Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Πανεπιστήμιο Θεσσαλίας

Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών
Ηλεκτρονικών Υπολογιστών

Αναφορά ειδικού θέματος χειμερινού εξαμήνου

ROOM MAPPING USING RC CAR CONTROLLED BY RASPBERRY PI

Φοιτητές:

Νικολάου Δημήτριος dimnikol@inf.uth.gr

Παναγιώτου Δημήτριος dimipana@inf.uth.gr

Επιβλέπων Καθηγητής:

Μπέλλας Νικόλαος

Σεπτέμβριος 2016

ΠΕΡΙΛΗΨΗ

Αντικείμενο του ειδικού θέματος είναι η χαρτογράφηση αγνώστου χώρου από τηλεκατευθυνόμενο όχημα τόσο στο δισδιάστατο επίπεδο, όσο και στο τρισδιάστατο, αντλώντας πληροφορίες από μία μόνο κοινή web camera. Αρχικά δημιουργήθηκε το όχημα και οι απαραίτητες για την τροφοδοσία του συνδεσμολογίες. Ακολούθησε ο προγραμματισμός για τη σωστή κίνηση του οχήματος και τα πρώτα πειράματα λειτουργίας του συνόλου του εξοπλισμού. Έπειτα πειραματιστήκαμε πάνω στο τρισδιάστατο επίπεδο, με χρήση ζευγων φωτογραφιών και επεξεργασία τους μέσω του εργαλείου OpenCV. Ωστόσο τα πειράματα και τα αποτελέσματα υπό τις συγκεκριμένες συνθήκες δεν κρίθηκαν αρκετά ικανοποιητικά. Για το λόγο αυτό η τελική υλοποίηση έγινε πάνω στο δισδιάστατο χώρο, όπου αντλούμε πληροφορίες από φωτογραφίες και τις αποτυπώνουμε σε ένα χάρτη δύο διαστάσεων, με τη μορφή κάτοψης, χαρτογραφώντας έτσι το χώρο και τα τυχόν εμπόδια που το όχημα συνάντησε.

ΕΙΣΑΓΩΓΗ

Βασικός στόχος της εργασίας μας ήταν να μελετήσουμε κατα πόσο είναι δυνατή η χαρτογράφηση αγνώστου χώρου, με τη χρήση μιας απλής web camera. Οι προσπάθειες μας αρχικά αφορούσαν την τρισδιάστατη αναπαράσταση, που θα αποτελούσε και την πληρέστερη αντιστοιχία με τον πραγματικό χώρο. Ο αλγόριθμός μας βγάζει ζεύγη φωτογραφιών, μετακινώντας ελαφρώς τη θέση του ρομπότ μετά την πρώτη λήψη. Τα ζεύγη αυτά υφίστανται επεξεργασία με τη βοήθεια συναρτήσεων της ανοιχτής βιβλιοθήκης OpenCV. Η έξοδος του προγράμματός μας είναι ένα αρχείο 3D το οποίο και βλέπουμε με τη βοήθεια του προγράμματος MeshLab. Ωστόσο, η αδυναμία του εξοπλισμού μας, καθώς και κάποια ζητήματα αποτελεσματικότητας αλγορίθμων μείωσαν τη δραστικότητα του τρόπου αυτού.
Για το λόγο αυτό, η τελική μας προσέγγιση αναφέρεται στο δισδιάστατο επίπεδο. Το ρομπότ μετακινείται από απόσταση και βγάζει φωτογραφίες κατά βούληση του χρήστη. Οι φωτογραφίες αυτές, αφού περάσουν μια σειρά από φίλτρα εικόνων, εξάγουν τα εμπόδια που υπάρχουν εντός της κοντινής περιοχής λήψης της κάμερας. Μόλις ολοκληρωθεί η κίνηση του ρομπότ και η λήψη φωτογραφιών, όλα τα σημεία εμπόδια αποτυπώνονται με ικανοποιητική ακρίβεια στο επίπεδο με μορφή κάτοψης χώρου, η οποία και αποθηκεύεται.

ΜΕΡΟΣ 1ο : ΜΗΧΑΝΙΚΑ ΜΕΡΗ ΤΟΥ ΡΟΜΠΟΤ, ΑΡΧΙΤΕΚΤΟΝΙΚΗ ΣΥΣΤΗΜΑΤΟΣ ΚΑΙ ΣΥΝΔΕΣΜΟΛΟΓΙΑ

1.1) Μηχανικά Μέρη και μονάδες συστήματος

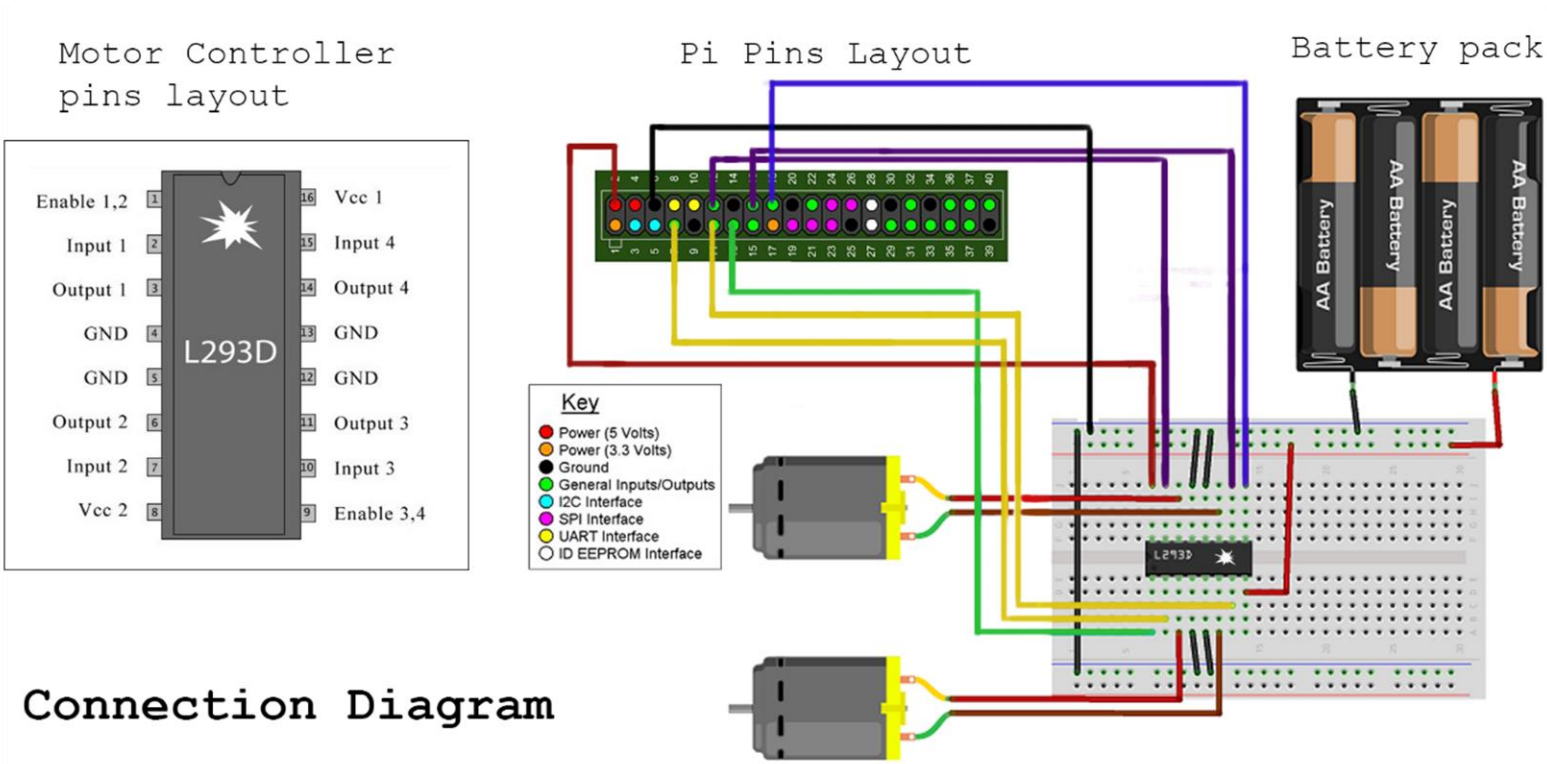
Το ρομπότ αποτελείται από τα εξής μηχανικά μέρη:

