ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΜΗΧΑΝΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΤΟΜΕΑΣ Μ.Κ. & Α.Ε.

Εργαστήριο Αυτομάτου Ελέγχου

Διπλωματική Εργασία

Σχεδιασμός λογισμικού οπτικής ανάδρασης θέσης με εφαρμογή στον έλεγχο διαστημικού εξομοιωτή

Πατσιαούρας Ηλίας

Επιβλέπων Καθηγητής: Ε. Γ. Παπαδόπουλος

ΑΘΗΝΑ 2015

Περίληψη

Η παρούσα διπλωματική εργασία αφορά την ανάπτυξη λογισμικού τεχνητής όρασης εντοπισμού θέσης και προσανατολισμού καθώς και του λογισμικού και υλικού ελέγχου θέσης/προσανατολισμού επίπεδου διαστημικού εξομοιωτή. Ο εν λόγω εξομοιωτής βρίσκεται στο Εργαστήριο Αυτομάτου Ελέγχου της σχολής Μηχανολόγων Μηχανικών ΕΜΠ, αποτελείται από δύο ρομπότ, τα οποία αιωρούνται σε διάκενο 10μm πάνω σε μια τράπεζα γρανίτη αμελητέας τραχύτητας. Σκοπός του εξομοιωτή είναι η πειραματική μελέτη της δυναμικής των ρομπότ σε συνθήκες που εξομοιώνουν την έλλειψη βαρύτητας στο επίπεδο, μέσω εξάλειψης της τριβής. Η πρωτοτυπία έγκειται στο ότι τα ρομπότ είναι μικρής μάζας, χαμηλού κόστους αλλά πλήρως αυτόνομα και με υποσυστήματα ανάλογα με αυτά ενός πραγματικού διαστημικού ρομποτικού συστήματος.

Η εργασία αυτή χωρίζεται σε δυο βασικά τμήματα. Το πρώτο αφορά το σχεδιασμό και την υλοποίηση ενός ολοκληρωμένου λογισμικού πακέτου με σκοπό τον οπτικό εντοπισμό των ρομπότ επί της τράπεζας του γρανίτη. Το δεύτερο τμήμα της εργασίας περιλανβάνει τον έλεγχος θέσης του νέου ρομπότ και παράλληλα τη δόκιμη του πακέτου xPC Target για την υλοποίηση αυτή ως συνεχεία της διπλωματικής εργασίας του Κων/νου Μαχαιρά. Κίνητρο για την ανάπτυξη ενός συστήματος σαν αυτό αποτελεί η συνεχής αυξανόμενη σπουδαιότητα των ρομποτικών συστημάτων στο διάστημα σε περιπτώσεις όπως: εξερεύνηση του διαστήματος, κατασκευή, συντήρηση και επιθεώρηση συστημάτων, προσέγγιση και πρόσδεση σε άλλα συστήματα που βρίσκονται σε τροχιά.

Τα ρομπότ του εξομοιωτή αποτελούνται από 3 υποσυστήματα: α) το μηχανολογικό: είναι κατασκευέ αλουμίνιο, διαθέτουν δύο βραχίονες δύο βαθμών ελευθερίας κι έναν σφόνδυλο αντίδρασης ως εναλλακτικό τρόπο κίνησης, β) το πνευματικό: με χρήση αέριου CO^2 για να αιωρούνται μέσω 3 αεροεδράνων αλλά και για να κινούνται στο επίπεδο μέσω 3 ζευγών προωθητήριων, γ) το ηλεκτρικό/ ηλεκτρονικό: προσδιορίζουν τη θέση τους μέσω δυο συστημάτων αισθητήρων (ενσωματωμένοι οπτικοί αισθητήρες και εξωτερική κάμερα) και μέσω αυτομάτου έλεγχου επενεργούν κατάλληλα στο περιβάλλον.

Χρησιμοποιώντας την τεχνική του σχεδιασμού βασισμένου στο μοντέλο, ο σχεδιασμός και η ανάπτυξη του νέου ρομπότ έγινε σε συνολικό επίπεδο. Το πακέτο λογισμικού που χρησιμοποιήθηκε ήταν το xPC Target, της Mathworks. Στο περιβάλλον αυτό, τα απλά μοντέλα προσομοίωσης Simulink μετατρέπονται άμεσα σε προγράμματα C, τα οποία εκτελούνται σε πραγματικό χρόνο κατευθείαν στο τελικό υλικό (hardware) του ρομπότ. Με τον τρόπο, έγινε δυνατή η αξιολόγηση και η εξέλιξη του σχεδιασμού σε όλη τη διάρκεια της ανάπτυξης. Έτσι, δόθηκε έμφαση στη καινοτομία, ενώ παρακάμφθηκαν πιθανά προβλήματα χαμηλού επίπεδου σε λογισμικό (software) και υλικό (hardware).

Abstract

The present diploma thesis deals with the design and implementation of computer vision software that estimates position and orientation of a space simulator's robot. It also describes the development of a control algorithm for the pose orient of the robot. The simulator is in the Control Systems Lab of the Mechanical Engineering Department of NTUA, and consists of robots hovering at a gap of 10 μm over a granite table of negligible roughness. The purpose of the simulator is the experimental study of the robots' dynamics in conditions that simulate zero gravity in 2D, through the elimination of friction. The novelty lies in the fact that the robots are of low mass, low cost, completely autonomous and carry subsystems that resemble those in a real space robot.

This work is devided to two major parts. The first focused on the design and implementation of a fully functional software suite with main purpose, the visual localization of the robots on the granite table and the broadcast to the local network of their attitude. The second part of this work presents the study of robot dynamic model and the design of a closed-loop attitude control algorithm that implement's a visual servoing controller using feedback from the first part of this work. This concept is going to be implemented on the xPC target platform in order to evaluate and test its capabilities. The motivation for the development of such a system is the increasing importance of space robots in cases such as: space exploration, construction, maintenance and inspection of systems in space, approach and attachment to other bodies in orbit.

The robots of the simulator consist of 3 subsystems: a) mechanical: they are made of aluminum, and they include two arms, with two joints each, and a reaction wheel as an alternative way of rotation, b) pneumatic: they use CO2 gas in order to hover using 3 air-bearings, and for planar motion using 3 pairs of thrusters, c) electric/ electronic: they estimate their posture using two systems of sensors (on board optical sensors and an external camera) and through automatic control act appropriately upon the environment.

Employing the technique of model-based design, the development of the new robot was carried out as a whole, since the design of algorithms, the software development and the final integration on the hardware were studied as interdependent parts. The software used was the xPC Target, from Mathworks. In this environment, C code is quickly generated from simple Simulink models, and is finally executed on the robot's hardware in hard real time. In this manner, validation and verification of the design were continuously performed throughout the development. Emphasis was placed on innovation, while at the same time potential low-level problems in hardware and software were overcame.

Ευχαριστίες

Θέλω να ευχαριστήσω το Καθηγητή κ. Ε. Παπαδόπουλο για την υποστήριξη του, την άμεση βοήθεια του σε ό,τι πρόβλημα αντιμετώπισα και την προτροπή του για καλύτερο αποτέλεσμα κατά την διάρκεια εκπόνησης της παρούσας διπλωματικής εργασίας. Με την βοήθεια και τη στήριξη του απέκτησα σημαντική εμπειρία στο τομέα των εφαρμογών υπολογιστών καθώς και του περιβάλλοντος xPC target της Mathworks. Θέλω να ευχαριστήσω επίσης όλα τα μέλη του εργαστηρίου που με βοήθησαν με διάφορους τρόπους να ολοκληρώσω την παρούσα εργασία. Ευχαριστώ ιδιαίτερα τον Ιωσήφ Παρασκευά για την πίστη του σε εμένα και τη δουλειά μου καθώς και τις πολύτιμες πληροφορίες που μου μετέδωσε, ως ο παλαιότερος του εργαστηρίου, οι οποίες έπαιξαν σημαντικότατο ρόλο στην ποιότητα της δουλειάς μου. Τέλος, ευχαριστώ την οικογένειά μου για την χωρίς αντάλλαγμα στήριξη της, καθώς και τους φίλους μου για την συμβολή τους στην ενίσχυση της αυτοεκτίμησης, και της δημιουργικότητάς μου.

Περιεχόμενα

Κατάλογος εικόνων

Κατάλογος διαγραμμάτων

Κατάλογος πινάκων

Κατάλογος σχημάτων

Πρόλογος

Η εξέλιξη της τεχνολογίας και των επιστημών έχει επιτρέψει στην ανθρωπότητα να θέσει σε τροχιά γύρω από την Γη ένα μεγάλο πλήθος δορυφόρων, οργάνων και διαστημικών σκαφών. Κατά τη διάρκεια της παραμονής τους σε τροχιά γύρω από τη Γη,είναι πολύ πιθανό οι διατάξεις αυτές να υποστούν βλάβες ενώ οι πόροι τους εξαντλούνται. Το κόστος κατασκευής τους, αλλά και εκείνο των διαστημικών αποστολών, είναι τέτοιο που καθιστά δύσκολη την αντικατάστασή τους ή την επισκευή τους. Επιπλέον, υπάρχει το ζήτημα των διαστημικών “σκουπιδιών”, άχρηστων αντικειμένων, συνήθως υπολείμματα προηγούμενων αποστολών, σε τυχαίες τροχιές και άκρως επικίνδυνων για τη ζωή έμψυχου και άψυχου υλικού.

Από τα προηγούμενα, γίνεται κατανοητό ότι η ύπαρξη ενός ρομποτικού συστήματος σε τροχιά που να επιτελεί εργασίες συντήρησης, συλλογής, και τοποθέτησης αντικειμένων σε ακίνδυνες τροχιές είναι πολύ σημαντική και χρήσιμη. Για να μπορέσει να κατασκευαστεί ένα τέτοιο ρομπότ σωστά και να είναι λειτουργικό στο χώρο δράσης του είναι αναγκαία η ύπαρξη επίγειου εξομοιωτή των συνθηκών του διαστήματος και της διαδικασίας των ενεργειών που θα εκτελέσει το ρομπότ.

Ένα τέτοιο σύστημα εξομοίωσης αποτελείται από υποσυστήματα για τη διεκπεραίωση λειτουργιών με σκοπό τη δημιουργία ένα εικονικού περιβάλλοντος που να παρέχει τη δυνατότητα δοκιμών και αξιολόγησης μεθόδων. Ένα τέτοιο υποσύστημα είναι η διάταξη προσδιορισμού και ανάδρασης της θέσης για την οπτική οδήγηση εξομοιωτή διαστημικού ρομπότ που περιγράφεται στον παρών σύγγραμμα.

1 Εισαγωγή

1.1 Σκοπός εργασίας

Η παρούσα διπλωματική εργασία εκπονήθηκε στο πλαίσιο των προπτυχιακών σπουδών του συντάκτη και αποτελεί μέρος ενός μεγαλύτερου εγχειρήματος που αποσκοπεί στην ανάπτυξη ενός εξομοιωτή διαστημικών ρομπότ (ρομπότ σε τροχιά). Ο εξομοιωτής βρίσκεται και αναπτύσσεται από το Εργαστήριο Αυτόματου Έλεγχου του ΕΜΠ στη Σχολή Μηχανολόγων Μηχανικών και αποτελείται από ρομπότ, τα οποία με τη χρήση αεροστατικών εδράνων αιωρούνται σε απόσταση μερικών μm πάνω σε μια τράπεζα γρανίτη αμελητέας τραχύτητας. Αυτό έχει ως αποτέλεσμα την εξάλειψη της τριβής μεταξύ ρομπότ και τράπεζας. Σκοπός του εξομοιωτή είναι η πειραματική μελέτη της δυναμικής των ρομπότ και η δημιουργία μια πλατφόρμας δοκιμών διαστημικών σεναρίων, όπως για παράδειγμα δοκιμές για την αξιολόγηση μοντέλων έλεγχου θέσης/προσανατολισμού, μοντέλων σχεδιασμού τροχιάς, μελέτη κρούσεων μεταξύ των ρομπότ και για την μελέτη της δυναμικής τους και πιο σύνθετων σεναρίων όπως την συνεργασία δυο ρομπότ για την σύλληψη και διακομιδή ενός αντικειμένου.

Για την μελέτη των παραπάνω σεναρίων απαιτείται ανάδραση θέσης, και επομένως πρέπει η θέση να είναι διαθέσιμη στα υπολογιστικά συστήματα των ρομπότ. Η διπλωματική εργασία αυτή περιγράφει την διαδικασία που ακολουθήθηκε για να επιτευχθεί η ανάδραση θέσης για την οπτική οδήγηση των ρομπότ του εξομοιωτή καθώς και την υλοποίηση μια τέτοιας οδήγησης θέσης σε ένα από τα δυο ρομπότ, σε περιβάλλον Matlab/Simulink.

Ο ορός *οπτική* αναφέρεται στη χρήση οπτικών μέσων που χρησιμοποιήθηκαν για τον προσδιορισμό της θέσης των ρομπότ. Πιο συγκεκριμένα τοποθετήθηκε πάνω από το χώρο εργασίας ψηφιακή κάμερα που παρατηρεί και εντοπίζει τα ρομπότ. Αυτό πραγματοποιείται από το λογισμικό που δημιουργήθηκε στο πλαίσιο της εργασίας αυτής και το οποίο αναλαμβάνει να συλλέγει στιγμιότυπα του χώρου εργασίας σε πραγματικό χρόνο, να τα επεξεργάζεται και να εξάγει συμπεράσματα για τη απόλυτη θέση και προσανατολισμό των ρομπότ. Αυτό πραγματοποιείται με την ύπαρξη επάνω στα ρομπότ φωτεινών πηγών LED που εντοπίζονται από τον αλγόριθμο. Τέλος η ανατροφοδότηση της θέσης στα ρομπότ γίνεται μέσω ασύρματου δικτυού WIFI με την βοήθεια μιας κάρτας δικτύου και μιας ασύρματης γέφυρας που διαθέτει κάθε ρομπότ.

Στην περιγραφή που προηγήθηκε υπάρχουν κάποιες καίριες παράμετροι που επηρεάζουν την υλοποίηση και την λειτουργία της διάταξης οπτικής ανάδρασης μέσω κάμερας. Κάποιες από αυτές είναι μη αναστρέψιμες αφού έχουν γίνει ήδη στο πλαίσιο παλαιότερων εργασιών όπως η επιλογή οπτικού αισθητήρα και φακού για την κάλυψη του χώρου. Οι υπόλοιπες παράμετροι παραμένουν μεταβλητές και είναι

αυτές που στο πλαίσιο αυτής της εργασίας θα τροποποιηθούν σε σχέση με τις προηγούμενες εργασίες για να επιτευχθεί ένα καλύτερο αποτέλεσμα. Αυτές είναι: η γλώσσα προγραμματισμού στην οποία θα γραφεί ο αλγόριθμος, με ποιο τρόπο θα πραγματοποιηθεί η μοντελοποίηση της κάμερας, οι παράμετροι λήψης στιγμιότυπων, η διαδικασία αναγνώρισης των ρομπότ καθώς και ποιες από αυτές θα γίνουν σε πραγματικό χρόνο και ποιες ως προεργασία. Όσον αφορά το δεύτερο μέρος της εργασίας που είναι η υλοποίηση του έλεγχου θέσης στο ρομπότ οι παράμετροι που δε είναι αναστρέψιμοι είναι τα τεχνικά χαρακτηριστικά του υπολογιστικού συστήματος του ρομπότ αφού έχουν ήδη επιλεχθεί σε προηγούμενες εργασίες ενώ οι ελεύθεροι παράμετροι που επηρεάζουν το τελικό αποτέλεσμα είναι: το λειτουργικό σύστημα των ρομπότ, ο τύπος έλεγχου που θα εφαρμοστεί, ο τρόπος βαθμονόμησης των επενεργητών του ρομπότ και των γεωμετρικών και δυναμικών χαρακτηριστικών του.

1.2 Βιβλιογραφική ανασκόπηση

Το πρώτο μέρος της παρούσας διπλωματικής εργασίας ασχολείται με την βαθμονόμηση της κάμερας, έτσι ώστε να μπορεί να προσδιοριστεί η θέση του ρομπότ με την μεγαλύτερη δυνατή ακρίβεια.

Οι πρώτες μέθοδοι βαθμονόμησης κάμερας βασίζονται στη χρήση ενός πλέγματος βαθμονόμησης με σημεία που οι 3-D συντεταγμένες τους είναι γνωστές. Τα σημεία αυτά ονομάζονται σημεία ελέγχου (“Control points”) και μπορούν να εντοπιστούν εύκολα από αλγορίθμους επεξεργασίας εικόνας (π.χ. γωνίες, κουκκίδες). Μόλις προσδιοριστούν τα σημεία ελέγχου η μέθοδος βαθμονόμησης με βάση την θέση τους στην εικόνα υπολογίζει τις βέλτιστες τιμές των εξωγενών παραμέτρων (μεταφορά και περιστροφή) καθώς και τον ενδογενών παραμέτρων (αναλογία pixel, εστιακή απόσταση, κλπ.). Από τις πρωταρχικές και σημαντικές εργασίες που χρησιμοποιούν την μέθοδο αυτή είναι του R.Y. Tsai το 1987 [1] και του Z. Zhang το 1999 [2], πάνω στην οποία έχουν αναπτυχθεί πολλές open-source εφαρμογές βαθμονόμησης.

Οι παραπάνω μέθοδοι χρησιμοποιούν το μοντέλο της μικροσκοπικής οπής, το οποίο σε περιπτώσεις μεγάλης μη-γραμμικής παραμόρφωσης του φακού μπορεί να οδηγήσει σε αυξημένα σφάλματα βαθμονόμησης. Μία άλλη μέθοδος βαθμονόμησης που ασχολείται κυρίως με το μοντέλο της παραμόρφωσης είναι αυτή του D.C Brown το 1971 [3].

Άλλη κατηγορία αποτελούν οι μέθοδοι αυτοβαθμονόμησης που προσπαθούν να υπολογίσουν τις παραμέτρους της κάμερας χρησιμοποιώντας πολλαπλές προβολές της ίδιας σκηνής και αξιοποιώντας την ακαμψία της σκηνής. Μία από τις σημαντικότερες μελέτες αυτής της κατηγορίας μπορεί να βρεθεί στο [4]. Επίσης, βαθμονόμηση της κάμερας μπορεί να επιτευχθεί αξιοποιώντας το γεγονός ότι οι ευθείες γραμμές μέσα στην εικόνα πρέπει μετά τον τελικό μετασχηματισμό να είναι πράγματι ευθείες [5].

Τέλος οι παραπάνω μέθοδοι προϋποθέτουν τον εντοπισμό ενός αντικειμένου στον 3D χώρο και για αυτό χρειάζονται πολλαπλές εικόνες με διαφορετικούς προσανατολισμούς. Στην περίπτωση που χρειάζεται ένα απλουστευμένο μοντέλο μίας ή δύο ελεύθερων παραμέτρων, η βαθμονόμηση μπορεί να γίνει μόνο με μία εικόνα του πλέγματος βαθμονόμησης όπως και στην παρούσα εργασία.

Το δεύτερο μέρος της εργασίας πραγματεύεται τον σχεδιασμό ελέγχου κλειστού βρόχου με ανάδραση θέσης προσανατολισμού.

Ο έλεγχος θέσης ενός ρομποτικού συστήματος βασίζεται στο ορισμό ενός σφάλματος που αποτελεί τη διαφορά επιθυμητής και πραγματικής θέσης του συστήματος. Έχουν αναπτυχθεί με τα χρόνια πολλοί τρόποι ελέγχου ρομποτικών συστημάτων, όπως ο Βέλτιστος Έλεγχος, ο Ασαφούς Λογικής (fuzzy logic), έλεγχος με νευρωνικά δίκτυα. Όλοι όμως βασίζονται στις αρχές του κλασικού έλεγχου ο οποίος αποτελεί την θεμελιώδη θεωρία ελέγχου και παρέχει τους πιο αξιόπιστους και διαδεδομένους τρόπους ελέγχου. Όλες οι μέθοδοι του κλασσικού έλεγχου βασίζονται στην ιδία αρχή, δηλαδή το προσδιορισμό κερδών τα οποία πολλαπλασιαζόμενα με το σφάλμα να προσδιορίζουν το σήμα ελέγχου που στέλνεται στους επενεργητές.

Ένα μέρος της θεωρίας του κλασσικού Έλεγχου είναι ο *έλεγχος βασισμένος στο μοντέλο* (model-based control) και αποτελεί την μεθοδολογία με την οποία γνωρίζοντας τα δυναμικά χαρακτηριστικά του συστήματος μπορούν προσδιοριστούν τα κέρδη για να προκύψει συγκεκριμένης μορφής απόκριση σφάλματος. Όλα αυτά περιγράφονται αναλυτικά στα [6] και [7].

Στην παρούσα εργασία θα αναλυθεί η χρήση ελέγχου PD.

1.3 Δομή εργασίας

Στον **πρόλογο και το πρώτο κεφάλαιο** είσαγεται το πρόβλημα των διαστημικών απορριμάτων και πως η ύπαρξη διαστημικών ρομπότ μπορεί να αποτελέσει λύση σε αυτό. Παρουσιάζονται τα ερευνητικά κίνητρα που εμπεριέχει η ανάπτυξη τους και η ανάγκη για την ύπαρξη εξομοιωτών διαστημικών ρομπότ στη Γη. Στη συνέχεια γίνεται η σύνδεση με το πρόβλημα οπτικής ανάδρασης θέσης και οδήγησης που είναι τα δυο βασικά ζητήματα που πραγματεύεται αυτή η εργασία.

Στο **δεύτερο κεφάλαιο,** παρουσιάζονται τα μαθηματικά μοντέλα και τα στοιχεία θεωρίας που χρησιμοποιήθηκαν για την επίλυση των ζητημάτων της εργασίας. Αυτά εντοπίζονται στο τομέα την τεχνητής όρασης και του αυτόματου ελέγχου.

Στο **τρίτο κεφάλαιο,** περιγράφεται αναλυτικά η διάταξη του εξομοιωτή, τα κύρια στοιχεία του, ο τρόπος λειτουργίας του και πως μπορεί να βοηθήσει στην ανάπτυξη των πραγματικών ρομπότ. Εξηγείται το υπάρχον σύστημα και οι φυσικοί περιορισμοί που έχει. Επίσης περιγράφεται συγκεκριμένα το σύστημα οπτικής ανάδρασης θέσης, πως οι επιλογές που έχουν γίνει επηρεάζουν την ανάπτυξη του συστήματος οπτικής ανάδρασης θέσης και τι περιορισμοί εισάγονται για τον σχεδιαστή του νέου συστήματος. Περιγράφεται το υπάρχον σύστημα και τα μειονεκτήματα του. Στην συνέχεια αναλύεται η διερεύνηση που έγινε και διατυπώνονται οι ανάγκες και οι απαιτήσεις των συστημάτων που καλούνται να αναπτυχθούν. Τέλος περιγράφεται το νέο λογισμικό.

Το **τέταρτο κεφάλαιο,** εστιάζει στην υλοποίηση όσων περιγράφθηκαν στο τρίτο κεφάλαιο. Πιο συγκεκριμένα περιγράφει την μοντελοποίηση της κάμερας και τη βαθμονόμηση της με το λογισμικό MATLAB. Επίσης, περιγράφονται οι τεχνικές που μελετήθηκαν και επιλέχθηκαν και παρουσιάζεται το τελικό αποτέλεσμα που είναι το γραφικό περιβάλλον του λογισμικού. Τέλος παρουσιάζονται αποτελέσματα από πειράματα που έγιναν με σκοπό την αξιολόγηση του νέου αλγόριθμου ως προς την ακρίβεια και την αξιοπιστία του.

Το **πέμπτο κεφάλαιο,** αφορά τον έλεγχο θέσης του ρομπότ και τις ενέργειες που έγιναν για να υλοποιηθεί. Αυτές περιλαμβάνουν την πειραματική μέτρηση και βαθμονόμηση των επενεργητών, την εξέταση και αξιολόγηση διαφόρων μεθόδων οδήγησης των βαλβίδων για την καλύτερη και ομαλότερη λειτουργία του ρομπότ. Επίσης περιλαμβάνει τον προσδιορισμό των χαρακτηριστικών αδράνειας του ρομπότ και την μοντελοποίηση του με σκοπό τον σχεδιασμό ελέγχου βασισμένου στο μοντέλο.

Το **έκτο κεφάλαιο** αναφέρεται στην υλοποίηση του ελέγχου και των υποσυστημάτων επικοινωνίας στο περιβάλλον Simulink του MATLAB για τον προγραμματισμό του ρομπότ. Περιγράφεται ο τρόπος με τον οποίο υλοποιηθήκαν όλα τα υποσυστήματα, τα προβλήματα και οι περιορισμοί αυτής της υλοποίησης.

Στο **έβδομο κεφάλαιο,** παρουσιάζονται τα συμπεράσματα όλης της εργασίας και αναφέρονται τα σημεία τα οποία κατά την γνώμη του συγγραφέα απαιτούν περαιτέρω μελέτη.

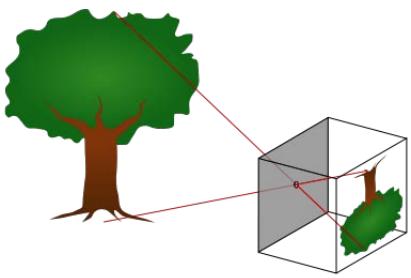
2 Στοιχεία θεωρίας

Σε αυτό το κεφάλαιο παρατίθενται όλα τα στοιχεία θεωρίας που χρησιμοποιήθηκαν για την εκπόνησης αυτής της εργασίας. Στην συνέχεια παρουσιάζονται: (α) το μαθηματικό μοντέλο που χρησιμοποιήθηκε για την μοντελοποίηση του οπτικού αισθητήρα, δηλαδή τις μαθηματικές σχέσεις οι όποιες συνδέουν τις μετρήσεις σε μονάδες εικονοστοιχείων px σε μονάδες μήκους mm. (β) παρουσιάζεται το μοντέλο παραμόρφωσης που εισάγεται από τον ευρυγώνιο φακό και έχει τοποθετηθεί για να είναι παρατηρήσιμος όλος ο χώρος εργασίας. Αυτό το μοντέλο συνδέει μαθηματικά τις συντεταγμένες ενός αντικείμενου στο επίπεδο της εικόνας με αυτές που θα είχε εάν είχαμε ένα αισθητήρα χωρίς φακό σε μεγαλύτερη απόσταση.

2.1 Μοντέλο κάμερας μικρής οπής

Το μοντέλο μικρής οπής ή αλλιώς "*Pin Hole model*" περιγράφει τη μαθηματική σχέση μεταξύ των συντεταγμένων ενός σημείου στο χώρο και της προβολής του επί του επιπέδου της εικόνας του *ιδανικού* αισθητήρα, όπου το διάφραγμα της κάμερας περιγράφεται ως ένα σημείο (μηδενική διάμετρος) χωρίς χρήση φακού. Το μοντέλο δεν περιλαμβάνει, για παράδειγμα, γεωμετρικές παραμορφώσεις ή θόλωση που προκαλείται από τους φακούς και από το άνοιγμα του διαφράγματος. Επίσης, δεν λαμβάνει υπόψη το γεγονός ότι πρακτικά οι αισθητήρες έχουν διακριτές συντεταγμένες στη εικόνα και η ακρίβεια του μοντέλου εξαρτάται από την ποιότητα της κάμερας και, σε γενικές γραμμές, μειώνεται από το κέντρο της εικόνας στα άκρα ως αποτέλεσμα της παραμόρφωσης του φακού.

Μερικά από τα δεδομένα που δεν λαμβάνει υπόψη το μοντέλο μπορούν να αντισταθμιστούν, με την εφαρμογή κατάλληλων μετασχηματισμών συντεταγμένων στις συντεταγμένες της εικόνας, ενώ κάποια να αμεληθούν ως μικρής σημασίας εάν χρησιμοποιείται μια κάμερα υψηλής ποιότητας. Αυτό σημαίνει ότι το μοντέλο αυτό μπορεί να χρησιμοποιηθεί στην τεχνητή όραση ως μια περιγραφή του πώς μια φωτογραφική μηχανή απεικονίζει μια τρισδιάστατη σκηνή (βλ. Σχήμα 2-1).

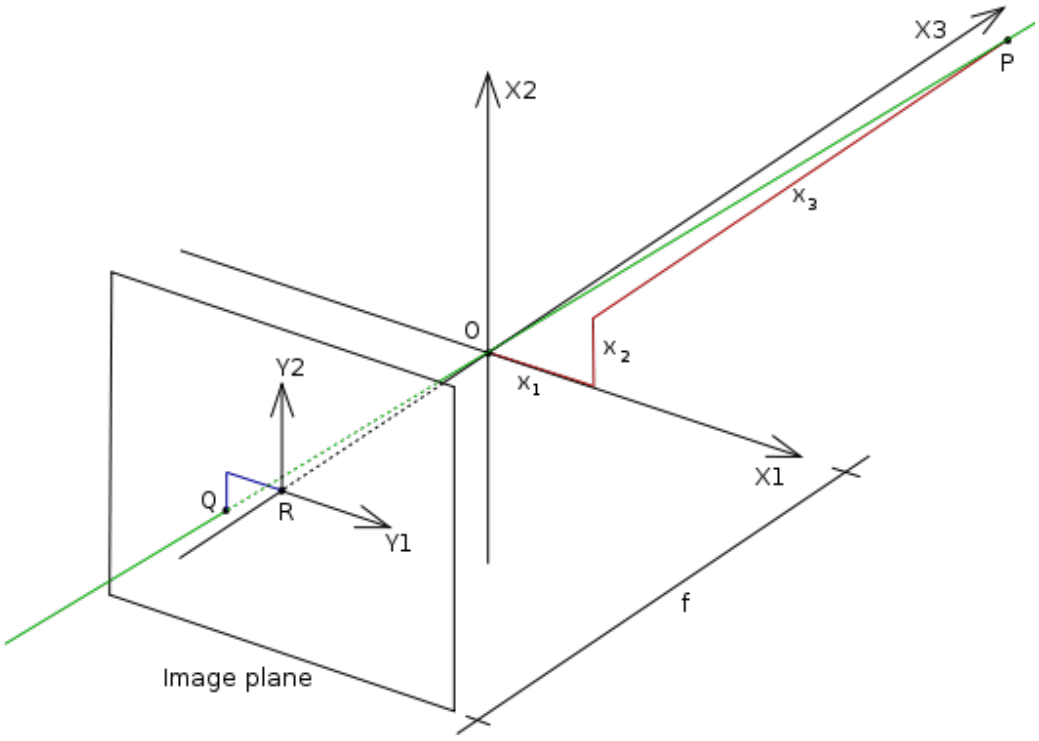


Σχήμα 2-1. Γραφική εξήγηση του μοντέλου.

2.1.1 Γεωμετρική ανάλυση

Η γεωμετρία και οι μαθηματικές σχέσεις που σχετίζεται με το μοντέλο οπής κάμερας παρουσιάζονται παρακάτω (βλ. Σχήμα 2-2). Η περιγραφή περιλαμβάνει τα ακόλουθα βασικά αντικείμενα:

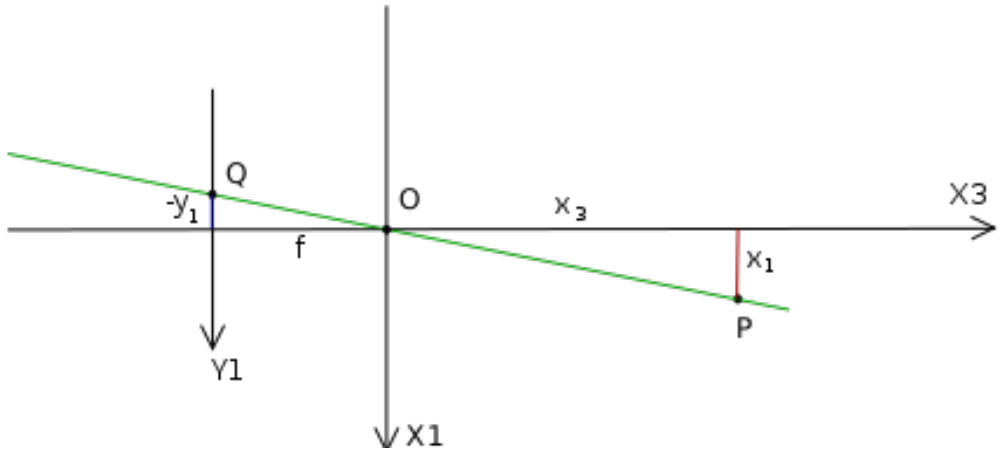
- Ένα τρισδιάστατο ορθοκανονικό σύστημα συντεταγμένων με αρχή το ***O***. Σε αυτό το σύστημα βρίσκεται και το διάφραγμα της κάμερας. Οι τρεις άξονες αναφέρονται ως X1, X2, X3. Ο άξονας X3 δείχνει την κατεύθυνση που βλέπει η κάμερα και ονομάζεται οπτικός άξονας ή *κύριος άξονας*. Ενώ η επιφάνεια που ορίζουν τα X1,X2 ονομάζεται *κυρία επιφάνεια*.
- Μια *επιφάνεια εικόνας* οπού ο 3Δ χώρος προβάλλεται μέσω του διαφράγματος της κάμερας. Η *επιφάνεια εικόνας* είναι παράλληλη στην *κύρια επιφάνεια* και απέχει απόσταση *f* από την αρχή ***O*** στην αρνητική κατεύθυνση του άξονα X3. Μια υλοποίηση του μοντέλου κάμερας είναι η *επιφάνεια εικόνας* να βρίσκεται στο *–f* του άξονα X3
- Ένα σημείο **R** στην τομή του οπτικού άξονα και της επιφάνειας εικόνας και ονομάζεται κύριο σημείο ή κέντρο εικόνας
- Ένα σημείο **P** κάπου στο χώρο με συντεταγμένες (x_1,x_2,x_3)
- Η γραμμή προβολής του σημείου στην κάμερα. Είναι η πράσινη γραμμή που περνάει από το **P** και το **O**
- Η προβολή του σημείου **P** στην *επιφάνεια εικόνας* ονομαζόμενη **Q**. Το σημείο **Q** βρίσκεται στην τομή της γραμμής προβολής με την *επιφάνεια εικόνας*
- Υπάρχει επίσης ακόμα ένα 2Δ σύστημα συντεταγμένων με αρχή το **R** με άξονες Y1,Y2 παράλληλους στα X1,X2 αντίστοιχα. Οι συντεταγμένες του **Q** είναι (y_1 , y_2) σε αυτό.



Σχήμα 2-2. Σύστημα συντεταγμένων του μοντέλου.

Το διάφραγμα στο μοντέλο αυτό θεωρείται σημειακό και αναφέρεται στην βιβλιογραφία ως εστιακό κέντρο.

Στη συνέχεια θέλουμε να κατανοήσουμε πώς οι συντεταγμένες (y_1 , y_2) του σημείου εξαρτόνται από τις συντεταγμένες (x_1 , x_2 , x_3) του σημείου P. Αυτό μπορεί να γίνει κατανοητό στο Σχήμα 2-3 το οποίο δείχνει την ίδια σκηνή με το προηγούμενο σχήμα, αλλά τώρα από πάνω, κοιτάζοντας προς τα κάτω με την αρνητική κατεύθυνση του άξονα X2.



Σχήμα 2-3. Εξήγηση προβολής.

Στο Σχήμα 2-3 βλέπουμε δύο όμοια τρίγωνα, και τα δύο έχουν τμήματα της γραμμής προβολής (πράσινη) ως υποτείνουσες τους . Η κάθετη του αριστερού τριγώνου είναι $-y_1$ και ***f*** και η κάθετη του ορθογωνίου τριγώνου είναι x_1 και x_3. Δεδομένου ότι τα δύο τρίγωνα είναι όμοια προκύπτει ότι:

$$-\frac{y_1}{f} = \frac{x_1}{x_3} \quad \rightarrow \quad y_1 = -f\frac{x_1}{x_3} \tag{2.1}$$

$$-\frac{y_2}{f} = \frac{x_2}{x_3} \quad \rightarrow \quad y_1 = -f\frac{x_2}{x_3} \tag{2.2}$$

2.1.2 Περιστροφή εικόνας και το επίπεδο της εικόνας

Η μετατροπή από 3Δ σε 2Δ συντεταγμένες που περιγράφεται από μοντέλο μικρής οπής είναι μια προοπτική προβολή που ακολουθείται από μια περιστροφή 180° στο επίπεδο της εικόνας, όπως φαίνεται στο Σχήμα 2-1. Αυτό συνδέεται με το πώς λειτουργεί μια πραγματική κάμερα pinhole, η προκύπτουσα εικόνα περιστρέφεται κατά 180° ενώ το μέγεθος των αντικειμένων εξαρτάται από την απόστασή τους από το εστιακό σημείο και το συνολικό μέγεθος της εικόνας εξαρτάται από την απόσταση f μεταξύ του επιπέδου της εικόνας και του εστιακού κέντρου. Για να μελετηθεί μια μη περιστραμμένη εικόνα, η οποία είναι ό,τι θα περιμέναμε από μια κάμερα:

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = -\frac{f}{x_3}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.3}$$

Περιστροφουμέ το σύστημα συντεταγμένων στο επίπεδο της εικόνας κατά 180°. Αυτο υλοποιείται πολύ εύκολα σε μια πραγματική κάμερα, εάν διαβάσουμε τα πίξελ της με τέτοια σείρα ωστε να σχηματιστεί η μη ανεστραμμενή εικόνα.

Για την μελετη της μη περιστραμένης εικόνας μετακινούμε το επίπεδο της εικόνας ώστε να τέμνει τον άξονα X3 στο *f* αντί για *-f* και με βάση τους προηγούμενους υπολογισμούς. Αυτό θα δημιουργήσει ένα εικονικό επίπεδο εικόνας που δεν μπορεί να εφαρμοστεί στην πράξη, αλλά παρέχει ένα απλοποιημένο θεωρητικό μοντέλο διατειρώντας την ισχύς του μοντέλου.

Προκύπτει λοιπόν σύμφωνα με τα παραπάνω η απλουστευμένη μορφή της σχέσης 2.1.

$$\begin{bmatrix} y_1 \\ y_2 \end{bmatrix} = \frac{f}{x_3} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \tag{2.4}$$

Για την καλύτερη εποπτεία του μαθηματικού μοντέλου μπορούμε να επαναδιατυπώσουμε τον τύπο σε μορφή μητρώων:

$$\lambda \begin{pmatrix} y_1 \\ y_2 \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{pmatrix} \tag{2.5}$$

Όπου $\lambda = X_3$ είναι ο ομογενής συντελεστής κλίμακας.

Τα περισσότερα από τα σημερινά συστήματα απεικόνισης καθορίζουν την αρχή των αξόνων στο επάνω αριστερά pixel της εικόνας. Ωστόσο, είχε προηγουμένως υποτεθεί ότι η αρχή του συστήματος συντεταγμένων εικόνας αντιστοιχεί στο κύριο σημείο **R**, που βρίσκεται στο κέντρο της εικόνας (βλ. Σχήμα 2-2). Μια μετατροπή των συντεταγμένων είναι απαραίτητη. Χρησιμοποιώντας ομογενείς συντεταγμένες, η θέση του **R** μπορεί εύκολα να ενσωματωθεί μέσα στη μήτρα προβολής. Η εξίσωση προβολής γίνεται:

$$\lambda \begin{bmatrix} y_1 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & R_x & 0 \\ 0 & f & R_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ 1 \end{bmatrix} \tag{2.6}$$

2.2 Μοντέλο παραμόρφωσης εικόνας

Οι πραγματικοί φακοί καμερών εισάγουν μη γραμμική παραμόρφωση της εικόνας. Αυτή η παραμόρφωση είναι ένας συνδυασμός λοξότητας, ακτινικής και περιστροφικής παραμόρφωσης. Στους σύγχρονους φακούς, η ακτινική παραμόρφωση αποτελεί το μεγαλύτερο τμήμα παραμόρφωσης, ενώ οι υπόλοιποι όροι μπορούν να παραληφθούν. Η ακτινική παραμόρφωση αναφέρεται αλλιώς και ως βαρελοειδής λόγω του χαρακτηριστικού να απεικονίζει τα πράγματα διογκωμένα στο κέντρο της εικόνας. Όπως δηλώνει και το όνομα της, εισάγει ένα σφάλμα στη θέση των εικονοστοιχείων που εξαρτάται από την απόσταση τους από το κέντρο της εικόνας. Η ακτινική παραμόρφωση του φακού εμφανίζεται εντονότερα στις άκρες της εικόνας, όπου η ακτινική απόσταση είναι μεγαλύτερη. Μία τυπική μοντελοποίηση της ακτινικής παραμόρφωσης μπορεί να περιγραφεί ως ακολούθως.

$$\begin{bmatrix} x_u - o_x \\ y_u - o_y \end{bmatrix} = L(r_d) \begin{bmatrix} x_d - o_x \\ y_d - o_y \end{bmatrix} \quad (2.7)$$

όπου:

$$L(r_d) = 1 + k_1 r_d^2 \quad (2.8)$$

$$r_d^2 = (x_d - o_x)^2 + \left(y_d - o_y\right)^2 \quad (2.9)$$

Όπου k_1: είναι ο συντελεστής παραμόρφωσης, με τα x_d,y_d να είναι οι παραμορφωμένες συντεταγμένες και x_u, y_u να είναι οι μη παραμορφωμένες συντεταγμένες. Μια τέτοια περιγραφή της ακτινικής παραμόρφωσης δε είναι επαρκής για μεγάλες παραμορφώσεις σαν αυτές που δημιουργούνται από ευρυγώνιους φακούς.

