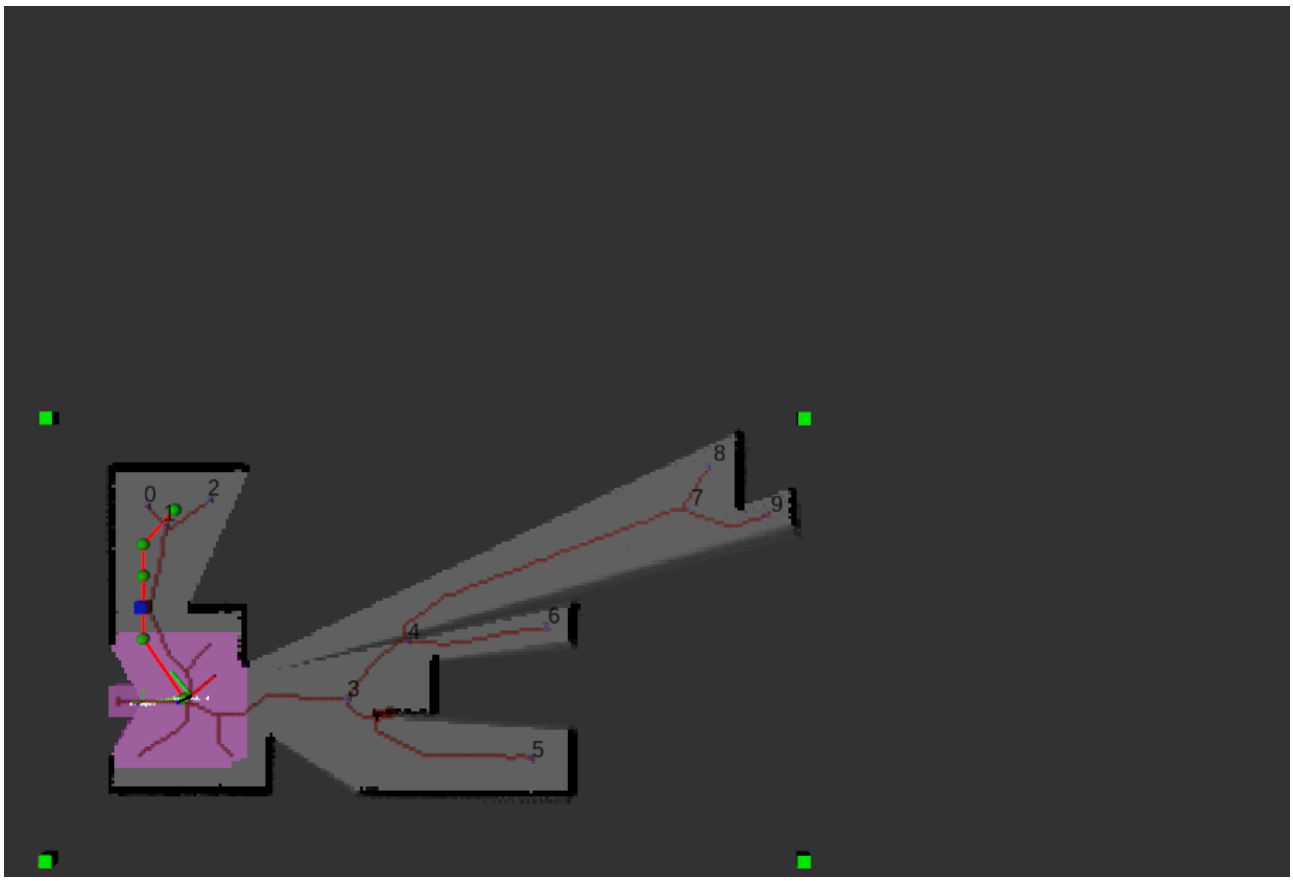
Challenge 1: *Laser-based obstacle avoidance*