- Shadow chassis εξωτερικό περίβλημα
- 2x Rubber wheels
- Breadboard Mini
- Jumper Wires
- 2x DC Gear Motor με χαρακτηριστικά:
    - RPM: 125r / minute
    - Current: 80 - 100mA
    - Reduction: 48:1
    - Output Torque: 0.8kg / cm
    - Dimensions: 70x22x18mm
    - Voltage: 3V
- L293D Motor Controller για να ελέγχει την κίνηση των motors
- 4 Battery Pack για την τροφοδοσία του Motor Controller

Ενώ το Raspberry που χρησιμοποιήθηκε είναι τύπου Raspberry Pi 2 Model B Με λειτουργικό Ubuntu Raspbian Jessie. Το Pi τέλος, τροφοδοτείται από ένα Power Bank ZALMAN ZM-PB84IW (8400MAH)

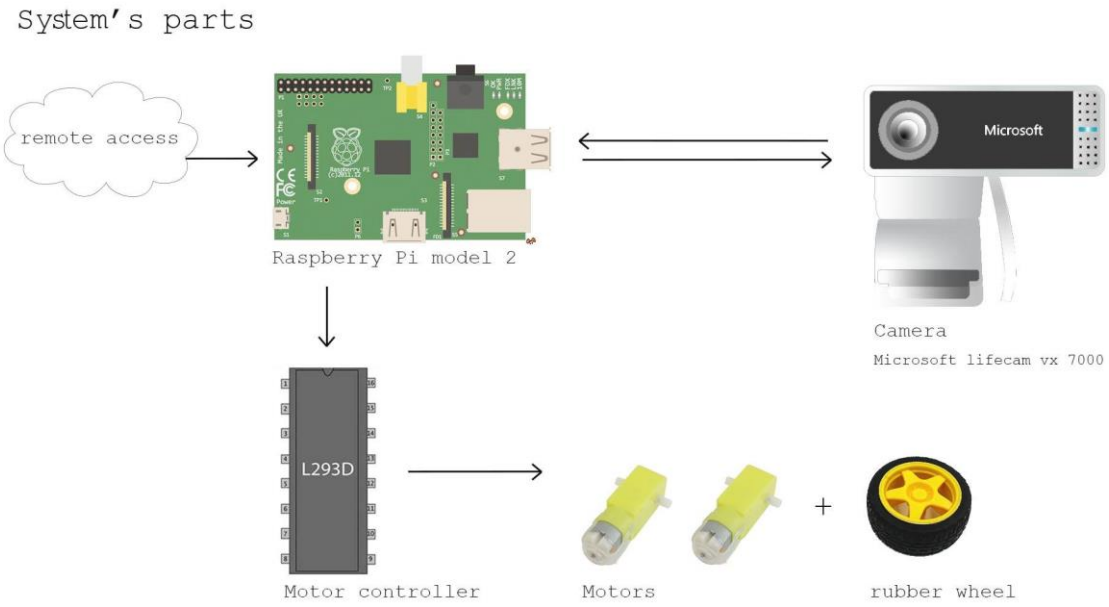
Για τη λήψη φωτογραφιών χρησιμοποιήθηκε μια web camera Microsoft VX7000.

Στην εικόνα 1 φαίνονται τα pin layouts του pi και του motor controller, καθώς και η συγκεκριμένη συνδεσμολογία που χρησιμοποιήσαμε.

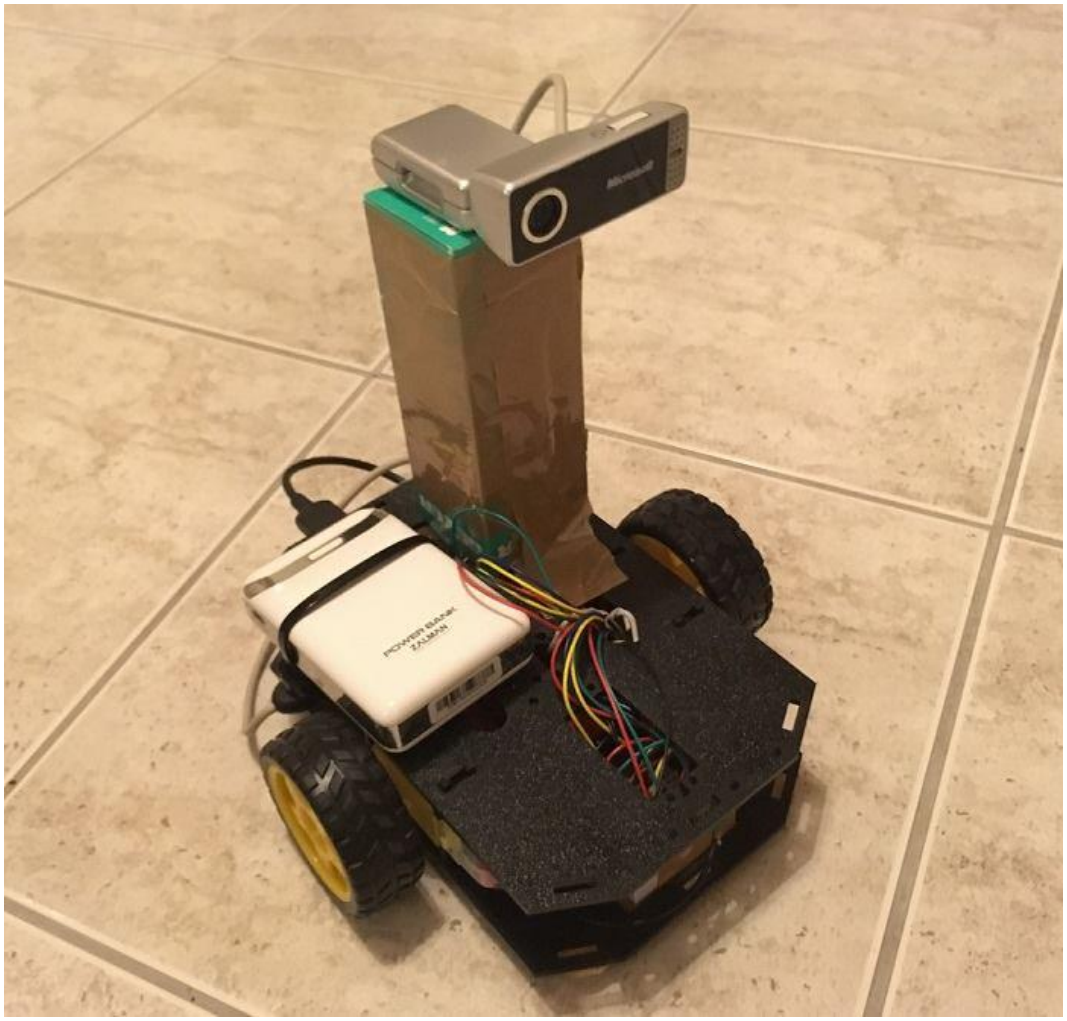


Εικόνα 1

1.2) **Αρχιτεκτονική Συστήματος**Το σύστημά μας αποτελείται ουσιαστικά από το Raspberry pi , την κάμερα, το motor controller, τα motors και τον χειριστή του Pi. Συγκεκριμένα η αρχιτεκτονική του συστήματος φαίνεται στην παρακάτω εικόνα .



Εικόνα 2 Πάνω: αρχιτεκτονική συστήματος Κατω: εικόνα του ρομπότ



1.3) ΜΕΡΟΣ 2^{ο} : ΤΡΙΣΔΙΑΣΤΑΤΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

2.1) ΥΛΟΠΟΙΗΣΗ

Η πρώτη μας υλοποίηση, που ταυτόχρονα είναι και η βέλτιστη, ήταν να αναπαραστήσουμε το χώρο στις 3 διαστάσεις. Για να το καταφέρουμε αυτό, ακολουθήσαμε τον ακόλουθο αλγόριθμο:

- Αρχικά το ρομπότ βγάζει μια φωτογραφία από την αρχική του θέση κοιτώντας προς μια τυχαία διεύθυνση
- Μετά το ρομπότ μετακινείται στο οριζόντιο επίπεδο κατα 6 εκατοστά, όσο δηλαδη είναι περίπου και η απόσταση μεταξύ των ματιών μας. Έπειτα ξαναβγάζει φωτογραφία.