Σύμφωνα με το μοντέλο που προτείνουν οι Janne Heikkilä και Olli Silvén [8] μπορούμε να μοντελοποιήσουμε την παραμόρφωση συμφώνα με την παρακάτω σχέση:

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = 1 + k_1 r^2 + k_2 r^4 + k_5 r^6 + dx \quad (2.10)$$

$$r^2 = x^2 + y^2 \quad (2.11)$$

Όπου: $x_{d,}\, y_d$, οι παραμορφωμένες συντεταγμένες

Και όπου dx είναι η εφαπτομενική παραμόρφωση στην όποια οφείλεται η λοξότητα της εικόνας.

$$dx = \begin{bmatrix} 2k_3 xy + k_4(r^2 + 2x^2) \\ k_3(r^2 + 2y^2) + 2k_4 xy \end{bmatrix} \quad (2.12)$$

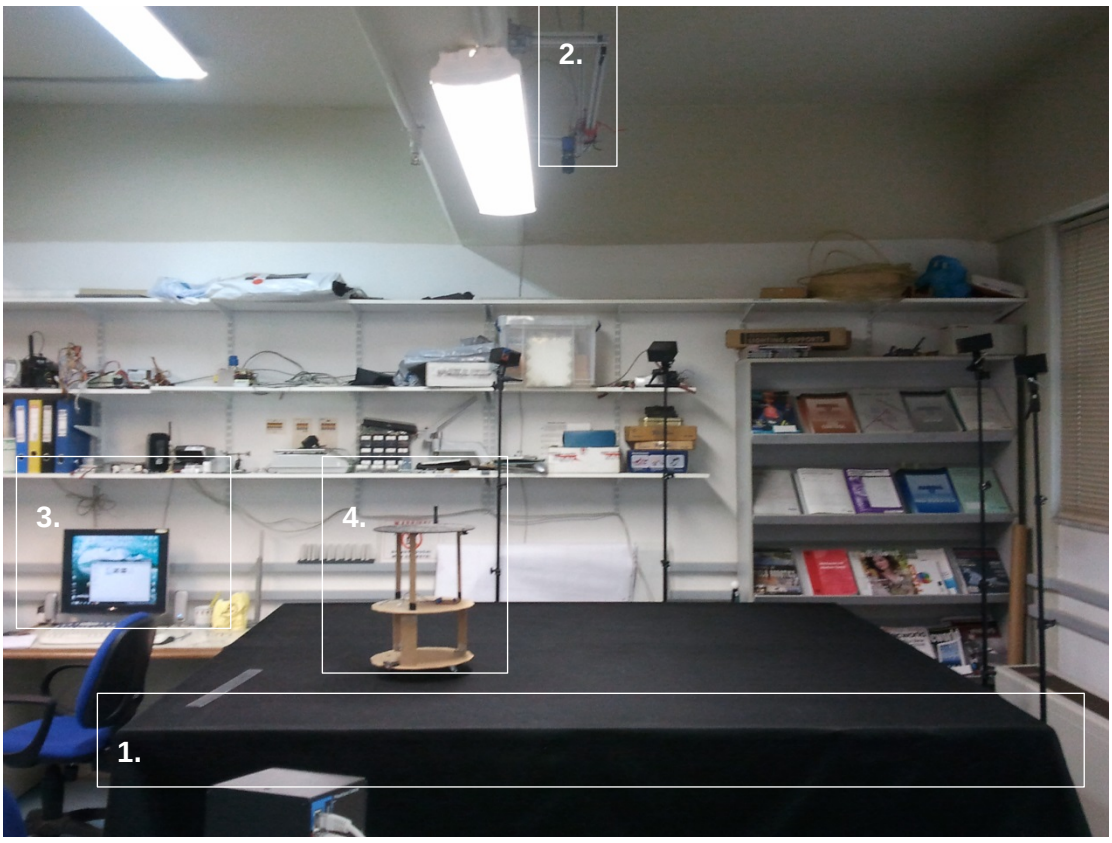
Αξίζει να σημειωθεί ότι αυτό το μοντέλο παραμόρφωσης εισήχθη για πρώτη φορά από τον Brown το 1966 και ονομάζεται μοντέλο ‘’Plumb Bob’’. Η εφαπτομενική παραμόρφωση οφείλεται στην κακή ευθυγράμμιση, ή κακό κεντράρισμα των εξαρτημάτων του φακού και άλλες ατέλειες κατασκευής σε ένα φακό.

3 Εντοπισμός θέσης ρομπότ εξομοιωτή μέσω τεχνητής όρασης

Στο κεφάλαιο αυτό περιγράφεται αρχικά ο διαστημικός εξομοιωτής του Εργαστηρίου Αυτομάτου Έλεγχου του ΕΜΠ με σκοπό ο αναγνώστης να μπορέσει να κατανοήσει το τρόπο λειτουργίας, τις σχεδιαστικές επιλογές που πάρθηκαν καθώς και τους περιορισμούς του συστήματος. Στη συνέχεια το ενδιαφέρον εστιάζεται στο σύστημα εντοπισμού θέσης των ρομπότ του εξομοιωτή, περιγράφεται η λειτουργία του και αναλύονται τα προβλήματα της υπάρχουσας υλοποίησης. Τέλος, παρουσιάζονται οι ιδέες της νέας υλοποίησης, βασισμένες στις νέες απαιτήσεις, η οποία θα παρουσιασθεί εκτενέστερα στο επόμενο κεφάλαιο.

3.1 Εγκατάσταση

Στην Εικόνα 1 παρουσιάζεται η εγκατάσταση του Διαστημικού Εξομοιωτή.



Εικόνα 1. Ο Διαστημικός Εξομοιωτής.

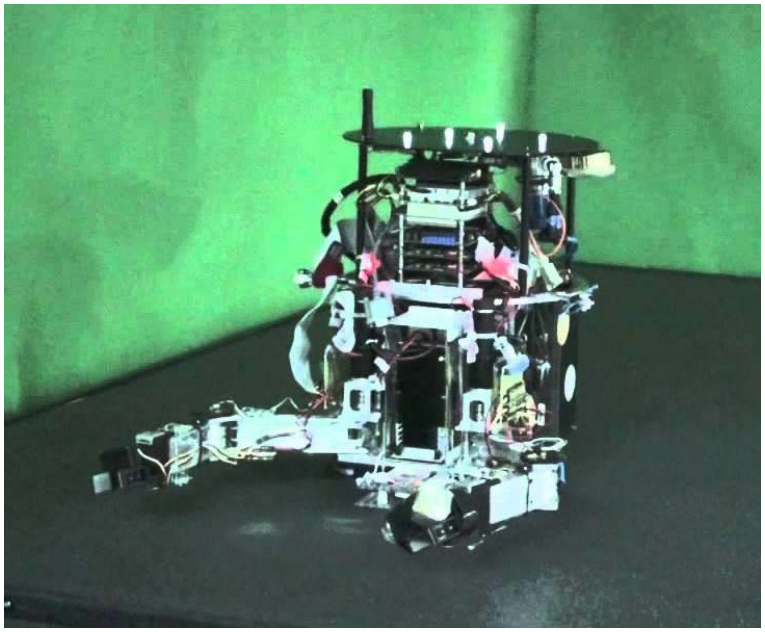
1. Τράπεζα γρανίτη, με τραχύτητα 5 μm.
2. Κάμερα, που χρησιμοποιείται από το κέντρο έλεγχου για τον οπτικό εντοπισμό των ρομπότ και άλλων αντικειμένων πάνω στη επιφάνεια διεξαγωγής του πειράματος. Καλύπτει όλη την επιφάνεια της τράπεζας και δίνει ακρίβεια ~1.5 mm.
3. Τον υπολογιστή που είναι κέντρο έλεγχου του πειράματος, είναι συνδεμένος με την κάμερα στέλνοντας στο ρομπότ τη θέση του, καθώς είναι και ο υπολογιστής που προγραμματίζει το ρομπότ, ελέγχοντας την ροή του προγράμματος.
4. Ρομπότ, που στυρίζονται κατακόρυφα με αεροέδρανα στην επιφάνεια της τράπεζας με πρακτικά μηδενική τριβή. Φέρουν εγκατάσταση CO_2 και μπαταρίες για την προώθηση τους και την λειτουργία των υπολοίπων συστημάτων έλεγχου αντίστοιχα. Υπάρχουν δυο ρομπότ συνολικά τα οποία φέρουν στο επάνω μέρος τους (LED) τα οποία βρίσκονται σε μοναδική διάταξη με σκοπό να αναγνωρίζονται από την κάμερα.

3.2 Λειτουργία

Σκοπός του εξομοιωτή είναι η δημιουργία περιβάλλοντος τροχιάς, δηλαδή κίνηση χωρίς εξωτερικές δυνάμεις και τριβές στο επίπεδο. Η ιδέα βασίζεται στη εξάλειψη της τριβής με σκοπό την δημιουργία ενός δισδιάστατου περιβάλλοντος μηδενικών τριβών. Αυτό επιτυγχάνεται με την χρήση αεροεδράνων πάνω σε λεία τράπεζα γρανίτη, τα οποία επιτρέπουν στα ρομπότ να αιωρούνται σε μια απόσταση 10-15 μm από την επιφάνεια του γρανίτη με αποτέλεσμα πρακτικά μηδενική τριβή. Επίσης με την επιπεδότητα της τράπεζας επιτυγχάνεται η πλήρης αντιστάθμιση της δύναμης της βαρύτητας χωρίς να δημιουργούνται συνιστώσες στις δυο άλλες κατευθύνσεις.

Με γνώμονα την εξάλειψη των εξωτερικών δυνάμεων, το ρομπότ απαιτείται να είναι πλήρως αυτόνομο χωρίς καμία μηχανική σύνδεση με σταθερό σημείο. Η απαίτηση αυτή οδήγησε στην μορφή του ρομπότ που φαίνεται στη Εικόνα 2. Το ρομπότ πλοηγείται στο χώρο με προωθητήρες αερίου CO_2, το οποίο φέρεται από το ρομπότ σε φιάλη υψηλής πίεσης. Επίσης για τον έλεγχο του προσανατολισμού του, υπάρχει και ένας αδρανειακός τροχός οδηγούμενος από ένα ηλεκτρικό κινητήρα, που συνεπικουρεί στην αλλαγή/διατήρηση προσανατολισμού. Τα δυο αυτά συστήματα ελέγχονται από έναν υπολογιστή μορφής PC104 εγκατεστημένο στο ρομπότ που τροφοδοτείται από δυο μπαταριές λίθου.

Για την πλοήγηση του ρομπότ χρησιμοποιούνται δυο συστήματα αναφοράς. Σε συμφωνία με τους πραγματικούς δορυφόρους ένα απόλυτο και ένα σχετικό. Το πρώτο υλοποιείται με την οπτική αναγνώριση του ρομπότ από μια κάμερα και το προσδιορισμό της θέσης του και του προσανατολισμού του σε συμφωνία με τους πραγματικούς δορυφόρους όπου υπολογίζουν την θέση τους με συστήματα οπτικής αναγνώρισης αστερισμών, ενώ το δεύτερο υλοποιείται με την χρήση οπτικών αισθητήρων από ποντίκια υπολογιστών, τα οποία δίνουν την σχετική κίνηση του ρομπότ και είναι σε συμφωνία με την χρήση των γυροσκοπικών αισθητήρων και μαγνητομέτρων που χρησιμοποιούνται στους δορυφόρους για την σχετική τους κίνηση.



Εικόνα 2. Ρομπότ εξομοιωτή.

3.3 Περιορισμοί

Από την παραπάνω περιγραφή του συστήματος γίνεται σαφές ότι η προσομοίωση συνθηκών διαστήματος περιέχει τους παρακάτω *περιορισμούς*:

- δισδιάστατη εξομοίωση
- περιορισμένη περιοχή εξομοίωσης 2200x1800mm
- Μέγιστος χρόνος πειράματος περιορισμένος λόγω εξάντλησης του CO_2 και της μπαταριάς.

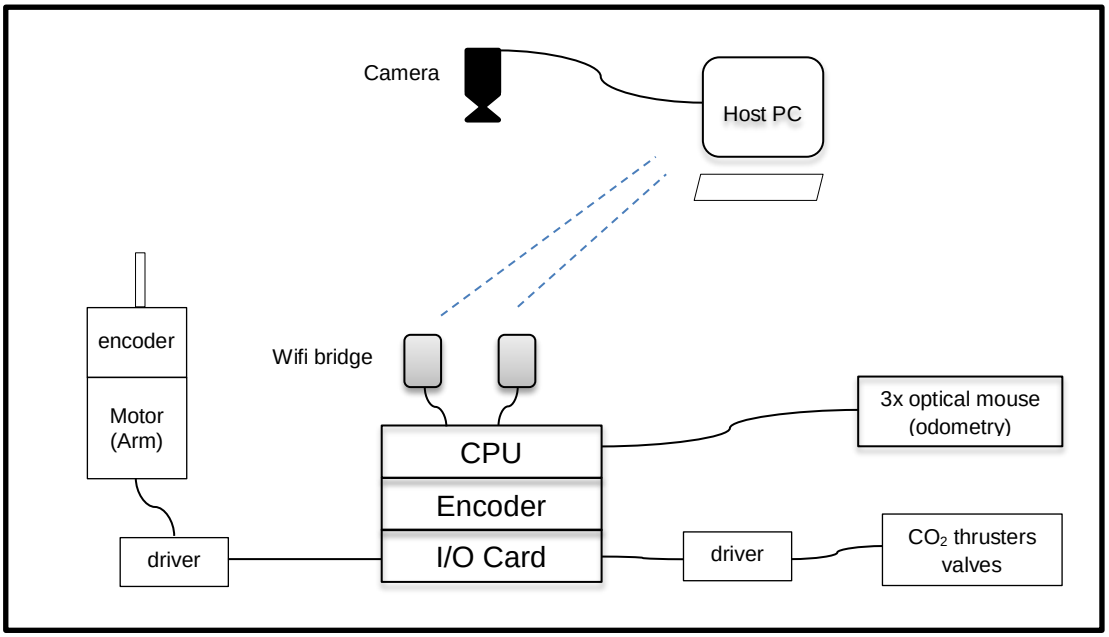
3.4 Εντοπισμός θέσης μέσω τεχνητής όρασης

Αναπόσπαστο κομμάτι κάθε ρομποτικού συστήματος είναι η γνώση της προς έλεγχο ιδιότητάς του. Ο σκοπός του εξομοιωτή είναι η μελέτη προβλημάτων επίπεδης δυναμικής άρα είναι επιθυμητή η γνώση της θέσης και του προσανατολισμού του ρομπότ σε κάθε χρονική στιγμή.

Για το σκοπό αυτό, μέρος του διαστημικού εξομοιωτή είναι και μια ψηφιακή κάμερα, τοποθετημένη επάνω από τον χώρο δράσης των ρομπότ, με στόχο τη λήψη διαδοχικών στιγμιότυπων αυτού. Με επεξεργασία των στιγμιότυπων αυτών, είναι δυνατός ο προσδιορισμός της θέσης και του προσανατολισμού των κινούμενων σωμάτων του διαστημικού εξομοιωτή (βλ. Σχήμα 3-1). Η μέθοδος αυτή ελέγχου ρομποτικών διατάξεων είναι γνωστή ως οπτική οδήγηση κλειστού βρόχου μέσω οπτικής ανάδρασης θέσης (visual servoing) και είναι αρκετά διαδεδομένη τα τελευταία χρόνια λόγω της αυξανόμενης ισχύος των υπολογιστικών συστημάτων και της κυκλοφορίας καμερών υψηλής ανάλυσης και χαμηλού κόστους.

Η μελέτη του συστήματος της κάμερας, η επιλογή, η αγορά και η τοποθέτηση των εξαρτημάτων έγινε από τον Ι. Κοντολάτη [9] στα πλαίσια της μεταπτυχιακής του

εργασίας. Εδώ θα αναφερθούν για λόγους πληρότητας τα βασικά χαρακτηριστικά του συστήματος και οι κύριοι λόγοι που αυτό έχει την παρούσα μορφή. Παρακάτω περιγράφεται συνοπτικά η διαδικασία εκλογής των στοιχείων του συστήματος, όπως έγινε από τον Ι. Κοντολάτη.



Σχήμα 3-1. Σχηματική αναπράσταση του εξομοιωτή.

3.5 Περιγραφή εγκατάστασης κάμερας

Αρχικά, η επιλογή του οπτικού αισθητήρα και του φακού της κάμερας έγιναν έτσι, ώστε τα στιγμιότυπα να καλύπτουν τον χώρο δράσης, η ανάλυση της εικόνας και η συχνότητα λήψης να είναι αποδεκτές και το κόστος να κυμαίνεται σε λογικά πλαίσια. Οπότε, οι φυσικοί παράμετροι της πειραματικής διάταξης έθεσαν τους περιορισμούς στην επιλογή του συστήματος της κάμερας αλλά και στον τρόπο στήριξής του. Πιο αναλυτικά, τα φυσικά χαρακτηριστικά είναι τα εξής: η τράπεζα, διαστάσεων 2200 x 1800 mm και βάρους 4 τόνων, είναι κατασκευασμένη από γρανίτη, με τραχύτητα επιφάνειας μικρότερη των 5 μ και αποτελεί την επιφάνεια κίνησης του ρομπότ. η άνω επιφάνεια της τράπεζας απέχει από την οροφή του χώρου περίπου 1800 [mm]. Η θέση της τράπεζας βρίσκεται κάτω από δοκό στήριξης του κτιρίου πάχους 300[mm] και ύψους 250 mm. η απόσταση της άνω επιφάνειας του γρανίτη από την επιφάνεια της δοκού είναι 1550 mm (βλ. Εικόνα 3).

Ο συνδυασμός του ύψους και της έκτασης της περιοχής δράσης οδήγησαν στην επιλογή ευρυγώνιου φακού. Ως ικανοποιητική ανάλυση καθορίστηκε εκείνη η οποία επιτρέπει τον καθορισμό της θέσης των αντικειμένων με ακρίβεια [mm], και είναι της τάξης των 4 εκατομμυρίων εικονοστοιχείων (4 Mpixels). Ο ελάχιστος επιθυμητός ρυθμός δειγματοληψίας στιγμιότυπων καθορίστηκε στο 1 fps (frame per second). Κρίθηκε ότι, το χρώμα δεν είναι απαραίτητη πληροφορία για την συγκεκριμένη εφαρμογή.

Εικόνα 3. Η ανάρτηση και η θέση της κάμερας στον εξομοιωτή.

Με βάση τα προαναφερθέντα, και με μεγαλύτερη βαρύτητα να έχει το κριτήριο του κόστους, επιλέχτηκε η κάμερα mvBlueFOX-124G της εταιρείας MATRIX VISION με αισθητήρα τύπου CCD, απεικόνισης στη κλίμακα του γκρι με μέγεθος στοιχείου 8bits, ανάλυσης 1600x1200 εικονοστοιχείων, ρυθμός λήψης 8 fps, πρωτόκολλο επικοινωνίας USB 1.1/2.0 (έως 480Mbits/sec), συμβατότητα με λειτουργικά συστήματα Microsoft Windows και Linux, και κόστος αγοράς 1640 € (βλ. Εικόνα 4). Πιο συγκεκριμένα ο αισθητήρας CCD είναι της εταιρείας Sony με κωδικό ICX274AL/AQ, διαγώνιο 8.923 mm, κατηγορίας 1/1.8”, με μέγεθος εικονοστοιχείου 4.4 x 4.4 μm. Επίσης, η δυνατότητα έκθεσης είναι από 50 μs έως 10 s με βήμα 1 μs. Η κάμερα συνοδεύεται από βιβλιοθήκη εντολών για τη ρύθμιση και τον έλεγχο πληθώρας παραμέτρων, έτοιμα προγράμματα ανοικτού κώδικα για τη κτήση και επεξεργασία εικόνας, αλλά και πραγματοποίηση ρυθμίσεων. Επιπλέον, διαθέτει εξωτερικό μεταλλικό περίβλημα με οπές για τη τοποθέτηση της σε βάση ενώ το βάρος της είναι 120gr. Τέλος, δέχεται οπτικούς φακούς τύπων C, CS και S προαιρετικά.

Εικόνα 4. Κάμερα Matrix Vision bluefox.

Ο οπτικός φακός που επιλέχτηκε είναι ο H2H0414C-MP της εταιρείας Computar, ο οποίος προορίζεται για χρήση σε κάμερες με αισθητήρα ανάλυσης εκατομμυρίων εικονοστοιχείων. Το κόστος του είναι 160 € και διαθέτει κυμαινόμενη εστιακή απόσταση f 4-8 mm για να επιτευχθεί η όσο το δυνατόν μεγαλύτερη κάλυψη της επιφάνειας δράσης με τη λιγότερη δυνατή παραμόρφωση. Το βάρος του είναι 72 gr, με διατομή 41.6 mm και μήκος 48.8 mm, ενώ είναι εδράσεις τύπου C. Ο φακός διαθέτει τρεις δακτυλίους για τη χειροκίνητη ρύθμιση της εστίασης (0,5 m ως άπειρο),

της εστιακής απόστασης (4 - 8 mm) και του ανοίγματος του κλείστρου (F1.4 – F16C). Η ανάλυση των 2MP παρέχει την ακρίβεια που απαιτείται, ενώ η ταχύτητα των 480Mbits/s που επιτυγχάνει το πρότυπο USB2.0 σε συνδυασμό με τις δυνατότητες του αισθητήρα CCD δίνουν ρυθμό ανανέωσης της εικόνας 8 fps, τιμή ικανοποιητική για την συγκεκριμένη εφαρμογή.

3.6 Το υπάρχον σύστημα

Το σύστημα οπτικής αναγνώρισης είναι κομμάτι της διπλωματικής εργασίας του Α. Καλγρεάδη [10] το όποιο έχει υποστεί μεταγενέστερες βελτιώσεις και τροποποιήσεις από τον Ι. Κοντολάτη [9] και Κ. Μαχαιρά [11].

Το σύστημα αποτελείτε από έναν υπολογιστή στο οποίο συνδέεται η κάμερα που παρατηρεί τα ρομπότ και ο οποίος τρέχει το λογισμικό που δημιουργήθηκε από τους παραπάνω. Το λογισμικό αυτό είναι γραμμένο σε γλώσσα C και αναλαμβάνει:

- το άνοιγμα και τη ρύθμιση της κάμερας,
- τη λήψη εικόνας,
- την αποπαραμόρφωση της εικόνας,
- την επεξεργασία εικόνας (image processing) για εύρεση του ρομπότ,
- τον υπολογισμό θέσης,
- την αποστολή θέσης μέσω δικτύου στο ρομπότ.

Το σύστημα προϋποθέτει επίσης την τοποθέτηση φωτεινών πηγών LED στα ρομπότ με συγκεκριμένη διάταξη, με σκοπό την αναγνώριση τους από το λογισμικό και εξαγωγή συμπερασμάτων για τη θέση και προσανατολισμό των ρομπότ.

3.6.1 Προβλήματα

Για την καλύτερη και σφαιρικότερη γνώση του προβλήματος οπτικής αναγνώρισης, όχι μόνο ως προς την θεωρία αλλά και την υλοποίηση, έγινε εκτενής χρήση του υπάρχοντος λογισμικού. Το παραπάνω σύστημα εμφάνιζε πολλά μειονεκτήματα και προβλήματα που παρουσιάζονται παρακάτω:

1. **Ακρίβεια**, το μεγαλύτερο πρόβλημα που εμφανίζεται στην υλοποίηση αυτή είναι η αφαίρεση της παραμόρφωσης που εισάγεται από τον ευρυγώνιο φακό ο οποίος είναι απαραίτητος για να παρατηρείται ολόκληρος ο χώρος εργασίας. Το υπάρχον λογισμικό δε αφαιρεί σε ικανοποιητικό βαθμό την παραμόρφωση, το σφάλμα διαφέρει σημαντικά με την θέση του ρομπότ (αυξάνει από το κέντρο της εικόνας προς τα άκρα και δεν υπάρχει μαθηματική εκτίμηση του παρά μόνο πειραματικά.
2. **Εποπτεία,** το λογισμικό δεν παρέχει στο χρήστη καμία πληροφορία για την λειτουργία του, με αποτέλεσμα να μην γνωρίζουμε την κατάσταση λειτουργίας και τις επιδόσεις του, ενώ η ρύθμιση των εσωτερικών παραμέτρων του καθίσταται αδύνατη. Επίσης ο κώδικας του είναι πολύ μεγάλος και μη βελτιστοποιημένος.

3. **Δυσκολία χειρισμού,** δεν υπάρχει γραφικό περιβάλλον αλληλεπίδρασης με το χρηστή (GUI), δεν παρέχει καμία παραμετροποίηση ή ρύθμιση για την λήψη εικόνας, την επεξεργασία και την αποστολή δεδομένων.
4. **Επιδόσεις,** ο ρυθμός ανανέωσης είναι 2,5 Hz ενώ η κάμερα έχει δυνατότητα έως 8 Hz. Επίσης παρατηρήθηκε μεγάλος χρόνος απόκρισης και μη εκμετάλλευση των σύγχρονων μεθόδων όπως, του αντικειμενοστραφούς προγραμματισμού, και την αρχιτεκτονική πολλών πυρήνων.
5. **Αξιοπιστία,** ευπάθεια στις ανακλάσεις, στις συνθήκες φωτισμού στον αριθμό των LED και στη περιοχή κίνησης των ρομπότ, κατά την όποια η αξιοπιστία μειωνόταν όσο το ρομπότ μετακινείται προς τα άκρα της εικόνας.
6. **Περιορισμοί,** ο αριθμός των ρομπότ είναι προκαθορισμένος και περιορισμένος σε δυο. Το μοντέλο αποπαραμόρφωσης αποτελεί κομμάτι του κώδικα και δεν μπορεί να μελετηθεί ανεξάρτητα.
7. **Λογική υλοποίησης/αναγνώρισης,** ο τρόπος εύρεσης των LED και στη συνεχεία των ρομπότ είναι πολύ χρονοβόρος και αδόμητος καθώς με αποτέλεσμα να απαιτεί περισσότερο υπολογιστικό χρόνο. Επίσης χρησιμοποιείται φίλτρο half toning χωρίς να είναι απαραίτητο ενώ προσθέτει σημαντικό αριθμό πράξεων.

3.7 Η νέα υλοποίηση

Με βάση τις παραπάνω παρατηρήσεις η βελτίωση του υπάρχοντος κώδικα (που θα απαιτούσε σημαντικό χρόνο για την κατανόηση του, λόγω κακής δόμησης) δεν αποτέλεσε ρεαλιστική επιλογή. Έτσι κρίθηκε σκοπιμότερη η εκ νέου σχεδίαση του αλγορίθμου. Με αφορμή αυτήν την επιλογή έγινε μια διερεύνηση, για τον προσδιορισμό των δυνατοτήτων και περιορισμών του υλικού και λογισμικού που θα χρησιμοποιηθεί και είναι διαθέσιμο, προτού καθοριστούν στόχοι και προδιαγραφές για το νέο λογισμικό. Η διερεύνηση αυτή αναλύεται στις ακόλουθες παραγράφους.

3.7.1 Επιλογή της C# για γλώσσα προγραμματισμού

Διερευνήθηκε πρώτα από όλα η επιλογή γλώσσας προγραμματισμού αφού θα καθορίσει με την σειρά της άλλες επιλογές. Έχοντας τις επιλογές VB, C#, C/C++ ως υποστηριζόμενες γλώσσες προγραμματισμού για την έτοιμη βιβλιοθήκη που προσφέρει η εταιρεία MATRIX VISION της κάμερας απορρίπτουμε τις C/C++ επειδή οι δυο πρώτες είναι σύγχρονες γλώσσες υψηλού επιπέδου που υποστηρίζουν αντικειμενοστραφής (object-oriented) προγραμματισμό που θα επιταχύνει, θα μειώσει και θα απλοποίησει την δομή του κώδικα. Επιπροσθέτως προσφέρουν πληθώρα εργαλείων για την ευκολία δημιουργία γραφικού περιβάλλοντος, ενώ ο προγραμματισμός σε αυτές τις γλώσσες είναι πολύ ευκολότερος μια και υποστηρίζονται από πολύ εξελιγμένα περιβάλλοντα προγραμματισμού τα όποια προτείνουν και διορθώνουν δυναμικά λάθη του προγραμματιστή. Ακόμη είναι ευκολότερη η χρήση εξωτερικών βιβλιοθηκών, όπως στην περίπτωση μας και ο τρόπος σύνταξης τους διευκολύνει και προάγει τον προγραμματισμό σε ανεξάρτητα κομμάτια το οποίο είναι σαφώς ευκολότερο, δίνει καλύτερη εποπτεία και δίνει την δυνατότητα για μελλοντικές προσθήκες χωρίς να απαιτείτα να κατανοήσει ο σχεδιαστής ολόκληρο τον κώδικα, παρά μόνο το επιμέρους τμήμα που πρόκειται να

τροποποιήσει. Λαμβάνοντας υπ' όψιν τα μειονεκτήματα που περιγράφτηκαν στην παράγραφο 3.6.1 η VB και C# συμβάλουν σημαντικά στην εξάλειψη κάποιων από αυτά. Αν και ισοδύναμες, τελικά επιλέχτηκε η C# ως πιο οικία γλωσσά (C/C++).

3.7.2 Εξοικείωση με την γλώσσα προγραμματισμού και την βιβλιοθήκη της κάμερας

Με την επιλογή της C# ως γλώσσα προγραμματισμού. Κρίθηκε σκόπιμο ως δεύτερο σταδιο να διερευνηθεί, η παρεχομένη από την MATRIX VISION βιβλιοθήκη [12] καθώς και τεχνικές προγραμματισμού της C# [13], με σκοπό την απόκτηση εμπειρίας γύρω από τα παρεχόμενα εργαλεία ώστε να αποφευχθούν μη υλοποιήσιμες ιδέες αλλά και να μπορέσουμε να προσαρμόσουμε τις ανάγκες μας στα υπάρχοντα εργαλεία για βελτιστοποίηση του χρόνου εκτέλεσης, την αύξηση της αξιοπιστίας μέσω σωστών τεχνικών αλλά και την μείωση του χρόνου που απαιτείται από τον προγραμματιστή για την υλοποίηση του λογισμικού.

Στα πλαίσια της εξοικείωσης με τη C# και το περιβάλλον προγραμματισμού της (Visual Studio), έγιναν οι παρακάτω στοχευμένες δοκιμές που εμπνεύστηκαν από μεμονωμένα προβλήματα που περιγράφονται στην Παράγραφο 3.6.1:

- Υλοποίηση γραφικού περιβάλλοντος που τρέχει παράλληλα με άλλο κώδικα υπολογισμών σε διαφορετικό πυρήνα,
- Εκτενείς δοκιμές με την έννοια του αντικειμενοστραφή προγραμματισμού και την αντιστοίχιση με φυσικά αντικείμενα,
- Χρήση ενσωματωμένων εργαλείων debugging (εύρεσης και αναγνώρισης λαθών),
- Διατήρηση τιμών και παραμέτρων του γραφικού περιβάλλοντος και μετά το κλείσιμο του προγράμματος,
- Υλοποίηση επικοινωνίας μέσω τοπικού δικτύου με τα πρωτοκόλλα TCP/UDP.

Στα πλαίσια της εξοικείωσης με την Βιβλιοθήκη της MATRIX VISION, έγιναν οι παρακάτω δοκιμές:

- Μια προς μια μελέτη των παρεχόμενων κλάσεων και συναρτήσεων (λειτουργία και σύνταξη),
- Χρονομέτρηση των συναρτήσεων που επιστρέφουν την εικόνα,
- Δυνατότητα τροποποίησης παραμέτρων λήψης εικόνας όπως: συχνότητα ρολογιού, χρόνος έκθεσης, τρόπος λήψης εικόνας (επικαλυπτόμενη, συνεχής, κατά απαίτηση),
- Δημιουργία αρχείων [.xml] ρυθμίσεων και φόρτωση τους στη κάμερα.

Η παραπάνω διερεύνηση μας οδήγησε στα παρακάτω **<u>συμπεράσματα</u>**:

- Η χρήση των συναρτήσεων που επιστρέφουν την εικόνα σε μορφή πινάκα είναι πολύ χρονοβόρες και πρέπει σε κάθε περίπτωση κατά το σχεδιασμό του νέου αλγορίθμου να περιορισθεί η χρήση των συναρτήσεων σε ελάχιστες φορές,
- Βρέθηκε κλάση που συλλέγει στατιστικά για την λειτουργία της κάμερας, άρα να συμπεριληφθεί στην υλοποίηση,

- Βρέθηκε ότι είναι εφικτό να αλλαχτούν οι παράμετροι λήψης στιγμιότυπου αλλά θα πρέπει να δοθούν σε μορφή αρχείου [*.xml]. Άρα πρέπει να γίνει η αντίστοιχη υλοποίηση στο γραφικό περιβάλλον του χρήστη,
- Εντοπίστηκε συνάρτηση που εμφανίζει ξεχωριστό παράθυρο που δείχνει την εικόνα που επεξεργάζεται ο αλγόριθμος χωρίς υπολογιστικό κόστος και Επιπρόσθετα βρέθηκε μια κλάση που επιτρέπει τη σχεδίαση βασικών σχημάτων πάνω στην εικόνα. Αυτά έδωσαν την ιδέα για οπτική ενημέρωση του χρήστη για το εάν έχουν βρεθεί τα ρομπότ και αν έχουν αναγνωριστεί σωστά.

Με βάση την παραπάνω διερεύνηση καθορίστηκαν οι παρακάτω **σχεδιαστικοί στόχοι** για την αλληλεπίδραση του λογισμικού με τον χρηστή και την επικοινωνία με την κάμερα.

- Δημιουργία γραφικού περιβάλλοντος πινάκα έλεγχου για την επιλογή παραμέτρων του κώδικα αναζήτησης ρομπότ και του τρόπου επικοινωνίας ρομπότ-υπολογιστή,
- Απομνημόνευση των παραμέτρων που έχει επιλέξει ο χρήστης,
- Έλεγχο ροής της διαδικασίας αναγνώρισης και ενημέρωση του χρήστη για την λειτουργιά του και τις επιδόσεις του, όπως π.χ. την συχνότητας δειγματοληψίας και την υστέρηση των δεδομένων.
- Εμφάνιση σε πραγματικό χρόνο των στιγμιότυπων που λαμβάνει η κάμερα σε ξεχωριστό παράθυρο και οπτική επισήμανση των εντοπισμένων αντικειμένων.

3.7.3 Εισαγωγή στην επεξεργασία εικόνας

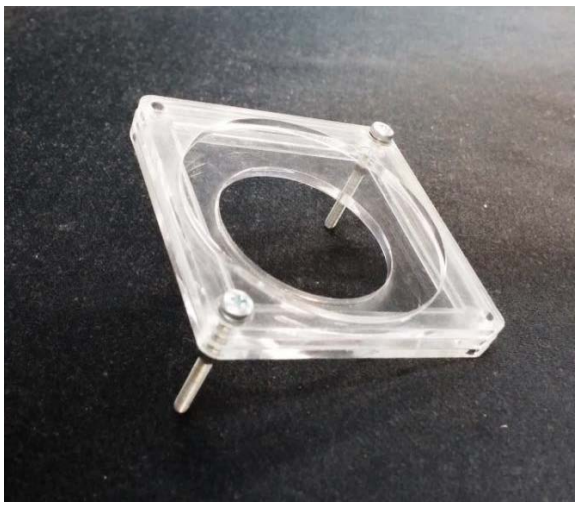
Στον υπάρχοντα κώδικα, η αναγνώριση των ρομπότ επί της τράπεζας γρανίτη γίνεται με την βοήθεια φωτεινών πηγών LED που είναι τοποθετημένα επάνω στα ρομπότ. Ο αλγόριθμος στην πραγματικότητα εντοπίζει αυτά τα LED και στην συνέχεια αποφασίζει εάν αυτά τα LED ανήκουν ή όχι σε κάποιο ρομπότ. Έτσι η διαδικασία της αναγνώρισης θέσης διαιρείται σε δυο μέρη, το πρώτο είναι η αναγνώριση των LED και το δεύτερο η ταυτοποίηση των LED με αυτά ενός ρομπότ. Τα δυο μέλη όπως θα αναλυθούν ξεχωριστά παρακάτω έχουν μεμονωμένα προβλήματα τα οποία περιγράφθηκαν στην Παράγραφο 3.6.1. Είναι φανερό όμως ότι η διαδικασία αναγνώρισης ρομπότ βασίζεται στη σωστή αναγνώριση των LED. Για την αντιμετώπιση του συγκεκριμένου προβλήματος εξετάστηκαν αρχικά δυο εναλλακτικές λύσεις.

Η πρώτη ήταν η χρήση βιβλιοθηκών τεχνητής όρασης έτσι ώστε το ρομπότ να εντοπίζεται σαν αντικείμενο χωρίς την βοήθεια των LED με πλεονέκτημα της ύπαρξης μόνο μιας διαδικασίας αναγνώρισης. Ο τρόπος αυτός όμως είχε τα εξής μειονεκτήματα:

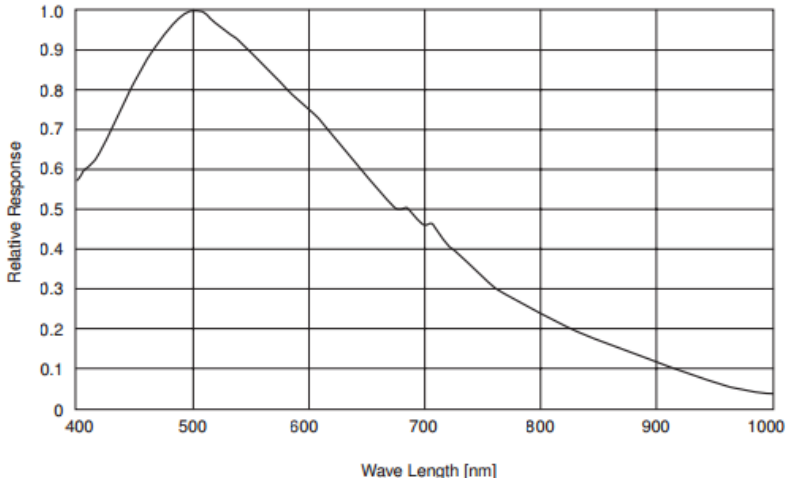
- Υπολογιστικό βάρος
- Μεγάλη πολυπλοκότητα
- Ανακριβής προσδιορισμός γεωμετρικών στοιχείων (π.χ. κέντρο ρομπότ)
- Μικρή αξιοπιστία χωρίς την τροποποίηση των ρομπότ
- Απαίτηση συγκεκριμένης μορφής ρομπότ

Η δεύτερη επιλογή που εξετάστηκε ήταν η χρήση υπέρυθρων LED σε συνδυασμό με φίλτρο ορατού φωτός στην κάμερα με σκοπό την μείωση της πληροφορίας που λαμβάνει η κάμερα από το περιβάλλον ώστε να αυξήσουμε την αξιοπιστία του κομματιού αναγνώρισης LED. Η ιδέα αυτή αποτέλεσε εναλλακτική αφού επιβεβαιώθηκε ότι ο φωτισμός του εργαστηρίου δε παράγει υπέρυθρο φως εφόσον παράγεται από λάμπες φθορίου όπου το φάσμα του εκπεμπόμενου φωτός που φαίνεται στο Σχήμα 3-4 περιορίζεται μόνο στη ορατή ακτινοβολία. Για την αξιολόγηση αυτής της ιδέας μελετήθηκε η ευαισθησία του αισθητήρα της κάμερας σε υπέρυθρο φωτισμό δηλαδή σε τιμές άνω των 700 nm. Όπως φαίνεται στο Σχήμα 3-2 η ευαισθησία της κάμερας πέρα από το οπτικό φάσμα πέφτει ραγδαία, επομένως πρέπει να επιλεχτεί ένας συνδυασμός φίλτρου και υπέρυθρου LED κοντά στα 700 [nm] (dark red light) , διαφορετικά θα απαιτείται πολύ μεγάλη ισχύς φωτισμού που δεν είναι διαθέσιμη αφού το ρομπότ τροφοδοτείται από μπαταρία.