Το συγκεκριμένο task υπολογίζει την γραμμική και περιστροφική ταχύτητα του ρομπότ με σκοπό το ρομπότ να περιπλανάται στο χώρο αποφεύγοντας τα εμπόδια που βρίσκει. Οι τελικές απόλυτες ταχύτητες δεν πρέπει να ξεπερνάνε τα 0.3 m/s. Η υλοποίηση αυτού του task γίνεται εντός της συνάρτησης produceSpeedsLaser(). Το ρομπότ διαθέτει έναν αισθητήρα Lidar ο οποίος λαμβάνει μετρήσεις από τις ακτίνες laser που εκπέμπει επιστρέφοντας στο πρόγραμμα μυνήματα τύπου sensor_msgs/LaserScan. Αρχικά στη συνάρτηση αρχικοποιούνται τα attributes του message σε κατάλληλες τοπικές μεταβλητές σε περίπτωση που χρησιμοποιηθούν αργότερα. Ορίζεται έπειτα μία lambda συνάρτηση map_fun η οποία μετατρέπει το εύρος (0 – μήκος του πίνακα scan) στο εύρος (angle_min – angle_max). Χρησιμοποιείται δηλαδή για να αντιστοιχεί τον δείκτη του πίνακα scan στην κατάλληλη γωνία της αντίστοιχης ακτίνας. Αρχικοποιούνται οι δύο ταχύτητες σε 0. Έπειτα, χρησιμοποείται μία for loop η οποία διατρέχει τον πίνακα με τις μετρήσεις (scan) στο εύρος 270 – 430 (προκύπτει μετά από πειράματα) οι οποίες ανταποκρίνονται στο μπροστινό μέρος του ρομπότ και έχουν μεγαλύτερη σημασία για τον υπολογισμό της γραμμικής ταχύτητας. Μέσα στη for loop αρχικά μετατρέπουμε το δείκτη i σε γωνία (rad) και υπολογίζουμε την γραμμική ταχύτητα από τον τύπο Σ(cos(θ) * scan(i)) όπου θ και scan(i), η γωνία και το μήκος της ακτίνας αντίστοιχα. Το συνημίτονο χρησιμοποιείται για να δώσει πρόσημο στον υπολογισμό της ταχύτητας του κάθε στοιχείου του αθροίσματος. Έπειτα, ελέγχουμε με μία for loop στο εύρος 200 – 500 αν το ρομπότ έχει φτάσει σε μία απόσταση μικρότερη από 0.2 m από μπροστινό εμπόδιο και θέτουμε την γραμμική ταχύτητα σε αυτη την περίπτωση ίση με 0 για να αποφύγει την σύγκρουση και στηριζόμαστε μόνο στην περιστροφική για να φύγει από το αδιέξοδο. Στη συνέχεια, με τον ίδιο τρόπο υπολογίζεται και η περιστροφική ταχύτητα με τη διαφορά ότι χρησιμοποιούμε ημίτονο για τον υπολογισμό. Έπειτα ελέγχουμε αν η γραμμική ταχύτητα είναι μηδενική πράγμα που σημαίνει ότι το ρομπότ έχει κολλήσει μπροστά από εμπόδιο και για αυτό αυξάνουμε την περιστροφική ταχύτητα για να μπορέσει να φύγει γρηγορότερα από το αδιέξοδο. Τέλος, εφαρμόζουμε στις δύο ταχύτητες την μαθηματική συνάρτηση tanh για να περιορίσουμε τις τιμές τους στο εύρος -0.3 έως 0.3 m/s και τις επιστρέφουμε. Η διαδικασία υπολογισμού των ταχυτήτων βασίστηκε στην λογική των σημειώσεων με τίτλο “Αυτόνομη εξερεύνηση και κάλυψη – Τεχνικές επιλογής στόχου” με μία παραλλαγή όπως φαίνεται άλλωστε και από τη σύγκριση των τύπων. Ενδεικτικός χάρτης που παράγεται από αυτό το ερώτημα φαίνεται στην Εικόνα 1.

Εικόνα 1: Χάρτης από την περιπλάνηση του ρομπότ πραγματοποιώντας αποφυγή εμποδίων.

Challenge 2: *Path visualization*

Το συγκεκριμένο task κάνει ορατό στο RViz το μονοπάτι που θα ακολουθήσει το ρομπότ. Η υλοποίση αυτή πραγματοποιείται στη συνάρτηση selectTarget(). Συγκεκριμένα, υπάρχει ήδη στον κώδικα μια “for loop” (for p in self.path) η οποία διατρέχει τα σημεία του μονοπατιού που θα ακολουθήσει το ρομπότ. Η μεταβλητή p περιέχει τις global συντεταγμένες x, y του κάθε σημείου του μονοπατιού. Το attribute resolution περιέχει την ανάλυση του χάρτη και το attribute origin περιέχει την απόσταση του συστήματος συντεταγμένων του χάρτη από το σύστημα συντεταγμένων του ρομπότ. Η δουλειά που κάνουμε εμείς είναι να μετατρέψουμε αυτές τις συντεταγμένες στο σύστημα συντεταγμένων του χάρτη στον RViz. Επομένως, πολλαπλασιάζουμε τις συντεταγμένες του p με το resolution και προσθέτουμε σε αυτό το origin. Ενδεικτική εικόνα του μονοπατιού που παράγεται από αυτό το ερώτημα φαίνεται στην Εικόνα 2.