- Το ζευγάρι αυτό φωτογραφιών περνάει από την εξής διαδικασία: Αρχικά γίνεται downsampling στις φωτογραφίες με τη χρήση της συνάρτησης pyrdown της βιβλιοθήκης OpenCV. Αυτό το βήμα γίνεται για να βελτιώσει το χρόνο των επόμενων πράξεων, αφού πρακτικά η ανάλυση δε χαλάει αρκετά για να αλλοιώσει το αποτέλεσμα.
  Έπειτα υπολογίζεται το disparity map και το disparity απο τις δυο αυτές φωτογραφίες χρησιμοποιώντας τις συναρτήσεις steroSGBM_create και stereo.compute αντίστοιχα.
  Η συνάρτηση stereoSGBM (Semi-global block matching) αντιστοιχεί blocks (και όχι pixels) των δύο φωτογραφιών και ως έξοδο επιστρέφει ένα αντικείμενο τύπου stereoSGBM, το οποίο με τη χρήση της συνάρτησης stereo.compute παίρνουμε τελικά τη φωτογραφία με το disparity, δηλαδή μια ασπρόμαυρη εικόνα όπου όσο πιο λευκό είναι κάθε pixel τόσο πιο κοντά βρίσκεται στην κάμερα.

- Τέλος, από το disparity map που φτιάξαμε και λαμβάνοντας υπόψην την εστιακή απόσταση(focal length) της κάμερας καθώς και την αριστερή αρχική εικόνα δημιουργούμε το τελικό 3d point cloud που αντιστοιχεί κάθε σημείο στο τρισδιάστατο επίπεδο. Όλα τα σημεία αποθηκεύονται σε αρχείο τύπου PLY(polygon file format) το οποίο ανοίγουμε με τη χρήση του προγράμματος MeshLab.

- Έπειτα το ρομπότ στρίβει και η διαδικασία επαναλαμβάνεται για όσες φορές θεωρήσουμε οτι αρκούν για να βγάλουν μια σαφή ένδειξη για το χώρο.

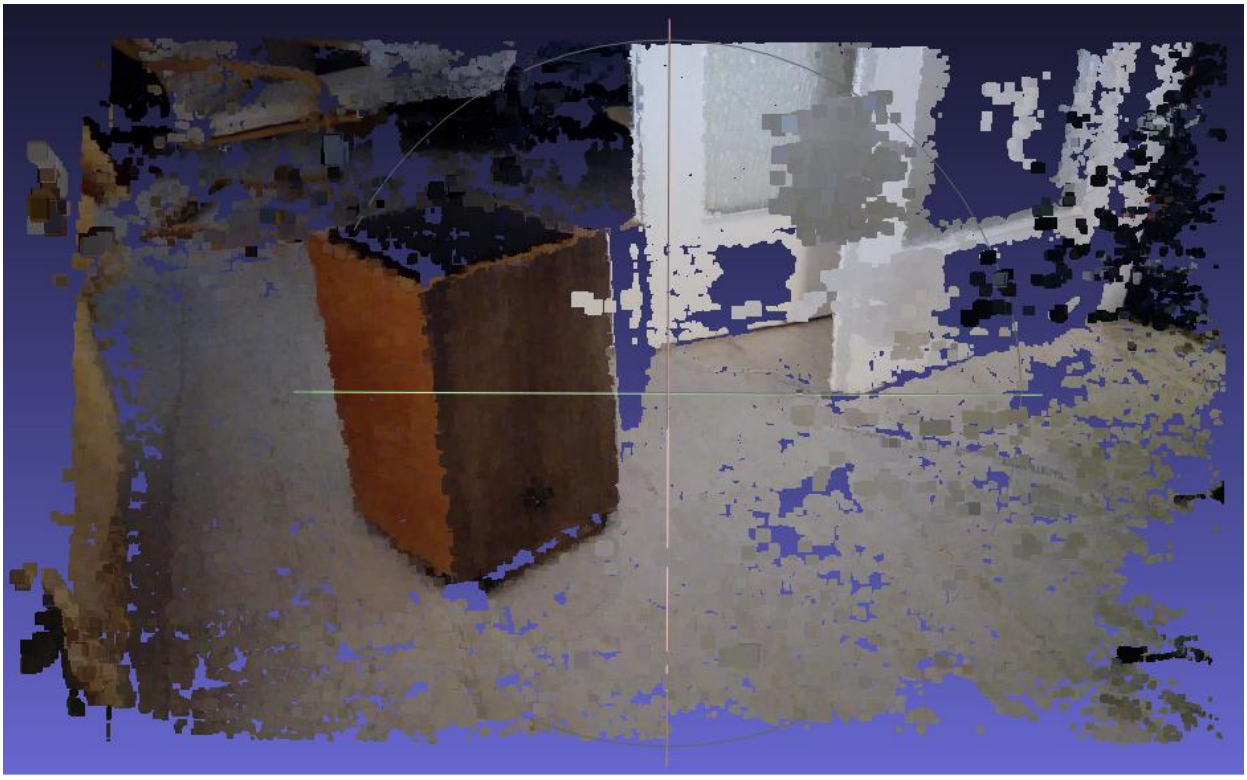
2.2) ΠΕΙΡΑΜΑΤΑ ΚΑΙ ΑΠΟΤΕΛΕΣΜΑΤΑ

Όπως φαίνεται και στις εικόνες 3-10, παρ'όλο που ο αλγόριθμος λειτούργησε αρκετά ικανοποιητικά πριν τοποθετήσουμε την κάμερα στο όχημα, μόλις η κάμερα τοποθετήθε η αποτελεσματικότητά του έπεσε κατακόρυφα.
Το πρόβλημα έγκειται στο ότι όταν η λήψη είναι από ψηλή θέση, υπάρχει μεγαλύτερη πληροφορία σχετικά με το βάθος κάποιων αντικειμένων και έτσι το disparity map είναι αρκετά ακριβές. Όταν όμως η λήψη γίνεται από τόσο χαμηλά, η πληροφορία αυτή έχει μειωθει με αποτελεσμα να υπάρχουν αρκετά κενά στο map.
Ένα δεύτερο πρόβλημα ήταν η ακρίβεια κίνησης του ρομπότ. Ο αλγόριθμος που χρησιμοποιήσαμε είναι αρκετά ευαίσθητος και για να λειτουργήσει με μία μονο κάμερα, πρέπει το ρομπότ να μετακινηθεί αποκλειστικά οριζόντια και για ακριβώς έξι εκατοστά. Το ρομπότ όμως, λόγω βάρους αλλά και δύο μόνο τροχών δεν έχει την ακρίβεια κίνησης που θα θέλαμε και έτσι προκαλεί σχετικά μεγάλα errors αποστάσεων. Στις παρακάτω εικόνες φαίνεται ξεκάθαρα το αποτέλεσμα του αλγορίθμου, όπου συγκρίνουμε το αποτέλεσμα απο μια λήψη απο ψηλα(εικόνες 3-6, με το αποτέλεσμα απο λήψη πάνω στο ρομπότ (εικόνες 7-10).

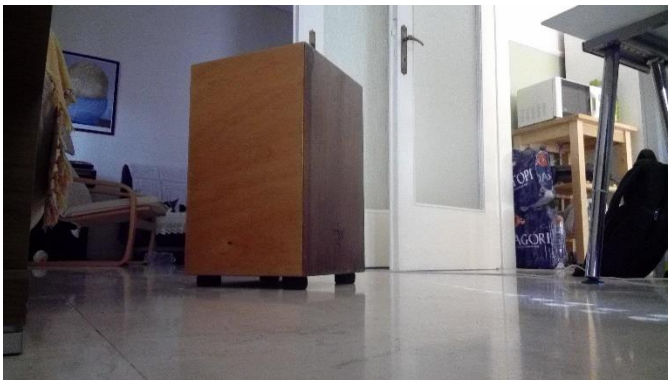
Εικόνα 3 Αριστερή φωτογραφία

Εικόνα 4 Δεξιά φωτογραφία

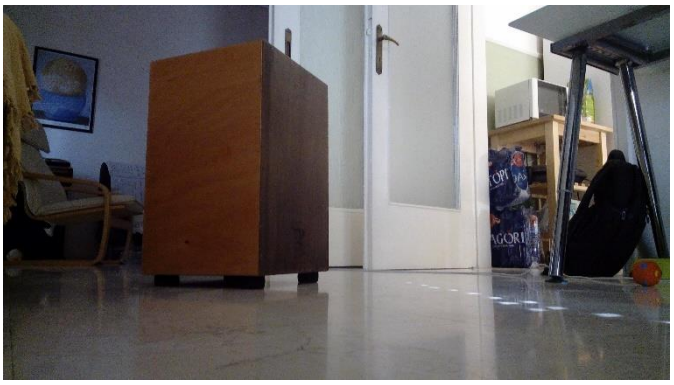
Εικόνα 4 Disparity map



Εικόνα 6 Τελική έξοδος αλγορίθμου.