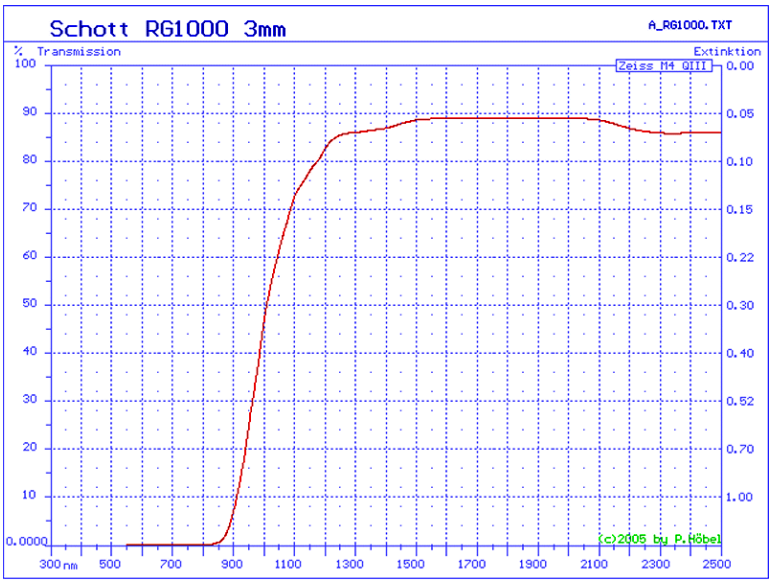
Η έρευνα αγοράς που έγινε έδειξε ότι υπάρχουν LED στα 740, 850 και 940 nm με σχετικά προσιτό κόστος από την άλλη μεριά όμως δε βρέθηκαν φίλτρα κοντά στα 700 nm πάρα μόνο ειδικής παραγγελίας με απαγορευτικό κόστος. Τα περισσότερα φίλτρα υπέρυθρων που υπάρχουν στο εμπόριο είναι διαπερατά άνω των 900 nm τιμή για την οποία έχουμε 10% ευαισθησία. Παρόλα αυτά, πραγματοποιήθηκε μια δοκιμή με φίλτρο 900 nm και δυο LED των 940 nm με σκοπό την απόκτηση εμπειρίας όσον αφορά την ευαισθησία του αισθητήρα που από το διάγραμμα είναι περίπου 5% για 940 nm. Για αυτό το λόγο κατασκευάστηκε μια διάταξη υπέρυθρων LED χαμηλής ισχύος μήκους κύματος 940 [nm] πάνω σε μια διάτρητη πλακέτα και μια κατασκευή από plexiglass για την γρήγορη στήριξη του φίλτρου RG1000 (50mm x50mm) στο φακό της κάμερας (βλ. Εικόνα 5). Το πείραμα είχε σχετική επιτυχία αφού τα LED ήταν τα μόνα ορατά σημεία της εικόνας αλλά όχι στην απόσταση λειτουργίας (~1m) με την ονομαστική τους ισχύ 1 w. Το συμπέρασμα είναι ότι είναι εφικτό με LED υψηλότερης ισχύος αλλά τέτοια LED δε είναι ευρέως διαθέσιμα και θα έκαναν την κατασκευή οποιουδήποτε αντικειμένου που πρέπει να αναγνωρίζεται από την κάμερα πολύ πιο περίπλοκη



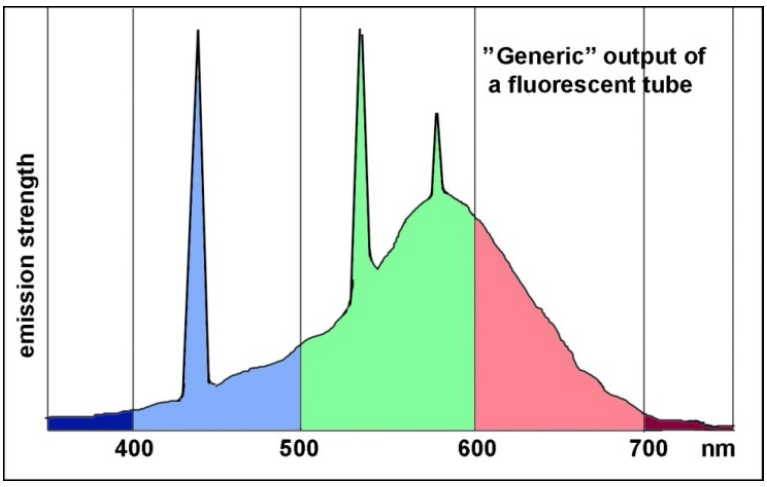
Εικόνα 5. Κατασκευή για την προσαρμογή φίλτρου υπερύθρων στην κάμερα.



Σχήμα 3-2. Ευαισθησία αισθητήρα κάμερας σε οπτική ακτινοβολία.



Σχήμα 3-3. Διαπερατότητα φίλτρου υπέρυθρων.



Σχήμα 3-4. Φάσμα φωτός λάμπας φθορίου.

Με τις παραπάνω εναλλακτικές να μην έχουν σημαντικά πλεονεκτήματα έναντι της υπάρχουσας προσέγγισης, επιλέχθηκε να ακολουθηθεί η υπάρχουσα αλλά με σημαντικές βελτιώσεις στον τομέα της υλοποίησης που αναλύονται παρακάτω.

3.7.4 Το σκέλος αναγνώρισης LED

Αρχικά θα σχολιασθεί ο υπάρχων κώδικας, θα επισημανθούν τα τμήματα που χρήζουν τροποποίησης και στη συνεχεία θα παρουσιαστεί ο νέος κώδικας που σχεδιάστηκε και πως αυτός αντιμετωπίζει τα προβλήματα του προηγούμενου. Τέλος αυτοί θα συγκριθούν έτσι ώστε να εκτιμηθεί η τελική βελτίωση που έγινε στο όλο σύστημα.

Στον υπάρχοντα κώδικα για την αναγνώριση των LED, ως μια περιοχή φωτεινών πίξελ, εντοπίζονταν πρώτα όλα αυτά τα πίξελ της εικόνας τα οποία πιθανόν να ανήκουν σε κάποιο από τα LED και μετά συγκρίνονταν όλες οι πιθανές αποστάσεις μεταξύ τους με σκοπό να εντοπιστούν τα γειτονικά και να θεωρηθούν ως ένα LED. Αυτό δε είναι αποτελεσματικό γιατί:

- Για να εντοπιστούν όλα τα πίξελ σαρώνεται όλη η εικόνα. Όπως όμως βρέθηκε από την διερεύνηση που έγινε, απαιτεί σημαντικό χρόνο με αποτέλεσμα τη μείωση του χρόνου ανανέωσης αυτό της εικόνας.
- Με δεδομένο ότι κάθε LED καταλαμβάνει κατά μέσο όρο 50 πίξελ, για ένα τυπικό πείραμα που απαιτεί δυο ρομπότ θα υπάρχουν στη εικόνα τουλάχιστον 200 πιξελ. Αυτό μας δίνει 200^2 αποστάσεις που πρέπει να υπολογιστούν και να συγκριθούν. Στην πραγματικότητα αν και απαιτούνται 40000 μαθηματικές πράξεις όπως η τετραγωνική ρίζα που είναι υπολογιστικά χρονοβόρες αυτός ο αριθμός δε είναι απαγορευτικός. Το πρόβλημα χειροτερεύει όταν η κάμερα εντοπίσει ανακλάσεις από το φωτισμό που συνήθως είναι μεγάλες περιοχές με μέσο όρο 2000 πιξελ, αυξάνοντας της πράξεις σε 2200^2 Μια τέτοια περίπτωση είναι αρκετά πιθανή και δημιουργεί αισθητή μείωση στην ταχύτητα εκτέλεσης.
- Δε έχουμε κανένα δεδομένο για τα LED όπως πλάτος ύψος μέγεθος, δεδομένα που μπορούν χρησιμοποιηθούν για την απόφαση του αλγορίθμου σχετικά με το εάν το σύνολο αυτό είναι κάποιο LED ή κάποια ανάκλαση.

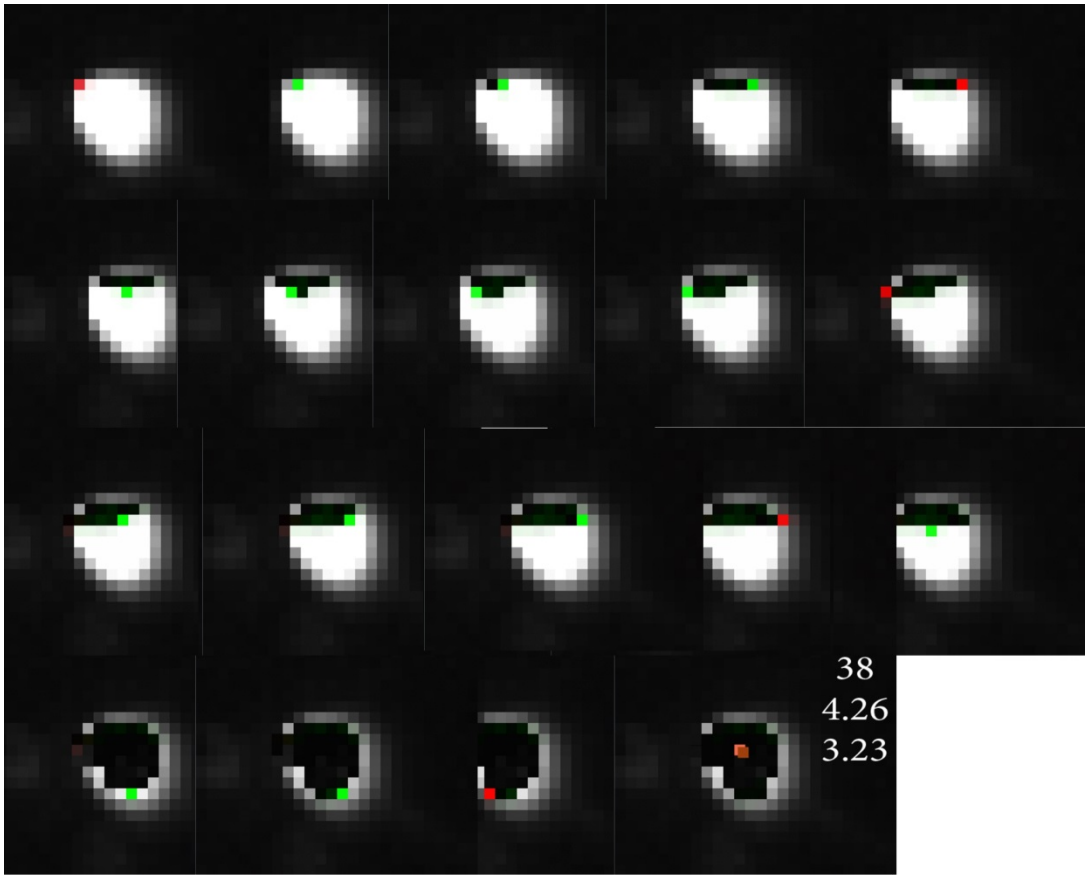
Για την αποφυγή των παραπάνω μειονεκτημάτων και έχοντας υπ’ όψιν αυτά της Παραγράφου 3.6.1 σχεδιάστηκε νέος κώδικας, ο οποίος σαρώνει την εικόνα από αριστερά προς τα δεξιά και από πάνω προς τα κάτω όπως και ο προηγούμενος, αλλά ξεκινώντας με κάποια περιοχή της εικόνας που έχει θέσει ο χρήστης αποκλείοντας εξ’ αρχής κάποιες περιοχές στα περιθώρια της εικόνας, όπου η κάμερα βλέπει το χώρο του εργαστηρίου και όχι το χώρο εργασίας των ρομπότ. Έτσι μειώνεται η περιοχή αναζήτησης άρα και ο χρόνος επεξεργασίας καθώς μειώνεται και σημαντικά η πιθανότητα εσφαλμένης αναγνώρισης LED αφού στις περιοχές αυτές έχουμε τις περισσότερες ανακλάσεις από αντικείμενα και εξοπλισμό που βρίσκεται μέσα στο εργαστήριο. Αναζητά πιξελ με φωτεινότητα μεγαλύτερη μιας δεδομένης τιμής (καθοριζόμενη από τον χρήστη). Βρίσκοντας ένα τέτοιο, εκκινεί η διαδικασία αναγνώρισης ενός LED (βλ. Εικόνα 6). Η οποία είναι η συνεχή αναζήτηση γειτονικών πίξελ που είναι εξίσου φωτεινά και όχι συγκρίνοντας τις αποστάσεις όλων των φωτεινών πίξελ τις εικόνας. Πρακτικά μηδενικό υπολογιστικό κόστος.

Έτσι λοιπόν με το που εντοπιστεί ένα φωτεινό LED διακόπτεται η σάρωση της εικόνας και δημιουργείται ένα υποψήφιο αντικείμενο LED και με αρχή τις συντεταγμένες του πίξελ αυτού εντοπίζει μια ευρύτερη συνεχή περιοχή φωτεινών πίξελ με τεχνική παρόμοια του αλγορίθμου *flood fill* αλλά τροποποιημένη για να γνωρίζουμε τον αριθμό πίξελ, το πλάτος, το ύψος και το γεωμετρικό κέντρο σε συντεταγμένες εικόνας αυτές μετατρέπονται άμεσα και σε παγκόσμιες συντεταγμένες στο χώρο εργασίας. Αυτή η μετατροπή θα αναλυθεί στο τέλος αυτής της παραγράφου. Θέτοντας κάποια κριτήρια για τις παραπάνω χαρακτηρίστηκες τιμές του υποψήφιου LED, ο αλγόριθμος το απορρίπτει αλλιώς το αποθηκεύεται σε ενα κατάλογο που περιλαμβάνει όλα τα πραγματικά LED.

Ο αλγόριθμος συνεχίζει από εκεί που είχε διακοπεί αναζητώντας νέα υποψήφια LED. Αυτή η διαδικασία επαναλαμβάνεται τόσες φορές όσες και τα LED που βρίσκονται μπροστά από την κάμερα είτε ανήκουν σε κάποιο ρομπότ είτε είναι απλές ανακλάσεις φωτεινών αντικειμένων του εργαστηρίου. Αφού περάσει η διαδικασία αναγνώρισης ρομπότ που ακολουθεί (θα περιγραφεί στη συνέχεια), κάποια από τα LED της παραπάνω λίστας θα αντιστοιχιστούν σε κάποιο ρομπότ, ιδανικά όλα. Όταν λοιπόν η όλη διαδικασία θα επαναληφθεί, ο αλγόριθμος θα αναζητήσει μόνο τα LED που έχουν αντιστοιχιστεί σε ρομπότ σαρώνοντας μια τετράγωνη περιοχή της εικόνας με κέντρο την προηγούμενη θέση του LED και πλάτος και ύψος που επιλέγεται από τον χρήστη (αναφέρεται ως smart scan). Με αυτήν την τεχνική μειώνουμε ραγδαία το υπολογιστικό βάρος όπου από 1600x1200 πράξεις περιορίζονται σε 50x50x4 ποσοστιαία μείωση των πράξεων κατά 99,47%. Επίσης με αυτό τον τρόπο, φωτεινές περιοχές που πιθανόν να θεωρήθηκαν LED, εφόσον δε αντιστοιχήθηκαν σε ρομπότ δε σαρώνονται και άρα δεν ‘’βαραίνουν’’ υπολογιστικά τον αλγόριθμο. Πιο συγκεκριμένα ο αλγόριθμος χωρίς αυτήν την τεχνική αναζήτησης έχει μέγιστο ρυθμό τα 2 Hz ενώ με αυτήν φτάνει στο όριο ανανέωσης της κάμερας όπου με ιδανικές ρυθμίσεις είναι 17,8 hz. Σε Αντιπαράθεση με τον προηγούμενο κώδικα που είχε 2,5 hz Έχουμε Αύξηση του ρυθμού ανανέωσης κατά 612%

Πίνακας 1. Σύγκριση επιδόσεων παλαιού με νέο αλγόριθμο

	Ρυθμός ανανέωσης	Αριθμός πράξεων	Αποσφαλμάτωση	Δομή κώδικα
Παλαιός αλγόριθμος	2	1.920.000	καμία	Συνεχής/ πεπλεγμένη
Νέος αλγόριθμος	17.8	10.000	Βασισμένη σε σχήμα & μέγεθος	Εναλλάξιμα τμήματα



Εικόνα 6. Διαδικασία αναγνώρισης LED (διαδοχικά προς τα δεξιά και κάτω).

Η αποπαραμόρφωση και μετατροπή των συντεταγμένων από πίξελ σε μονάδες απόστασης στο σύστημα συντεταγμένων του γρανίτη περιγράφεται στην Παράγραφο 4. Η διαδικασία αυτή όπως θα δούμε στη συνέχεια είναι υπολογιστικά μη αποδέκτη γιατί απαιτεί αριθμητική μέθοδο για τον υπολογισμό των συντεταγμένων που κατά μέσο όρο απαιτεί 17 επαναλήψεις. Προφανώς κάτι τέτοιο δε μπορεί να γίνει σε πραγματικό χρόνο για κάθε LED που υπάρχει στην εικόνα. Ο τρόπος που επιλέχθηκε είναι να δημιουργηθεί ένας πινάκας αντιστοίχισης των πίξελ σε αντίστοιχες πραγματικές συντεταγμένες στο χώρο. Με αυτό τον τρόπο αποφεύγονται οι περισσότερες αριθμητικές πράξεις και πλέον, η ακρίβεια του λογισμικού είναι πλήρως ανεξάρτητη από αυτόν αφού όλη η μοντελοποίηση και λύση του προβλήματος έχει γίνει εκτός βρόχου και οπότε μπορεί να τροποποιηθεί και να βελτιωθεί χωρίς να είναι απαραίτητη η γνώση προγραμματισμού και η τροποποίηση του κώδικα του λογισμικού. Η δημιουργία αυτού του πίνακα έγινε με την χρήση μιας βιβλιοθήκης του MATLAB ‘’*Camera Calibration Toolbox for Matlab’’* [14] που έχει δημιουργηθεί από την INTEL. Με χρήση των εργαλείων της δημιουργήθηκαν δύο πίνακες ο *‘’fileX.txt’’* και *‘’fileY.txt’’* με διάσταση 1200 γραμμές και 1600 στήλες (όση και η ανάλυση της κάμερας) όπου δίνουν σε χιλιοστά την τετμημένη και την τεταγμένη αντίστοιχα ενός σημείου που βρίσκεται 430 mm επάνω από την επιφάνεια του γρανίτη. Στην ουσία έχει ευρεθεί μια μοναδική θέση στο πραγματικό σύστημα για κάθε σημείο στην εικόνα υποθέτοντας ότι το πραγματικό σημείο στο χώρο ανήκει στο επίπεδο 430 mm πάνω από την επιφάνεια του γρανίτη.

3.7.5 Αναγνώριση Ρομπότ

Η αναγνώριση και ο εντοπισμός των ρομπότ επεται της διαδικασίας εντοπισμού των LED κατά την οποία τα LED που έχουν ευρεθεί θεωρείται ότι είναι υποψήφια LED κάποιου ρομπότ. Αρχικά ο αλγόριθμος υπολογίζει με βάση τον αριθμό των καταχωρημένων ρομπότ πόσα LED πρέπει να υπάρχουν τουλάχιστον ώστε να μπορούν να εντοπιστούν τα ρομπότ και τα συγκρίνει με τα ευρεθέντα, αν ο αριθμός επαρκεί συνεχίζει. Συγκρίνει τις αποστάσεις σε mm μεταξύ τους, εάν βρεθούν τα LED που αντιστοιχούν σε ένα ρομπότ τότε δημιουργείται μια οντότητα ρομπότ. Η απόσταση των LED κάθε ρομπότ αποτελεί το χαρακτηριστικό ταυτοποίησης τους από τον αλγόριθμο και πρέπει να είναι μοναδικό. Στη συνέχεια καταχωρείται το μεγαλύτερο από τα δύο LED ως κεντρικό και ύστερα χαρακτηρίζονται και τα δύο ως αντιστοιχισμένα και ενημερώνεται η εικόνα ότι έχει ευρεθεί το ρομπότ κυκλώνοντας το κέντρο του με ένα λευκό κύκλο (βλ. Εικόνα 7). Παράλληλα έχουν ευρεθεί η θέση του και ο προσανατολισμός του, και αντίστοιχα η γραμμική και γωνιακή του ταχύτητα. Στο επόμενο κύκλο το λογισμικό δεν ξανά ακλουθεί την ιδία διαδικασία. Ο αλγόριθμος αναγνώρισης LED έχει τρέξει και έχει ενημερώσει της θέσεις των LED που είναι αντιστοιχισμένα επομένως ο Αλγόριθμος αναγνώρισης ρομπότ υπολογίζει μοναχά τη νέα θέση και προσανατολισμό του ρομπότ και πραγματοποιεί έναν έλεγχο της απόστασης των LED του κάθε ρομπότ για να πιστοποιήσει ότι είναι ακόμα το σωστό ρομπότ. Σε περίπτωση που κάποιο από αυτά τα LED φύγει από την περιοχή που σκανάρει ο αλγόριθμος, επειδή για παράδειγμα το ρομπότ κινήθηκε πολύ γρήγορα ή επειδή το LED έσβησε, τότε ο αλγόριθμος αναγκάζεται να επαναλάβει την διαδικασία από την αρχή αναζητώντας τα LED χωρίς γνώση της θέσης τους και αναζητά εκ νέου τα ρομπότ ενημερώνοντας το χρήστη για το συμβάν.

Εικόνα 7. Κύκλος αναγνώρισης ρομπότ και τα δύο LED.

4 Υλοποίηση οπτικής ανάδρασης θέσης

Στο κεφάλαιο αυτό αναλύεται η μοντελοποίηση της τεχνητής όρασης που επιτρέπει την την εξαγωγή δεδομένων θέσης από την εικόνα. Στη συνέχεια περιγράφεται η βαθμονόμηση της κάμερας και εμφανίζονται τα αποτελέσματα της σχολιάζοντας την ακρίβεια που επιτεύχθηκε. Ύστερα παρουσίαζεται η εφαρμογή που υλοποιεί την όλη διαδικασία αναγνώρισης και εντοπισμού των ρομπότ. Περιγράφεται η διαδικασία που απαιτείται από τον χρήστη για την χρήση της εφαρμογής και εξηγούνται τα πέδια που μπορούν να τροποποιηθούν. Τέλος παρατίθενται και σχολιάζονται αποτελέσματα από το πείραμα που έγινε για την αξιολόγηση της συνολικής υλοποίησης.

4.1 Μοντελοποίηση τεχνητής όρασης

Τα μοντέλα που χρησιμοποιήθηκαν για την μετατροπή των δεδομένων θέσης που εξάγονται από τις διαδικασία αναγνώρισης και εντοπισμού των ρομπότ είναι το μοντέλο κάμερας μικρής οπής που αναλύθηκε στο Κεφάλαιο 2.1 και το μοντέλο παραμόρφωσης του Κεφαλαίου 2.2.

Η διαδικασία που ακολουθείται είναι η εξής: Αφού έχουν πραγματοποιηθεί η αναγνώριση των LED, ο αλγόριθμος έχει προσδιορίσει το κέντρο τους σε συντεταγμένες εικόνας px. Γνωρίζουμε ότι αυτές οι συντεταγμένες είναι παραμορφωμένες και συνδέονται με της πραγματικές μέσω της παρακάτω σχέσης.

$$\begin{bmatrix} x_d \\ y_d \end{bmatrix} = 1 + k_1 r^2 + k_2 r^4 + k_5 r^6 + dx \tag{4.1}$$

Όπου: x_d, y_d: Παραμορφωμένες συσντεταγμένες, k_1, k_2, k_5,: Συντελεστές ακτιικής παραμόρφωσης, k_3, k_4: συντελεστές εφαπτομενικής παραμόρφωσης.

Και, $r^2 = x^2 + y^2$, των απαραμόρφωτων συντεταγμένων

Όπου dx η εφαπτομενική παραμόρφωση στην όποια οφείλεται η λοξότητα της εικόνας.

$$dx = \begin{bmatrix} 2k_3 xy + k_4(r^2 + 2x^2) \\ k_3(r^2 + 2y^2) + 2k_4 xy \end{bmatrix} \tag{4.2}$$

Είναι φανερό, ότι η σχέση (4.1) αποτελεί ενα περίπλοκο μη γραμμικό σύστημα αφού περιλαμβάνει το *r* με διάφορους εκθέτες. Γίνεται κατανοητό, ότι δε μπορεί να λυθεί ως προς τις απαραμόρφωτες συντεταγμένες με κάποια από τις γνωστές μεθοδολογίες επίλυσης μη γραμμικών συστημάτων. Προτιμήθηκε λοιπόν, η επίλυση μέσω αριθμητικής μεθόδου. Αύτη η διαδικασία απαιτεί περίπου 17 επαναλήψεις για να συγκλίνει αλλά λόγω των πολλών μαθηματικών υπολογισμών είναι ακατάλληλη για χρήση σε αλγορίθμους που τρέχουν σε πραγματικό χρόνο.

Αφού υπολογιστούν οι πραγματικές συντεταγμένες της εικόνας μέσω των σχέσεων του μοντέλου μικρής οπής, μπορούμε να υπολογίσουμε της παγκόσμιες συντεταγμένες μέσω ενός πίνακα περιστροφής που θα υπολογιστεί παρακάτω.

Οι συντεταγμένες των πίξελ της εικόνας πρέπει να είναι εκφρασμένες με αρχή το κέντρο της εικόνας προτού εφαρμόσουμε το μοντέλο μικρής οπής. Άρα έχουμε την σχέση (4.3) με y_r, x_r, να είναι οι συντεταγμένες του κέντρου και y_n, x_n οι νεές συντεταγμένες εκφρασμένες ως προς αυτό.

$$\begin{matrix} y_n \\ x_n \end{matrix} = \begin{matrix} y - y_R \\ x - x_R \end{matrix} \tag{4.3}$$

Μέσω του μεγέθους του πίξελ *τ* μετατρέπουμε τα πίξελ σε mm και μέσω της εστιακής απόστασης f και της γνωστής απόστασης των ρομπότ από την κάμερα Z έχουμε της παγκόσμιες συντεταγμένες του ρομπότ:

$$\begin{bmatrix} y_G \\ x_G \end{bmatrix} = \tau \frac{Z}{f} \begin{bmatrix} y_n \\ x_n \end{bmatrix} \tag{4.4}$$

Οι y_G, x_G είναι εκφρασμένες σε ένα σύστημα αξόνων όπου το επίπεδο X,Y είναι παράλληλο με αυτό της εικόνας και βρίσκεται σε απόσταση Z. Τελικά μένει μονό ένα μετασχηματισμός μέσω μήτρας μεταφοράς από το παγκόσμιο σύστημα στο τοπικό σύστημα της τράπεζας του γρανίτη.

4.2 Βαθμονόμηση κάμερας

Με τον όρο *''βαθμονόμηση της κάμερας''* αναφέρεται όλη η προεργασία που πρέπει να γίνει για να καθοριστούν οι παράμετροι της μοντελοποίησης ώστε το μοντέλο να είναι σε θέση να εξάγει αριθμητικά αποτελέσματα. Αυτό αποτελεί θεμελιώδες μέρος της εργασίας, γιατί σχετίζεται άμεσα με την ακρίβεια των μετρήσεων οι οποίες συνυπολογίζονται από διάφορες εργασίες μέσα στο λογισμικό, όπως την αναγνώριση των ρομπότ άλλα αποτελούν και την κύρια πληροφορία που θα λάβουν τα ρομπότ. Εύκολα συμπεραίνει κανείς ότι η αποτυχία βαθμονόμησης συνεπάγεται συνολική αποτυχία. Στην συνέχεια θα περιγραφεί η διαδικασία της βαθμονόμησης, τα εργαλεία που χρησιμοποιήθηκαν και θα γίνει μια ανάλυση των αποτελεσμάτων που προέκυψαν. Θα σχολιαστούν με σκοπό την ανάδειξη των περιορισμών του συστήματος.

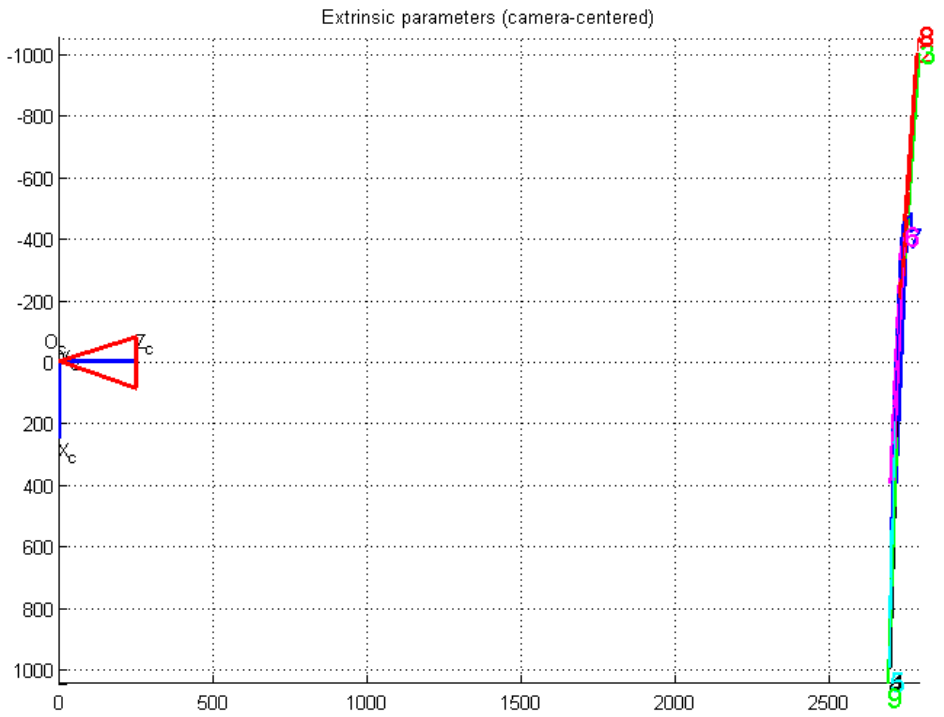
4.2.1 Μέθοδοι και εργαλεία

Αρχικά έγινε μια διερεύνηση των μεθόδων που χρησιμοποιούνται για βαθμονόμηση κάμερας. Βρέθηκε ότι όλες βασίζονται στην ίδια αρχή, δηλαδή τη συλλογή σημείων της εικόνας για τα οποία γνωρίζουμε τη θέση τους στον πραγματικό κόσμο. Στην συνέχεια με αριθμητικές μεθόδους προσπαθούν να υπολογίσουν τις τιμές των παραμέτρων της μοντελοποίησης προκύπτοντας το μικρότερο στατιστικό σφάλμα επαναπροβολής.

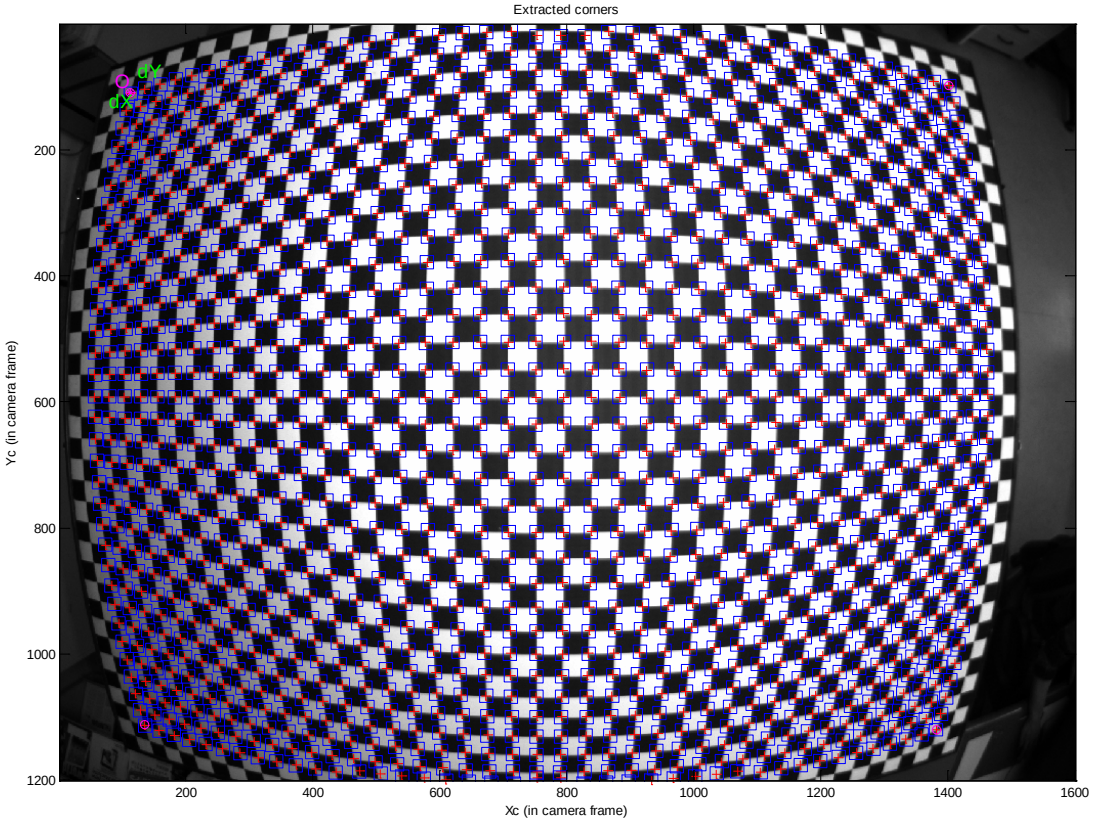
Για την διαδικασία αυτή χρησιμοποιήθηκε μια βιβλιοθήκη του MATLAB με τίτλο Camera Calibration Toolbox for Matlab [14] που δημιουργήθηκε από τον Jean-Yves Bouguet. Αύτη η βιβλιοθήκη έδωσε τα καλυτέρα αποτελέσματα και προσφέρει

εργαλεία για την αξιολόγηση των αποτελεσμάτων. Ένας πολύ σημαντικός λόγος που επιλέχθηκε αυτή η βιβλιοθήκη είναι η συμβατότητα της με το περιβάλλον του MATLAB το οποίο χρησιμοποιήθηκε ευρέως στην εργασία αυτή. Ο τρόπος με τον οποίο συλλέγει τα δεδομένα της αυτή η βιβλιοθήκη είναι εντοπίζοντας στην εικόνα επίπεδα τα οποία έχουν τη μορφή σκακιέρας. Πάνω σε κάθε τέτοιο επίπεδο εντοπίζει τις κορυφές του κάθε τετραγώνου και έτσι συλλέγει ένα σύνολο σημείων για τα οποία γνωρίζει την κατακόρυφη και οριζόντια απόσταση τους. Αυτή η διαδικασία επαναλαμβάνεται για διαφόρους προσανατολισμού του επιπέδου της σκακιέρας. Στα πλαίσια αυτής της εργασίας επιλέχθηκε να γίνει μόνο σε επίπεδο παράλληλο του γρανίτη αφού έδωσε καλυτέρα αποτελέσματα. Επίσης επειδή οι μετρήσεις που καλείται να κάνει το λογισμικό δεν είναι στο επίπεδο του γρανίτη αλλά 430mm ψηλότερα, δηλαδή το ύψος των ρομπότ πάνω στο οποίο είναι τοποθετημένα τα LED, για την βαθμονόμηση, αντί να τοποθετηθεί μια σκακιέρα στο επίπεδο αυτό κατεβάσαμε τη κάμερα κατά 430 mm. Στην Εικόνα 9 φαίνεται η σκακιέρα και οι εντοπισμένες κορυφές. Στην Εικόνα 10 φαίνεται το συστήμα συντεταγμένων με το επίπεδο XY παράλληλο στο γρανίτη το οποίο θα χρησιμοποιηθεί και ως σύστημα συντεταγμένων των ρομπότ. Επίσης προσφέρει μια οπτική απεικόνιση του σφάλματος μαρκάροντας τα πραγματικά σημεία της εικόνας με '+' και τις υπολογισμένες θέσεις αυτών με 'o' με βάση την παραμόρφωση που μετρήθηκε.

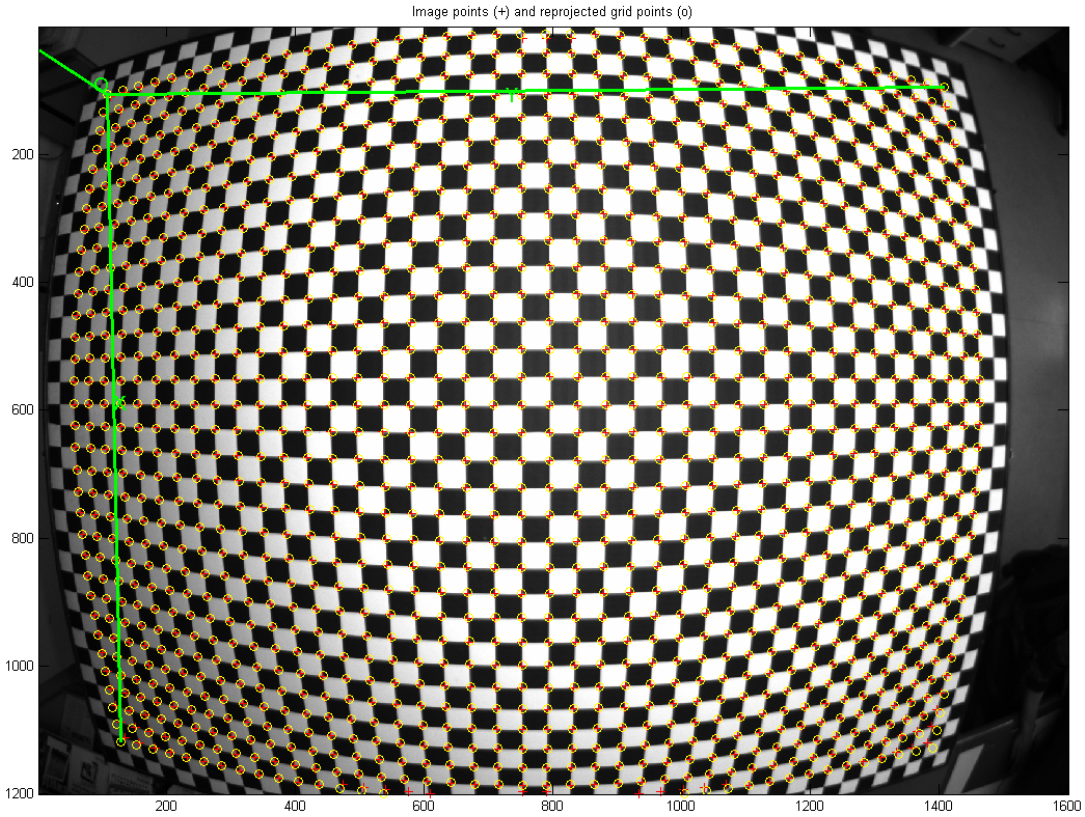
Στην Εικόνα 8 φαίνεται πως έχει αναπλάσει την πραγματική διάταξη η βιβλιοθήκη, δηλαδή την τράπεζα και την κάμερα. Είναι εύκολο να παρατηρηθεί ότι η κάμερα δεν έχει τοποθετηθεί εντελώς κάθετα πάνω από τον γρανίτη για αυτό και στην Εικόνα 11 γίνεται αντιληπτή μια προοπτική στις απαραμόρφωτες εικόνες.



Εικόνα 8. 3Δ απεικόνιση της διάταξης της σκακιέρας.



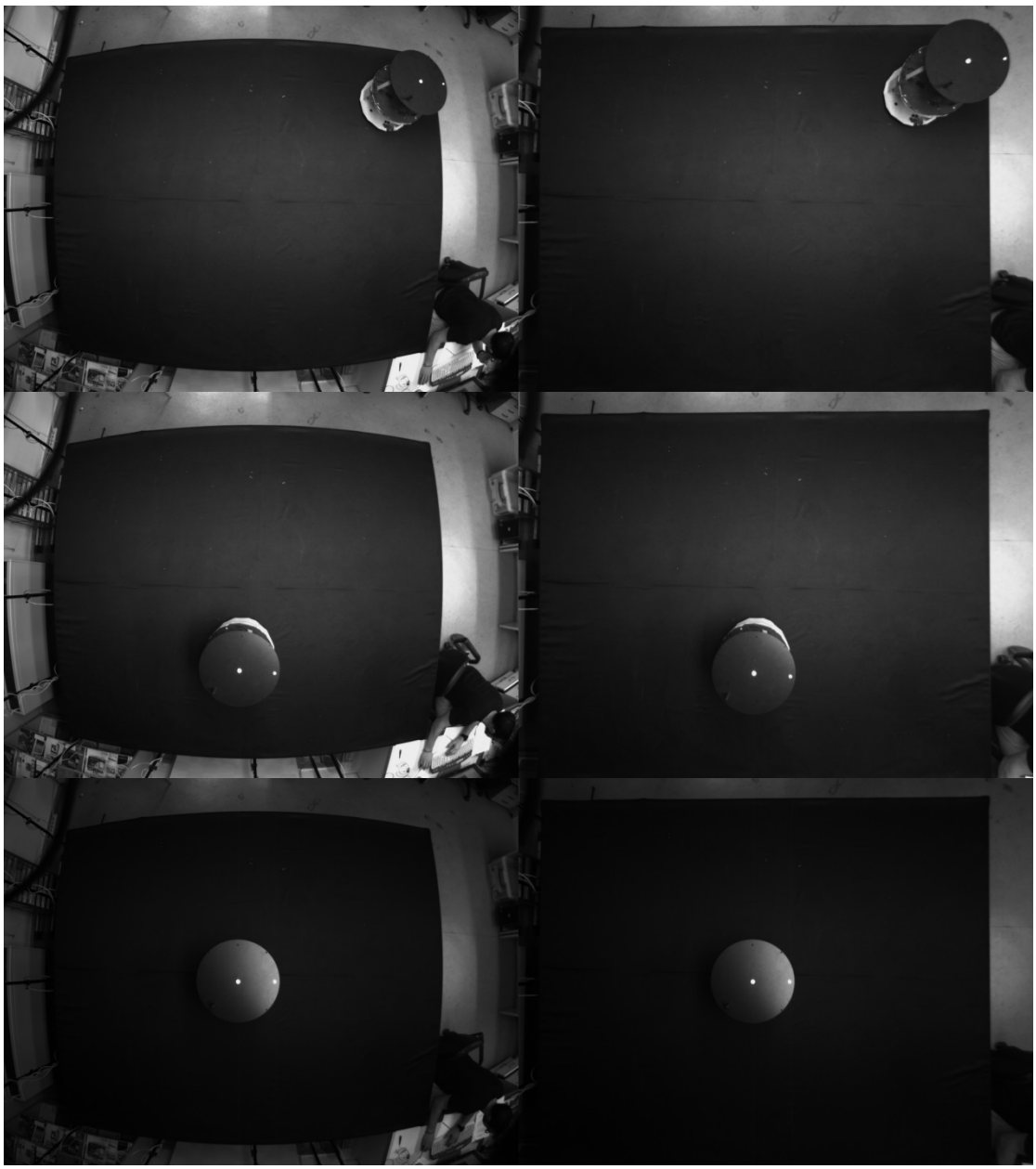
Εικόνα 9. Εντοπισμός κορυφών.