Εικόνα 2: Το μονοπάτι που παράγεται στον RViz.

Challenge 3: *Path following*

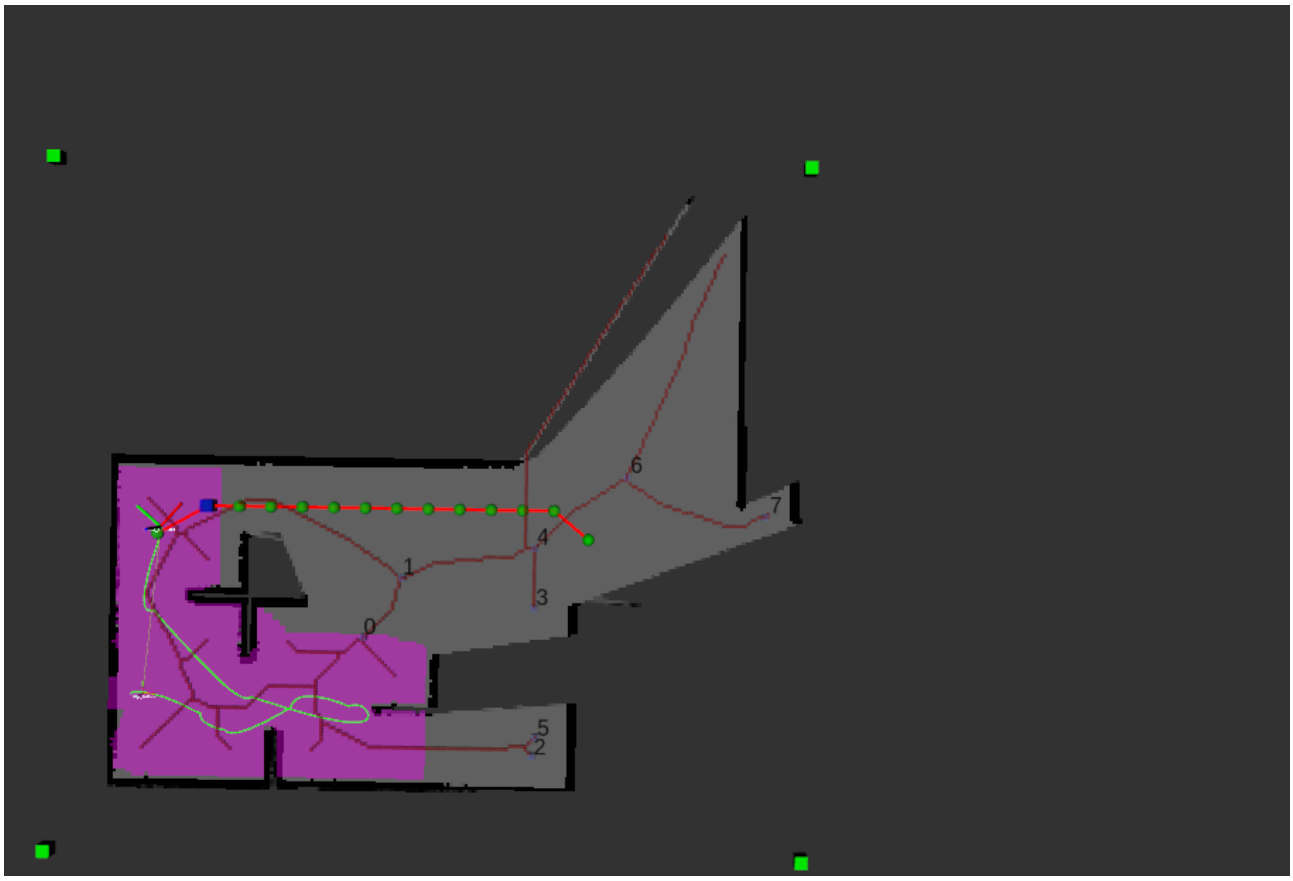
Το task αυτό παράγει τις σωστές ταχύτητες προκειμένου το ρομπότ να ακολουθεί το μονοπάτι που παράγεται. Και σε αυτή την περίπτωση οι απόλυτες ταχύτητες δεν θα πρέπει να ξεπερνάνε τα 0.3 m/s. Αρχικά, στις μεταβλητές st_x και st_y έχουν αποθηκευτεί οι συντεταγμένες του επόμενου σημείου – στόχου που πρόκειται να ακολουθήσει το ρομπότ. Εφόσον έχουμε στη μεταβλητή theta την yaw γωνία στην οποία βρίσκεται το ρομπότ θα τη χρησιμοποιήσουε για να βρούμε την θέση x, y έτσι ώστε μετά να υπολογίσουμε τη γωνία σε rad χρησιμοποιώντας τον τύπο atan2(y/x). Έπειτα στη μεταβλητή delta_theta αποθηκεύουμε τη διαφορά ανάμεσα στη γωνία του σημείου - στόχου και τη γωνία του ρομπότ. Στη συνέχεια, για τον υπολογισμό της γωνιακής και της γραμμικής ταχύτητας ακολουθείται πιστά η λογική των σημειώσεων με τίτλο “Αυτόνομη εξερεύνηση και κάλυψη – Τεχνικές επιλογής στόχου” σελίδα 93. Για να αποφύγουμε το overshoot πρόβλημα μετά από πειράματα καταλήξαμε να θέσουμε τη δύναμη n ίση με 25. Τέλος, με κατάλληλους ελέγχους περιορίζουμε τις ταχύτητες στο εύρος που αναφέρθηκε παραπάνω. Σύμφωνα με τις σημειώσεις η γωνιακή ταχύτητα θα έπρεπε να πολλαπλασιαστεί με 0.3 δηλαδή την μέγιστη τιμή της αλλά το αποτέλεσμα ήταν πολύ αργό οπότε αφέθηκε όπως ήταν.