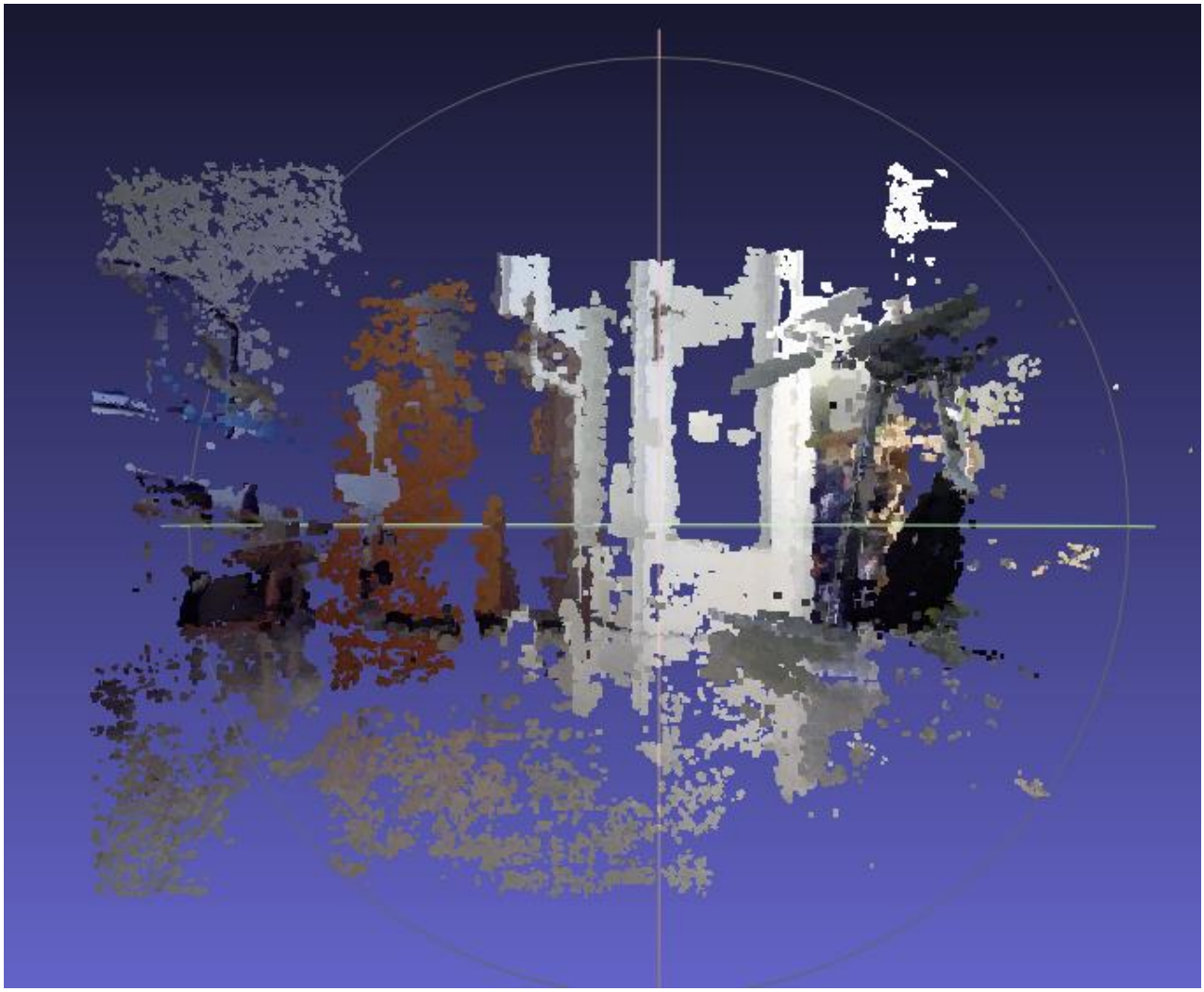
Εικόνα 7: Αριστερή φωτογραφία από λήψη ρομπότ



Εικόνα 8: Δεξιά φωτογραφία από λήψη ρομπότ

Εικόνα 9: Disparity map απο λήψη ρομπότ

Εικόνα 10: Τελική έξοδος αλγορίθμου για είσοδο από λήψη ρομπότ.



ΜΕΡΟΣ 3ο : ΔΙΣΔΙΑΣΤΑΤΗ ΑΝΑΠΑΡΑΣΤΑΣΗ

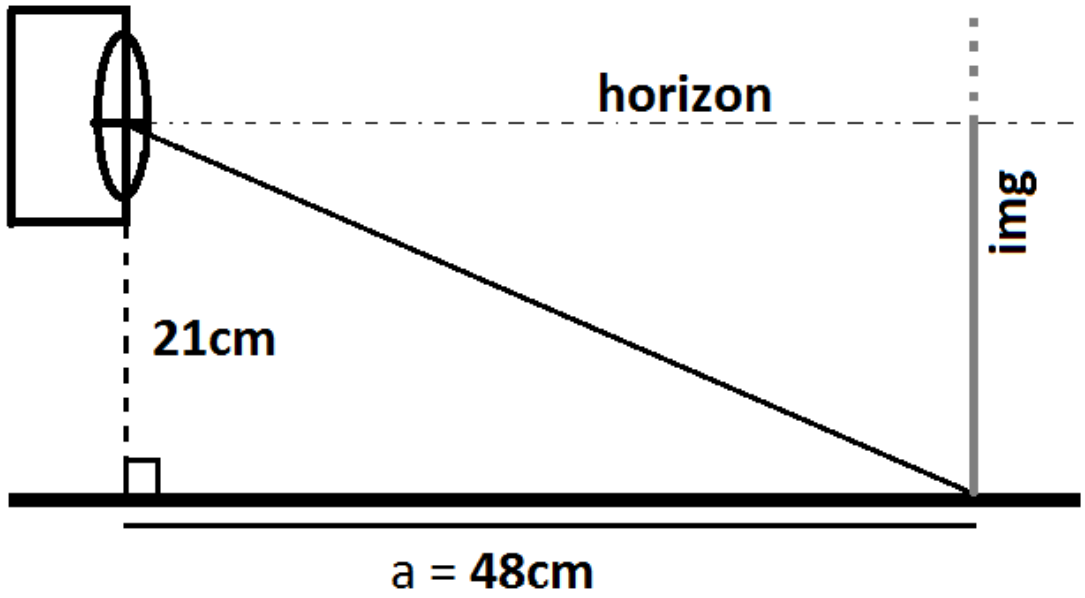
3.1) Η ΙΔΕΑ

Η αναγνώριση βάθους σε μη ιδανικές και άγνωστες συνθήκες απαιτεί είτε δύο κάμερες ή ζεύγη εικόνων. Όμως σε εξιδανικευμένες συνθήκες μπορεί να επιτευχθεί μία βασική αναγνώριση βάθους, ακόμα και με μονές φωτογραφίες. Οι παραδοχές που γίνονται για να δουλέψει αυτή η μεθοδολογία είναι οι παρακάτω:

- Πρέπει να γνωρίζουμε την απόσταση της κάμερας από το έδαφος
- Πρέπει να γνωρίζουμε την κλίση της κάμερας
- Πρέπει το έδαφος να είναι επίπεδο και χωρίς κλίση
- Τέλος, πρέπει να γνωρίζουμε πόσο απέχει από την κάμερα το χαμηλότερο σημείο της φωτογραφίας

Στην δική μας περίπτωση ισχύει ότι:

- Η κάμερα βρίσκεται σε ύψος 21 cm
- Η κάμερα είναι παράλληλη με το έδαφος που σημαίνει ότι ο ορίζοντας βρίσκεται ακριβώς στη μέση της φωτογραφίας
- Η απόσταση του χαμηλότερου σημείου της φωτογραφίας είναι 48 cm



***Εικόνα 3.1**: Η θέση της κάμερας στον χώρο*

Με δεδομένα τα παραπάνω, ο αλγόριθμος εφαρμόζει διάφορα φίλτρα στην φωτογραφία και χρησιμοποιεί βασικές τριγωνομετρικές ταυτότητες για να υπολογίσει τις αποστάσεις. Λειτουργεί ως εξής:

1) Βγάζει φωτογραφία και εφαρμόζει τα φίλτρα ώστε να ξεχωρίσουν τα εμπόδια

2) Σκανάρει την εικόνα από κάτω προς τα πάνω και από αριστερά προς τα δεξιά

3) Μόλις εντοπίσει ένα εμπόδιο τότε υπολογίζει την απόσταση και την γωνία του εμποδίου