Εικόνα 10. Επαναπροβολή του επιπέδου.

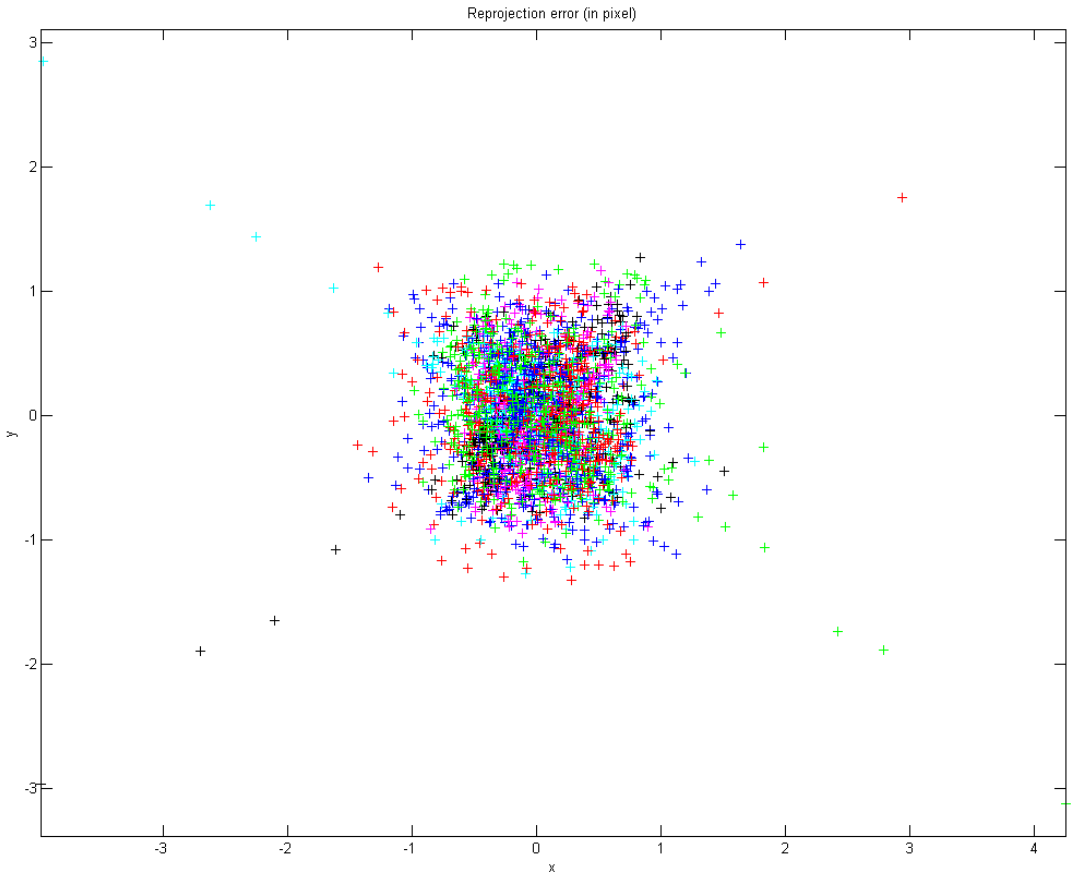
4.2.2 Αποτελέσματα και σχολιασμός βαθμονόμησης

Τα αποτελέσματα της βαθμονόμησης της κάμερας φαίνονται εποπτικά στην Εικόνα 11 όπου δίνονται ενδεικτικά τρεις θέσεις του ρομπότ πριν και μετά την αφαίρεση της παραμόρφωσης. Είναι εμφανές ότι ή μεγάλη παραμόρφωση που προκαλείται στα άκρα της εικόνας έχει σχεδόν εξαλειφτεί. Παρόλα αυτά φαίνεται μια προοπτική της τράπεζας που υποδηλώνει ότι το επίπεδο του γρανίτη δεν είναι απόλυτα παράλληλο με αυτό του αισθητήρα του φακού, πράγμα που δεν επηρεάζει την ακρίβεια εφόσον έχει ληφθει υπ' όψιν από την βιβλιοθήκη.



Εικόνα 11. Παραμορφωμένες και απαραμόρφωτες εικόνες.

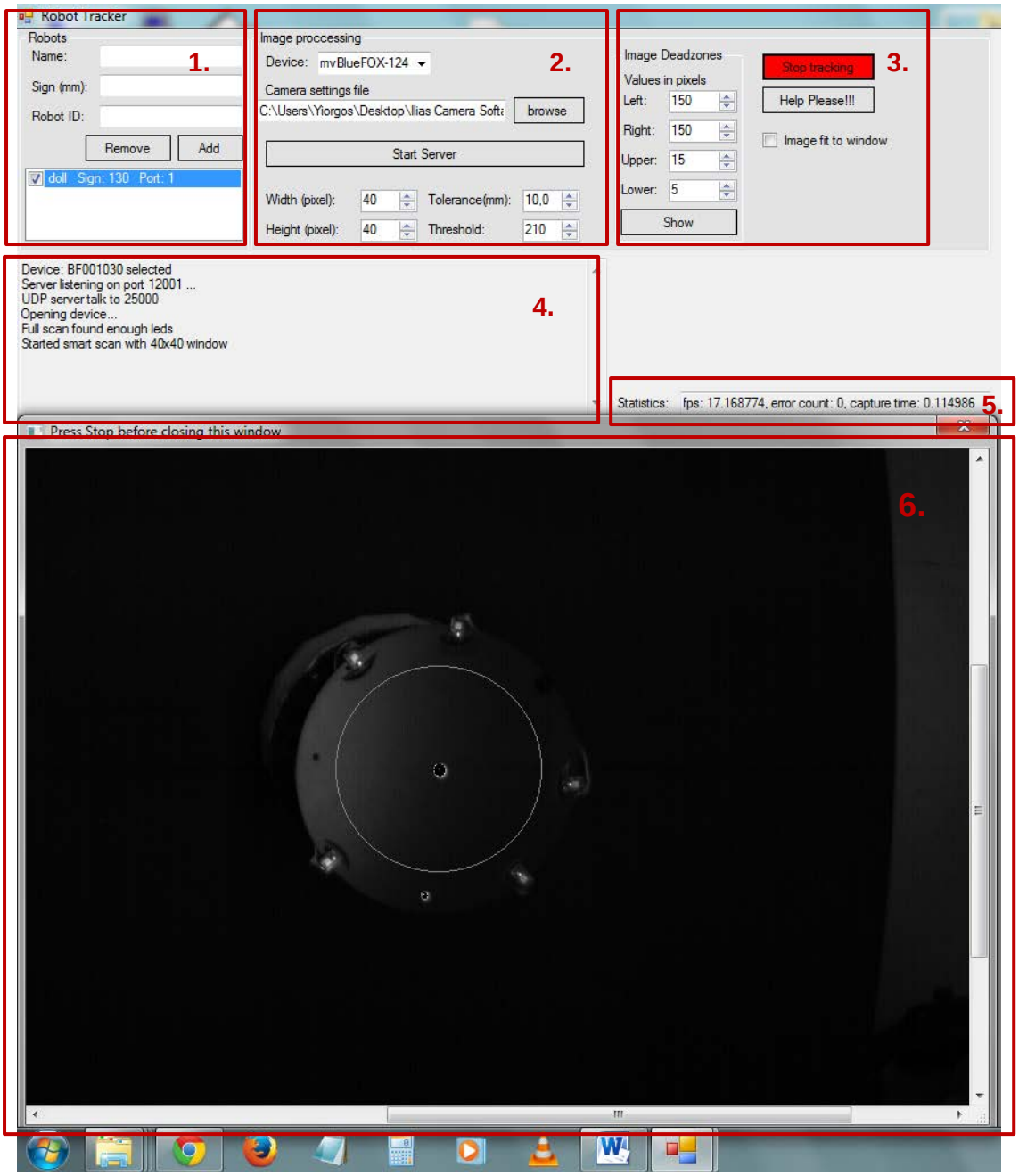
Τα αριθμητικά αποτελέσματα της βαθμονόμησης παρουσιάζονται παρακάτω στο Σχήμα 4-1 Αυτή απεικονίζει, για κάθε σημείο της εικόνας που λήφθηκε υπ' όψιν κατά την βαθμονόμηση, την απόσταση σε πίξελ από το θεωρητικό σημείο που υπολογίζεται από το μοντέλο. Βλέπουμε ότι το σφάλμα περιορίζεται στην περιοχή του 1px που αντιστοιχεί σε ακρίβεια 1,3mm ενώ φτάνει για πολύ λίγα σημεία τα 3 px,που αντιστοιχεί σε απόσταση μικρότερη των 5mm στην θέση των ρομπότ.



Σχήμα 4-1. Σφάλμα επαναπροβολής των σημείων βαθμονόμησης.

4.3 Περιγραφή της εφαρμογής

Σε αυτή την παράγραφο περιγράφεται η λειτουργία και η δυνατότητες του γραφικού περιβάλλοντος που αναπτύχθηκε για το λογισμικό τεχνητής όρασης. Ακολουθεί επεξήγηση των πεδίων και των παραμέτρων που καλείται ο χρήστης να συμπληρώσει ή τροποποιήσει. Στη συνέχεια γίνεται μια ανάλυση για τις επιλογές που έγιναν ώστε να προκύψει το γραφικό περιβάλλον που φαίνεται στην Εικόνα 12. Τέλος παρουσιάζεται βήμα προς βήμα η διαδικασία εκκίνησης, ρύθμισης και λειτουργιάς του λογισμικού.



Εικόνα 12. Γραφικό περιβάλλον λογισμικού τεχνητής όρασης.

Στην Εικόνα 12 φαίνεται το παράθυρο του λογισμικού τεχνητής όρασης που αναπτύχθηκε στα πλαίσια αυτής της εργασίας. Έχει χωριστεί σε έξι βασικά τμήματα για την διευκόλυνση της επεξήγησης του.

1. Στο πρώτο τμήμα ο χρήστης εισάγει, αφαιρεί και επιλέγει τα ρομπότ που θα αναγνωρίζει ο αλγόριθμος. Αποτελείται από τα πεδία *Name* όπου είναι ένα όνομα που επιλέγει ο χρήστης για το ρομπότ, το *Sign* που είναι η απόσταση σε χιλιοστά των δύο LED του ρομπότ και *Robot ID* που είναι το αριθμητικό αναγνωριστικό που χρησιμοποιεί ο κώδικας ως ετικέτα των συντεταγμένων των ρομπότ κατά την ασύρματη αποστολή τους μέσω δικτύου. Επίσης υπάρχει ένας κατάλογος με τα ρομπότ που έχει εισάγει ο χρήστης από τον

οποίο μπορεί να επιλέξει τα ρομπότ που αναζητά ο αλγόριθμος. Επίσης εισάγει νέα ρομπότ μέσω του κουμπιού *Add* και αφαιρεί μέσω του κουμπιού *Remove*.

2. Στο δεύτερο τμήμα επιλέγεται η συνδεμένη κάμερα που πρόκειται να χρησιμοποιηθεί από τον κώδικα και επιλέγεται το αρχείο ρυθμίσεων που θα φορτωθεί στην κάμερα. Εδώ περιλαμβάνεται και το κουμπί το οποίο ξεκινά την αποστολή δεδομένων στα ρομπότ και τον απομακρυσμένο έλεγχο του λογισμικού. Επίσης υπάρχουν τα πεδία *Width* και *height* τα οποία αποτελούν το πλάτος και το ύψος σε πίξελ της περιοχής που αναζητά ο αλγόριθμος για ένα LED όταν γνωρίζει την θέση του. Το πεδίο *Threshold* περιέχει μια τιμή από 0-255 η οποία αποτελεί την φωτεινότητα των πίξελ ο αλγόριθμος αναγνωρίζει ως LED. Τέλος το πεδίο *Tolerance* που περιγράφει την διαφορά που επιτρέπεται μεταξύ της πραγματικής απόστασης των LED ενός ρομπότ και της υπολογιζόμενης από τον αλγόριθμο για να ταυτοποιηθεί το ρομπότ.
3. Στο τρίτο τμήμα ο χειριστής επιλέγει στα πεδία *Deadzones* την περιοχή μέσα στην εικόνα που σκανάρεται από τον αλγόριθμο για να ευρεθούν τα ρομπότ, και ορίζεται θέτοντας την απόσταση των πλευρών της από της πλευρές της εικόνας σε αριθμό πίξελ. Με το πάτημα του κουμπιού *show*, ο χειριστής μπορεί να δει οπτικά την περιοχή αυτή. Στο τμήμα αυτό υπάρχει επίσης η επιλογή *Image fit to window* επιλέγοντας αυτή, η εικόνα της κάμερας προσαρμόζεται στο παράθυρο με αριθμό 6 (όχι όπως στην Εικόνα 12). Το κουμπί *Start tracking* εκκινεί και διακόπτει τη διαδικασία αναζήτησης των ρομπότ.
4. Στο τέταρτο τμήμα γίνεται ενημέρωση του χρήστη για την κατάσταση του λογισμικού με σκοπό την αντιμετώπιση προβλημάτων ως επί το πλείστον.
5. Στο πέμπτο τμήμα γίνεται ενημέρωση για τις επιδόσεις του λογισμικού όσον αφορά την ταχύτητα ανανέωσης σε καρέ ανά δευτερόλεπτο, την χρονική απόκριση της κάμερας και τον αριθμό των σφαλμάτων κατά την επικοινωνία υπολογιστή-κάμερας.
6. Το παράθυρο όπου προβάλλεται σε πραγματικό χρόνο η εικόνα της κάμερας εμπλουτισμένη με την επισήμανση των ρομπότ, εάν έχουν βρεθεί, με μορφή κύκλου με κέντρο το μεγάλο LED του ρομπότ. Χρησιμοποιείται κυρίως για την ενημερώσει του χρήστη εάν τα ρομπότ και κατ' επέκταση τα LED τους έχουν βρεθεί.

4.3.1 Ιδιότητες και παράμετροι του λογισμικού

Στην προηγούμενη παράγραφο περιγράφηκε το περιβάλλον του λογισμικού και τι είναι σε θέση να επιλέξει και να τροποποιήσει ο χρήστης. Σε αυτή την παράγραφο θα αναλυθεί η επιλογή της μορφής του γραφικού περιβάλλοντος και των παραμέτρων που επιλέγει ο χρήστης.

Ξεκινώντας χρησιμοποιήθηκε ο προηγούμενος κώδικας που είχε αναπτυχτεί στο Εργαστήριο με σκοπό να εντοπιστούν μειονεκτήματα και δυσκολίες στη χρήση του. Αυτές περιγράφονται στην Παράγραφο 3.6.1. Πιο συγκεκριμένα τα κύρια προβλήματα και οι ελλείψεις του προηγούμενου λογισμικού συνοψίζονται παρακάτω:

- Κανένα είδος παραμετροποίησης της λογικής αναζήτησης, δηλαδή αλλαγή φωτεινότητας κατωφλιού, ορισμός μέρους της εικόνας ως ενεργή.

- Έλλειψη δυνατότητας αλλαγής του τρόπου λήψης στιγμιοτύπων, δηλαδή αλλαγή χρόνου έκθεσης (exposure), ταχύτητας λήψης και τρόπου λήψης π.χ. συνεχής, αλληλοκαλυπτόμενη, οδηγούμενη από γεγονός.
- Δυνατότητα ύπαρξης μη προκαθορισμένου αριθμού ρομπότ και εύκολη εναλλαγή μεταξύ αυτών.
- Μονόδρομη λειτουργία του λογισμικού, όπου για την εκ νέου αναζήτηση απαιτείται επανεκκίνηση του.
- Καμία ενημέρωση του χρήστη για την κατάσταση λειτουργίας του, π.χ. εάν βρέθηκαν ή όχι όλα τα ρομπότ εάν τα LED που βρέθηκαν επαρκούν ή εάν ενδεχομένος ότι κάποιο ρομπότ δεν είναι ορατό.
- Έλλειψη γραφικού περιβάλλοντος και εικόνας από την κάμερα.

Έχοντας κατά νου τα προαναφερθέντα, προχωρήσαμε στην ανάπτυξη και το σχεδιασμό του γραφικού περιβάλλοντος. Εύκολα εντοπίσθηκε ότι η μεγαλύτερη ανάγκη ήταν η δυνατότητα ρύθμισης της κάμερας, αφού τα πειράματα πρέπει να είναι ανεξάρτητα των συνθηκών φωτισμού και οι εργοστασιακές ρυθμίσεις της κάμερας δε ήταν επαρκείς για την συγκεκριμένη εφαρμογή, ενώ έπρεπε να επιταχυνθεί η λήψη στιγμιοτύπων και να μειωθεί δραματικά ο χρόνος έκθεσης. Έτσι ενσωματώθηκε η δυνατότητα επιλογής αρχείου ρυθμίσεων μέσα από το γραφικό περιβάλλον που δημιουργήθηκε. Τα αρχεία αυτά ρυθμίσεων μπορεί ο χρήστης να τα δημιουργήσει εύκολα με το λογισμικό της MATRIX VISION. Επίσης μια άλλη σημαντική παράμετρος την οποία ο χρήστης πρέπει να έχει πρόσβαση, για να υπάρχει ανεξαρτησία από τις συνθήκες φωτισμού, είναι η τιμή κατωφλιού δηλαδή η τιμή φωτεινότητας των πίξελ πάνω από την οποία θεωρούνται μέρος κάποιου πίξελ καθώς και η δυνατότητα ορισμού υποπεριοχής αναζήτησης με σκοπό την αφαίρεση περιοχών στα άκρα της εικόνας όπου εντοπίζονται πιο συχνά ανακλάσεις φωτισμού πάνω σε έπιπλα και άλλα αντικείμενα του εργαστηρίου.

Επίσης είναι απαραίτητο να μπορεί ο χρήστης να αποθηκεύσει και να επιλέξει τα ρομπότ τα οποία αναζητά ο αλγόριθμος, προσθέτοντας και αφαιρώντας ρομπότ χωρίς να χρειάζεται επανεκκίνηση του λογισμικού. Επίσης στα πλαίσια του νέου αλγορίθμου αναζήτησης είναι απαραίτητη η γνώση της απόστασης των LED ενός ρομπότ για να γίνει ταυτοποίηση και απαιτείται και ένας μοναδικός αριθμός που αντιπροσωπεύει το ρόμποτ και χρησιμοποιείται στα μυνήματα συντεταγμένων που στέλνει το λογισμικό για να γνωρίζουμε ποιου ρομπότ η θέση είναι, μια και κάθε ρομπότ λαμβάνει τις συντεταγμένες όλων για πιθανή μελλοντική χρήση.

Στο πλαίσιο του νέου επιταχυμένου κώδικα που δημιουργήθηκε, ο οποιός αναζητά τα LED με βάση την προηγούμενη τους θέση, αφέθηκε στο χειριστή η δυνατότητα ρύθμισης του εύρους αυτής αφού εξαρτάται από την ταχύτητα που κινείται ένα αντικείμενο και ενδεχόμενως στο μέλλον να χρησιμοποιηθεί ο εξομοιωτής για μελέτη διακίνησης παθητικών αντικειμένων από τους βραχίονες των ρομπότ άρα να απαιτείται δυνατότητα παρακολούθησης αντικειμένων με μεγαλύτερες ταχύτητες.

Τέλος έχει γίνει μεγάλη πρόοδος στην ενημέρωση του χρήστη για την κατάσταση του λογισμικού. Ο χρηστής γνωρίζει:

- εάν ο αλγόριθμος έχει βρει τα ρομπότ και αναζητά την νέα θέση τους με βάση την προηγούμενη, αναγράφοντας, *Started smart scan with 40x40 window*

- εάν δεν υπάρχει επαρκής αριθμός LED με βάση τα ρομπότ που πρέπει να βρεθούν, αναγράφοντας, *Not enough Leds, n leds missing,*
- Εάν πραγματοποιήθηκε σάρωση χωρίς αρχικές τιμές, αναγράφοντας, *Full scan found enough Leds,*
- Για το άνοιγμα της κάμερας και της επικοινωνίας με το λογισμικό απομακρυσμένου ελέγχου που έχει αναπτυχθεί για τον έλεγχο των ρομπότ.

Μια απαραίτητη προσθήκη ήταν η προβολή της εικόνας της κάμερας στο χρηστή ο οποίος με αυτό το τρόπο αποκτά μεγάλη εποπτεία με αποτέλεσμα να μην υπάρχει ανάγκη για γραπτή ενημέρωση για προβλήματα που σχετίζονται με το πώς βλέπει η κάμερα το χώρο εργασίας. Επίσης μέσω αυτού ο χρήστης ενημερώνεται για την εύρεση των ρομπότ οπτικά προβάλλοντας στην εικόνα της κάμερα ένα λευκό κύκλο γύρω από τα αναγνωρισμένα ρομπότ.

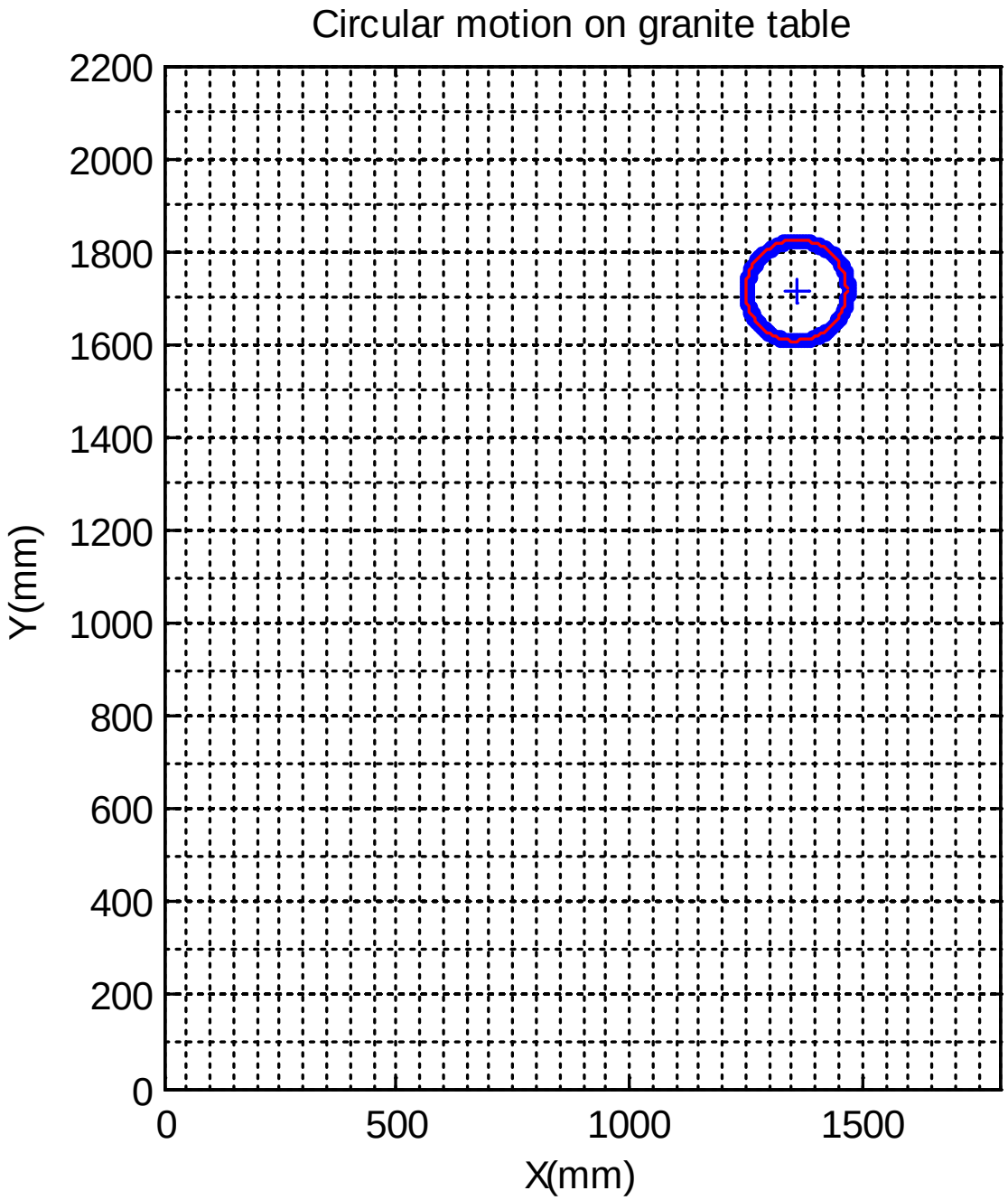
Οι κώδικες που περιγράφονται στην Παράγραφο 3.7.4 και 3.7.5 βρίσκονται στο Παράρτημα Β Κώδικες.

4.4 Πείραμα και σχολιασμός αποτελεσμάτων

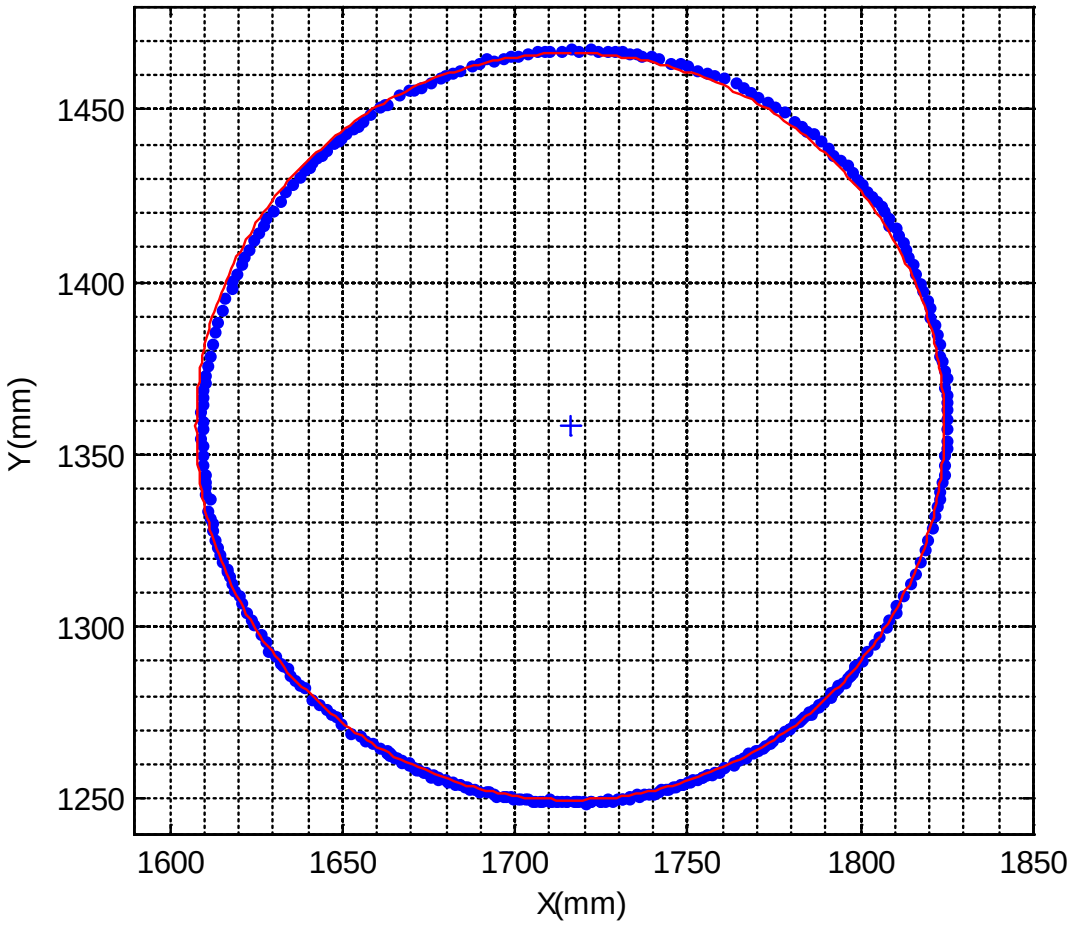
Για την επαλήθευση της ποιότητας της βαθμονόμησης που περιγράφεται στη Παράγραφο 4.2.2 έγινε ένα συνολικό πείραμα, που θα περιγραφεί στην παρούσα παράγραφο. Όπως είδαμε στην Παράγραφο 4.2.2 το σφάλμα που προκύπτει από την βαθμονόμηση είναι της τάξης του 1 με 1,5 px τιμή που αντιστοιχεί σε 1,3 με 1,9 [mm] για κάποιες ακραίες περιπτώσεις φτάνει και τα 3 px που αντιστοιχεί σε 5 mm.

Για την επαλήθευση των παραπάνω τιμών έγινε το εξής πείραμα: Τοποθετήθηκε ένα ομοίωμα του ρομπότ του οποίου ένα στήριγμα από τα τρία εξυπηρετούσε περιστροφική άρθρωση με σκοπό το ρομπότ να κάνει μια κυκλική κίνηση γύρω από αυτό το σημείο. Με αυτό το τρόπο εξάχθηκαν δεδομένα που περιγράφουν την ακρίβεια απόλυτης και σχετικής θέσης, την λοξότητα και παραμόρφωση καθώς και την ακρίβεια προσανατολισμού.

Τα δεδομένα που συλλέχτηκαν είναι οι συντεταγμένες του κέντρου του ρομπότ (δηλαδή του κεντρικού του LED), ο προσανατολισμός του, που προκύπτει από τη κλίση της ευθείας που ενώνει τα δύο LED, καθώς και την μεταξύ τους απόσταση που γνωρίζουμε ότι είναι 130 mm. Στο Σχήμα 4-2 φαίνεται το τραπέζι του γρανίτη και η κυκλική κίνηση που έγινε από το ρομπότ. Για να μετρηθεί η απόλυτη ακρίβεια έπρεπε να εντοπιστεί ένα σημείο στο επίπεδο του γρανίτη και όχι των LED. Με την κυκλική κίνηση γνωρίζουμε το πραγματικό κέντρο σε σχέση με τη τράπεζα του γρανίτη αφού είναι η θέση της άρθρωσης και μπορεί εύκολα να υπολογιστή από τα δεδομένα ως το κέντρο του παραγόμενου κύκλου. Με βάση την ελαχιστοποίηση της τυπικής απόκλισης των δεδομένων προκύπτουν οι συντεταγμένες τους κέντρου του κύκλου και η μέση ακτίνα.



Σχήμα 4-2. Κυκλική κίνηση ρομπότ στη τράπεζα γρανίτη.

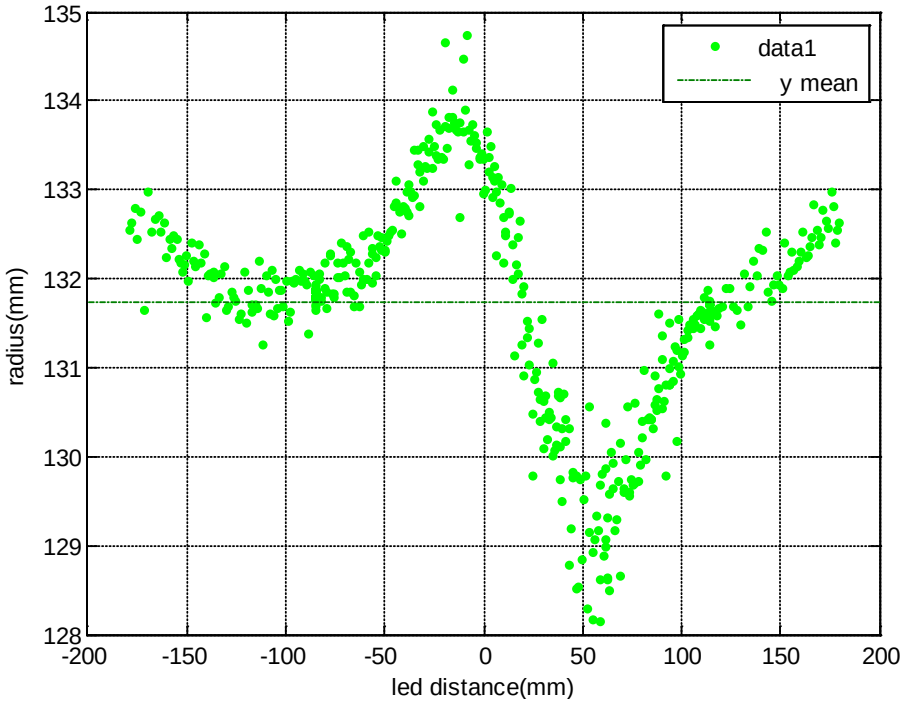


Σχήμα 4-3. Θέση του ρομπότ (μπλε) μεσος κύκλος (κοκκινο).

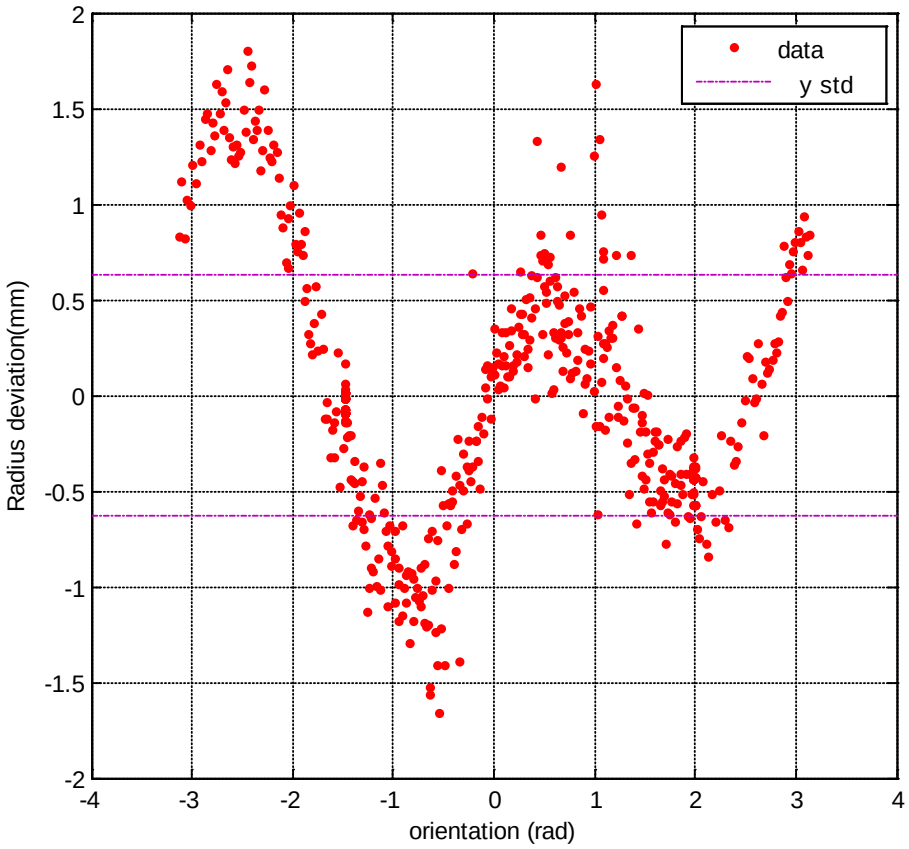
Προς επαλήθευση των παραπάνω τιμών μετρήθηκαν οι πραγματικές αποστάσεις από τα άκρα του τραπεζιού και προέκυψαν τα αποτελέσματα του Πίνακας 2. Τα αποτελέσματα είναι πολύ εντυπωσιακά για τον άξονα X ενώ δε δικαιολογείτε το σφάλμα στο Y πιθανόν να οφείλεται σε λάθος κατά την βαθμονόμηση, διότι εάν παρατηρηθεί ο κύκλος είναι σφάλμα μεταφοράς και εφόσον ο κύκλος δε έχει την μορφή έλλειψης. Άρα δε επηρεάζεται το πείραμα εφόσον αποτελεί μια απλή μετατόπιση του συστήματος αναφοράς. Έτσι προχωρούμε στο σχολιασμό των υπόλοιπων δεδομένων.

Πίνακας 2. Συγκριση τιμών από κάμερα με τις πραγματικές.

Κάμερα	Πραγματικές	Διαφορά
$x_{center} = 1359\ mm$	$x_{center} = 1359\ mm$	$0\ mm$
$y_{center} = 1717\ mm$	$y_{center} = 1697\ mm$	$20\ mm$
$R_{average} = 108\ mm$		



Σχήμα 4-4. Απόσταση LED του ρομπότ συναρτήσει του προσανατολισμού του.



Σχήμα 4-5. Διακύμανση ακτίνας κατά την περιστροφική κίνηση.

Στο Σχήμα 4-4 απεικονίζεται η απόσταση των δύο το LED του ρομπότ η οποία αποτελεί και το χαρακτηριστικό με το οποίο το λογισμικό της κάμερας ταυτοποιεί το ρομπότ. Σε αυτό το πείραμα η πραγματική τιμή του είναι 130 mm. Παρατηρώντας το διάγραμμα βλέπουμε μια διακύμανση 6 mm με μέση τιμή 131,7 mm. Επίσης εντοπίζεται μια μεγάλη κλίση στην περιοχή 50-60^{O}, ύστερα από μελέτη προκύπτει ότι:

$$\tan^{-1}\frac{1600}{1200} \cong 53° \tag{4.5}$$

Όπου 1600X1200 είναι η αναλόγια διαστάσεων του οπτικού αισθητήρα, άρα είναι λογικό μιας και εκεί μεγιστοποιείται το συνδυασμένο σφάλμα και των δύο αξόνων. Ομοίως αυτό το σφάλμα μεγιστοποιείται και για γωνίες: -53°, -127°, και 127° κάτι που δεν γίνεται σαφές από το διάγραμμα. Αυτό μας οδηγεί στο συμπέρασμα ότι το σφάλμα απόστασης των LED επηρεάζεται και από την θέση του ρομπότ αφού κατά την περιστροφή του αλλάζει και η θέση του αφού δε περιστρέφεται γύρο από το μέσο των LED.

Στο Σχήμα 4-5 απεικονίζεται η διαφορά της μετρούμενης ακτίνας από την μέση ακτίνα που υπολογίστηκε. Παρατηρείται ότι το μέγιστο σφάλμα είναι μόλις 1,5 mm ενώ το μεγαλύτερο κομμάτι του δείγματος είναι εντός της περιοχής 0-0,5 mm.

5 Έλεγχος θέσης Ρομπότ

Σκοπός του εργαστηρίου είναι ο σχεδιασμός ρομποτικών συστημάτων που κινούνται σε τροχιά στο διάστημα. Ο σχεδιασμός όμως τέτοιων συστημάτων ξεφεύγει από τον τυπικό σχεδίασμο, διότι οι συνθήκες δράσης ενός τέτοιου ρομπότ είναι πολύ διαφορετικές από τις επίγειες και πολλές φορές δεν μπορούν να γίνουν αντιληπτές από τον σχεδιαστή. Ο εξομοιωτής προσφέρει μια εξοικείωση και ένα μέσο αξιολόγησης των συστημάτων που σχεδιάζονται.

Αρχικός στόχος με βάση τα παραπάνω τέθηκε η δημιουργία ενός ελεγκτή θέσης κλειστού βρόχου βασισμένου στην ανάδραση θέσης από το σύστημα της κάμερας που περιγράφηκε στο Κεφάλαιο 4. Ο σκοπός του συστήματος ελέγχου είναι να αποτελέσει τη βάση για υλοποίηση έλεγχου τροχιάς μεταγενέστερα, να αξιολογήσει το σύστημα της κάμερας και να επιβεβαιώσει τις επιλογές που έγιναν όσον αφορά το σύστημα προώθησης με CO_2 του ρομπότ. Ο ελεγκτής υλοποιεί έλεγχο PD και σχεδιάστηκε βασισμένος στο αναλυτικό μοντέλο του ρομπότ. Στο παρόν κεφάλαιο θα αναλυθεί το σύστημα επενεργητών και πως αυτό συνεργάζεται με τον έλεγχο θέσης .