Challenge 4: *Path following & obstacle avoidance*

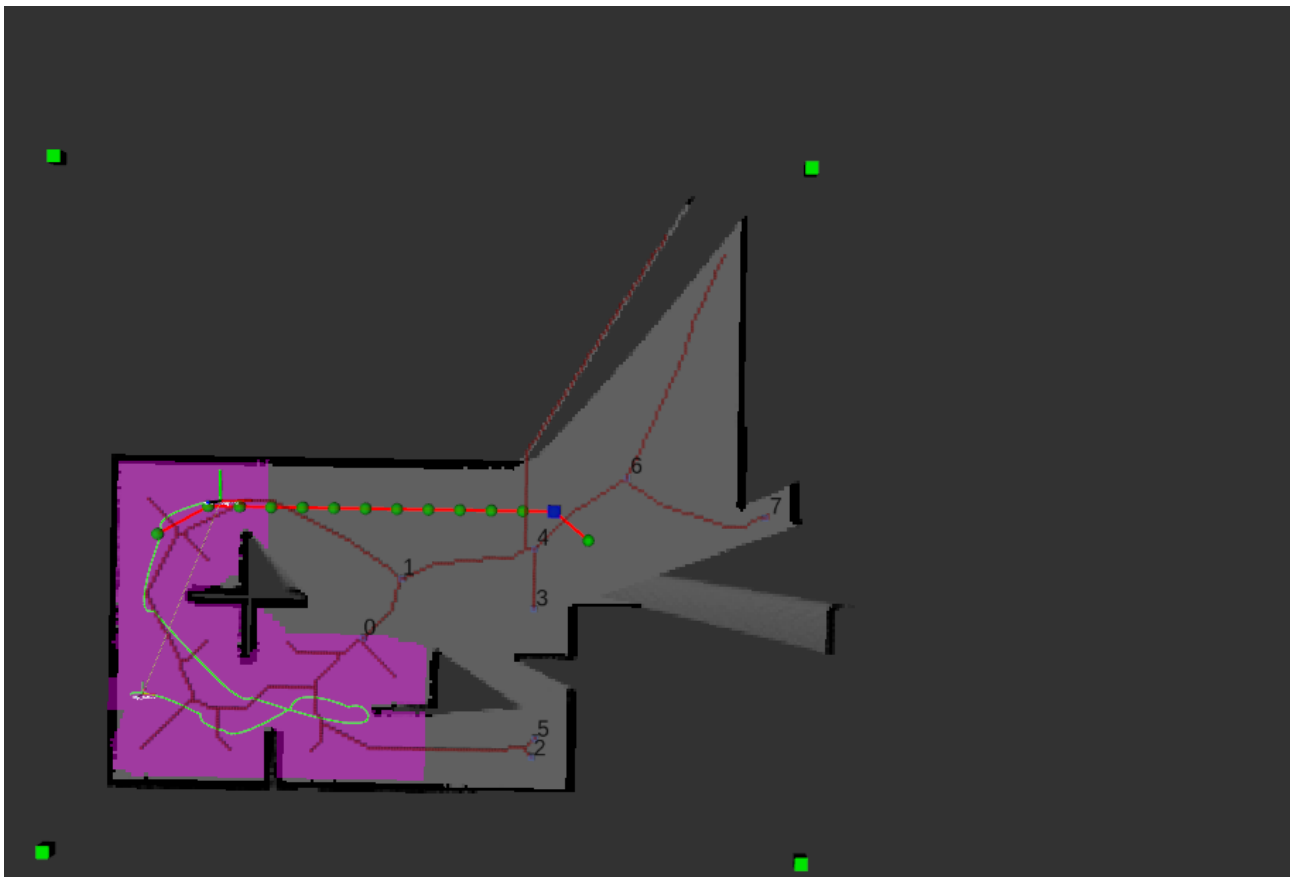
Σε αυτό το task θα πρέπει να υλοποιηθεί ένα συνδυαστικό μοντέλο για την υπολογισμό των ταχυτήτων δηλαδή να συνδυάζει τις ταχύτητες που υπολογίστηκαν για την υλοποίηση του obstacle avoidance και του path following. Στην υλοποίηση μας χρησιμοποιούμε δύο περιπτώσεις οι οποίες χωρίζονται από μια δομή if/else. Η γενική περίπτωση είναι αυτή όπου η γραμμική ταχύτητα που παράγεται από τον υπολογισμό του obstacle avoidance δεν είναι μηδενική. Σε αυτή την περίπτωση ορίζουμε δύο παραμέτρους parameter_1 και parameter_2 οι οποίες ορίζουν το βαθμό συμμετοχής της κάθε ταχύτητας στον συνολικό υπολογισμό. Η γραμμική και η περιστροφική ταχύτητα ορίζονται σαν ο σταθμισμένος μέσος των δύο ταχυτήτων. Στη γενική περίπτωση η ταχύτητα του path following συμμετέχει με βαθμό 0.8 ενώ η ταχύτητα του obstacle avoidance με 0.2. Επομένως, δίνουμε μεγαλύτερη προτεραιότητα στην ακολούθηση του μονοπατιού. Οι τιμές αυτές αποφασίστηκαν μετά από δοκιμές. Η δεύτερη περίπτωση είναι αυτή όπου η γραμμική ταχύτητα του obstacle avoidance είναι μηδενική. Σε αυτή την περίπτωση το ρομπότ βρίσκεται μπροστά από κάποιο εμπόδιο και θα πρέπει να αποφευχθεί η σύγκρουση. Για τον λόγο αυτό δίνουμε τιμές 0 και 1 στις δύο παραμέτρους λαμβάνοντας έτσι υπόψιν μόνο την ταχύτητα του obstacle avoidance για να αποφύγουμε τη σύγκρουση.