4) Το σκανάρισμα συνεχίζεται μέχρι ο υπολογισμός να φτάσει στο δεξί όριο της εικόνας

5) Στο τέλος εξάγεται μια λίστα με τις αποστάσεις και τις γωνίες όλων των εμποδίων

6) Ο χειριστής μπορεί να επιλέξει την μετακίνηση του ρομπότ και την επανάληψη της παραπάνω διαδικασίας με νέα φωτογραφία από άλλο σημείο

7) Αλλιώς ο χειριστής μπορεί να επιλέξει τον τερματισμό του αλγορίθμου. Τότε οι λίστες με τα εμπόδια ζωγραφίζονται σε ένα τετράγωνο που αποτελεί και την κάτοψη του χώρου

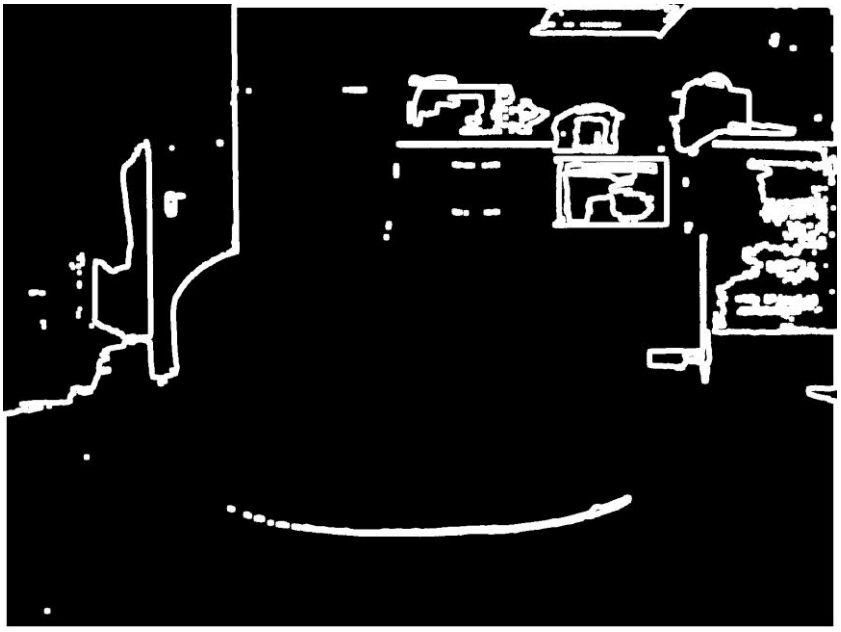
3.2) ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ – ΕΦΑΡΜΟΓΗ ΦΙΛΤΡΩΝ

Για να ξεχωρίσουμε τα εμπόδια σε μια φωτογραφία χρησιμοποιούμε τα παρακάτω φίλτρα.

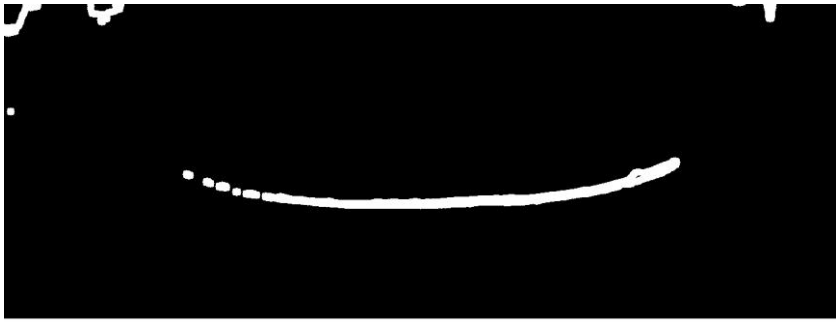
- Η εικόνα από έγχρωμη μετατρέπεται σε ασπρόμαρη
- Εφαρμόζεται ένα φίλτρο **threshold** για edge detection
- Αποκόπτεται ένα μέρος της εικόνας για πιο γρήγορους και ακριβείς υπολογισμούς

Οι παρακάτω εικόνες δείχνουν την διαδικασία της επεξεργασίας εικόνας:

Εφαρμογή φίλτρου gray



Εφαρμογή edge detection

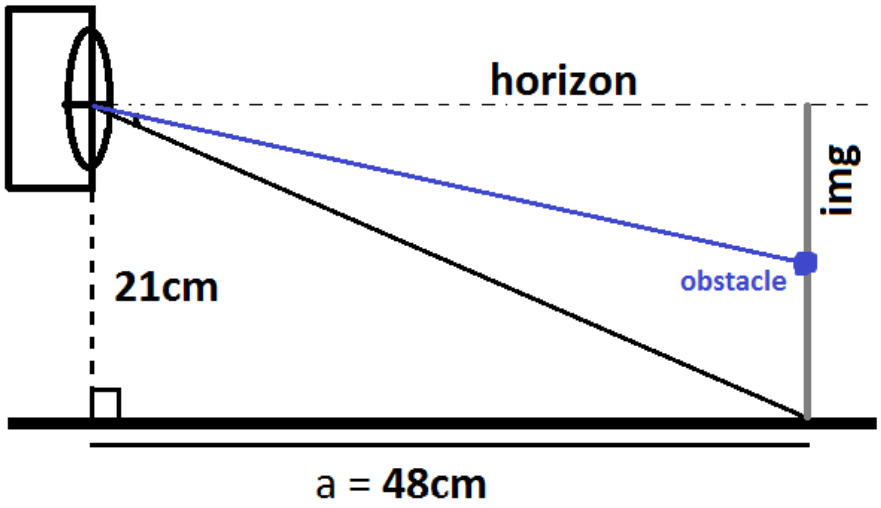


Εφαρμογή image crop

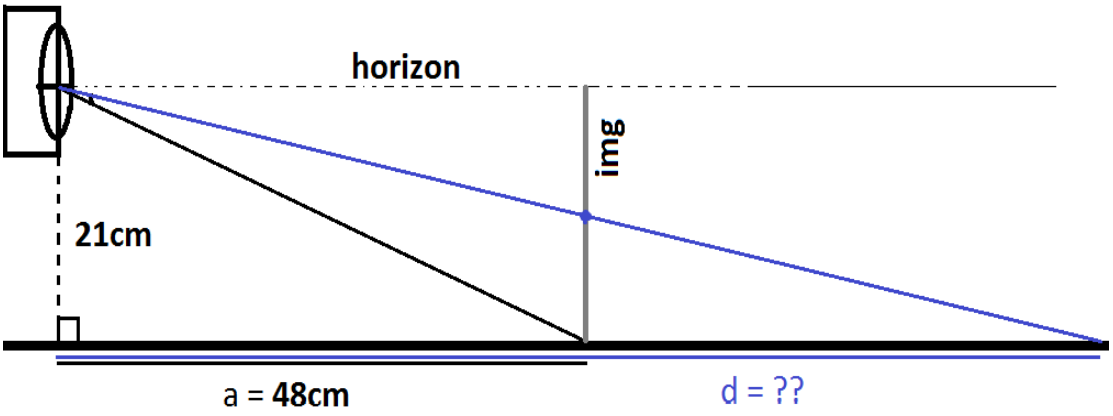
3.3) ΥΠΟΛΟΓΙΣΜΟΣ ΑΠΟΣΤΑΣΗΣ – ΓΩΝΙΑΣ ΕΜΠΟΔΙΩΝ

Χάρη στις παραδοχές που κάναμε για τη θέση της κάμερας, μπορούμε να χρησιμοποιήσουμε απλές τριγωνομετρικές ταυτότητες για να υπολογίσουμε την απόσταση και την γωνία των εμποδίων με ικανοποιητική ακρίβεια. Παρακάτω περιγράφεται λεπτομερώς η διαδικασία υπολογισμού.

Έστω ότι έχουμε μια φωτογραφία που απεικονίζει ένα εμπόδιο. Μπορούμε να ανάγουμε το πρόβλημα στην απλούστερη διδιάστατη έκδοσή του, όπως φαίνεται στην επόμενη εικόνα.