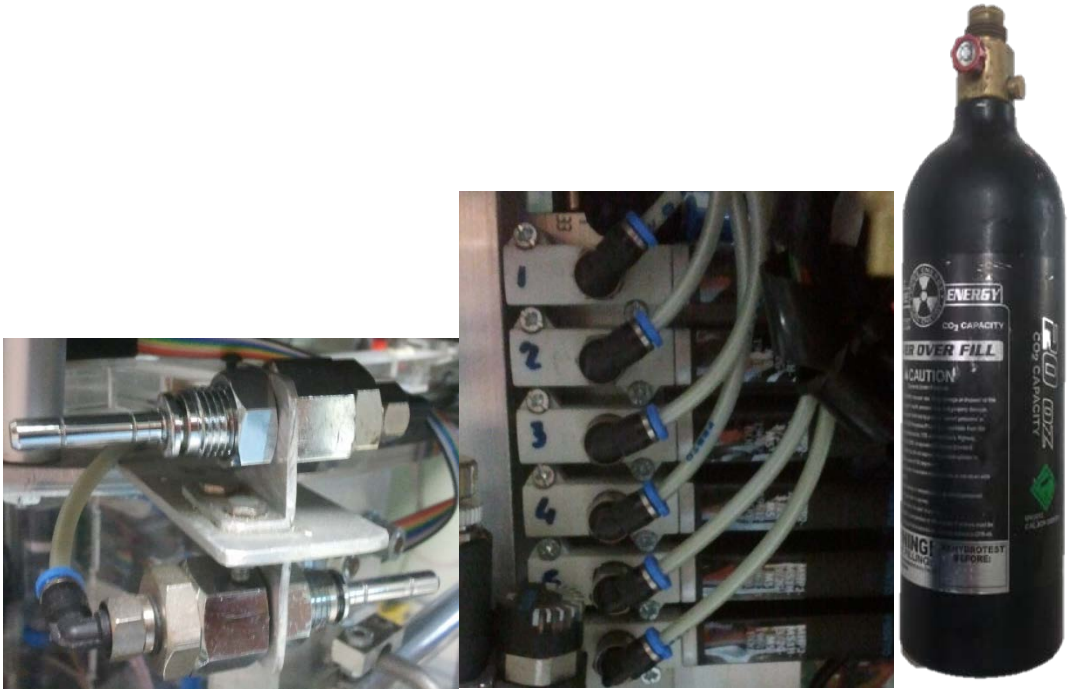
5.1 Επενεργητές

Το ρομπότ που έχει σχεδιαστεί και κατασκευαστεί από τον Κ. Μαχαιρά [11] χρησιμοποιεί για την προώθηση του 6 ακροφύσια σε ζεύγη (αντίθετης ενέργειας) που είναι τοποθετημένα περιμετρικά του κυκλικού κορμού του ρομπότ σε γωνία 120° μεταξύ τους. Αυτά συνδέονται σε ένα κύκλωμα υψηλής πίεσης αερίου CO_2 που τροφοδοτείται από μια αποσπώμενη φιάλη (βλ. Εικόνα 13).

Για τον έλεγχο της ώσης που παράγεται, τα ακροφύσια ενεργοποιούνται από βαλβίδες οι οποίες έχουν την δυνατότητα πολύ γρήγορης λειτουργίας έτσι μέσα σε ένα χρονικό διάστημα αναφοράς μεταβαλλόντας τη διάρκεια που παραμένει ανοιχτή η βαλβίδα. Εάν αυτό γίνεται επαναλαμβανόμενο με συχνότητα πολύ μεγαλύτερη της ιδιοσυχνότητας του συστήματος τότε μπορούμε να έχουμε μια ψευδό-αναλογική ώση. Αυτού του είδους αναλογικός έλεγχος ονομάζεται PWM (pulse width modulation) και το ποσοστό που αντιστοιχεί στο χρόνο που είναι ενεργή η έξοδος προς τον συνολικό χρόνο αναφοράς ονομάζεται duty cycle και συμβολίζεται στο παρόν σύγραμμα $\boldsymbol{\tau}$.

$$\boldsymbol{\tau} = \frac{t_{opened}}{T} \tag{5.1}$$

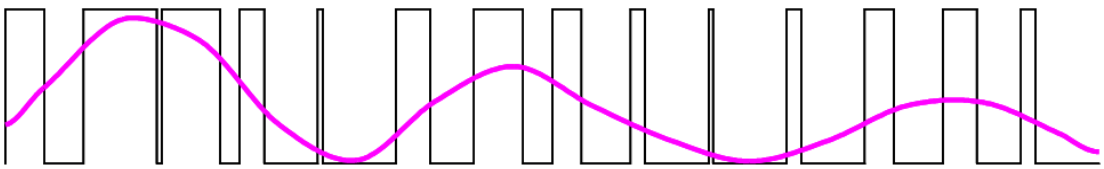
Αυτός ο τρόπος αναλογικού ελέγχου χρησιμοποιείται ευρέως σε ψηφιακά συστήματα που θέλουμε να μιμηθούμε μια αναλογική έξοδο. Αυτή η μέθοδος περιγράφεται πιο αναλυτικά παρακάτω.



(α) (β) (γ)

Εικόνα 13. (α) Ακροφύσια, (β) βαλβίδες έλεγχου, (γ)φιάλη CO_2.

Στο Σχήμα 5-1 παρουσιάζεται ένας παλμός PWM και η αντίστοιχη αναλογική τιμή που μιμείται με ροζ χρώμα. Ένας τέτοιος παλμός μπορεί να εφαρμοστεί στις βαλβίδες του ρομπότ αλλά με κάποιους περιορισμούς που εισάγονται από τα ηλεκτρομηχανικά χαρακτηριστικά της βαλβίδας που είναι ο χρόνος ανοίγματος, κλεισίματος και η μεγίστη συχνότητά της. Το πρόβλημα αυτό περιγράφεται παρακάτω.



Σχήμα 5-1. Παλμός PWM και η αντίστοιχη αναλογική τιμή του.

5.1.1 Περιορισμοί του κλασικού ελέγχου PWM

Ο έλεγχος PWM χρησιμοποιείται ευρέος με συχνότητες της ταξης χιλιάδων Hz σε ηλεκτρικά συστήματα, όπως για παράδειγμά για τον έλεγχο φωτεινότητας ενός LED. Το LED μπορεί ανάβει και να σβήνει 500 φορές σε ένα δευτερόλεπτο, φαινόμενο που δε μπορεί να γίνει αντιληπτό από τον άνθρωπο με αποτέλεσμα να έχουμε συνεχούς έντασης φωτεινότητα. Αυτό είναι εφικτό εφόσον η δυναμική των συστημάτων αυτών

είναι αμελητaία σε αυτές τις συχνότητες. Για την εφαρμογή τέτοιου ελέγχου σε ένα μηχανικό σύστημα, όπως οι βαλβίδες του ρομπότ, πρέπει να ληφθεί υπόψιν η δυναμική του συστήματος γιατί είναι αυτή που θα καθορίσει την συχνότητα του PWM.

Από τον κατασκευαστή γνωρίζουμε ότι μια βαλβίδα για να ανοίξει πλήρως απαιτείται χρόνος 1,8 ms, ενώ για το κλείσιμό της 2 ms. Γίνεται κατανοητό ότι εάν εφαρμοστεί PWM με περίοδο μικρότερη από 5,7 ms δε θα έχουμε κανένα αποτέλεσμα εφόσον ο χρόνος δεν επαρκεί ώστε η βαλβίδα να ανοίξει και να κλείσει. Ακόμη επείδη η λειτουργία του PWM βασίζεται στην σχέση μεταξύ της διάρκειας κατα την οποία η βαλβίδα είναι ανοιχτή και της διάρκειας κατά την οποία είναι κλειστή, πρέπει αυτές να είναι μεγαλύτερες από τις παραπάνω τιμές ώστε να εξασφαλίζεται ότι η βαλβίδα ανοιξέ και έκλεισε πλήρως και αντιστιχά το ακροφύσιο παρήγαγε την προβλεπόμενη ώση. Αυτό θα συνέβαινε εάν οι παραπάνω τιμές ήταν απλά υστέρηση, όμως στην πραγματικότητα κατά το μεταβατικό στάδιο της κατάστασης της βαλβίδας δε μπόρει να υπολογισθεί με ακρίβεια η παραγώμενη ώση. Προκύπτει λοιπόν ότι οι διάρκειες που θα απαιτεί ο έλεγχος PWM πρέπει να είναι αρκέτα μεγαλύτερες απο αυτές ώστε το φαινόμενο που περιγράφτηκε να μπορεί να αμεληθεί. Όμως η αναλογική τιμή του ελέγχου PWM ισούται με το πηλίκο της διάρκεια κατά την οποία είναι ανοιχτή η βαλβίδα προς την περίοδο επανάληψης. Άρα εφόσον η δίαρκεια αυτή δεν μπορεί να είναι μηδέν λόγω του παραπάνω περιορισμού προκύπτει μια εύρος τιμών το οποίο δε μπορεί να επιτύχει ο ελεγχος PWM. Αυτές οι περιοχές ονομάζονται περιοχές κορεσμού και εντοπίζονται στα άκρα του εύρους του PWM. Για την καλύτερη κατανόηση των παραπάνω δίνεται ένα παραδείγμα:

Έστω η συχνότητα του PWM 10 Hz και η ελάχιστες διάρκειες κατα τις οποίες η βαλβίδα πρέπει να μείνει ανοιχτή και κλειστή αντίστοιχα.

$$t_{on} = 8\,ms \tag{5.2}$$

$$t_{off} = 10\,ms \tag{5.3}$$

η περίοδος του παλμού είναι T=1/10 = 100 ms
Άρα η ελάχιστη τιμή $\boldsymbol{\tau}$ που μπορει να παραχθεί είναι:

$$\boldsymbol{\tau_{min}} = \frac{t_{on}}{T} = 8\% \tag{5.4}$$

και αντίστοιχα

$$\boldsymbol{\tau}_{max} = 1 - \frac{t_{off}}{T} = 1 - f \cdot t_{off} = 100\% - 10\% = 90\% \tag{5.5}$$

Οι παραπάνω τιμές είναι η μέγιστη και η ελάχιστη αναλογική τιμή που μπορούμε να μιμηθούμε, είναι φανερό ότι στην περίπτωση των διαστημικών ρομπότ οπού δε υπάρχει κανένα είδος τριβής πρέπει ο έλεγχος θέσης να είναι σε θέση να ασκήσει πολύ μικρές δυναμείς, διαφορετικά θα υπάρχει μόνιμη ταλαντωτική κίνηση γύρω από την επιθυμητή θέση.

Προκύπτει λοιπόν ότι το κάτω όριο θα πρέπει να μειωθεί. Αυτό συμβαίνει αυξάνοντας την περίοδο. Μειώνοντας όμως τη συχνότητα αρχίζει να γίνεται πολύ διακριτός ο έλεγχος δύναμης και παύει να έχει ψευδό-αναλογική λειτουργιά. Επίσης δημιουργείται μεγάλη υστέρηση στο σύστημα ελέγχου το οποίο λειτουργεί σε

μεγαλύτερη συχνότητα. Για να επιτευχθεί συνεχής λειτουργιά και μεγάλο εύρος σχεδιάστηκε ένας αλγοριθμος ο οποίος έχει μεταβλητή συχνότητα με σκοπό να αυξήση το οφέλιμο εύος του κλασικού ελέυχου PWM. Αυτός περιγράφερεται στην Παράγραφο 5.1.2

5.1.2 Μεταβλητής συχνότητας PWM

Ο αλγόριθμος PWM μεταβλητής συχνότητας βασίστηκε στην εξής ιδέα. Παρατηρώντας την εξίσωση 5.1 μπορεί το ***τ*** να επιλέγεται αλλάζοντας την περίοδο και κρατώντας σταθερό τον χρόνο που μένει ενεργή η έξοδος. Με αυτό το τρόπο εξασφαλίζεται οτί η έξοδος θα μείνει για καθορισμένο χρόνο ενεργή. Αντίστοιχα μπορεί αυτό να εφαρμοστεί παράλληλα και για την εξασφάλιση της διάρκειας που η έξοδος μένει ανενεργή. Προκύπτουν λοιπόν οι περιρισμοί:

$$t_{opened} > t_{on} \tag{5.6}$$

$$t_{closed} > t_{off} \tag{5.7}$$

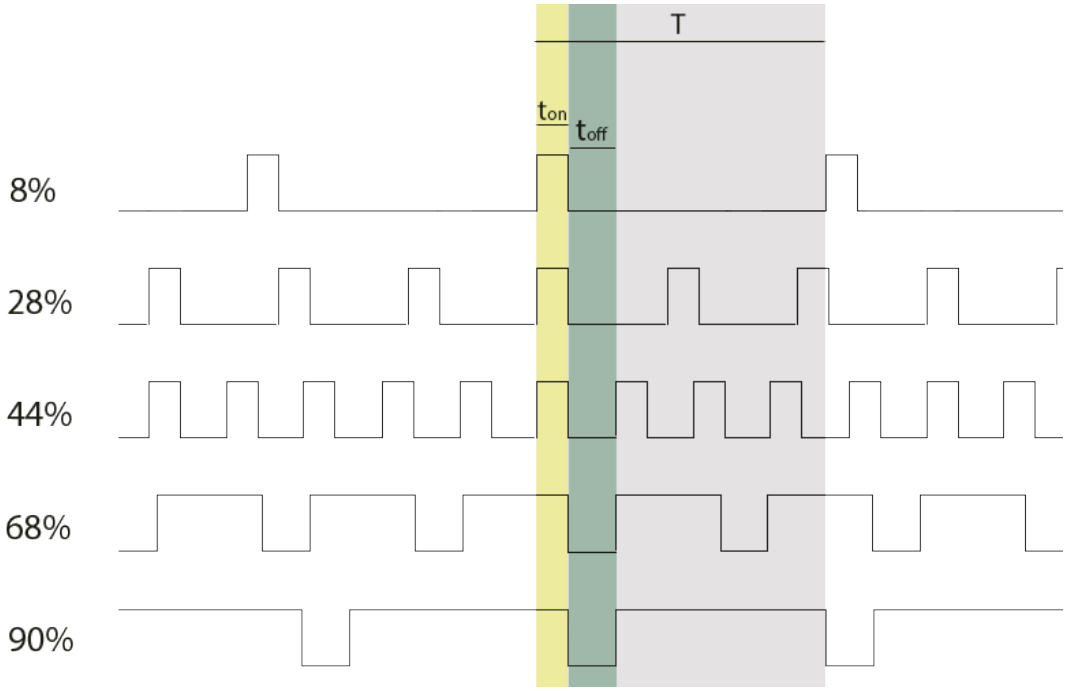
Όμως:

$$\mathrm{T} = t_{opened} + t_{closed} \tag{5.8}$$

Άρα

$$\mathrm{T} > t_{on} + t_{off} \tag{5.9}$$

Ο αλγόριθμος λειτουργεί ως εξής: ξεκινωντας από την ελάχιστη τίμη **τ** με σταθερό t_{opened} μειώνει συνεχώς την T μέχρι $T = t_{on} + t_{off}$, όπου για να συνεχίσει να αυξάνεται το **τ** διατηρείται πλεον σταθερο το t_{off} και ξανα αυξάνεται η περίοδος Τστο Σχήμα 5-2 φαίνονται οι παλμοί PWM για διαφορα ***τ***.



Σχήμα 5-2. Παλμός PWM για κάποιες ενδεικτικές τιμές ***τ***.

Με βάση τα παραπάνω μπορούμε να υπολογίσουμε το $\boldsymbol{\tau}$ για το οποίο το T γίνεται ελάχιστο, αυτό ορίζουμε $\boldsymbol{\tau_{threshold}}$ δηλαδή για:

$$T = t_{on} + t_{off} \tag{5.10}$$

Όπου,

$\boldsymbol{\tau} < \boldsymbol{\tau_{threshold}}$, t_{on} παραμένει σταθερό,

- $\boldsymbol{\tau} > \boldsymbol{\tau_{threshold}}$ t_{off} παραμένει σταθερό

Το $\boldsymbol{\tau_{threshold}}$ είναι φανερό ότι υπολογίζεται από την σχέση:

$$\boldsymbol{\tau_{threshold}} = t_{on} / (t_{on} + t_{off}) \tag{5.11}$$

Παρατηρώντας τη σχέση (5.1) το T μπορεί να πάρει πολύ μεγάλες τιμές, έτσι ορίζοντας ένα μέγιστο χρόνο περιόδου προκύπτουν η ελάχιστη και μέγιστη τιμή του $\boldsymbol{\tau}$ όπου πέρα από αυτές ο αλγόριθμος κλείνει ή ανοίγει μόνιμα τη βαλβίδα.

Άρα για T_{max}=0,5s, βάσει τις τιμές του προηγούμενου παραδείγματος έχουμε:

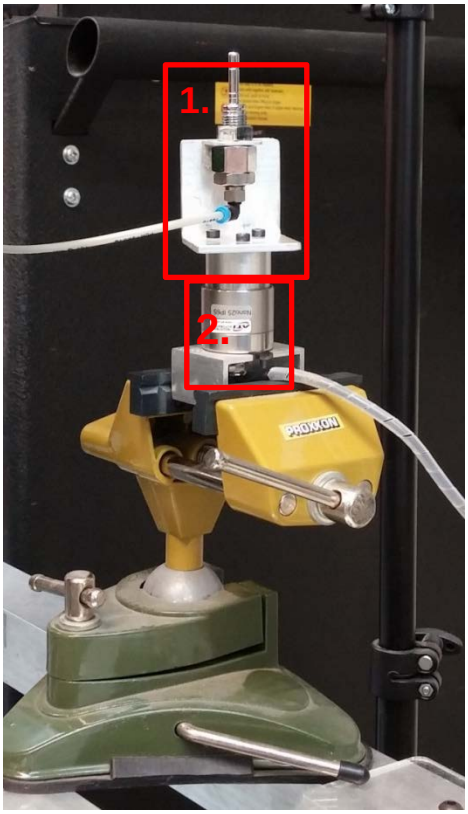
$$\boldsymbol{\tau_{min}} = \frac{t_{on}}{T} = \frac{0.008}{0.5} = 1.6\% \tag{5.12}$$

$$\boldsymbol{\tau_{max}} = \mathbf{1} - \frac{t_{off}}{T} = \frac{0.0010}{0.5} = 98\% \tag{5.13}$$

Η υπόθεση ότι κατά το άνοιγμα της βαλβίδας δε έχουμε πίεση στο ακροφύσιο είναι λανθασμένη διότι κατά το άνοιγμα της βαλβίδας ελευθερώνεται μια ποσότητα αερίου. Αυτή η ποσότητα δε είναι εύκορα υπολογίσιμη για αυτο το λόγο επιλέγονται τιμές $t_{on} \gg t_{on\,πραγματικο}$ ώστε να θεωρείται αμελητέα αυτή η ποσότητα. Η ιδία τεχνική ακολουθείται και για χρόνο κλεισίματος.

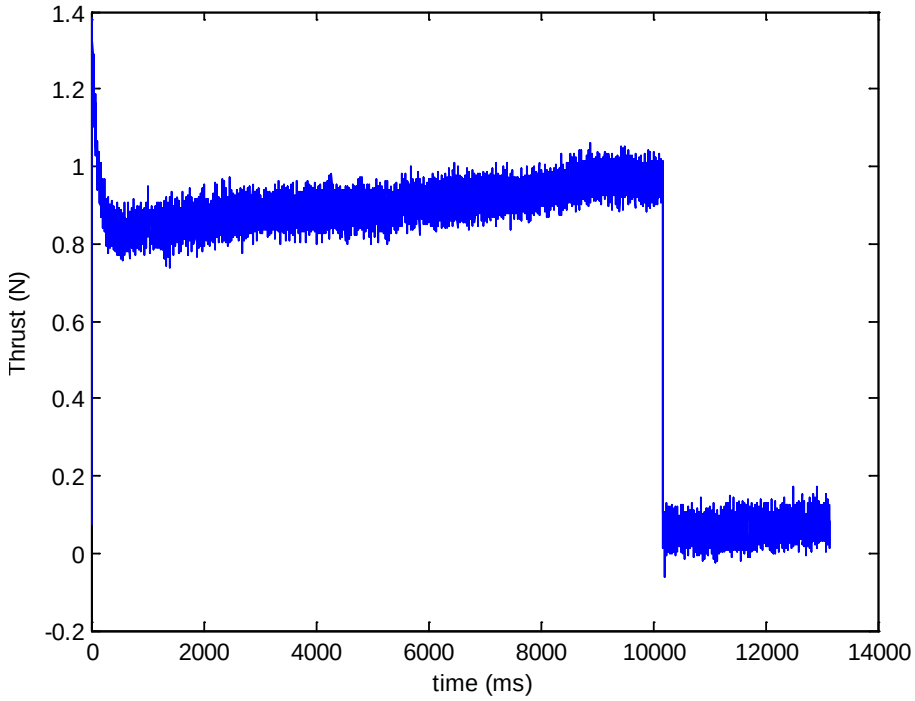
5.1.3 Σύστημα επενεργητών και προσδιορισμός μηχανικών περιορισμών ελέγχου

Για να εφαρμόσουμε έλεγχο κλειστού βρόχου στα ρομπότ, χρειάζεται να γνωρίζουμε τις δυνατότητες και τους περιορισμούς των προωθητήρων. Επικεντρώνουμε την προσοχή μας στον προσδιορισμό της μεγίστης δύναμης που μπορεί να ασκήσει ένα ακροφύσιο, καθώς και της αναλογίας δύναμης με $\boldsymbol{\tau}$ (duty cycle), δηλαδή μια σύγκριση της επιθυμητής ψευδό-αναλογικής με την πραγματική. Η μετρήσεις έγιναν με ένα αισθητήρα δύναμης ροπής της εταιρείας ATI που ήταν προσδεμένος στο ακροφύσιο (βλ. Εικόνα 14) με δειγματοληψία 10000 hz επαρκές για την μέτρηση της υστέρησης της βαλβίδας που είναι 1.8 και 2 ms. Η στήριξη του ακροφυσίου στον αισθητήρα έγινε με τέτοιο τρόπο ώστε η δύναμη να ασκείται μόνο στην αξονική κατεύθυνση του αισθητήρα ώστε να απλοποιηθούν και να είναι πιο αξιόπιστες οι μετρήσεις. Τα δεδομένα συλλέχθηκαν από την εφαρμογή της ATI και στην συνέχεια μελετήθηκαν με την βοήθεια του λογισμικού MATLAB.

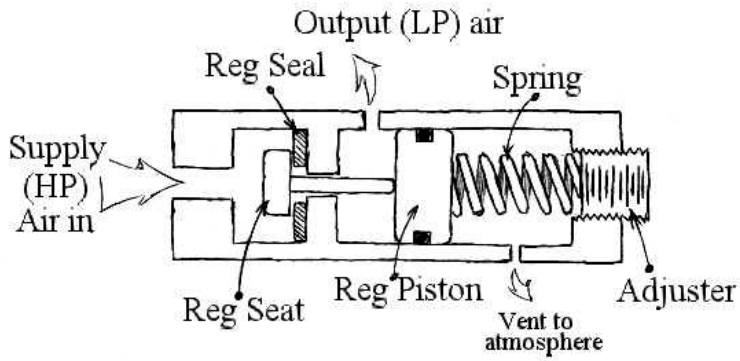


Εικόνα 14. 1) Ακροφύσιο, 2) αισθητήρας δύναμης /ροπής.

Στο Σχήμα 5-3 φαίνεται η απόκριση της δύναμης για πλήρες άνοιγμα της βαλβίδας για 10 δευτερόλεπτα. Παρατηρείται μια κορυφή στην αρχή που διαρκεί 200 ms περίπου και στην συνέχεια μια ανοδική τάση. Η κορυφή στην αρχή οφείλεται στη ρυθμιστική βαλβίδα που αναλαμβάνει την μείωση της πίεσης πριν αυτή φτάσει στης βαλβίδες των ακροφυσίων. Από το τον τρόπο λειτουργίας της βαλβίδας συμπεραίνουμε ότι είναι λογική αυτή η κορυφή αφού δημιουργείται με το άνοιγμα της βαλβίδας μεγάλη πτώση πίεσης στη κατάθλιψη της με αποτέλεσμα την μεγάλη επιτάχυνση του δρομέα της βαλβίδας όπου με την απουσία στοιχείου απόσβεσης έχουμε φαινόμενα υπερακόντισης μέχρι να επέλθει ισορροπία. Όσον άφορα την ανοδική τάση που παρατηρείται είναι εφικτό να οφείλεται στην πτώση της θερμοκρασίας που προκαλείται από την εκτόνωση του διοξειδίου αυξάνοντας την πυκνότητα του αέρα με αποτέλεσμα την αύξηση της ώσης για δεδομένη παροχή. Ο μηχανισμός της ρυθμιστικής βαλβίδας φαίνεται στην Εικόνα 15.

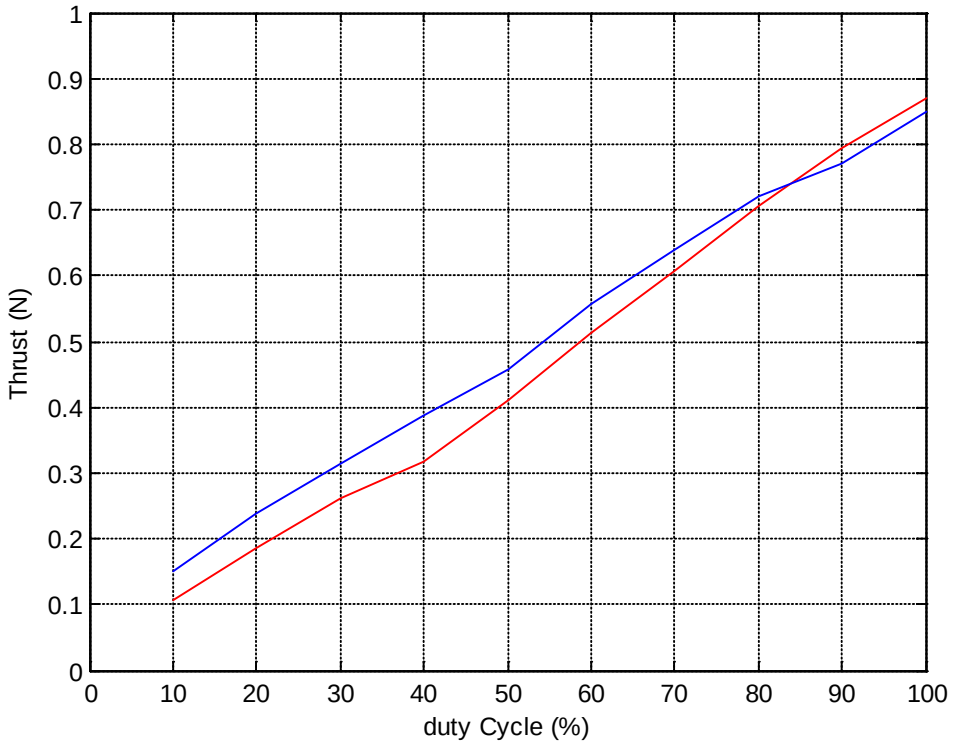


Σχήμα 5-3. Δύναμη ακροφυσίου ανοιχτό 10 δευτερόλεπτα.

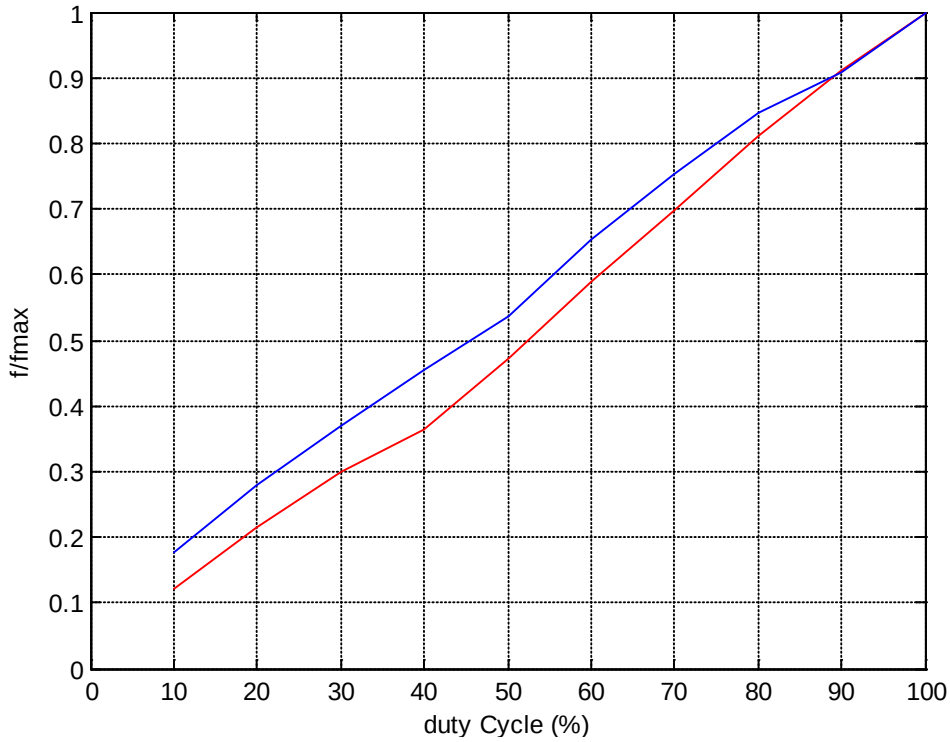


Εικόνα 15. Μηχανισμός ρυθμιστικής βαλβίδας.

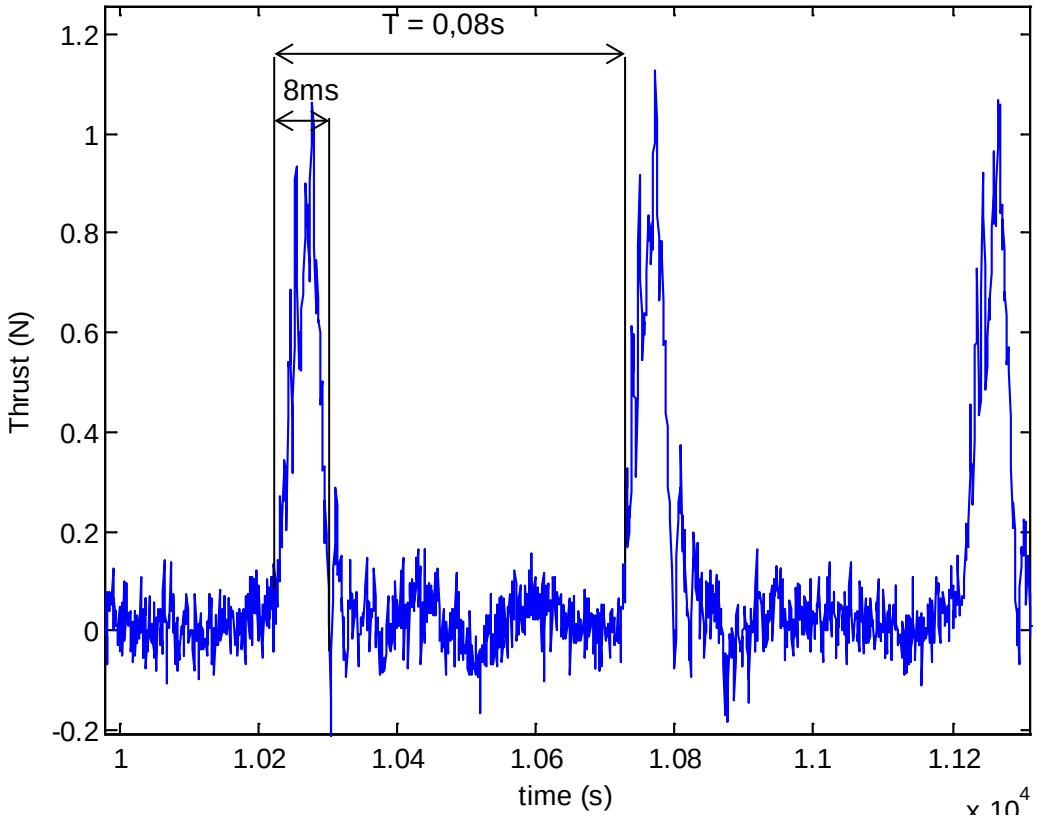
Στο Σχήμα 5-6 παρουσιάζεται η δύναμη του ακροφυσίου συναρτήσει του χρόνου για duty cycle 10. Στο Σχήμα 5-7 παρουσιάζεται η δύναμη του ακροφυσίου για duty cycle 50%.Είναι άξιο παρατήρησης συγκρίνοντας το Σχήμα 5-6 με το Σχήμα 5-7 ότι ενώ τα δύο σχήματα έχουν την ίδια διάρκεια φαίνονται λιγότεροι παλμοί στο πρώτο, ένω παράλληλα το πλάτος ανοίγματος ειίναι ίδιο, πράγμα που δείχνει τη λειτουργία του αλγόριθμου PWM μεταβλητής συχνότητας. Για κάθε τέτοιο σχήμα υπολογίζουμε μια μέση τιμή δύναμης για το αντίστοιχο duty cycle και τα απεικονίζουμε στο Σχήμα 5-4 και στο Σχήμα 5-5 όπου ο κατακόρυφος άξονας είναι η δύναμη και η αδίαστατη Δύναμη αντίστοιχα. Έγιναν δυο πειράματα χρησιμοποιώντας τον παραπάνω αλγόριθμο της Παραγράφου 5.1.2 με τιμές 7/5ms και 9/8 ms για τα t_{on}, t_{off} αντίστοιχα. Όπως φαίνεται στα διαγράμματα, στο πρώτο πείραμα η δύναμη είναι αρκετά γραμμική και όπως ακριβώς θα περιμέναμε.



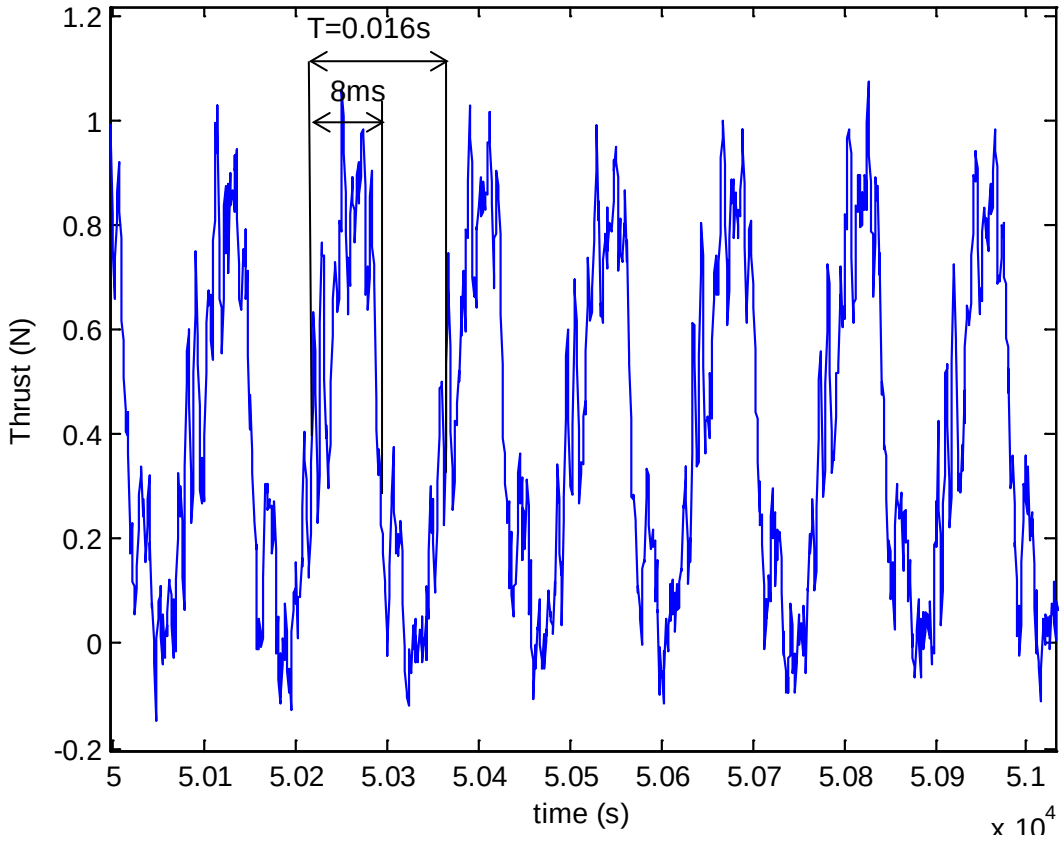
Σχήμα 5-4. Δύναμη προς ποσοστιαία μεγίστη δυνατή.



Σχήμα 5-5. Αδιάστατη δύναμη προς επιθυμητή.



Σχήμα 5-6. Δύναμη ακροφυσίου για 20% duty cycle.



Σχήμα 5-7. Δύναμη ακροφυσίου για 50% duty cycle.

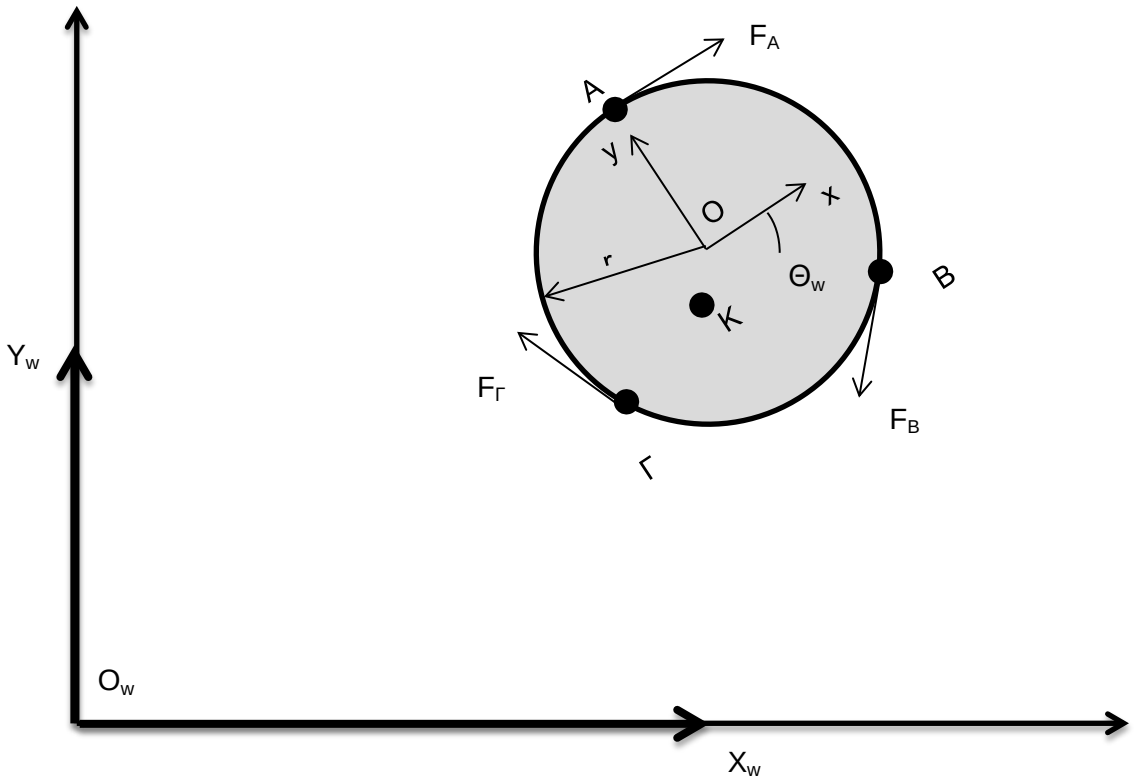
5.1.4 Βαθμονόμηση επενεργητών.

Το γεγονός ότι το ρομπότ και ο έλεγχος του γίνονται στο περιβάλλον του MATLAB έκανε πολύ απλή τη βαθμονόμηση και τον έλεγχο των επενεργητών μιας και τα δεδομένα του πειράματος δηλαδή το Σχήμα 5-4 μπορούν να εισαχθούν ως πίνακας αντιστοίχισης (lookup table) στην έξοδο του ελεγκτή μετατρέποντας τις μονάδες δύναμης σε *τ* (duty cycle). Με δεδομένο ότι το σύστημα δε θα ζητήσει ποτέ περισσότερη δύναμη από την παρεχόμενη ο ελεγκτής δε χρειάζεται να τροποποιηθεί. Αυτός ο περιορισμός προέκυψε ως απαίτηση του πνευματικού συστήματος και περιγράφεται στο Κεφάλαιο 5.3.

5.2 Μοντελοποίηση

Για τον προσδιορισμό των κερδών ελέγχου βασισμένου στο μοντέλο έγιναν δυο προσεγγίσεις. Η πρώτη απλουστευμένη, υποθέτει το ρομπότ ως ένα ομοιογενή κύλινδρο του όποίου το κέντρο μάζας βρίσκεται στο γεωμετρικό κέντρο των κυλίνδρων. Εφόσον το ρομπότ κινείται στο επίπεδο το πρόβλημα είναι δυσδιάστατο. Η δεύτερη προσέγγιση έγινε υποθέτοντας ότι το κέντρο μάζας του ρομπότ δεν συμπίπτει με το γεωμετρικό του κέντρο και άρα κατά την περιστροφή του το ρομπότ θα περιστραφεί γύρω από το κέντρο μάζας του και όχι το γεωμετρικό του κέντρο. Για να συμβεί αυτό πρέπει να ασκηθεί μια ακόμη δύναμη που θα παίξει το ρόλο της κεντρομόλου ώστε το ρομπότ να περιστραφεί γύρω από άλλο σημείο. Η μαθηματική διατύπωση των δύο μοντέλων ακολουθεί στη παράγραφο 5.2.1

5.2.1 Μαθηματικά μοντέλα



Σχήμα 5-8. Συστήματα συντεταγμένων.