Challenge 5: *Smarter subgoal checking*

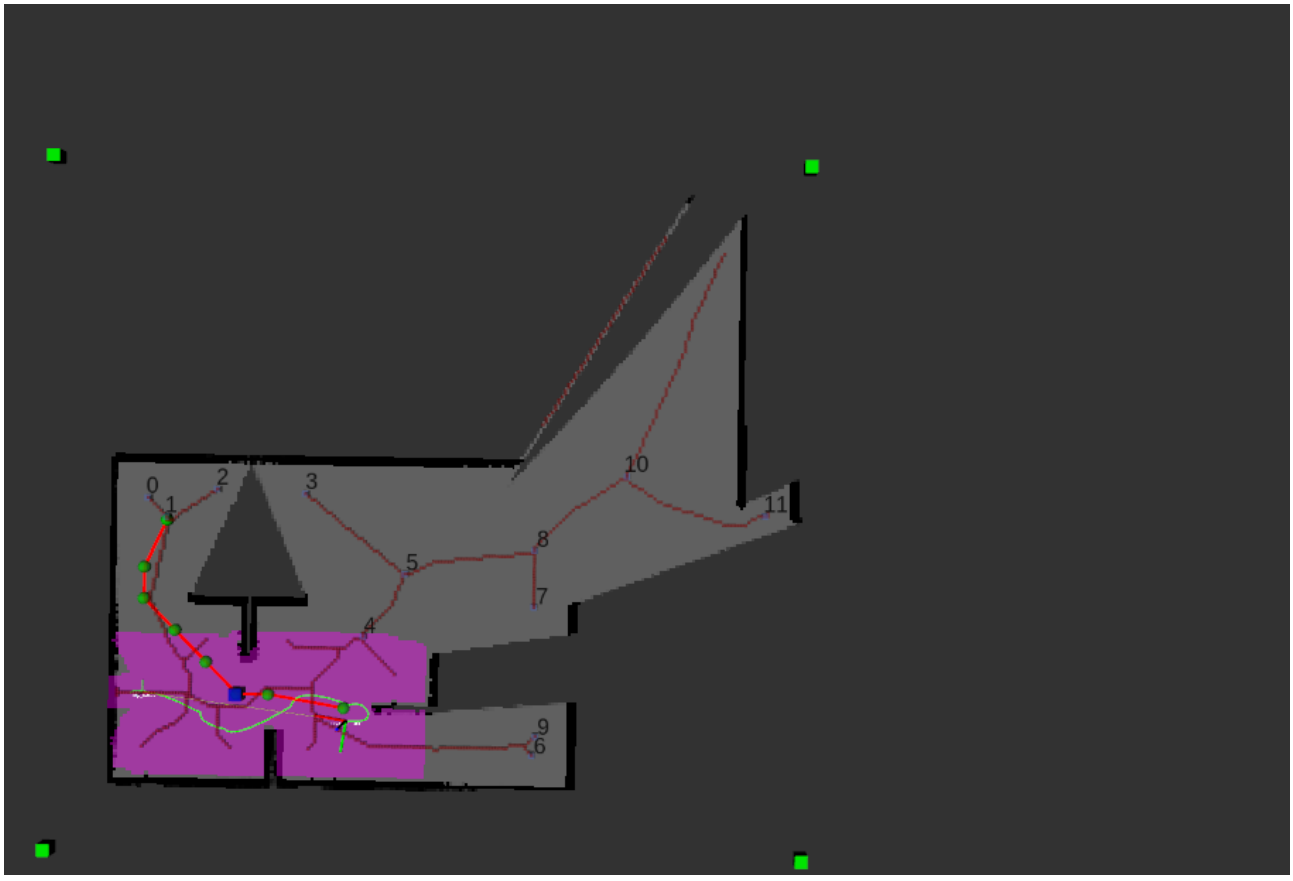
Το συγκεκριμένο task υλοποιεί έναν εξυπνότερο τρόπο επιλογής του επόμενου subgoal που θα ακολουθήσει το ρομπότ. Αρχικά, διατρέχουμε όλα τα subtargets με μία δομή “for loop” για να υπολογίσουμε τις κλίσεις των σημείων ανά δύο. Με αυτό τον τρόπο θα μπορούμε να ελέγχουμε πόσα subtargets ανήκουν στην ίδια ευθεία το οποίο σημαίνει ότι το ρομπότ θα μπορεί να τα προσπεράσει και να μεταβεί κατευθείαν στο τελυταίο σημεό της ευθείας. Κρατάμε στις μεταβλητές slope1 και slope2 τις κλίσεις τριών διαδοχικών σημείων και έπειτα ελέγχουμε τη διαφορά τους. Σε περίπτωση που αυτή η διαφορά είναι μικρότερη από 0.3 θεωρούμε ότι ανήκουν σχεδόν στην ίδια ευθεία οπότε το ρομπότ μπορεί να διανύσει την απόσταση με ασφάλεια. Οι εικόνες 3 και 4 δείχνουν την εφαρμογή αυτής της λογικής. Επίσης, παίρνουμε μία μέτρηση από το laser για να ελέγξουμε αν υπάρχει εμπόδιο στο μπροστινό μέρος του οχήματος. Αυτό γίνεται σε περίπτωση που τα subtargets δεν ανήκουν στην ίδια ευθεία αλλά δεν υπάρχει και κάποιο εμπρόσθιο εμπόδιο στο ρομπότ οπότε διανύει τα subtargets ανά δύο. Η εικόνα 5 δείχνει την εφαρμογή αυτής της λογικής.



Εικόνα 3: Επιλογή ακολούθησης πρώτου subtarget.



Εικόνα 4: Επιλογή ακολούθησης δωδέκατου subtarget εφόσον όλα τα ενδιάμεσα σημεία ανήκουν στην ίδια ευθεία.



Εικόνα 5: Επιλογή ακολούθησης δεύτερου subtarget εφόσον ανάμεσα στο πρώτο και το δεύτερο δεν υπάρχει κάποιο εμπόδιο.