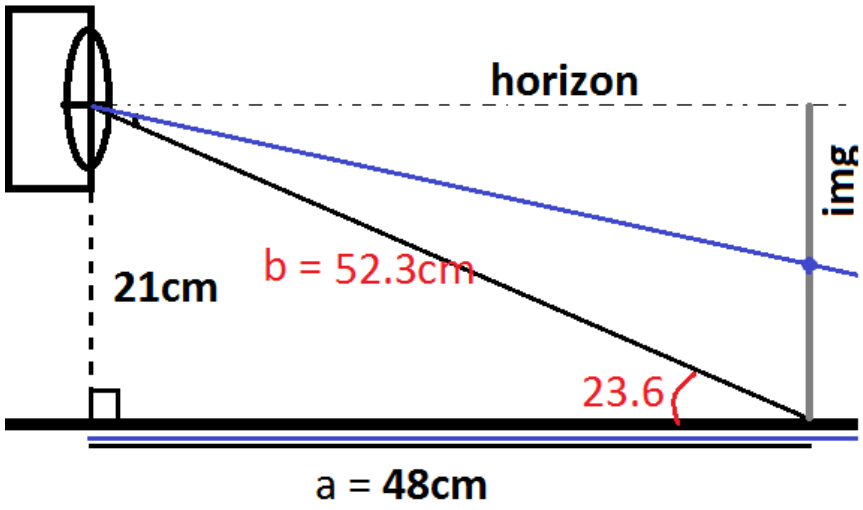


Οπότε εμείς ψάχνουμε την απόσταση **d,** η οποία είναι η απόσταση του εμποδίου απ’ την βάση της κάμερας. Αυτό δείχνει η παρακάτω εικόνα.

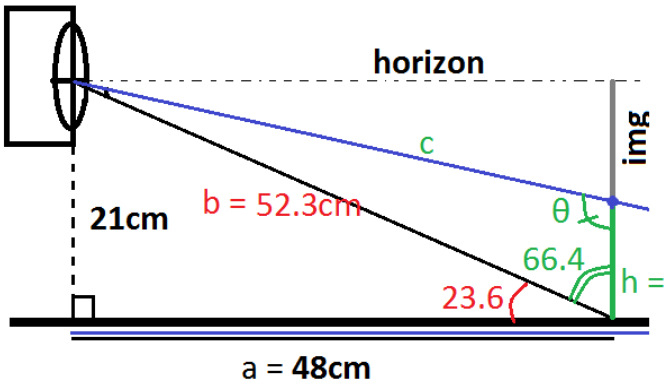


Για να βρούμε το **d** λοιπόν ακολουθούμε τα παρακάτω βήματα.

- Βρίσκουμε ό,τι λείπει απ' το αρχικό τρίγωνο



- Λύνουμε το δεύτερο τρίγωνο



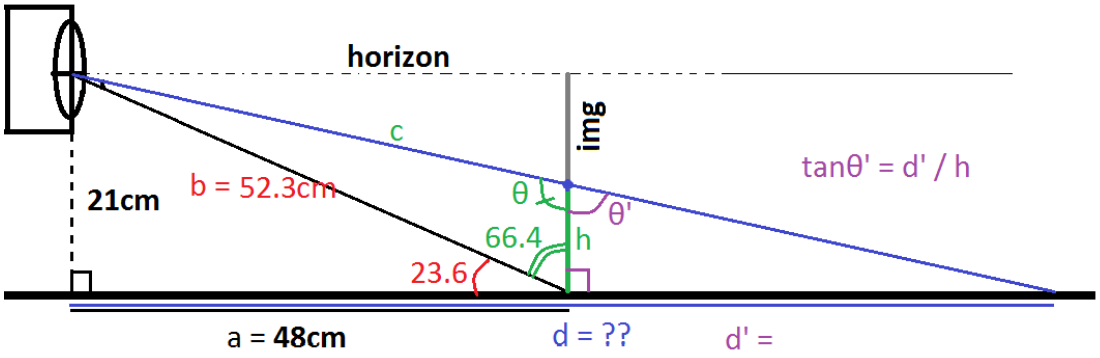
Law of cosines:

$c^2 = a^2 + h^2 - 2ah*\cos(66.4)$

Law of sines:

$\sin\theta / b = \sin(66.4) / c$

- Λύνουμε το δεξί τρίγωνο όπου βρίσκουμε το **d'**



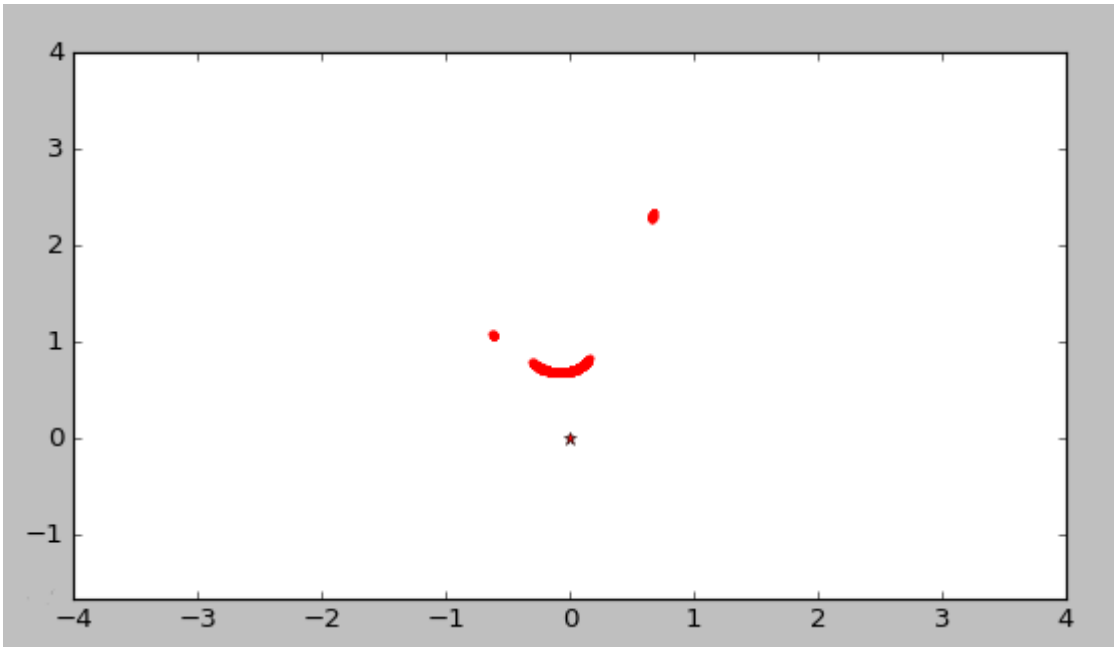
$\tan\theta' = d' / h$

- Η τελική απόσταση είναι το **d' + a**

Με παρόμοιο τρόπο αλλά χρησιμοποιώντας τα οριζόντια pixel υπολογίζουμε την γωνία των εμποδίων (δηλαδή το πόσο αριστερά ή δεξιά βρίσκεται το εμπόδιο σε σχέση με το ρομπότ).

3.4) 2D ΑΠΕΙΚΟΝΙΣΗ ΤΩΝ ΕΜΠΟΔΙΩΝ

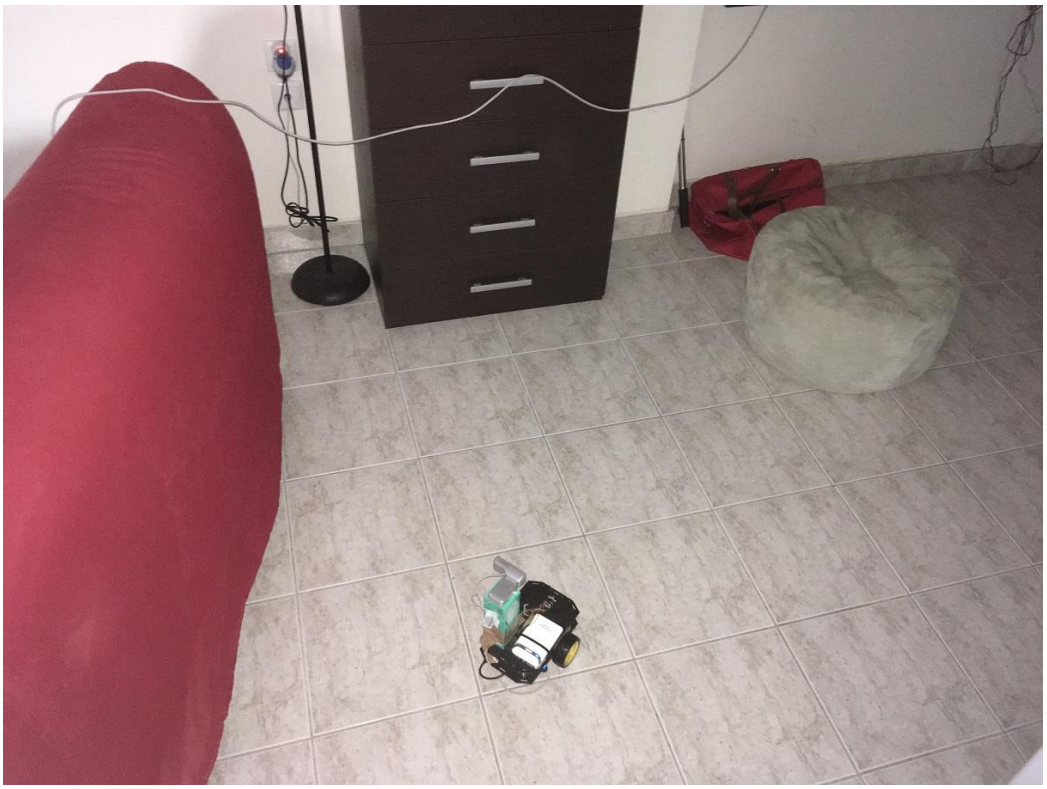
Στο τέλος ο αλγόριθμος παίρνει τις λίστες με τα εμπόδια και τα απεικονίζει σε ένα διάγραμμα. Παρακάτω φαίνεται το διάγραμμα που αντιστοιχεί στην προηγούμενη φωτογραφία. Οι άξονες του διαγράμματος δείχνουν την απόσταση σε μέτρα. Με κόκκινο χρώμα φαίνονται τα εμπόδια ενώ το αστεράκι απεικονίζει τη θέση του ρομπότ.