Για την μοντελοποίηση του ρομπότ ορίζουμε τα παρακάτω:

- Το σταθερό σύστημα αναφοράς με αρχή το O_w
- Το σωματόδετο σύστημα αναφοράς με αρχή το σημείο Ο
- Τη στάση του ρομπότ $\mathbf{O_wO}$ (Y_w,Y_w,θ) περιλαμβάνει συντεταγμένες του σημείου Ο και την περιστροφή του σωματόδετου ως προς το σταθερό σύστημα συντεταγμένων
- Τις δυνάμεις από τα ακροφύσια $\mathbf{F_A}$, $\mathbf{F_B}$, $\mathbf{F_Γ}$
- Ένα πίνακα περιστροφής **R** από το **O** στο $\mathbf{O_w}$

Υποθέτοντας το ρομπότ ως κύλινδρο γραφούμε την εξίσωση κίνησης του ρομπότ στο σταθερό σύστημα συντεταγμένων.

Ως μεταβλητές καταστάσης θεωρούνται θέση και προσανατολισμό στο επίπεδο.

$$\boldsymbol{f_w} = m\ddot{\boldsymbol{x}} \tag{5.14}$$

$$\begin{bmatrix} f_{x\,w} \\ f_{y\,w} \\ M_W \end{bmatrix} = \boldsymbol{M} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\theta} \end{bmatrix} \tag{5.15}$$

Εκφράζοντας την δύναμη στο σύστημα συντεταγμένων του ρομπότ:

$$\boldsymbol{R}(\theta) \begin{bmatrix} f_x \\ \mathrm{f_y} \\ M \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} \tag{5.16}$$

$$\boldsymbol{R}(\theta) \begin{bmatrix} f_{A_\chi} + f_{B_\chi} + f_{\Gamma_\chi} \\ f_{A_y} + f_{B_y} + f_{\Gamma_y} \\ (f_A + f_B + f_\Gamma)r \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} \tag{5.17}$$

Όπου:

$$f_{A_x} = f_A \cos(180°) \tag{5.18}$$

$$f_{A_y} = f_A \sin(180°) \tag{5.19}$$

$$f_{B_x} = f_B \cos(180° + 120°) \tag{5.20}$$

$$f_{B_y} = f_B \sin(180° + 120°) \tag{5.21}$$

$$f_{\Gamma_x} = f_\Gamma \cos(180° + 240°) \tag{5.22}$$

$$f_{\Gamma_y} = f_\Gamma \sin(180° + 240°) \tag{5.23}$$

Και άρα:

$$\boldsymbol{R}(\theta) \begin{bmatrix} -f_A + \dfrac{f_B}{2} + \dfrac{f_\Gamma}{2} \\ +\dfrac{\mathrm{f_B}\sqrt{3}}{2} - \dfrac{\mathrm{f_\Gamma}\sqrt{3}}{2} \\ (f_A + f_B + f_\Gamma)r \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} \tag{5.24}$$

$$\boldsymbol{R}(\theta)\begin{bmatrix} -1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ r & r & r \end{bmatrix}\begin{bmatrix} f_A \\ f_B \\ f_\Gamma \end{bmatrix} = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}\begin{bmatrix} \ddot{x_w} \\ \ddot{y_w} \\ \ddot{\theta} \end{bmatrix} \quad (5.25)$$

Για την δεύτερη μοντελοποίηση όπου δε συμπίπτει το γεωμετρικό κέντρο με το κέντρο μάζας. Αφού αποδείχτηκε ότι η ροπή που ασκείται απο τα ακροφύσια στο ρομπότ ειναι ανεξάρτητη του σήμειου ως προς το οποίο μετράται. Γενικότερα αποδείχθηκε ότι δυνάμεις που ασκούνται εφαπτόπτενες σε κύκλο ανά ίσες γωνίες παράγουν ροπή ανεξάρτητη του σημείου αναφοράς. Το μοντέλο που προκύπτει για το ρομπότ είναι το ίδιο με τη διαφορά ότι η ροπή αδράνειας του είναι διαφορετική και ίση με:

$$J_{offcenter} = J + m(OK)^2 \quad (5.26)$$

Άρα για την πλήρη γνώση του μοντέλου απαιτείται η γνώση των αδρανειακών παραμέτρων του μοντέλου, δηλαδή μάζα και ροπή αδράνειας.

5.2.2 Προσδιορισμός στοιχείων αδράνειας.

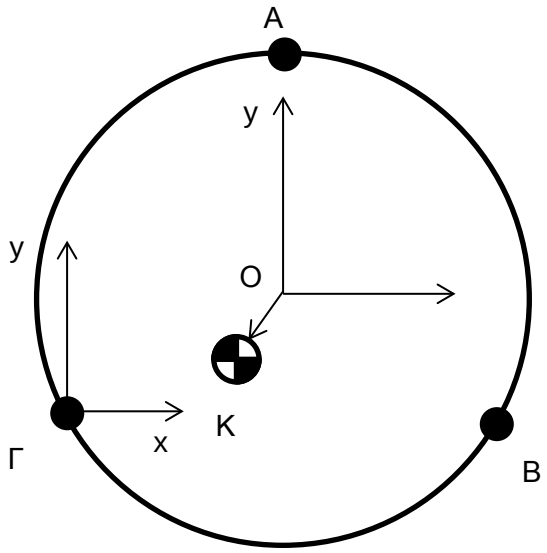
Στην παραπάνω μοντελοποίηση όπου το κέντρο μάζας δε έχει υποτεθεί στο γεωμετρικό κέντρο του (κέντρου του κυκλικού του σώματος) πρέπει να καθοριστεί η θέση του κέντρου μάζας του πειραματικά. Για τον υπολογισμό τοποθετήθηκε το ρομπότ πάνω σε τρεις ζυγούς για να μετρηθούν η αντιδράσεις στηρίξεις και να υπολογιστεί η θέση του. Για το πείραμα χρησιμοποιήθηκαν τρεις ίδιου τύπου ζυγοί όπου ο κάθενας στήριζε ένα από τα τρία στηρίγματα του ρομπότ. Επειδή παρατηρήθηκαν μικρές αποκλείσεις το πείραμα επαναλήφθηκε 8 φορές εναλλασωντας τους ζυγούς. Τα δέδομένα του πειράματος παρουσιάζονται στο Πίνακας 3.

Πίνακας 3. Μετρήσεις βάρους για τον υπολογισμό κέντρου μάζας.

μέτρηση	W_A	W_Γ	W_B	
1	2416	3517	3467	
2	2420	3536	3446	
3	2445	3543	3413	
4	2457	3509	3428	
5	2455	3437	3408	
6	2455	3456	3379	
7	2426	2460	3391	
8	2472	3413	3403	Σύνολο
Μέση τιμή	2443.2	3358.87	3416.87	9219

Στο Σχήμα 5-9 φαίνεται η κάτοψη του ρομπότ. Τα Α, Β, Γ είναι οι στυρίξεις του ρομπότ, το σύστημα συντεταγμένων με αρχή το Ο είναι το κύριο που χρησιμοποιείται σε όλη την εργασία και με αρχή το Γ είναι το βοηθητικό σύστημα συντεταγμένων που χρησιμοποιείται για τον υπολογισμό της θέσης του κέντρου μάζας Κ. Ο υπολογισμός της θέσης του κέντρου μάζας Κ στο σωματόδετο σύστημα συντεταγμένων του ρομπότ γίνεται αντιμετωπίζοντας το πρόβλημα δισδιάστατα. Αυτό είναι εφικτό εφόσον το ρομπότ μπόρει να περιστραφει μόνο γύρω από τον άξονα z. Με αυτό την προσέγγιση δεν απαιτείται ο υπολογισμός της συνηστώσας του κέντρου μάζας στο z άξονα, πρέπει όμως κατά την ζύγιση του ρομπότ οι στυρίξεις του να είναι στο ίδιο επίπεδο το οποίο θα είναι παράλληλο με το έδαφος.

Έχουμε λοιπόν τα διανύσματα θέσης των στηρίξεων στο σύστημα συντεταγμένων με αρχή το Γ και άξονες όπως φαίνονται στο σχήμα:



Σχήμα 5-9. Σώμα ρομπότ και το σύστηματα συντεταγμένων.

$$\boldsymbol{\Gamma A} = \begin{bmatrix} 142.5 \\ 246.8172 \end{bmatrix} \tag{5.27}$$

$$\boldsymbol{\Gamma B} = \begin{bmatrix} 285 \\ 0 \end{bmatrix} \tag{5.28}$$

$$\boldsymbol{\Gamma K} = \frac{\boldsymbol{\Gamma A} \cdot W_A + \boldsymbol{\Gamma B} \cdot W_B + 0 \cdot W_\Gamma}{W_A + W_B + W_\Gamma} \tag{5.29}$$

Εφόσον τα Α, Β, Γ και οι αντιδράσεις W_A, W_B, W_Γ είναι γνωστά.

$$\boldsymbol{\Gamma K} = \begin{bmatrix} 140.9839 \\ 64.1005 \end{bmatrix} mm \tag{5.30}$$

Άρα στο σωματόδετο σύστημα αναφοράς με αρχή Ο, το κέντρο μάζας είναι το διάνυσμα **OK** που δίνεται από την σχέση:

$$\boldsymbol{OK} = \boldsymbol{\Gamma K} - \boldsymbol{\Gamma O}$$

$$\boldsymbol{OK} = \begin{bmatrix} 140.9839 \\ 64.1005 \end{bmatrix} - \begin{bmatrix} 142.5 \\ 82.2724 \end{bmatrix} \quad (5.31)$$

$$\boldsymbol{OK} = \begin{bmatrix} -1.5161 \\ -18.1719 \end{bmatrix} mm \quad (5.32)$$

Άρα,

$$|\boldsymbol{OK}| = 18{,}235\ mm$$

Άρα έχουμε μάζα και ροπή αδράνειας:

$$m = 9{,}219\ kg \quad (5.33)$$

$$J = \frac{1}{2}mR^2 = 9{,}219 \cdot 0{,}15^2 = 103{,}714\ kgm^2 \quad (5.34)$$

$$J_{offcenter} = J + m(OK)^2 = 103{,}714 + 9219 \cdot 0{,}018235^2 = 106{,}78\ kgm^2 \quad (5.35)$$

$$J_{offcenter} = 106{,}78\ kgm^2 \quad (5.36)$$

5.2.3 Προωστήρες

Τα ακροφύσια δεν θεωρούνται ότι έχουν κάποια δυναμική. Στα πλαίσια της μοντελοποίησης τους ανήκει μόνο η μετατροπή της δύναμης σε **τ** duty cycle μέσω του πίνακα αντιστοίχισης που έχει δημιουργηθεί από τα πειραματικά δεδομένα της παραγράφου 5.1.4 .

5.3 Επιλογή ελέγχου

Στο πλαίσιο της εργασίας αυτής, ο έλεγχος θέσης που υλοποιήθηκε δε αποτελεί το βασικό έλεγχο που θα χρησιμοποιηθεί στο μέλλον για τα πειράματα αλλά καλείται να επιβεβαιώσει το σύστημα οπτικής ανάδρασης θέσης μέσω κάμερας, την μαθηματική μοντελοποίηση του ρομπότ, καθώς και το σύστημα των ακροφυσίων ως επενεργητες δύναμης. Με βάση τα παραπάνω επιλέχθηκε ένας έλεγχος PD λόγω της απλότητας του και της αξιοπιστίας του. Ο ελεγκτής σχεδιάζεται βασισμένος στο μοντέλο, δηλαδή θα είναι προσαρμοσμένος στην δυναμική του συστήματος, επιλέγοντας την απόκριση που επιθυμούμε. Στην περίπτωση μας δεν επιτρέπεται υπερακόντιση γιατί σε περιπτώσεις προσέγγισης (Docking) κάτι τέτοιο θα προκαλούσε σύγκρουση διαταράσσοντας όλο το πείραμα. Έτσι επιλέχτηκε *ζ=1*. Επίσης παρατηρήθηκε ότι σε περίπτωση πλήρους ανοίγματος κάποιου ακροφυσίου δημιουργείται πτώση πίεσης τέτοια ώστε τα αεροεδρανα να μην λειτουργούν σωστά με αποτέλεσμα το ρομπότ να έχει τριβή με το γρανίτη και να μην μπορεί να μετακινηθεί. Με βάση την παραπάνω παρατήρηση επιλέχθηκε το κέρδος του ελεγκτή έτσι ώστε να μην υπάρχει κορεσμός στα ακροφύσια, δηλαδή να ζητείται πάντα δύναμη ελάχιστα μικρότερη από την μεγίστη δυνατή. Με αυτό το τρόπο αποφεύγεται το πλήρες άνοιγμα των ακροφυσίων και παράλληλα επιτυγχάνεται οικονομία CO_2 αρά μεγαλύτερη αυτονομία.

Διατυπώνοντας την εξίσωση κίνησης του κλειστού βρόχου:

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} = K_P \boldsymbol{e} + K_d \dot{\boldsymbol{e}}$$

$$K_d \left(\begin{bmatrix} \dot{x}_{Wt} \\ \dot{y}_{Wt} \\ \dot{\theta}_t \end{bmatrix} - \begin{bmatrix} \dot{x_W} \\ \dot{y_W} \\ \dot{\theta} \end{bmatrix} \right) + K_p \left(\begin{bmatrix} x_{Wt} \\ y_{Wt} \\ \theta_t \end{bmatrix} - \begin{bmatrix} x_W \\ y_W \\ \theta \end{bmatrix} \right) = \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} \tag{5.37}$$

$$\begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} - K_d \left(\begin{bmatrix} \dot{x}_{Wt} \\ \dot{y}_{Wt} \\ \dot{\theta}_t \end{bmatrix} - \begin{bmatrix} \dot{x_W} \\ \dot{y_W} \\ \dot{\theta} \end{bmatrix} \right) - K_p \left(\begin{bmatrix} x_{Wt} \\ y_{Wt} \\ \theta_t \end{bmatrix} - \begin{bmatrix} x_W \\ y_W \\ \theta \end{bmatrix} \right) = 0 \tag{5.38}$$

$$\begin{aligned} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} - \begin{bmatrix} K_d & 0 & 0 \\ 0 & K_d & 0 \\ 0 & 0 & K_{dR} \end{bmatrix} \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}^{-1} \left(\begin{bmatrix} \dot{x}_{Wt} \\ \dot{y}_{Wt} \\ \dot{\theta}_t \end{bmatrix} - \begin{bmatrix} \dot{x_W} \\ \dot{y_W} \\ \dot{\theta} \end{bmatrix} \right) \\ - \begin{bmatrix} K_P & 0 & 0 \\ 0 & K_P & 0 \\ 0 & 0 & K_{PR} \end{bmatrix} \begin{bmatrix} m & 0 & 0 \\ 0 & m & 0 \\ 0 & 0 & J \end{bmatrix}^{-1} \left(\begin{bmatrix} x_{Wt} \\ y_{Wt} \\ \theta_t \end{bmatrix} - \begin{bmatrix} x_W \\ y_W \\ \theta \end{bmatrix} \right) = 0 \end{aligned} \tag{5.39}$$

$$\begin{aligned} \begin{bmatrix} \ddot{x_w} \\ \ddot{y}_w \\ \ddot{\Theta} \end{bmatrix} - \begin{bmatrix} \frac{K_d}{m} & 0 & 0 \\ 0 & \frac{K_d}{m} & 0 \\ 0 & 0 & \frac{K_{dR}}{J} \end{bmatrix} \left(\begin{bmatrix} \dot{x}_{Wt} \\ \dot{y}_{Wt} \\ \dot{\theta}_t \end{bmatrix} - \begin{bmatrix} \dot{x_W} \\ \dot{y_W} \\ \dot{\theta} \end{bmatrix} \right) \\ - \begin{bmatrix} \frac{K_P}{m} & 0 & 0 \\ 0 & \frac{K_P}{m} & 0 \\ 0 & 0 & \frac{K_{PR}}{J} \end{bmatrix} \left(\begin{bmatrix} x_{Wt} \\ y_{Wt} \\ \theta_t \end{bmatrix} - \begin{bmatrix} x_W \\ y_W \\ \theta \end{bmatrix} \right) = 0 \end{aligned} \tag{5.40}$$

Εάν η μητρωική μορφή της σχέσης (5.40) μετατραπεί σε τρείς σχέσεις έχουμε:

$$\ddot{x_w} - \frac{K_d}{m} (\dot{x}_{Wt} - \dot{x_W}) - \frac{K_P}{m} (x_{Wt} - x_W) = 0 \tag{5.41}$$

$$\ddot{y}_w - \frac{K_d}{m} (\dot{y}_{Wt} - \dot{y_W}) - \frac{K_P}{m} (y_{Wt} - y_W) = 0 \tag{5.42}$$

$$\ddot{\Theta} - \frac{K_{dR}}{J} \left(\dot{\theta}_t - \dot{\theta} \right) - \frac{K_{PR}}{J} (\theta_t - \theta) = 0 \tag{5.43}$$

Αυτές οι σχέσεις εχουν την τυπική μορφή:

$$s^2 + 2s\zeta\omega + \omega^2 = 0 \tag{5.44}$$

Όπου,

$$\frac{K_P}{m} = \omega^2 \tag{5.45}$$

$$\frac{K_{PR}}{m} = \omega_R^2 \tag{5.46}$$

$$\frac{K_d}{m} = \zeta\omega \tag{5.47}$$

$$\frac{K_{dR}}{m} = \zeta\omega_R \tag{5.48}$$

Επιλέγουμε τα κέρδη ώστε να έχουμε κρίσιμη απόσβεση ζ=1 και να μην υπάρχει κορεσμός στις βαλβίδες (εξηγήθηκε) δηλαδή για μέγιστο σφάλμα να απαιτείτε από τον έλεγχο ελάχιστα μικρότερη από τη μέγιστη παρεχόμενη δύναμη.

Αφού έχουν υπολογιστεί τα κέρδη, υπολογίζονται οι δυνάμεις και ροπές που απαιτεί το σύστημα και στην συνέχεια υπολογίζονται οι δυνάμεις που πρέπει να ασκήσουν τα ακροφύσια για την παραγωγή αυτών.

$$K_P\boldsymbol{e} + K_d\dot{\boldsymbol{e}} = \begin{bmatrix} f_X \\ f_Y \\ \mathrm{T} \end{bmatrix} = \boldsymbol{R}(\theta)\boldsymbol{T}\begin{bmatrix} f_A \\ f_B \\ f_\Gamma \end{bmatrix} \tag{5.49}$$

Ο πίνακας T συσχετίζει τις δυναμεις και ροπές που ασκούνται στο ρομπότ. εκφράσμένες στο σωματόδετο συστημα συντεταγμένων του με αρχή το O (βλ. Σχήμα 5-9) που ασκούνται στο ρομπότ με τις δυνάμεις που ασκούν τα ακροφύσια.

Όπου,

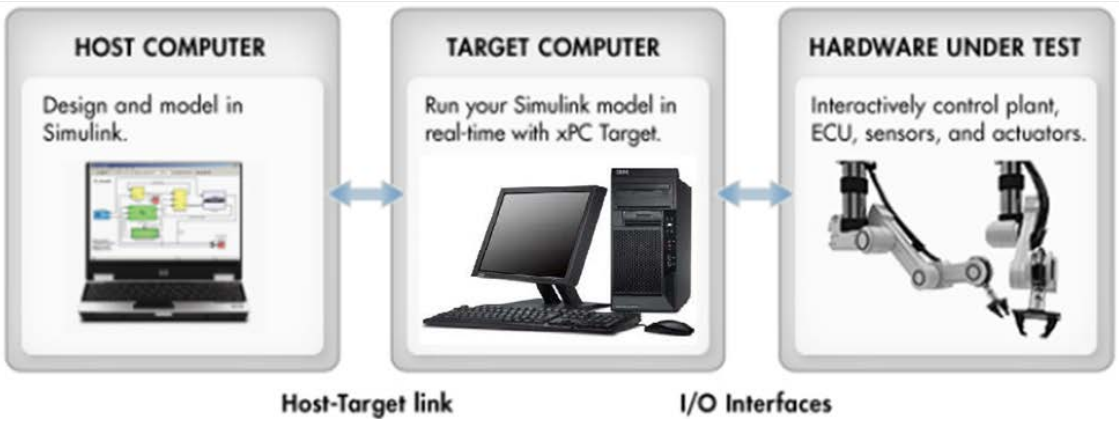
$$\boldsymbol{T} = \begin{bmatrix} -1 & \frac{1}{2} & \frac{1}{2} \\ 0 & \frac{\sqrt{3}}{2} & -\frac{\sqrt{3}}{2} \\ r & r & r \end{bmatrix} \tag{5.50}$$

Για να υπολογίσουμε τελικά τα f_A, f_B, f_Γ πρέπει να αντιστραφεί ο πινάκας T όμως δε είναι αντιστρέψιμος επομένως χρησιμοποιούμε το ψευδό-αντίστροφο του για τον προσδιορισμό των δυνάμεων. Ο ψευδό-αντίστροφος ελαχιστοποιεί την νόρμα άρα:

$$\begin{bmatrix} f_A \\ f_B \\ f_\Gamma \end{bmatrix} = \boldsymbol{R^{-1}}(\theta)\boldsymbol{T^{-1}}(\, K_P\boldsymbol{e} + K_d\dot{\boldsymbol{e}}\,) \tag{5.51}$$

6 Υλοποίηση στο xPC target

Το xPC Target [15] είναι μια λύση για τη δημιουργία πρωτοτύπων, δοκιμών και ανάπτυξη συστημάτων πραγματικού χρόνου χρησιμοποιώντας κοινούς υπολογιστές. Είναι ένα περιβάλλον που χρησιμοποιεί δυο υπολογιστές, έναν υπολογιστή (target PC) για την εκτέλεση εφαρμογών σε πραγματικό χρόνο και έναν που τρέχει το περιβάλλον του Simulink και ελέγχει τον Target PC. Δίνει τη δυνατότητα να συνδεθούν μοντέλα Simulink® και Stateflow® με φυσικά συστήματα και να εκτελούνται σε πραγματικό χρόνο σε συμβατικό υπολογιστή. Είναι ιδανικό για την ταχεία προτυποποίηση και αξιολόγηση των συστημάτων ελέγχου hardware-in-the-loop και επεξεργασίας δεδομένων. Επιτρέπει τη χρήση των φυσικών εισόδων/εξόδων ενός υπολογιστή μέσω μπλοκ τα οποία συνδέονται με τα υπόλοιπα μπλοκ του Simulink®, δημιουργεί αυτόματα κώδικα που με τη σειρά του κατεβάζεται στο δεύτερο υπολογιστή (xPC Target) που τρέχει σε πραγματικό χρόνο. Το xPC target τρέχει σε συμβατό υλικό όπως Intel, AMD, και άλλους x86-συμβατούς επεξεργαστές σε πραγματικό χρόνο. Ο υπολογιστής-Target μπορεί να είναι ένα επιτραπέζιος υπολογιστής, ένας βιομηχανικός υπολογιστής, ένα PC/104 ή PC/104+, CompactPCI, ή ένα υπολογιστή πλακέτας PC (βλ. Εικόνα 17).



Εικόνα 16. Η λογική του xPC Target.

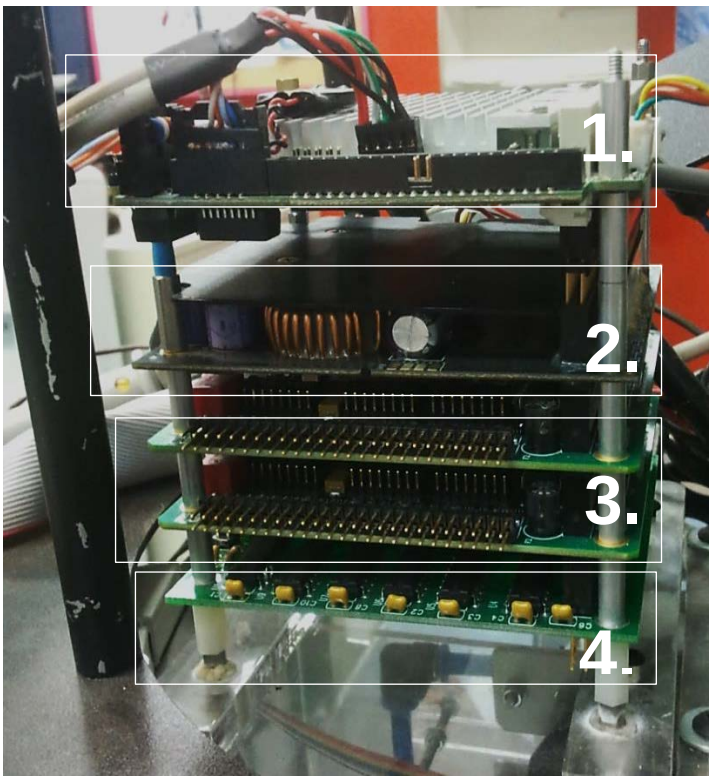
6.1 Περιγραφή διάταξης

Για την εφαρμογή μας επιλέχτηκε η χρήση ενός υπολογιστή μορφής PC/104 για τους παρακάτω λόγους:

- Μικρό μέγεθος, ώστε να μπορεί να ενσωματωθεί στο ρομπότ
- Χαμηλή κατανάλωση ενέργειας, ώστε να λειτουργεί αυτόνομα με μπαταρία
- Εύκολη προτυποποίηση καθώς δίνει την δυνατότητα να προσθέτονται λειτουργίες ενώνοντας συμβατές κάρτες σε μορφή πύργου
- Υποστήριξη των καρτών αυτών με έτοιμα μπλοκ στο περιβάλλον του xPC target

Η διάταξη αποτελείται από τον πύργο των καρτών του υπολογιστή του ρομπότ που φαίνεται στην Εικόνα 17 και από τις δυο ασύρματες γέφυρες δικτύου που χρησιμοποιούνται.

1. Ο υπολογιστής πλακέτας τύπου PC/104 PCM-3370 της Advantech,
2. τροφοδοτικό συμβατό με HE104 της TRI-M που αναλαμβάνει να παρέχει τις απαραίτητες τάσεις του υπολογιστή,
3. Δυο κάρτες DM6914 της RTD για την αποκωδικοποίηση αισθητήρων θέσης που χρησιμοποιούνται στους βραχίονες,
4. Κάρτα ψηφιακών εισόδων/εξόδων PC104-DO48H της measurement computing, για έλεγχο προωστήρων και μελλοντικών συστημάτων,
5. Δυο γέφυρες δικτύου WNCE2001 της NETGEAR μια για την επικοινωνία με την κάμερα και μια με το Host υπολογιστή για τον έλεγχο της ροής του προγράμματος.



Εικόνα 17. Πύργος PC/104 εγκατεστημένος στο ρομπότ.

Η διάταξη αυτή αποτελεί τον υπολογιστή xPC target του συστήματος και αναλαμβάνει την υλοποίηση του PD ελεγκτή θέσης/προσανατολισμού που σχεδιάστηκε παραπάνω. Συνδέεται με την πλακέτα έλεγχου των βαλβίδων για να πραγματοποιήσει την ψευδοαναλογική λειτουργία των βαλβίδων που περιγράφηκε στην Παράγραφο 5.1.2. Λαμβάνει μέσω του δικτύου του εργαστηρίου μέ πρωτοκόλλο UDP τις συντεταγμένες, τον προσανατολισμό, την παράγωγο των προηγούμενων καθώς και την χρονική καθυστέρηση της πληροφορίας αυτής και ένα αναγνωριστικό για την αντιστοίχιση των τιμών αυτών με το σωστό ρομπότ.

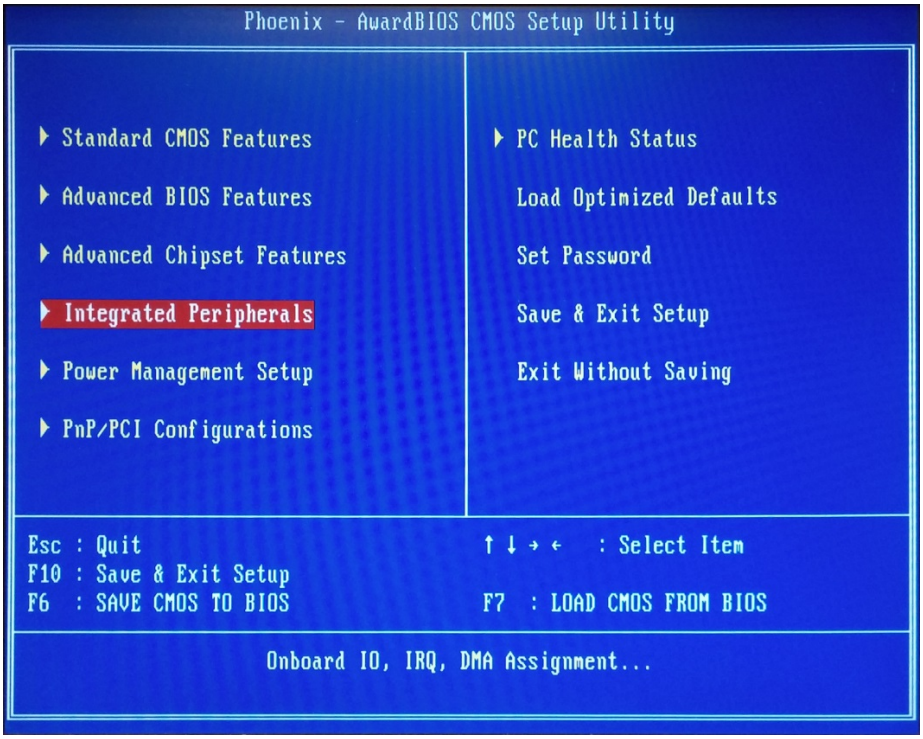
Πιο συγκεκριμένα ο υπολογιστής αυτός για να υλοποίησει τα παραπάνω τρέχει ένα βασικό λειτουργικό του xPC από ένα flash drive. Αυτό εκκινεί κάποια βασικά στοιχειά του όπως η κάρτα δικτύου για την επικοινωνία με το Host υπολογιστή και ένα γραφικό περιβάλλον το οποίο παρέχει πληροφορίες για την όλη διαδικασία φορτώματος και εκτέλεσης του προγράμματος. Το σύστημα έχει ρυθμιστεί να τρέχει με περίοδο 0.01 [ms] ώστε να υπάρχει επαρκής διαφοροποίηση κατά τον ψευδοαναλογικο έλεγχο τον βαλβίδων μιας και αποτελεί μια διαδικασία που είναι ευαίσθητη στο χρόνο.

6.1.1 Εγκατάσταση και ρύθμιση καρτών PC/104

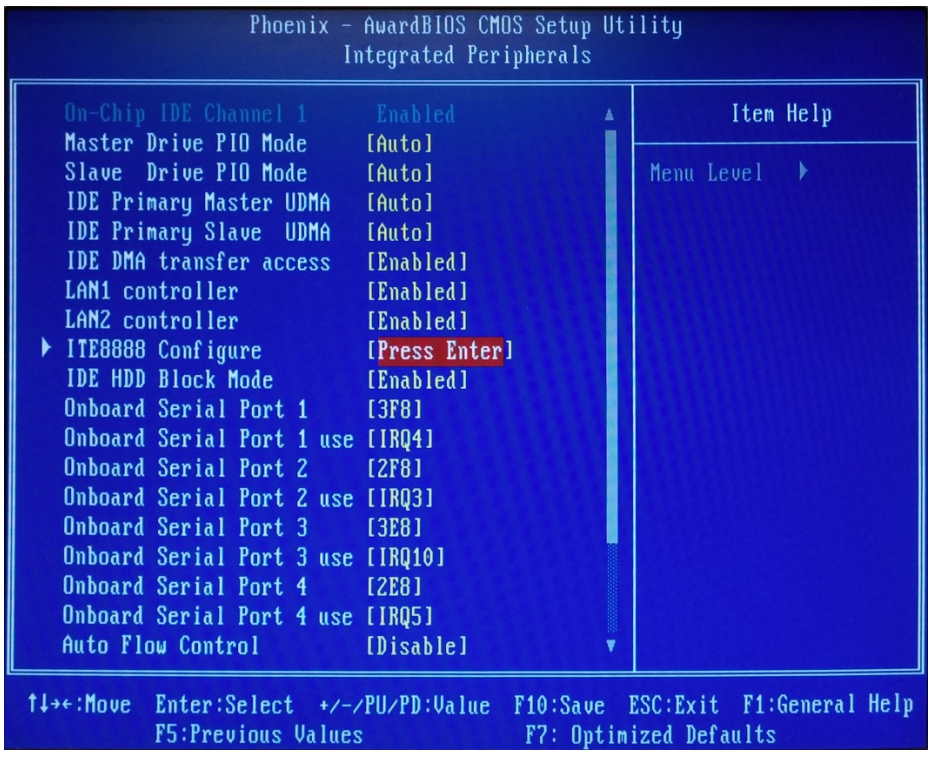
Το παρόν υπολογιστικό σύστημα, για να είναι σε θέση να αλληλεπιδρά με τα ηλεκτρομηχανικά συστήματα του ρομπότ, είναι εξοπλισμένο με κάρτες που υλοποιούν συγκεκριμένες λειτουργίες (βλ. Εικόνα 17). Αυτές οι κάρτες επικοινωνούν με τον επεξεργαστή μέσω του πρωτοκόλλου ISA BUS. Αυτό το πρωτόκολλο δεν υποστηρίζεται απευθείας από τον επεξεργαστή που διαθέτει το σύστημα αλλά έρχεται μαζί με μια ενσωματωμένη περιφερειακή συσκευή IT-8888 που υλοποιεί αυτό το πρωτόκολλο και ρυθμίζεται από τα BIOS του συστήματος. Για να είναι σε θέση μια εφαρμογή να χρησιμοποιήσει μια τέτοια κάρτα, πρέπει αυτή να έχει μια μοναδική διεύθυνση και να είναι καταχωρημένη στη περιφερειακή συσκευή που υλοποιεί το πρωτόκολλο.

Η διαδικασία καταχώρησης της κάρτας είναι η εξής:

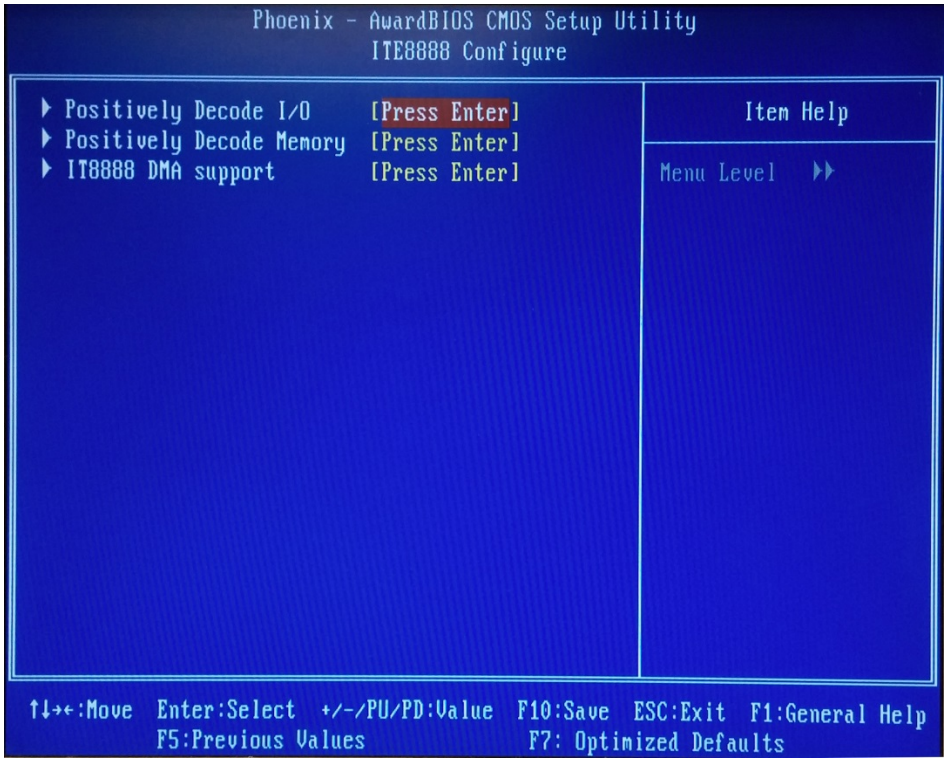
- Ανοίγουμε το μενού των BIOS του υπολογιστή του ρομπότ (βλ. Εικόνα 18), επιλέγουμε το πεδίο *Intergated Peripherals*, στη συνέχεια εντοπίζουμε την συσκευή IT8888 (βλ. Εικόνα 19), ύστερα επιλέγουμε *Positively Decode I/O* δηλαδή επικοινωνία με συσκευή εισόδων/εξόδων (βλ. Εικόνα 20),
- Επιλέγουμε με βάση τις διευθύνσεις που μπορεί να πάρει η κάρτα, σύμφωνα με τον κατασκευαστή μια που δεν χρησιμοποιείται από άλλη συσκευή. (βλ. Παράρτημα Α Εγχειρίδια),
- Εντοπίζουμε στο εγχειρίδιο της κάρτας το μήκος bit που απαιτείται για την επικοινωνία με αυτήν,
- Καταχωρούμε τη διεύθυνση της κάρτας, το μέγεθος της μνήμης σε bit και την ταχύτητα επικοινωνίας, για την οποία ο κατασκευαστής του επεξεργαστή συνιστά την επιλογή *Medium Speed* έναντι της προκαθορισμένης *Fast Speed* που όπως εντοπίστηκε κατά τις δοκιμές προκαλεί πάγωμα του συστήματος,
- Επιλέγουμε, όπως αναγράφεται στο εγχειρίδιο της κάρτας τη διεύθυνση που επιλέξαμε μέσω των αντίστοιχων μικροδιακοπτών επάνω στην κάρτα.
- Τοποθετούμε τη κάρτα στην στοίβα του PC/104.



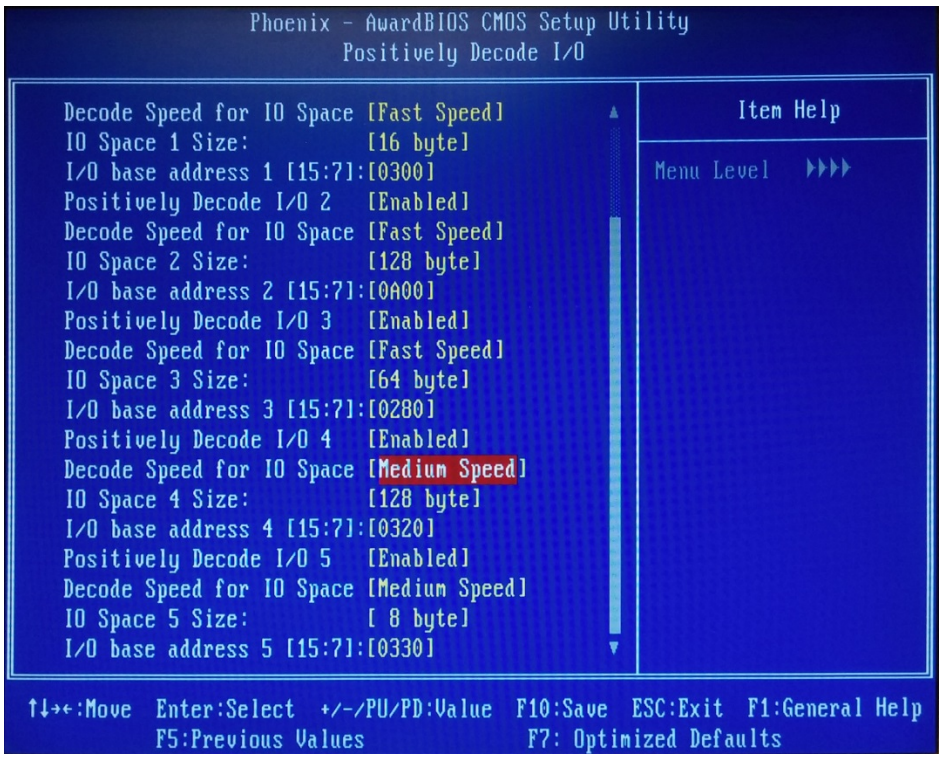
Εικόνα 18. Αρχικό μενού ρυθμίσεων στο BIOS του Target PC.



Εικόνα 19. Μενού ενσωματωμένων συσκευών στο BIOS του Target PC.