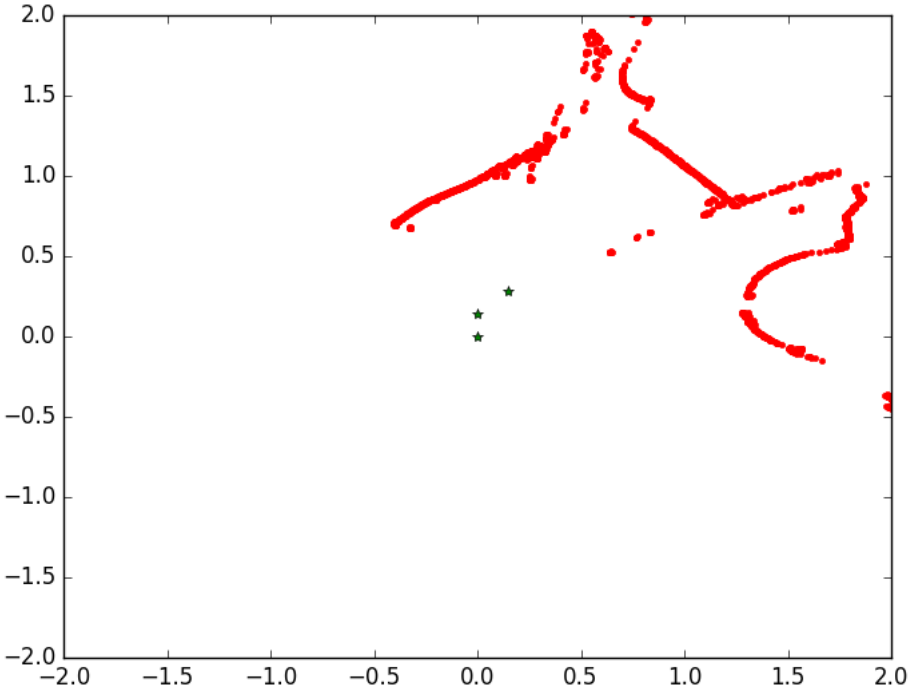


3.5) TEST – ΜΕΤΡΗΣΕΙΣ

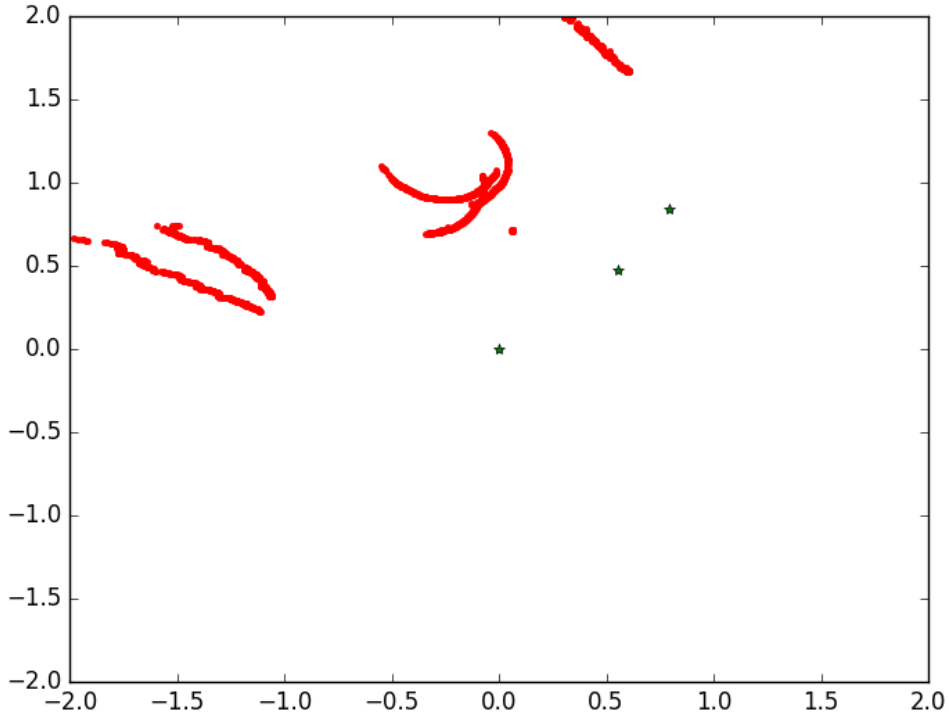
Παρακάτω παρουσιάζονται 3 τεστ που έγιναν σε διάφορες διαφορετικές περιπτώσεις.

1) Το ρομπότ βγάζει τρεις φωτογραφίες με κατεύθυνση αριστερά, μπροστά και δεξιά κάνοντας μόνο στροφές και μικρές μετακινήσεις. Ο χώρος και ο χάρτης που παράγεται φαίνονται παρακάτω. Βλέπουμε ότι τα βασικά σχήματα των εμποδίων καθώς και οι αποστάσεις τους υπολογίζονται σε ικανοποιητικό βαθμό.

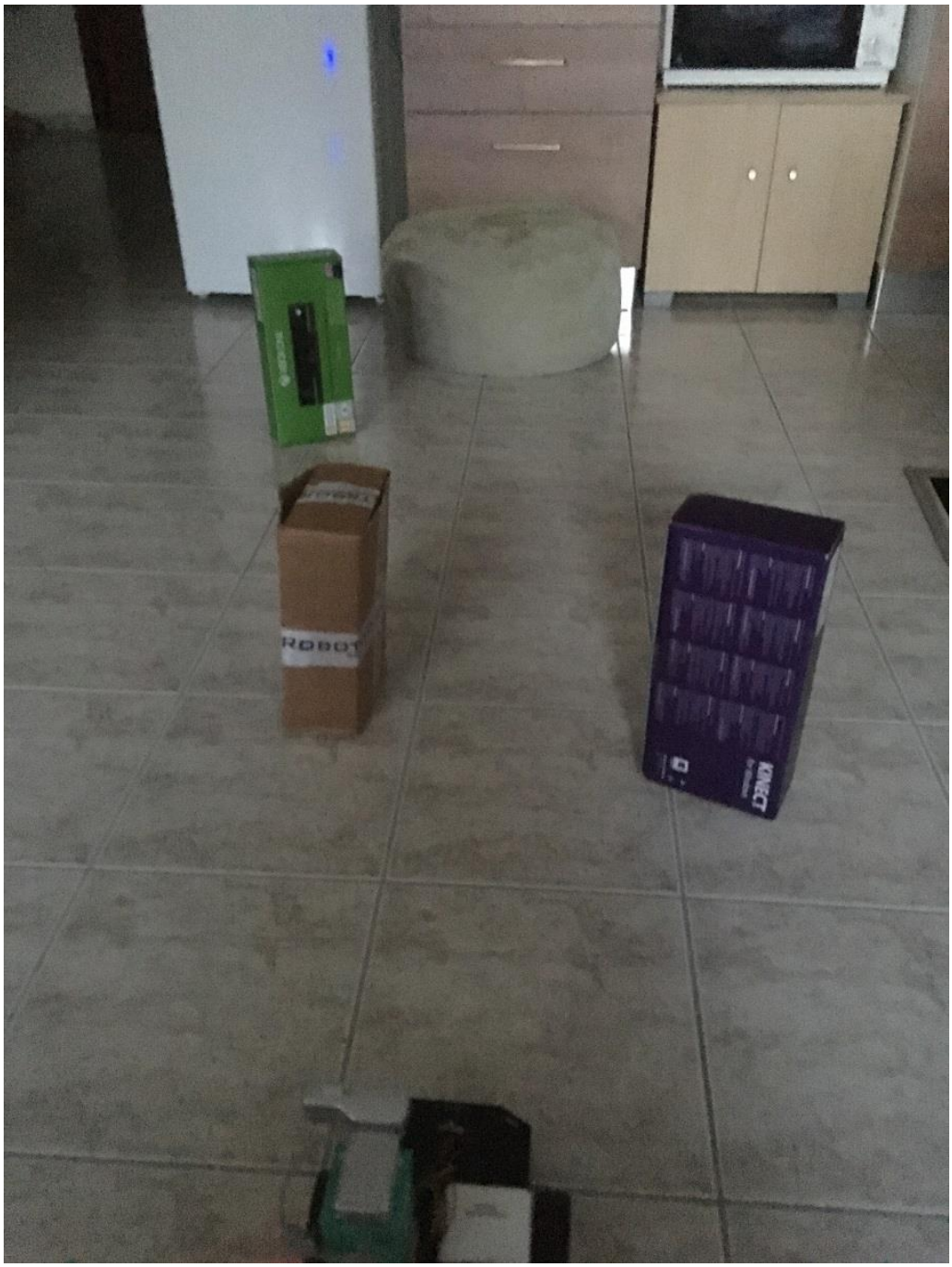


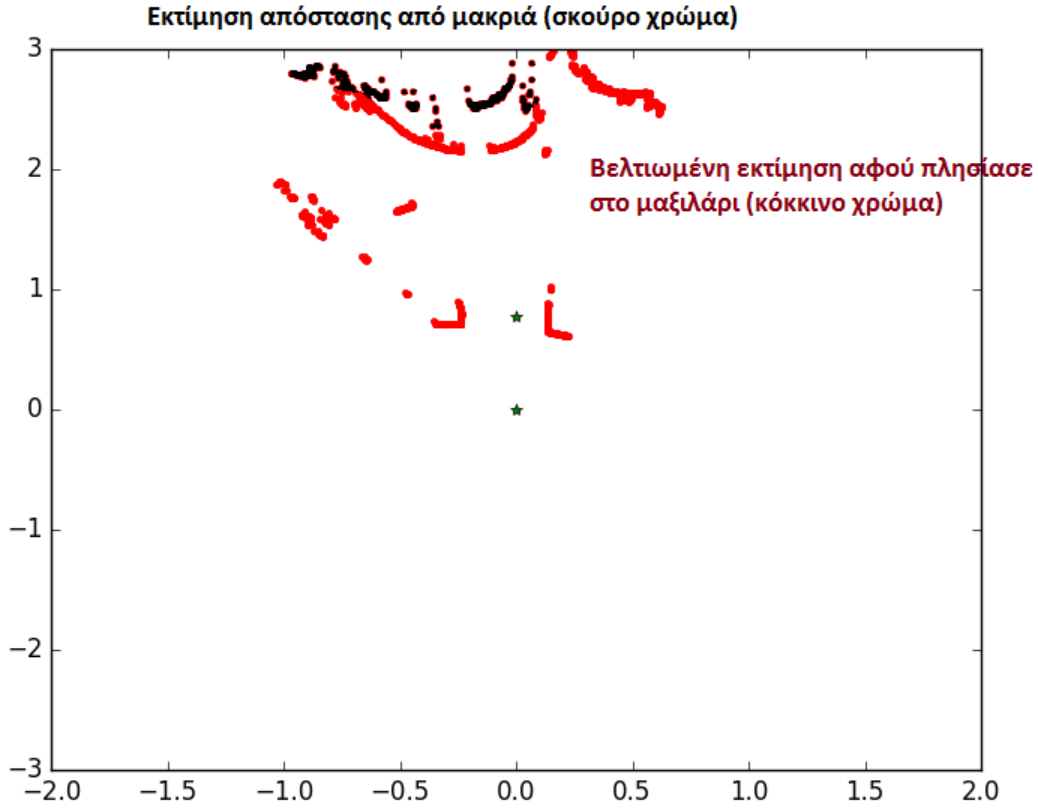

2.0
1.5
1.0
0.5
0.0
−0.5
−1.0
−1.5
−2.0
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

2) Το ρομπότ βγάζει φωτογραφία ένα αντικείμενο προχωρώντας τριγύρω του. Εδώ παρατηρούμε ότι εξαιτίας της παρατεταμένης κίνησης το ρομπότ δεν μπορεί υπολογίσει με μεγάλη ακρίβεια την θέση του, με αποτέλεσμα το εμπόδιο να φαίνεται κάπως παραμορφωμένο.



3) Σε αυτή την περίπτωση το ρομπότ προχωράει ευθεία ανάμεσα από διάφορα εμπόδια. Εδώ φαίνεται ότι όσο το ρομπότ πλησιάζει προς ένα εμπόδιο, τόσο πιο ακριβείς υπολογισμούς κάνει για την απόστασή του. Συγκεκριμένα η θέση του μαξιλαριού αρχικά υπολογίζεται πιο μακρυά απ' όσο πραγματικά είναι.





3.6) ΠΡΟΒΛΗΜΑΤΑ – ΔΥΣΚΟΛΙΕΣ

Ένα πρόβλημα που παρουσιάζεται είναι ότι εξαιτίας θορύβου στην εικόνα, υπολογίζονται εμπόδια που δεν υπάρχουν όπως ρογμές από τα πλακάκια, αντανακλάσεις απ' το φως, αλλαγές στα χρώματα. Σε αυτό συμβάλει η μικρή ανάλυση της κάμερας (1600x1200, οι κακές συνθήκες φωτισμού αλλά και οι αδυναμία των αλγορίθμων επεξεργασίας εικόνας.

Άλλο πρόβλημα υπάρχει στην καταγραφή της κίνησης του ρομπότ. Η απόσταση που διανύει ή η γωνία που διαγράφει όταν στρίβει δεν είναι πάντα ίδια. Αυτό οφείλεται στη φύση του συγκεκριμένου συστήματος controller – κινητήρων – τροχών το οποίο δεν μπορεί να παρέχει μεγάλη ακρίβεια.
Συγκεκριμένα, μετά από δοκιμές πάνω στην ακρίβεια της κίνησης, παρατηρήθηκε

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΜΕΛΛΟΝΤΙΚΗ ΔΟΥΛΕΙΑ

Αν και ο αρχικός στόχος της 3D αναπαράστασης αγνώστου χώρου μάλλον είναι αδύνατο να υλοποιηθεί με μια απλή κάμερα, τουλάχιστον από ένα χαμηλό σημείο λήψης και με τη συγκεκριμένη τεχνολογία και υλικό που χρησιμοποιήσαμε, η δημιουργία μιας κάτοψης είναι εφικτή.

Τα αποτελέσματα των πειραμάτων μας έδειξαν ότι ακόμα και με μία κάμερα μπορεί να γίνει αναπαράσταση χώρου στο δισδιάστατο επίπεδο, αρκεί οι συνθήκες φωτισμού να το επιτρέπουν και τα μηχανικά μέρη του ρομπότ να έχουν σχετικά μικρό error στις κινήσεις τους. Επίσης η βιβλιοθήκη OpenCV, παρ'όλη τη χρησιμότητά της, δεν μπορεί να φτάσει στο τέλειο αποτέλεσμα χρησιμοποιώντας μια γλώσσα αρκετά υψηλού,και άρα αφαιρετικού, επιπέδου όπως η Python που χρησιμοποιήσαμε, καθώς ο θόρυβος στις εικόνες, ακόμα και με διάφορες επεξεργασία και φίλτρα, πάντα θα αλλοιώνει το αποτέλεσμα.

Τα αποτελέσματα της δουλειάς μας μπορούν να γίνουν πολύ καλύτερα, αν αντικαταστήσουμε τα μηχανικά μέρη του ρομπότ με καλύτερα, κυρίως με τέσσερις τροχούς αντί των δύο υπαρχόντων. Επίσης ήδη με το Raspberry Pi 3 που κυκλοφορεί οι αλγόριθμοι θα έτρεχαν πολύ γρηγορότερα καθώς επιτρέπει πολλά περισσότερα threads, κάτι που λείπει από τον κώδικά μας.