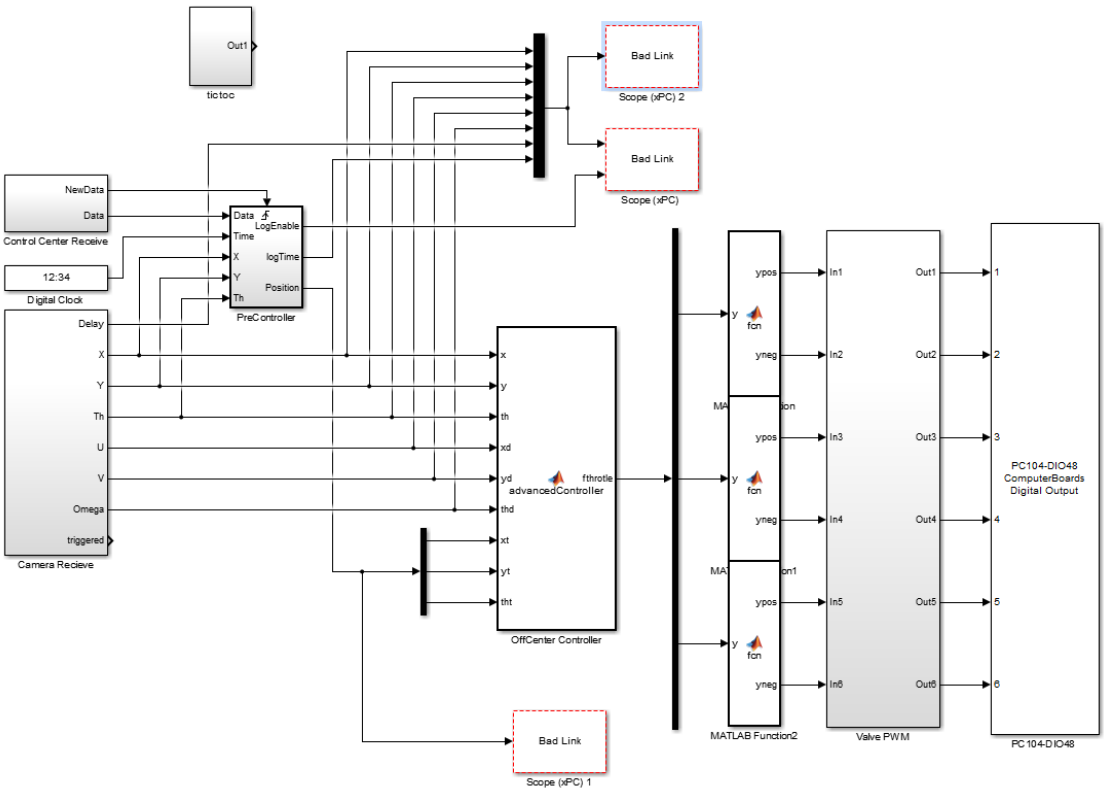
Εικόνα 20. Μενού συσκευής IT8888 στο BIOS του Target PC.



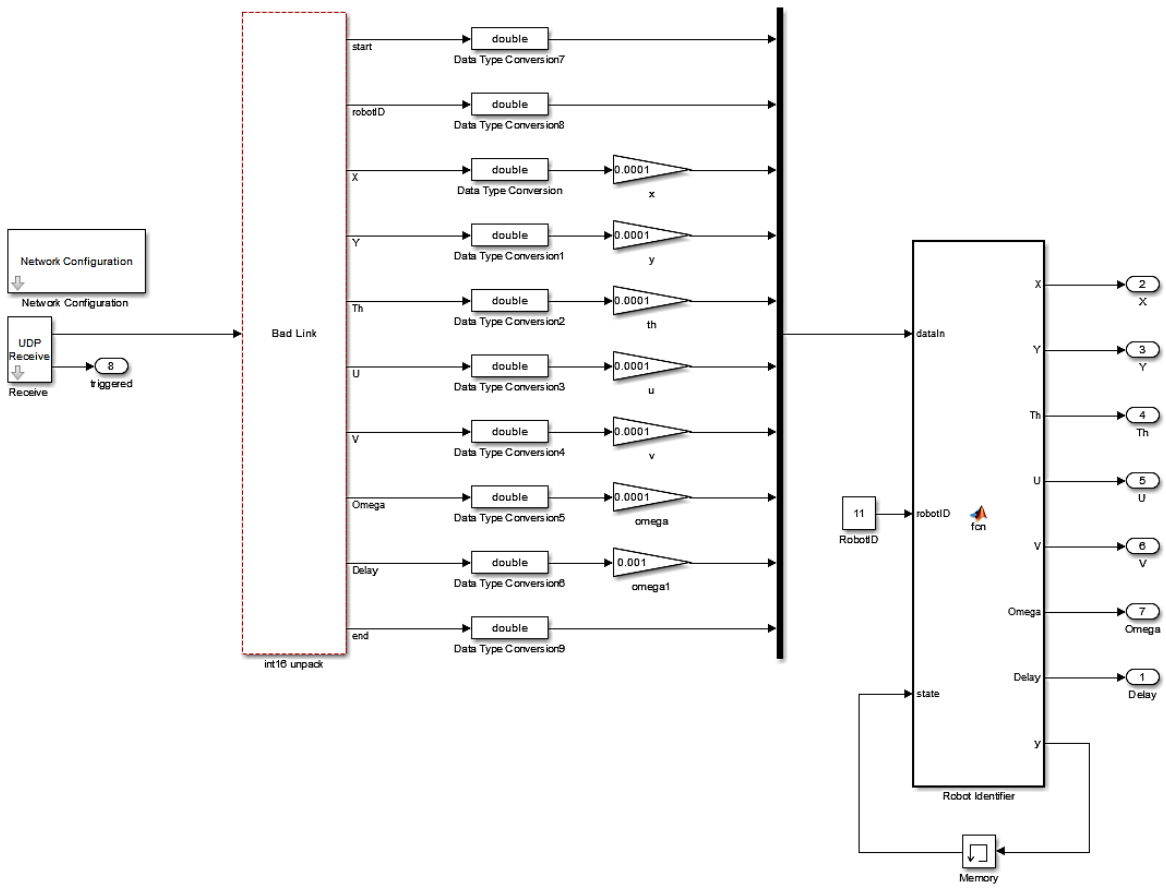
Εικόνα 21. Ρύθμιση καρτών PC104 στο BIOS του Target PC.

6.2 Υλοποίηση έλεγχου στο περιβάλλον Simulink

Σε αυτή την παράγραφο θα αναλυθεί ο τρόπος με τον οποίο υλοποιήθηκε ο έλεγχος θέσης στον υπολογιστή του ρομπότ καθώς και η υλοποίηση των επικοινωνιών του ρομπότ με το λογισμικό ανάδρασης θέσης, με το λογισμικό ελέγχου του ρομπότ και την υλοποίηση του ψευδοαναλογικού σήματος με τον αλγόριθμο που παρουσιάσθηκε στη Παράγραφο 5.1.2. Στο Διάγραμμα 6-1 φαίνεται το διάγραμμα το οποίο υλοποιεί τις παραπάνω λειτουργίες του ρομπότ. Αποτελείται από επί μέρους τμήματα (blocks) το καθένα από τα οποία υλοποιούν μια συγκεκριμένη λειτουργία και είναι μεταξύ του συνδεμένα για τη διακίνηση δεδομένων. Κάθε τέτοιο block έχει εισόδους ή/και εξόδους. Αυτές μπορεί να αντιστοιχούν σε φυσικές συνδέσεις (εισόδους/εξόδους) του υπολογιστή του ρομπότ, τμήματα κώδικα μορφής συνάρτησης δηλαδή έξοδοι που συνδέονται μέσω μαθηματικών σχέσεων με τις εισόδους, σε παραμέτρους του προγράμματος, σε τμήματα κώδικα που υλοποιούν λειτουργίες όπως γραφικά, αποθήκευση δεδομένων και άλλα.



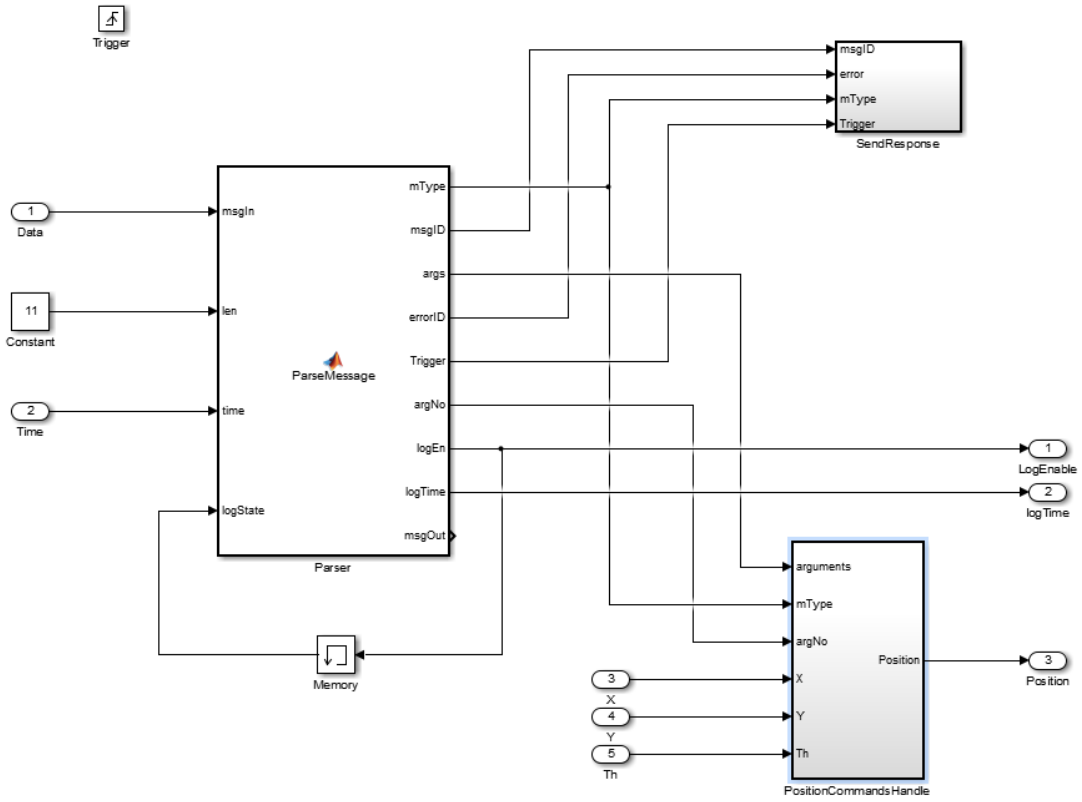
Διάγραμμα 6-1. Διάγραμμα προγραμματισμού του ρομπότ στο Simulink.



Διάγραμμα 6-2. Block λήψης και μορφοποίησης δεδομένων κάμερας.

Αρχίζοντας την περιγραφή, το block με όνομα “*Camera Receiver*” έχει την θέση, τον προσανατολισμό, την γωνιακή και γραμμική ταχύτητα του ρομπότ καθώς και την υστέρηση της πληροφορίας αυτή. Αποτελείται από ένα σύνολο blocks τα οποία φαίνονται στο Διάγραμμα 6-2 και υλοποιούν:

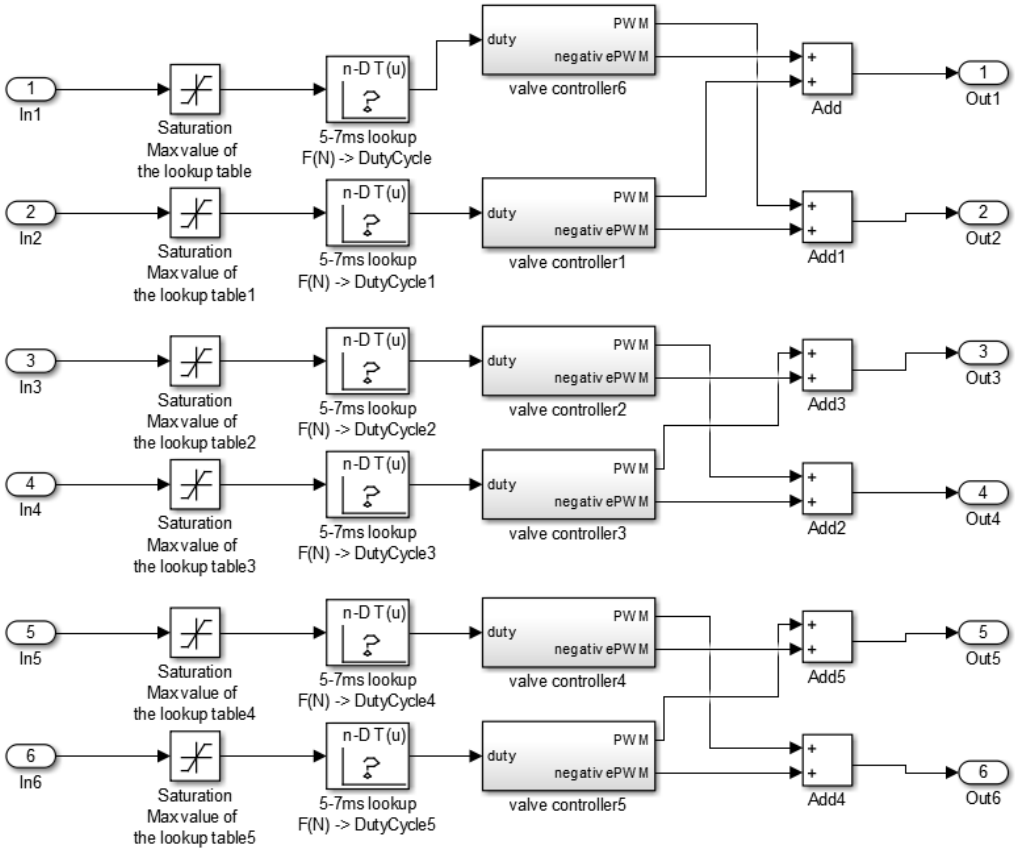
- Την επιλογή και ρύθμιση της δεύτερης κάρτας δικτύου για την σύνδεση με το λογισμικό της κάμερας
- Τη λήψη των δεδομένων μέσω πρωτοκόλλου UDP
- Τη μετατροπή των σημάτων σε μεταβλητές με φυσική σημασία (θέσης, ταχύτητας χρόνου)
- Την αναγνώριση με βάση το αναγνωριστικό του ρομπότ του σήματος και ανανέωση της θέσης του και του προσανατολισμού εάν αναφέρεται σε αυτό.



Διάγραμμα 6-3. Block επικοινωνίας με λογισμικό ελέγχου ρομπότ.

Το block με όνομα *"PreController"* έχει ως είσοδο δεδομένα που λήφθηκαν με το πρωτόκολλο TCP μέσω της κύριας κάρτας δικτύου του ρομπότ, τον χρόνο εκτέλεσης του προγράμματος και την παρούσα θέση/προσανατολισμό του ρομπότ. Δίνει ως έξοδο την επιθυμητή θέση του ρομπότ και ένα σήμα για εκκίνηση αποθήκευσης των δεδομένων θέσης καθώς και την χρονική στιγμή αυτή. Το block αυτό φαίνεται στο Διάγραμμα 6-3 και υλοποιεί:

- Μετατροπή των δεδομένων που λ;hφθηκαν σε μηνύματα συγκεκριμένης μορφής
- Διεκπεραίωση των μηνυμάτων, και κίνησης προς κάποια κατεύθυνση, περιστροφής άμεσου σταματήματος, έναρξης/λήξης καταγραφής δεδομένων θέσης
- Ανανέωση της επιθυμητής θέσης με βάση τις παραπάνω εντολές
- Αποστολή απάντησης στο δίκτυο για επικύρωση καλής επικοινωνίας με το λογισμικό έλεγχου του ρομπότ.



Διάγραμμα 6-4. Block οδήγησης βαλβίδων.

Το block με όνομα *"ValvePWM"* έχει ως είσοδο τις δυνάμεις που υπολογίστηκαν από τον έλεγχο των ακροφυσίων. Ως έξοδο έχει την οδήγηση των βαλβίδων που με το block της κάρτας PC/IO 48 μετατρέπονται σε πραγματικά σήματα. Το Block αυτό φαίνεται στο Διάγραμμα 6-4 και υλοποιεί για κάθε ακροφύσιο τον αλγόριθμο που περιγράφτηκε στην παράγραφο 5.1.2:

- Περιορισμό της δύναμης στα πραγματικά όρια,
- Μετασχηματισμό της δύναμης σε κατάλληλα σχηματισμένο παλμό για την οδήγηση των βαλβίδων με βάση τα δεδομένα που συλλέχθηκαν,
- Την μαθηματική διαδικασία του αλγορίθμου.

Το block αυτό παραλαμβάνει μόνο θετικές τιμές δύναμης αφού έχει προηγηθεί μια προεργασία κατά την οποία οι αρνητικές δυνάμεις μετατρέπονται σε θετικές εφόσον τα ακροφύσια είναι μονής δράσεως και χρησιμοποιούνται σε ζεύγη. Τα σήματα αυτού του block καταλήγουν σε ένα άλλο το οποίο είναι η γραφική αναπαράσταση μιας πραγματικής κάρτας εξόδων με την οποία τα σήματα μετατρέπονται σε πραγματικά σήματα.

Τα Blocks με όνομα "*scopes*" αναλαμβάνουν τη γραφική αναπαράσταση των δεδομένων που λαμβάνουν. Στην περίπτωση μας την θέση/προσανατολισμό του ρομπότ και οι αντίστοιχες παραγώγους των τιμών αυτών καθώς και των επιθυμητών. Το πιο σημαντικό block είναι αυτό με το όνομα *"offCneterController"* το οποίο αποτελεί το καθαρό κομμάτι του PD ελέγχου θέσης του ρομπότ.

7 Συμπεράσματα και μελλοντική εργασία

7.1 Συμπεράσματα

Κατά την υλοποίηση του λογισμικού αναγνώρισης ρομπότ μέσω κάμερας έγινε σαφές ότι η ακρίβεια των μετρήσεων του συστήματος εξαρτάται σε πολύ μεγάλο βαθμό από τον τρόπο με τον οποίο θα γίνει η βαθμονόμηση. Παρατηρήθηκε ότι η βαθμονόμηση της κάμερας με ένα μόνο στιγμιοτυπο του επιπέδου χώρου εργασίας στο σωστό ύψος (430mm πάνω από το επίπεδο του γρανίτη) παρέχει πολύ μεγαλυτερη ακρίβεια έναντι της προτεινόμενης βαθμονόμησης που είναι με πολλά στιγμιότυπα επίπεδων διαφόρων προσανατολισμών.

Επίσης η ακρίβεια του συστήματός επηρεάζεται και από την ασάφεια της θέσης της κάμερας πάνω από το χόρο εργασίας, η οποία προκαλείται κατά την βαθμονόμηση όπου απαιτείται η κατακόρυφη μετακίνηση της κάμερας για να βαθμονόμηθεί σε διαφορετικό επίπεδο από αυτό του γρανίτη. Άλλος παράγοντας είναι το ύψος των ρομπότ και η επιπεδότητα της κορηφής των ρομπότ όπου είναι τοποθετημένα τα LED.

Μια παρατήρηση που αξίζει να σημειωθεί είναι οτι παρόλο που το το νέο λογισμικό παρέχει τα δεδομένα θέσης των ρομπότ με 18 Hz το όλο σύστημα οπτικής ανάδρασης περιορίζεται από την χρήση των ασύρματων γεφύρων όπου για την επικοινωνία του υπολογιστή του λογισμικού και των ρομπότ παρεμβάλονται η κάρτα δικτύου του ρομπότ και του υπολογιστή, η γέφυρα ασυρματου δικτύου του δρομολογητή και η ασύρματη γέφυρα δικτύου του ρομπότ. Με όλα αυτά τα δεδόμένα φτάνουν στο ρομπότ με συχνότητα 4 Hz.

Κατά την υλοποίηση του ελέγχου και τις δοκιμές που έγιναν, είναι σαφές ότι το εύρος ώσης των ακροφυσίων πρέπει να μειωθεί, ώστε να γίνεται πιο λεπτός έλεγχος και το ρομπότ να μπορεί να κάνει πιο λεπτομερείς κινήσεις.

Η χρήση του μεταβλητού PWM που δημιουργήθηκε για την αύξηση του ευρους της ωσης των ακροφυσίων δεν έδωσε σημαντικά καλύτερα αποτελέσματα κατά τον έλεγχο θέσης. Αντίθετα μεγαλύτερη σημασία φάνηκε να έχει η ορθή επιλογή των νεκρών περιοχών (δηλαδή το όριο ώσης κατώ από το οποίο δε ενεργοποιείται καθόλου το ακροφύσιο).

Τέλος η χρήση του xPC target για τον προγραμματισμό του ρομπότ δε αποτελεί βέλτιστη λύση διότι λόγω της περιορισμένης υπολογιστικής ισχύς του υπολογιστή του ρομπότ και του αυστηρώς πραγματικού χρόνου λειτουργικό πόλυ συχνά το πρόγραμμα σταματάει να λειτουργεί. Το xPC target απευθύνεται σε ισχυρότερους υπολογιστές και για αυτό χρησιμοποιεί πολλούς πόρους. Χωρίς την δυνατότητα μείωσης του υπολογιστικού του βάρους δε θα μπορέσει να αποτελέσει το

λειτουργικό με το οποίο θα χρησιμοποιηθεί το ρομπότ όταν στο μέλλον θα πρόστεθουν αλγόριθμοι ελέγχου βραχιόνα και σχεδιασμού διαδρομής.

7.2 Μελλοντική Εργασία

Στα πλαίσια βελτίωσης του συστήματος τεχνητής όρασης, είναι η τροποποίηση του αλγόριθμου αναζήτησης led με κάποιον πιο ευέλικτο και πιο αξιόπιστο όπως τον αλγόριθμο *flood fill.* Στοχεύοντας με αυτό το τρόπο στον αποδοτικότερο εντοπισμό των LED και σε καλύτερη απόρριψη φωτεινών περιοχών που δεν είναι LED.

Μια πιθανή σημαντική βελτίωση θα ήταν να επανασχεδιαστεί ο κώδικας του λογισμικού χρησιμοποιώντας την τελευταία έκδοση της βιβλιοθήκης της MATRIX VISION [12] για την κάμερα που χρησιμοποιούμε διότι προσφέρει πιο συμπτυγμένο κώδικα, ευκολότερη ανάγνωση και κατανόηση από τρίτους, νέες δυνατότητες όπως ο σχεδιασμός και άλλων βασικών σχημάτων στην εικόνα και κειμένου μέσω υπορουτίνων και συναρτήσεων. Τέλος θα διακοπεί η υποστήριξη της υπάρχουσας βιβλιοθήκης μέχρι το τέλος 2015 από την εταιρία. Όλα αυτά μπορούν να συμβάλουν στην επιτάχυνση, την μείωση και την ευελιξία του κώδικα.

Μια από τις λειτουργίες του εξομοιωτή είναι και η μελέτη σεναρίων αποφυγής παθητικών αντικειμένων και συλλογή αυτών από τα ρομπότ. Επομένως είναι απαραίτητο τα αντικείμενά αυτά να έχουν τα αντίστοιχα συστήματα εντοπισμού (LED σε συγκεκριμένη απόσταση και ύψος από το επίπεδο του γρανίτη). Είναι προφανές ότι κάτι τέτοιο αυξάνει το κόστος αυτών και περιορίζει την μορφή τους. Θα ήταν εύχρηστο να αναπτυχτεί σαν μέρος του λογισμικού αυτού και μια ακόμη μέθοδος εντοπισμού παθητικών αντικειμένων χωρίς να απαιτούνται LED και συγκεκριμένο ύψος αλλά να βασίζεται εξολοκλήρου σε τεχνικές τεχνητής όρασης και να μην απαιτεί η γνώση τις ύπαρξης των αντικειμένων αυτών από το κώδικά.

Τέλος απαραίτητη βελτίωση είναι η αλλαγή του συστήματος στήριξης της κάμερας στο ταβάνι με σκοπό την αύξηση της στιβαρότητας της και την προσθήκη της δυνατότητας καθετής κίνησης προς την τράπεζα του γρανίτη χωρίς να χάνεται η ευθυγράμμιση της και το σημείο στο οποίο στέκεται για την διευκόλυνση και αύξηση της ακρίβειας βαθμονόμησης.

Βιβλιογραφία

[1] R. Y. Tsai, *A Versatile Camera Calibration Techniaue for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses.,* Robotics and Automation, 1987.

[2] Z. Z., *Flexible Camera Calibration By Viewing a Plane From Unknown Orientations,* Redmond, USA: One Microsoft Way, 1999.

[3] D. C. Brown, *Close range camera calibration,* Photgrammetric engineering, 1971.

[4] O. Faugeras, Q. T. Luong και S. Maybank, *Camera self-calibration: Theory and experiments,,* Springer Berlin/Heidelberg: Computer Vision—ECCV'92, 1992.

[5] F. Devernay και F. Olivier, *Straight lines have to be straight.,* Machine Vision and Applications,, 2001.

[6] Ε. Παπαδόπουλος και Κ. Κυριακόπουλος, *Συστήματα Ευφυούς Ελέγχου & Ρομποτική,* Αθήνα: Εκδόσεις ΕΜΠ, 2004.

[7] J. Craig J, *Introduction to Robotics, Mechanics and Control,* Addison-Wesley Publishing Company, 1989.

[8] J. Heikkila και O. Silven, *A Four-step Camera Calibration Procedure with Implicit Image Correction,* Oulu: Infotech Oulu and Department of Electrical Engineering University, 1997.

[9] Ι. Κοντολάτης, *Ανάδραση θέσης για την Οπτική οδήγηση Εξόμοιωτη Διαστημικού*

Ρομπότ, Αθήμα: Εθνικό Μετσόβιο Πολυτεχνείο, 2008.

[10] Ά. Καλγρεάδης, *Ανάπτυξη αλγορίθμων Επεξεργασίας Εικόνας και Επεκταμένου Φίλρου Kalman (EKF) για εντοπισμό Θέσης Ρομπότ Διαστημικού Εξομοιωτή,* Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο, 2013.

[11] Κ. Μαχαιράς, *Σχεδιασμός και υλοποίηση Ηλεκτρικού/ Ηλεκτρονικού Υποσυστήματος, και προγραμματισμός Πραγματικού Χρόνου Ρομπότ Διαστημικού Εξομοιωτή,* Αθήνα: Εθνικό Μετσόβειο Πολυτεχνείο, 2013.

[12] MatrixVision, «mvBlueFOX Technical Documentation,» Matrix Vision, [Ηλεκτρονικό]. Available: http://www.matrix-vision.com/manuals/mvBlueFOX/.

[13] Microsoft, «C# Programming Guide,» [Ηλεκτρονικό]. Available: https://msdn.microsoft.com/en-us/library/67ef8sbd.aspx.

[14] J. Y. Bouguet, «Camera Calibration Toolbox for Matlab,» [Ηλεκτρονικό]. Available: http://www.vision.caltech.edu/bouguetj/calib_doc/.

[15] Mathworks, «Simulink Real-Time,» [Ηλεκτρονικό]. Available: http://www.mathworks.com/products/simulink-real-time/features.html#real_time_simulation.

[16] Microsoft, «Microsoft Virtual Academy,» [Ηλεκτρονικό]. Available: http://www.microsoftvirtualacademy.com/training-courses/c-fundamentals-for-absolute-beginners.

[17] T. A. Clarket και J. G. Fryer, *The Development Of Camera Calibration Methods And Models,* London, UK: Department of Electrical, Electronic and Information Engineering, City University.

[18] Ι. Παρασκευάς, *Ανάπτυξη Συστήματος Οπτικών αισθητήρων για εντοπισμό θέσης Ρομπότ σε επίπεδη Κίνηση,* Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο, 2008.

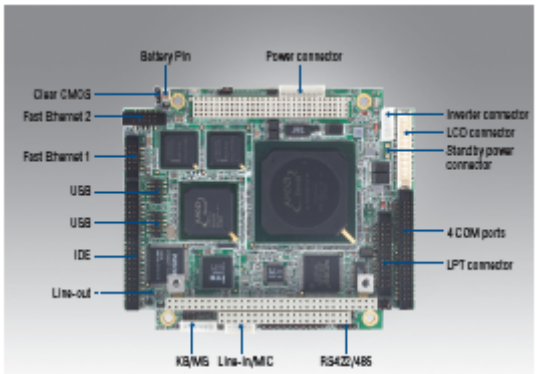
[19] Θ. Φλέσσα, *Έλεγχος και Προγραμματισμός Ρομπότ Επιπέδου Διαστημικού Εξομοιωτή,* Αθήνα: Εθνικό Μετσόβιο Πολυτεχνείο, 2009.

[20] P. Sturm και J. Stephen, *On Plane-Based Camera Calibration: A General Algorithm, Singularities, Applications,* Maybank Computational Vision Group Department of Computer Science The University of Reading Whiteknights.

Παράρτημα Α Εγχειρίδια

PCM-4153

AMD LX800 PC/104-Plus SBC, Onboard Memory/ Flash, VGA, TTL, Extended Temp. -40 ~ 85° C



Features

- AMD low power LX800 500 MHz processor
- Supports extended temperature range -40 ~ 85° C
- PC/104-plus expansion
- Onboard 1 GB Flash and onboard DDR 333MHz 512 MB memory
- Supports SUSIAccess and Embedded Software APIs

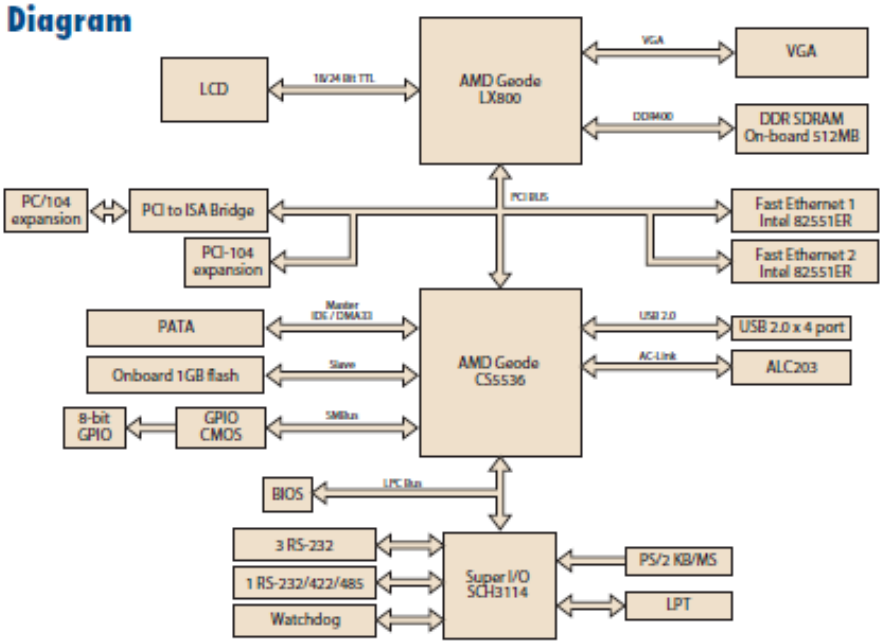
Software APIs: SMBus, H/W Monitor, Brightness, Watchdog, GPIO, Backlight On/Off

Utilities: BIOS flash, Monitoring

Specifications

Processor System	CPU	AMD Geode™ LX800, 500 MHz
	Frequency	500 MHz
	L2 Cache	128 KB
	System Chipset	AMD Geode LX800, with AMD CS5536
	BIOS	Award 4-Mbit
Memory	Technology	DDR 333 MHz
	Max. Capacity	512 MB
	Onboard memory	512 MB
Display	Chipset	AMD Geode LX800
	VRAM	Optimized Shared Memory Architecture up to 64 MB system memory
	VGA	Supports up to 1920 x 1440 x 32 bpp at 85 Hz Supports up to 1600 x 1200 x 32 bpp at 100 Hz
	TTL LCD	Supports up to 1600 x 1200 x 32 bpp at 60 Hz for 24-bit single channel TFT
	Dual Display	VGA+TTL
Ethernet	Speed	10/100 Mbps
	Controller	Fast Ethernet1: Intel 82551ER Fast Ethernet2: Intel 82551ER
	Connector	Box Header
Audio	Chipset	Realtek ALC203
Watchdog Timer		Output System Reset Programmable counter from 1 ~ 255 minutes/ seconds
Storage	PATA	1 Channel
	Onboard Flash	1 GB (up to 4 GB)
	Floppy	1, 82077AA compatible (transfer from LPT, available by custom BIOS)
Internal I/O	USB	4 x USB 2.0
	Serial	3 RS-232 from COM1/3/4, 1 RS-232/422/485 from COM2 (ESD protection for RS-232: Air gap ±15kV, Contact ±8kV)
	Parallel (LPT)	1, IEEE 1284, EPP, and ECP compatible (FDD mode supported)
	SMBus	1
	Keyboard/Mouse	1
	GPIO	8-bit general purpose input/output
Expansion	PC/104-Plus Slot	1
Power	Power Type	AT
	Power Supply Voltage	5V ± 5% only to boot up (12 V is optional for LCD inverter and add on card)
	Power Consumption (Typical)	1.35 A @ +5 V, 0.1 A @ +12 V (7.95 Watt)
	Power Consumption (Max. test in HCT)	1.51 A @ +5 V, 0.1 A @ +12 V (8.75 Watt)
	Power Management	ACPI/APM1.2
Environment	Operational	0 ~ 60° C (32 ~ 140° F) (Operational humidity: 40° C @ 85% RH non-condensing)
	Non-Operational	-40° C ~ 85° C and 60° C @ 95% RH non-condensing
Physical Characteristics	Dimensions (L x W)	96 x 115 mm (3.8" x 4.5")
	Weight	0.574 kg (1.26 lb) (with heat-sink)
	Height	Top Side: 11.45 mm, Bottom Side: 10.6 mm

Block Diagram



Ordering Information

Part No.	CPU	Onboard Memory	VGA	TTL	Fast Ethernet	USB2.0	RS-232	RS-232/ 422/485	Onboard Flash	Audio	Expansion	Thermal Solution	Operating Temp.
PCM-4153F-L0A2E	AMD LX800	512 MB	Yes	18/24 bit	2	4	3	1	1 GB	Yes	PC/104+	Passive	0 – 60° C
PCM-4153FZ-L0A2E	AMD LX800	512 MB	Yes	18/24 bit	2	4	3	1	1 GB	Yes	PC/104+	Passive	-20 – 80° C
PCM-4153FZ2-L0A2E	AMD LX800	512 MB	Yes	18/24 bit	2	4	3	1	1 GB	Yes	PC/104+	Passive	-40 – 85° C

Note: Passive = fanless; Active = with fan

Packing List

Part No.	Description	Quantity
	PCM-4153 SBC	
	Startup Manual	
	Utility CD	
1700000898	VGA cable D-SUB 15P(F)/12P-1.25 mm 15 cm	1
1700000918	Audio cable 10 cm	
1700003491	AT power cable 1 x 8P-2.0/B4P-5.08 x 2 15 cm	1
1700060202	Cable 6P-6P-6P PS/2 KB & Mouse 20 cm	1
1700260250	LPT port cable 25P to 26P 2.0 mm 25 cm	1
1701100202	LAN flat cable IDC10P 2.0 mm/RJ-45 20 cm	1
1701400181	COM 4 ports flat cable 18 cm IDC40P 2.0 mm	1
1701440350	IDE cable 44P/44P/44P 35 cm	1
1703040157	RS-422/485 W/D-SUB COM 4P 15 cm	1
1703060053	PS2 cable 6P (MINI-DIN)-6P (Wafer 2.0 mm) 6 cm	1
1703100121	USB 2-port cable 10P 12 cm IDC 2.0 mm	1
9660104000	PC/104 screw and copper post package	1
1960005764	Heatsink for PCM-4153 (80.44 x 77.97 x 7.82 mm)	1

Optional Accessories

Part No.	Dexcription
1653130421	PCI-104 connector 120-pin (Long pin)
1653132228	PC/104 connector 64-pin (Long pin)
1653120228	PC/104 connector 40-pin (Long pin)
1700001531	LPT to FDD cable

Embedded OS/API

Embedded OS/API	Part No.	Description
WinCE	2070000729	Image GX3 CE 5.0 Pro Plus Eng
	2070001612	CE 6.0 Pro GX3 4 Com V1.0 ENG
Win XPE	2070001576	XPE FP2007 GX3 (LX800) V3.0 ENG
	2070003216	XPE FP2007 GX3 Group V3.1 ENG (717.22 MB)
	2070003557	XPE FP2007 GX3 Group V3.0 CHS (641.41 MB)
QNX		V6.3.2/ 6.4.1
Software API	205E000019	SUSI 3.0 SW API for ESBC B: 20091116 XP

ISA Protocol interpreter module FAQ's

ITE 聯陽半導體 ITE Tech. Inc.

IT8888 Q&A v1.1

Aug. 2, 2007

Revision History

Revision	Change List	Note
1.0	Initial version	2005/04/26
1.1	1. Modify some wordings. 2. Add item-16 and item-17 3. Add Annex-A (AMD's FAQ for DDMA function on AMD's Geode+IT8888 reference platform).	2007/08/02

1. **Is the SYSCLK of ISA bus synchronous with the PCI Clock ?**
   Ans.: No, it is asynchronous.

2. **Is the Frequency of the ISA's SYSCLK programmable ?**
   Ans.: No, ISA's SYSCLK is fixed at the frequency of 1/4 PCICLK (around 8.25 MHz).

3. **Is the number of the wait state at the ISA Bus programmable ?**
   Ans.: No, it is not programmable. These two pins are usually for escaping or inserting the wait state by ISA slave devices -- NOWS# (No wait states signal) is to shorten ISA cycle and IOCHRDY is to insert wait state.

4. **Does the Bridge support ISA Master ?**
   Ans.: Yes, IT8888 supports 16-bit ISA Master.

5. **Is the IOCHRDY signal edge or level sensitive ?**
   Ans.: It is level sensitive.

6. **Are the DRQs edge or level sensitive ?**
   Ans.: The DMA request input signals of IT8888 (DRQ 7~5, 3~0) are level sensitive.

7. **At which timing does the bridge sense the IOCHRDY signal, and can you provide the Timing Diagram ?**
   Ans.: Yes, there are timing diagrams provided in IT8888's datasheet which are illustrated in Fig 7-11, 7-14 and 7-20.

8. **Are the IRQ pins with schmit trigger ?**
   Ans.:Yes, IRQ pins of IT8888 are with schmit trigger type.

9. **Have you done a test with the old IBM AT 8MHz ISA Card ?**
   Ans.: No, ITE didn't test the ISA bridge with IBM AT ISA card.

10. **Do you have a list indicating which ISA Cards passed the test and which failed the test ?**
    Ans.: Please refer to table -1.

Table-1: Test list for IT8888

ISA Adapter	Manufacture	Pass/Fail	Note
Super I/O Card	UMC UM82C865F	Pass	
	IT8680F-A	Pass	
Network Card	Realtek RTL8019AS	Pass	
Sound Card	ESS1868	Pass	
	Creative SB16	Pass	
	Creative AW32	Pass	
SCSI Card	AHA-1542 Mastercard	Pass	
	AHA-1520B	Pass	
Video Card	Trident TVGA 8800CS	Pass	
	UMC UM85C408F	Pass	
	Tseng Labs ET4000AX	Pass	

11. **What specification is your design based on ?**
    Ans.: ITE designed the ISA bridge based on Intel's Moon ISA bridge.

12. **Are there any demo boards or motherboards you recommend as a reference for customers ?**
    Ans.: It's recommended for customers to refer to the motherboards of ASUS/P3C200, Gigabyte/6WXM7,etc, where IT8888 is embedded.

13. **Is there any reference schematics ?**
    Ans.:Yes, IT8888's reference schematics is available. ITE can provide it to customers.

14. **Has ITE ever applied IT8888 to any south bridge ?**
    Ans.: ITE has ever tested IT8888 with Intel's 810, BX+PIIX4, ICH, ICH2 and ICH4.
    In addition, ITE tested our ISA bridge with VT8233 chipset and found some PPDMA problems in the ISA master cycle. It's also found that VT8233's subtractive decode function couldn't be disabled which conflicted with IT8888's subtractive decode function. ITE suggests customers to check this issue with VIA before applying IT8888 to VIA's chipset.

15. How many ISA slots does IT8888 support and what's the ISA's driving current ?
Ans.: IT8888 supports four ISA slots and the driving current for its ISA I/F is 8 mA.

16. What's the power current of Icc5 and Icc3 on IT8888 ?
Ans.: IT8888's power current (typical) is -- Icc5=28 mA, Icc3=2 mA while doing DMA transfer with 16-bit ISA device (AHA-1542CF and AHA-1520B) at 25 degree C when IT8888's VCC5=5V and VCC3=3.3V.

17. Is it possible to run ISA device with DMA support while applying IT8888 to AMD's Geode SOC ?
Ans.: Yes, it's possible. ITE provides some technical supports to AMD and then AMD implements the software porting (VSA DDMA module in BIOS) for 8/16-bit ISA DMA support on Geode with IT8888's DDMA function by himself.
AMD provides ITE a FAQ for DDMA function on AMD's Geode + IT8888 reference design. ITE includes it in this document as Annex-A and customers can refer more detailed information by it.
Thus, it's necessary to obtain the VSA DDMA module and related software technical support from AMD if customers need ISA device's DMA support on Geode + IT8888 platform.

Annex-A
FAQ for Geode + IT8888 reference platform

One solution to support ISA DMA function on Geode is to use the ITE8888 device. This device supports 8/16-bit I/O and memory cycles. It also supports 8/16-bit DMA if the available VSA DDMA module is built into the BIOS. A downside to using this solution is that the legacy I/O space (0-3FFh) is always subtractively decoded by the CS5535/CS5536. This requires that one or more of the six available I/O positive decode windows be configured in the IT8888 bridge. The default positive decode windows that are recommended:
100h-1DFh (implemented as 3 windows, 100h-17Fh, 180h-1BFh, 1C0h-1DFh).
200h-27Fh.
300h-35Fh (implemented as 2 windows, 300h-33Fh, 340h-35Fh).

In addition, AMD recommends the following register settings when implementing a IT8888 based bridge solution.
1) Configure the IT8888 I/O and memory positive decode windows to have them positively decode their spaces at "medium" speed.
2) Configure the IT8888 to subtractively decode the non-legacy I/O space (i.e., 400h-FFFFh) by setting bit 0 of PCI config space index 50h.
3) Disable the non-legacy subtractive decode feature in the CS5535/CS5536 by setting SDOFF to enabled.
4) Enable subtractive decoding in the IT8888 by setting the NOGO/CLKRUN# mux to CLKRUN# by setting bit 4 of PCI config space index 56h.

Note: The contents of Annex-A are provided by AMD.

PC104-DIO48

User's Manual

MEASUREMENT COMPUTING™

Revision 4
April, 2001

1 SOFTWARE INSTALLATION

The board has switches and jumpers to set before installing the board in your computer. By far the simplest way to configure your board is to use the *Insta*Cal™ program provided as part of your software package. *Insta*Cal™ will show you all available options, how to configure the various switches and jumpers (as applicable) to match your application requirements, and will create a configuration file that your application software (and the Universal Library) will refer to so the software you use will automatically know the exact configuration of the board.

Please refer to the *Extended Software Installation Manual* regarding the installation and operation of *Insta*Cal™. The following hard copy information is provided as a matter of completeness, and will allow you to set the hardware configuration of the board if you do not have immediate access to *Insta*Cal™ and/or your computer.

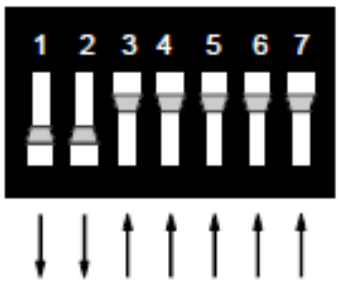
2 HARDWARE INSTALLATION

2.1 BASE ADDRESS

The PC104-DIO48 employs the PC bus for power, communications and data transfer. As such it draws power from the PC, monitors the address lines and control signals and responds to it's I/O address, and it receives and places data on the eight data lines.

The base address is the starting location that software writes to and reads from.

The base address switch is the means for setting the base address. Each switch position corresponds to one of the PC bus address lines. The down position activates that address bit.



SWITCH	HEX
1	200
2	100
3	80
4	40
5	20
6	10
7	08

BASE ADDRESS SWITCHES – 300h Shown.

Figure 2-1. Base Address Switches

The actual address is constructed by calculating the HEX or decimal number Base Address Select Switches which corresponds to the base address bits the PC104-DIO48 will respond to. For example, in Figure 2-1, switches 1 and 2 down, all others are up. Switch 1 = 200 hex (512 decimal) and switch 2 = 100 hex (256 decimal). When added together they equal 300 hex (768 decimal).

Certain address are reserved for use by the PC (Table 2-1). Others are free and can be used by the PC104-DIO48 and other expansion boards. We recommend that BASE = 300 hex (768 decimal) be tried first. See Figure 2-2 for the orientation of the switch block.

Table 2-1. PC I/O Addresses

HEX RANGE	FUNCTION	HEX RANGE	FUNCTION
000-00F	8237 DMA #1	2C0-2CF	EGA
020-021	8259 PIC #1	2D0-2DF	EGA
040-043	8253 TIMER	2E0-2E7	GPIB (AT)
060-063	8255 PPI (XT)	2E8-2EF	SERIAL PORT
060-064	8742 CONTROLLER (AT)	2F8-2FF	SERIAL PORT
070-071	CMOS RAM & NMI MASK (AT)	300-30F	PROTOTYPE CARD
080-08F	DMA PAGE REGISTERS	310-31F	PROTOTYPE CARD
0A0-0A1	8259 PIC #2 (AT)	320-32F	HARD DISK (XT)
0A0-0AF	NMI MASK (XT)	378-37F	PARALLEL PRINTER
0C0-0DF	8237 #2 (AT)	380-38F	SDLC
0F0-0FF	80287 NUMERIC CO-P (AT)	3A0-3AF	SDLC
1F0-1FF	HARD DISK (AT)	3B0-3BB	MDA
200-20F	GAME CONTROL	3BC-3BF	PARALLEL PRINTER
210-21F	EXPANSION UNIT (XT)	3C0-3CF	EGA
238-23B	BUS MOUSE	3D0-3DF	CGA
23C-23F	ALT BUS MOUSE	3E8-3EF	SERIAL PORT
270-27F	PARALLEL PRINTER	3F0-3F7	FLOPPY DISK
2B0-2BF	EGA	3F8-3FF	SERIAL PORT

The PC104-DIO48 BASE switch may be set for address in the range of 000 to 3F8 so it should not be hard to find a free address area for your PC104-DIO48. Once again, if you are not using IBM prototyping cards or some other board which occupies these addresses, then 300-31Fh are free to use. Addresses not specifically listed, such as 390-39Fh, are free.

2.2 INSTALLING THE BOARD

1. Turn the power off.

2. Push the board firmly down into the expansion bus connector. If it is not seated fully it may fail to work and could short circuit the PC bus power onto a PC bus signal. This could damage the motherboard in your PC as well as the PC104-DIO48.

2.3 CABLING TO THE DIO48 CONNECTOR

The connector is a standard 50-pin, male, header connector. A mating female connector (C50FF-##) may be purchased from Measurement Computing.

2.4 SIGNAL CONNECTION

All the digital outputs/inputs on the connector are CMOS TTL. TTL is an electronics industry term, short for Transistor Transistor Logic, which describes a standard for digital signals which are either 0V or 5V (nominal).

Under normal operating conditions, the voltages on the 82C55 pins range from near 0 volts for the low state to near 5.0 volts for the high state. Before connecting the PC104-DIO48 to external devices, review the electrical specification in this manual to ensure that the boards input voltage specifications are not exceeded. In addition to voltage and load matching, digital signal sources often need to be de-bounced. More details on digital interfacing are in the section on Interface Electronics in this manual.

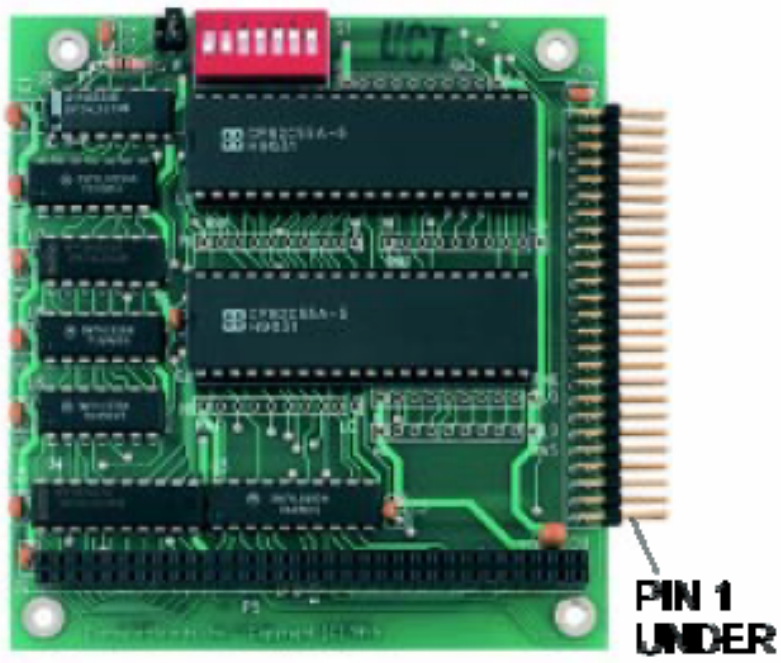


Figure 2-2. PC104-DIO48 Board Layout and Pin 1 Location

IMPORTANT NOTE: The PC104-DIO48 uses two 82C55 digital chips for digital I/O. The 82C55 digital I/O chip initializes all ports as inputs on power up and reset. A TTL input is a high impedance input. If you connect another TTL input device to the 82C55 it will probably be turned ON every time the 82C55 is reset, or, it might be turned OFF instead. Remember, an 82C55 which is reset is in INPUT mode.

To safeguard against unwanted signal levels, all devices being controlled by an 82C55 should be tied low (or high, as required) by a 2.2K ohm resistor.

You will find positions for pull up and pull down resistor packs on your PC104-DIO48 board. To implement these, please turn to the application note on pull up/down resistors.

2.5 UNCONNECTED INPUTS

Keep in mind that unconnected inputs *float*. If you are using a PC104-DIO48 board for input, and have unconnected inputs, ignore the data from those lines.

In other words, if you connect bit A0 and not bit A1, do not be surprised if A1 stays low, stays high or tracks A0. It is unconnected and so, is not specified. The 82C55 is not malfunctioning. In the absence of a pull-up/down resistor, any input which is unconnected is unspecified.

You do not have to tie input lines, and unconnected lines will not affect the performance of connected lines. Just make sure that you mask out any unconnected bits in software.

2.6 CONNECTOR DIAGRAM

The connector accepts female 50-pin header connectors, such as those on the C50FF-2, a 2-foot cable with connectors.

If frequent changes to signal connections or signal conditioning is required, please refer to the information on the CIO-TERM100, CIO-SPADE50 and CIO-MINI50 screw terminal boards.

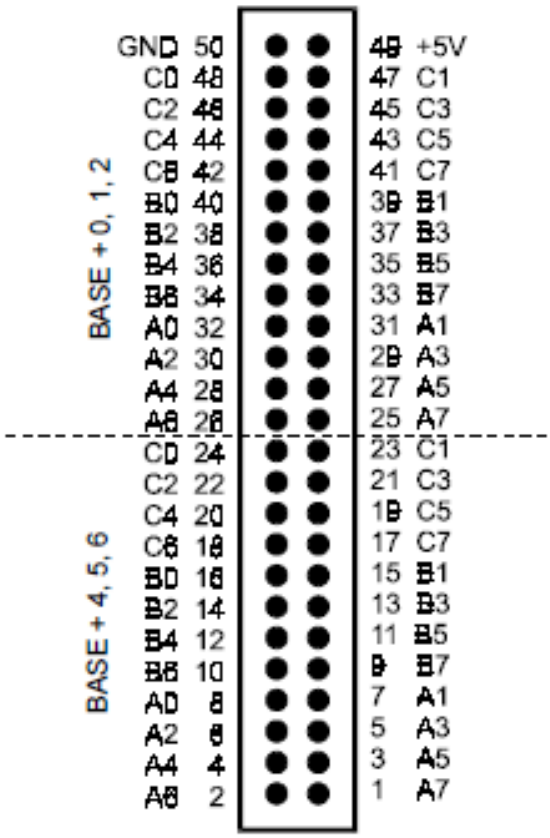


Figure 2-2 I/O Connector

Παράρτημα Β Κώδικες

Κώδικας τεχνητής όρασης

```
using mvIMPACT_NET
using mvIMPACT_NET.acquire;
using System;
using System.IO;
using System.Globalization;
using CameraServo;
using System.Diagnostics;
using System.Collections.Generic;
using System.Windows.Forms;


namespace IliasPatsiaourasImageProccessing
{
  public class ImageProccessing
  {

    public class Robot
    {
      public string name;
      public double sign;
      public int ID;
      public LED bigLed;
      public LED smallLed;
      public bool Exist;
      public double calculatedSign;
      public double x, y;//robot position (mm)
      public double x_old, y_old; //robot z^-1 position (mm)
      public double u, v; //robot speed (mm/s)
      public double theta;
      public double th_old;
      public double omega;

      //robot constructor
      public Robot(string Name, double Sign, int Port)
      {
        this.name = Name;
        this.ID = Port;
        this.sign = Sign;
        bigLed = new LED();
        smallLed = new LED();
        Exist = false;
      }

      public override string ToString()
      {
        return (name + "   Sign: " + sign.ToString() + "   ID: " + ID.ToString());
      }
    }

    public class LED
    {
      public int xSum, ySum; //variables used to store the sum of x,y for center of weight calc
      public int px_count;
      public int max_width;
      public int height;
      public double x, y;  //image frame Pixel coords
```

```
    public double xw, yw; //real world coords (mm)
    public bool assigned;
    public bool isUptodate;

    //instance constructor
    public LED ()
    {
        xSum = 0;
        ySum = 0;
        px_count = 0;
        max_width = 0;
        height = 0;
        xw = -1000.0;//something not realistic
        yw = -1000.0;
        x = -1000.0;
        y = -1000.0;
        assigned = false; //indicates if this led is led of a robot
        isUptodate = false;
    }

    //copy constructor
    public LED (LED led)
    {
        xSum = led.xSum;
        ySum = led.ySum;
        px_count = led.px_count;
        max_width = led.max_width;
        height = led.height;
        x = led.x;
        y = led.y;
        xw = led.xw;
        yw = led.yw;
        isUptodate = led.isUptodate;
        assigned = led.assigned;
    }

    //Assign Reference of led this led and marked as assigned
    public LED assign()
    {
        this.assigned = true;
        return this;
    }
}

// Flow control values
public bool fullScanRequired;
public int ledNum;
public int rbtNum;
public List<LED> ledFound;
public Robot[] rbt;

// Robot finding Values
public double tolerance;

// Image Proccessing Values
public byte threshold;
public byte saw_px;
```

```
public int r_dzone;
public int l_dzone;
public int d_dzone;
public int u_dzone;

//files
double[,] xCoord = new double[1600, 1200];
double[,] yCoord = new double[1600, 1200];

//constructor
public ImageProccessing(int ledNumPerRobot,Robot[] robots)//params double[] robotSignslist)
{
   //Init Flow control values
   fullScanRequired  = true; // Start with full scan
   ledNum    = ledNumPerRobot*robots.Length;//need check !!!!!!!!!!!!!!!!!!!!!!!!

   ledFound  = new List<LED>();
   ledFound.Capacity = ledNum + 50;

   rbtNum = robots.Length;

   // default values that is editable
   // Robot finding Values
   tolerance = 10.0;//mm
   //Init Image Proccessing Values
   threshold = (byte)210;
   saw_px    = (byte)1;
   r_dzone = 150;//150
   l_dzone = 150;//150
   d_dzone = 5;//5
   u_dzone = 15;//15

   //initialize robots
   rbt = new Robot[rbtNum];
   for (int s = 0; s < rbtNum; s++)
   {
      for (int t = s + 1; t < rbtNum; t++)
      {
         //check that every sign is different enough
         if (Math.Abs(robots[s].sign - robots[t].sign) <= tolerance)
         {
            MessageBox.Show(String.Format("\t\t\tWARNING:\n"+
               "{0} and {1} has ambiguous signs,\n"+
            "differnce must be at least: {2}mm (tolerance value)"
               , robots[s].name, robots[t].name, tolerance));
         }
      }
      rbt[s] = new Robot(robots[s].name, robots[s].sign, robots[s].ID);
   }


   //load lookup table files
   loadlookupFile("fileX.txt", "fileY.txt");
}


public void loadlookupFile(string filenameX, string filenameY) // run in the constructor of the Class
```

```
{

    int counterX = 0;
    int counterY = 0;
    string line;

    try
    {
        message(String.Format("loading calibration files."));

        System.IO.StreamReader fileX = new System.IO.StreamReader(".\\" + filenameX);
        while ((line = fileX.ReadLine()) != null)
        {
            string[] values = line.Split('\t');
            for (int x = 0; x < 1600; x++)
            {
                xCoord[x, counterX] = double.Parse(values[x], CultureInfo.InvariantCulture);
            }
            counterX++;
        }

        fileX.Close();

        message(".");

        System.IO.StreamReader fileY = new System.IO.StreamReader(".\\" + filenameY);
        while ((line = fileY.ReadLine()) != null)
        {
            string[] values = line.Split('\t');
            for (int x = 0; x < 1600; x++)
            {
                yCoord[x, counterY] = double.Parse(values[x], CultureInfo.InvariantCulture);
            }
            counterY++;
        }

        fileY.Close();

        message(".");

        if (counterY != 1200)
            message("fileY corrupted");
        if (counterX != 1200)
            message("fileX corrupted");
    }
    catch (Exception ex)
    {
        message((String.Format("exeption: {0}", ex)));
    }
    //debug
    //message(String.Format("DEBUG: x[1598, 1198]: {0}" + "y[1598, 1198]: {1}", xCoord[1598, 1198], yCoord[1598, 1198]));
}


public void scanForLeds(Image image)//update the ledFound List
{
    ledFound.Clear();//reset list for clean data after every run of this
```

```
        int padding = 0;
        int icenter = 0;
        int left = 0;
        int right = 0;
        int width = 0;
        double center = 0.0;
        int n = 0, rej = 0;

        //lock buffer and grant Access to buffer
        image.prepareReadWriteAccess();

        for (int y = u_dzone + 1; y < image.height -1 - d_dzone; y++)
        {
          for (int x = l_dzone + 1; x < image.width -1 - r_dzone; x++)
          {
            //led finding
            if (image.getPixel(x, y) >= threshold)
            {
              //reset values for the new led
              padding = 0;
              center = 0.0;
              icenter = 0;
              ledFound.Insert(n, new LED()); //add a fresh led to the list/ or if the previous rejected re-intialize it

              //px of led finding
              while (image.getPixel(x + icenter, y + padding) >= threshold)
              {
                //image.drawPoint(new Point(x + icenter, y + padding), saw_px);
                image.setPixel(x + icenter, y + padding, saw_px);
                ledFound[n].ySum += y + padding;                    //adding the coords for center of led calc
                ledFound[n].xSum += x + icenter;

                //right search
                right = 0;
                while (image.getPixel(x + icenter + right + 1, y + padding) >= threshold)
                {
                  //image.drawPoint(new Point(x + icenter + right + 1, y + padding), saw_px);
                  image.setPixel(x + icenter + right + 1, y + padding, saw_px);
                  ledFound[n].ySum += y + padding;                  //adding the coords for center of led calc
                  ledFound[n].xSum += x + icenter + right + 1;
                  right++;
                  if (x + icenter + right + 1 > image.width -1 - r_dzone)    //compares the next index value to prevent index
ovflow
                    break;
                }

                //left search
                left = 0;
                while (image.getPixel(x + icenter - left - 1, y + padding) >= threshold)
                {
                  //image.drawPoint(new Point(x + icenter -left -1, y + padding), saw_px);
                  image.setPixel(x + icenter - left - 1, y + padding, saw_px);
                  ledFound[n].ySum += y + padding;                  //adding the coords for center of led calc
                  ledFound[n].xSum += x + icenter - left - 1;
                  left++;
                  if (x + icenter - left - 1 < l_dzone)          //compares the next index value to prevent index ovflow
                    break;
```

```
            }

            width = left + right + 1;
            ledFound[n].px_count += width;

            if (width > ledFound[n].max_width) // statistics
               ledFound[n].max_width = width;

            center += (right - left) / 2.0;
            icenter = (int)center;//?? it must work similar to Math.Truncate()

            padding++;
            if (y + padding > image.height -1 - d_dzone) break;

            ledFound[n].height = padding;
         }//end px of led finding

         //center of led calc: subtracted the coords sum with count
         ledFound[n].y = (double)ledFound[n].ySum / (double)ledFound[n].px_count;
         ledFound[n].x = (double)ledFound[n].xSum / (double)ledFound[n].px_count;
         image.setPixel((int)ledFound[n].x, (int)ledFound[n].y, threshold - 1);// drawpoint similar method

         //led check
         if ((ledFound[n].px_count > 4) && (ledFound[n].px_count < 1000))
         {
            ledFound[n].isUptodate = true;
            n++; //this will create a new LED in next round or if its the last one it will convert zero-based index to led count
         }
         else
         {
            ledFound.RemoveAt(n); //ensures that ledFound.length corresponds only in real led and no rejected pixels
            rej++;//rejected counter
         }


      }//end led finding

      /*
      if (ledFound.Count >= ledNum)//this check make full scan a bit faster if it finds all led just end the search
      { //try it!! when robot is in upper left corner is fast when in down right has to scan all image
         image.releaseAccess();
         return;
      }*/

    }//end x loop
  }//end y loop

  //if didnt found all leds exit from here

  //unlock buffer and stop Access to buffer
  image.releaseAccess();
  //ledFound = n;
}


public void updateLed(Image image, LED led, int winWidth, int winHeight) //returns number of leds
{
```

```
int padding = 0;
int icenter = 0;
int left = 0;
int right = 0;
int width = 0;
double center = 0.0;
int n=0, rej = 0;

int bufferLength = 5;

LED[] ledh = new LED[bufferLength];
for (int i = 0; i < bufferLength; i++)
{
   ledh[i] = new LED();
}

//lock buffer and grant Access to buffer
image.prepareReadWriteAccess();

//???????the following limits is not checked if they conforms with the dead zones so
//???????? near deadzones there is an oscilation between full scan and history based scan
//???????? due to the break;s in the while of this method
for (int y = (int)led.y - winHeight / 2; y <= ((int)led.y + winHeight / 2); y++)
{
   for (int x = (int)led.x - winWidth / 2; x <= ((int)led.x + winWidth / 2); x++)
   {
      //led finding
      if (image.getPixel(x, y) >= threshold)
      {
         //reset values for the new led
         padding = 0;
         center = 0.0;
         icenter = 0;

         //px of led finding
         while (image.getPixel(x + icenter, y + padding) >= threshold)
         {
            //image.drawPoint(new Point(x + icenter, y + padding), saw_px);
            Debug.WriteLine(String.Format("n: {0}", n));
            image.setPixel(x + icenter, y + padding, saw_px);
            ledh[n].ySum += y + padding;                    //adding the coords for center of led calc
            ledh[n].xSum += x + icenter;

            //right search
            right = 0;
            while (image.getPixel(x + icenter + right + 1, y + padding) >= threshold)
            {
               //image.drawPoint(new Point(x + icenter + right + 1, y + padding), saw_px);
               image.setPixel(x + icenter + right + 1, y + padding, saw_px);
               ledh[n].ySum += y + padding;                 //adding the coords for center of led calc
               ledh[n].xSum += x + icenter + right + 1;
               right++;
               if (x + icenter + right + 1 > image.width -1 - r_dzone)     //compares the next index value to prevent index
ovflow
                  break;
            }
```

```
                //left search
                left = 0;
                while (image.getPixel(x + icenter - left - 1, y + padding) >= threshold)
                {
                  //image.drawPoint(new Point(x + icenter -left -1, y + padding), saw_px);
                  image.setPixel(x + icenter - left - 1, y + padding, saw_px);
                  ledh[n].ySum += y + padding;                    //adding the coords for center of led calc
                  ledh[n].xSum += x + icenter - left - 1;
                  left++;
                  if (x + icenter - left - 1 < l_dzone)           //compares the next index value to prevent index ovflow
                    break;
                }

                width = left + right + 1;
                ledh[n].px_count += width;

                if (width > ledh[n].max_width)
                  ledh[n].max_width = width;

                center += (right - left) / 2.0;
                icenter = (int)center;//?? it must work similar to Math.Truncate()

                padding++;
                if (y + padding > image.width -1 - d_dzone) break;

                ledh[n].height = padding;
              }//end px of led finding


              ledh[n].y = (double)ledh[n].ySum / (double)ledh[n].px_count;          //center of led calc: subtracted the coords sum with count
              ledh[n].x = (double)ledh[n].xSum / (double)ledh[n].px_count;//?????? maybe needs to be double for accurate center or doesnt work
              image.setPixel((int)ledh[n].x, (int)ledh[n].y, threshold - 1);
              //image.drawPoint(new Point(led[n].x, led[n].y), threshold - 1);

              //led check

              if ((ledh[n].px_count > 4) && (ledh[n].px_count < 3000))
              {
                n++;
                if (n >= bufferLength) return;//preventing from throw exception: ledh out of index
              }
              else
                rej++;//rejected counter


            }//end led finding

            //if (n == 1)          //IDEA: check if the led is found to return faster
              //return ledh[0];

          }//end x loop
        }//end y loop

        image.releaseAccess();         //unlock buffer and stop Access to buffer
```

```
    if (n == 1)
    {
        led.x = ledh[0].x;              //is alway the first the real one
        led.y = ledh[0].y;
        led.xSum = ledh[0].xSum;
        led.ySum = ledh[0].ySum;
        led.max_width = ledh[0].max_width;
        led.px_count = ledh[0].px_count;
        led.height = ledh[0].height;
        led.isUptodate = true;
        //led.assigned
    }
    else
    {
        led.isUptodate = false;
        //if a led dissapear and is not belong to a robot is ignored
        //led = new LED(); // reset led
        //ledFound.Remove(led);
        //robot update check if led isUptodate and raise fullScanRequired flag
    }
}


public void calcLedsRealCoords()//reference pass
{
    foreach (LED l in ledFound)
    {
        if (l.isUptodate) //prevent this method to waist time in out-dated leds
        {
            //BILINEAR INTERPOLATION

            //if l.x or l.y is integer Bilinear interpolation will fail (zero devide)
            if (  ((l.x % 1)==0)  ||  ((l.y % 1)==0)  )
            {
                l.xw = xCoord[(int)Math.Round(l.x), (int)Math.Round(l.y)];
                l.yw = yCoord[(int)Math.Round(l.x), (int)Math.Round(l.y)];
            }
            else
            {
                // bilinear interpolation
                int x0 = (int)Math.Floor(l.x);
                int x1 = (int)Math.Ceiling(l.x);
                int y0 = (int)Math.Floor(l.y);
                int y1 = (int)Math.Ceiling(l.y);

                double a0 = xCoord[x0, y1];
                double a1 = xCoord[x1, y1];
                double a2 = xCoord[x0, y0];
                double a3 = xCoord[x1, y0];

                double b0 = yCoord[x0, y1];
                double b1 = yCoord[x1, y1];
                double b2 = yCoord[x0, y0];
                double b3 = yCoord[x1, y0];

                double N = (x1 - x0) * (y1 - y0);
                double Na = (x1 - l.x) * (l.y - y0) / N;
```

```
                double Nb = (l.x - x0) * (l.y - y0) / N;
                double Nc = (xl - l.x) * (yl - l.y) / N;
                double Nd = (l.x - x0) * (yl - l.y) / N;

                l.xw = a0 * Na + a1 * Nb + a2 * Nc + a3 * Nd;
                l.yw = b0 * Na + b1 * Nb + b2 * Nc + b3 * Nd;
              }

              message(String.Format("real Coords: x:{0} , y:{1}", l.xw, l.yw));
            }
          }
        }

        public void ScanForRobots() //this assignes the led to robots searching all found LEDs
        {
          //numLeds number of leds
          double[,] dist = new double[ledFound.Count, ledFound.Count];

          //creates an lower trigonic array with all available distances
          for (int j = 0; j < ledFound.Count; j++)
          {
            for (int i = j + 1; i < ledFound.Count; i++)
            {
              dist[i, j] = Math.Sqrt(Math.Pow(ledFound[i].xw - ledFound[j].xw, 2) + Math.Pow(ledFound[i].yw - ledFound[j].yw, 2));

              for (int k = 0; k < rbtNum; k++)
              {
                //Robot identification
                if (Math.Abs(dist[i, j] - rbt[k].sign) <= tolerance)
                {
                  if (ledFound[i].px_count > ledFound[j].px_count)
                  {
                    rbt[k].bigLed = ledFound[i].assign(); //pass ledfound by reference to bigLed and change led property
"assigned" to true
                    rbt[k].smallLed = ledFound[j].assign();
                  }
                  else
                  {
                    rbt[k].bigLed = ledFound[j].assign();
                    rbt[k].smallLed = ledFound[i].assign();
                  }
                  rbt[k].Exist = true;
                  rbt[k].calculatedSign = dist[i, j];
                  message(String.Format("Scan found robot: {0}, bigled.assigned:{1}, smallLed.assigned:{2}", k,
rbt[k].bigLed.assigned, rbt[k].smallLed.assigned));
                }
              }
            }
          }
        }

        public void updateRobot(Robot rob, double refreshRate)
        {
          if (rob.bigLed.isUptodate && rob.smallLed.isUptodate)
          {
            rob.calculatedSign = Math.Sqrt(Math.Pow(rob.smallLed.xw - rob.bigLed.xw, 2) + Math.Pow(rob.smallLed.yw -
rob.bigLed.yw, 2));
```

```
            if (Math.Abs(rob.calculatedSign - rob.sign) <= tolerance) //checks that is still the correct robot
            {
                rob.Exist = true;

                //position
                rob.x = rob.bigLed.xw;
                rob.y = rob.bigLed.yw;
                rob.theta = Math.Atan2(rob.smallLed.yw - rob.bigLed.yw, rob.smallLed.xw - rob.bigLed.xw);
                message(String.Format("{0} exist and updated with    X:{1}   Y:{2}   Th:{3}   calculated Sign:{4}",rob.name, rob.x,
rob.y, rob.theta, rob.calculatedSign));

                //speed
                if (rob.x_old == null || rob.y_old == null || rob.th_old == null)
                {
                    //first time scenario
                    rob.u = 0;
                    rob.v = 0;
                    rob.omega = 0;
                }
                else
                {
                    rob.u = (Math.Round(rob.x - rob.x_old) * refreshRate);
                    rob.v = (Math.Round(rob.y - rob.y_old) * refreshRate);
                    rob.omega = (Math.Round(rob.theta - rob.th_old, 2) * refreshRate);
                }

                //prepare next step
                rob.x_old = rob.x;
                rob.y_old = rob.y;
                rob.th_old = rob.theta;
            }
            else
            {
                rob.Exist = false;
                //speed consideration????????
                fullScanRequired = true;
                message(String.Format("Robot Sign Verification Failed"));
            }

        }
        else
        {
            rob.Exist = false;
            fullScanRequired = true;
            message(String.Format("one or both Robot led is not up-to-date"));
        }

    }

    public void message(string str)
    {
        Debug.WriteLine(str);
    }

  }
}
```

Κώδικας λογισμικού Γραφικού περιβαλλοντος και ελέγχου κάμερας

```
using System;
using System.Globalization;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using mvIMPACT_NET;
using mvIMPACT_NET.acquire;
using IliasPatsiaouras.ImageProccessing;
using CameraServo.Common;
using CameraServo.TCPServerTest;
using System.Net;
using System.Net.Sockets;
using System.Diagnostics;

namespace CameraServo
{

  public partial class Form1 : Form
  {
    string settingsFilename;

    string LogFilename;
    static System.IO.FileStream logFileStream;

    DeviceManager devMgr;
    int devIdx;
    Device dev;
    Window win;
    tcpThreadedServer mytcpserver;

    //UDP attributes
    IPEndPoint UDPgroupEP;
    static IPEndPoint UDPreceiveEndPoint;
    static UdpClient udpClient;



    public Form1()
    {
      InitializeComponent();

      try
      {
        devMgr = new DeviceManager();
        //fill the drop down menu with all supported camera devices
        if (devMgr.deviceCount() == 0)
        {
          debug.AppendText("No MATRIX VISION device found!\n");
        }
        else
        {
```

```
            for (uint i = 0; i < devMgr.deviceCount(); i++)
            {
               Device tempDev = devMgr.getDevice(i);
               if (tempDev != null)
               {
                  //Console.WriteLine("[{0}]: {1}({2}, family: {3})", i, tempDev.serial.read(), tempDev.product.read(),
tempDev.family.read());
                  comboBoxDevice.Items.Add(tempDev.product.read());
               }
               tempDev.Dispose();
            }
         }

      }
      catch (ImpactException e)
      {
         MessageBox.Show(e.errorString);
      }
   }

   void Form1_Load(object sender, EventArgs e)
   {
      winWidth.Value = Properties.Settings.Default.winWidth;
   }

   void Form1_FormClosing(object sender, System.Windows.Forms.FormClosingEventArgs e)
   {
      Properties.Settings.Default.winWidth = (int)winWidth.Value;
      Properties.Settings.Default.Save();

      //throw new System.NotImplementedException();
      debug.AppendText("Closing Communication\n");
      try { UDPDisconnect(25000); }
      catch { }

      try {mytcpserver.Stop();}
      catch { }

   }

   delegate void AppendLineCallback(string text);

   public void AppendLine(string text)
   {
      // InvokeRequired required compares the thread ID of the
      // calling thread to the thread ID of the creating thread.
      // If these threads are different, it returns true.
      if (this.debug.InvokeRequired)
      {
         AppendLineCallback d = new AppendLineCallback(AppendLine);
         this.Invoke(d, new object[] { text });
      }
      else
      {
         this.debug.AppendText(text);
      }
   }
```

```
        private void buttonAddRbt_Click(object sender, EventArgs e)
        {
            if (textBoxName.Text != "" && textBoxSign.Text != "" && textBoxPort.Text != "")
            {
                string name = textBoxName.Text;
                double sign = double.Parse(textBoxSign.Text);
                int port = int.Parse(textBoxPort.Text);
                //add robot to the list
                RobotCheckList.Items.Add(new ImageProccessing.Robot(name, sign, port));

                //reset the input textboxes
                textBoxName.Clear();
                textBoxSign.Clear();
                textBoxPort.Clear();
            }
            else
            {
                MessageBox.Show("All fields must be filled");
            }
        }

        private void buttonRemoveRbt_Click(object sender, EventArgs e)
        {
            //if is something selected
            if (RobotCheckList.SelectedIndex != -1)
            {
                //remove this robot from checklist
                RobotCheckList.Items.RemoveAt(RobotCheckList.SelectedIndex);//always this last cause the following change the index
num
            }
        }

        private void comboBoxDevice_SelectedIndexChanged(object sender, EventArgs e)
        {
            //select device from drop down menu
            devIdx = comboBoxDevice.SelectedIndex;
            dev = devMgr.getDevice((uint)devIdx);
            debug.AppendText(String.Format("Device: {0} selected\n", dev.serial.read()));
        }

        private void buttonStart_Click(object sender, EventArgs e)
        {
            if (buttonStart.Text == "Start tracking")
            {
                //do START
                //if a device is selected
                if (dev != null)
                {
                    buttonStart.Text = "Stop tracking";
                    buttonStart.BackColor = Color.Red;

                    stat.Clear();
                    win = new Window(new mvIMPACT_NET.Image());
                    win.title = "Press Stop before closing this window";
                    if (fit.Checked)//fit to page selection check
                        win.zoomMode = ZoomingMode.zmFitAspect;
```

```
            //start image proccessing thread
            imageProcWorker.RunWorkerAsync();
        }
        else
        {
            MessageBox.Show("Please select Device");
        }
    }
    else
    {
        //do STOP
        buttonStart.Text = "Start tracking";
        buttonStart.BackColor = Color.LimeGreen;

        imageProcWorker.CancelAsync();
        if (checkBoxLogging.Checked)
        {
            logFileStream.Close();
        }
    }

}

private void imageProcWorker_DoWork(object sender, DoWorkEventArgs e)
{
    //debug.Invoke(new Action(() => debug.Clear()));
    debug.Invoke(new Action(() => debug.AppendText("Opening device...\n")));
    try
    {
        dev.open();
    }
    catch (ImpactAcquireException ex)
    {
        // this e.g. might happen if the same device is already opened in another process...
        MessageBox.Show("An error occurred while opening the device "
            + dev.serial.read() + "(error code: " + ex.Message + ").");
    }


    // establish access to the statistic properties
    Statistics statistics = new Statistics(ref dev);
    // create an interface to the device found
    FunctionInterface fi = new FunctionInterface(ref dev);

    try
    {
        fi.loadSetting(settingsFilename, TStorageFlag.sfFile);
    }
    catch
    {
        MessageBox.Show("Could not load camera settings");
        fi.loadSettingFromDefault();
    }

    //can overload with the specific request object. if not just use one of the available
    fi.imageRequestSingle();//fps: 6.95 capture time (s): 0.11075
    fi.imageRequestSingle();//fps: 17.05 capture time (s): 0.111
```

```
//fi.imageRequestSingle();//fps: 17.05 capture time (s): 0.167
//fi.imageRequestSingle();//fps: 17.05 capture time (s): 0.168

// run thread loop
Request request;
const int timeout_ms = 2000;
int requestNr = -1;
int lastRequestNr = -1;
int cnt = 0;

//if (RobotList.Items.Count == 0) //needs implementation

//crating a vector named 'input' that contains the selected robots to feed them to the image proccessing constructor
ImageProccessing.Robot[] input = new ImageProccessing.Robot[RobotCheckList.CheckedIndices.Count];
RobotCheckList.CheckedItems.CopyTo(input, 0);

//initiate imageproccessing
ImageProccessing imProc = new ImageProccessing(2, input);// This method also load the lookup tables
imProc.l_dzone = (int)leftDzone.Value;
imProc.r_dzone = (int)rightDzone.Value;
imProc.u_dzone = (int)upDzone.Value;
imProc.d_dzone = (int)lowDzone.Value;
imProc.threshold = (byte)threshold.Value;
imProc.tolerance = (double)tolerance.Value;

int winheight = (int)winHeight.Value;
int winwidth = (int)winWidth.Value;

//the LOOP
while (!imageProcWorker.CancellationPending)
{
   // wait for results from the default capture queue blocking method
   requestNr = fi.imageRequestWaitFor(timeout_ms);

   if (fi.isRequestNrValid(requestNr))
   {
      request = fi.getRequest(requestNr);
      if (fi.isRequestOK(ref request))
      {
         ++cnt;
         // here we can display some statistical information every 50th image
         if (cnt % 20 == 0)
         {
            stat.Invoke(new Action(() => stat.Text ="\nfps: " + statistics.framesPerSecond.readS() + ", error count: " +
               statistics.errorCount.readS() + ", capture time: " + statistics.captureTime_s.readS() + "\n"));
         }
         // This call is fast, as it uses the request memory to create the IMPACT image. However
         // the IMPACT image will become invalid as soon as the request buffer is unlocked via
         // fi.imageRequestUnlock( requestNr );
         mvIMPACT_NET.Image image = request.getIMPACTImage(TImpactBufferFlag.ibfUseRequestMemory);
         // This call would be slower as it copies the image data. This image remains valid even when the
         // request has been unlocked again...
         // Image image = pRequest.getIMPACTImage();


/*****************************************************************************************
                                IMAGE PROCCESSING (region)
```

```
                *
*****************************************************************************************/

            #region ImageProccessing
            //history based (smart) scan
            if (imProc.fullScanRequired == false)
            {
               //update only assigned leds
               foreach (ImageProccessing.LED ledi in imProc.ledFound.Where(led => led.assigned==true))
               {
                  imProc.updateLed(image, ledi, winwidth, winheight);
               }

               /*
               //upadate all founded leds
               for (int i=0; i < imProc.ledFound.Count; i++)
               {
                  imProc.updateLed(image, imProc.ledFound[i], winwidth, winheight);
                  //if a led lost but fullScanRequired_flag didnt raised this means that is not assigned to any robot,
                  //so we remove it from the list

                  //if is not uptodate
                  if (!imProc.ledFound[i].isUptodate)
                  {
                     imProc.ledFound.RemoveAt(i);
                     i--;
                  }
               }*/

               imProc.calcLedsRealCoords();
               //update all robots
               foreach (ImageProccessing.Robot rob in imProc.rbt)
               {
                  imProc.updateRobot(rob, statistics.framesPerSecond.read());
               }

            }
            //full scan
            else
            {
               imProc.scanForLeds(image); //create a list findLed.ledFound with all leds camera can see
               if (imProc.ledFound.Count >= imProc.ledNum) // chech that there is at least as many led required
               {
                  imProc.fullScanRequired = false;
                  imProc.calcLedsRealCoords();
                  imProc.ScanForRobots();
                  foreach (ImageProccessing.Robot rob in imProc.rbt)
                  {
                     imProc.updateRobot(rob, statistics.framesPerSecond.read());
                  }
                  debug.Invoke(new Action(() => debug.AppendText("Full scan found enough leds\n")));
               }
               else
               {
                  debug.Invoke(new Action(() => debug.AppendText(String.Format("full scan can't find enough leds.{0} leds
missing\n", imProc.ledNum - imProc.ledFound.Count))));
               }
```

```
                if (imProc.fullScanRequired==false)
                    debug.Invoke(new Action(() => debug.AppendText(String.Format("Started smart scan with {0}x{1} window\n",
winwidth, winheight))));
                }
                #endregion


                //if someone close the win window without "stop tracking" fisrt the followings crash and run catch
                try
                {
                  if (imProc.rbtNum > 0)//if there are robot selected
                  {
                    for (int i = 0; i < imProc.rbtNum; i++) //for the selected robot
                    {
                      if (imProc.rbt[i].Exist) //if image proccessing has found it draw a circle around it
                      {
                        var pt = new mvIMPACT_NET.Point((int)imProc.rbt[i].bigLed.x, (int)imProc.rbt[i].bigLed.y);
                        image.drawEllipse(pt, 90, 90, (int)imProc.rbt[i].sign);
                      }
                      //else not needed, every image is new
                    }
                    UDPBroadcastRobots(imProc.rbt, statistics.captureTime_s.read());

                    if (checkBoxLogging.Checked)
                    {
                      LogWrite(String.Format("{0}\t{1}\t{2}\t{3}\n", imProc.rbt[0].x, imProc.rbt[0].y, imProc.rbt[0].theta,
imProc.rbt[0].calculatedSign));
                    }
                  }

                  win.buffer = image;
                  win.update();
                }
                catch
                {
                  //do STOP
                  buttonStart.Invoke(new Action(() => buttonStart.Text = "Start tracking"));
                  buttonStart.Invoke(new Action(() => buttonStart.BackColor = Color.LimeGreen));

                  imageProcWorker.CancelAsync();
                  if (checkBoxLogging.Checked)
                  {
                    logFileStream.Close();
                  }
                }
                //image.Dispose();
              }
              else
              {
                debug.Invoke(new Action(() => debug.AppendText(String.Format("Error: " + request.requestResult.readS() +
"\n"))));
              }
              if (fi.isRequestNrValid(lastRequestNr))
              {
                // this image has been displayed thus the buffer is no longer needed...
                fi.imageRequestUnlock(lastRequestNr);
              }
```

```
            lastRequestNr = requestNr;
            // send a new image request into the capture queue
            fi.imageRequestSingle();
        }
        else
        {
            // this should not happen in this sample, but may happen if you wait for a request without
            // sending one to the driver before or if a trigger is missing while the device has been
            // set up for triggerd acquisition. Please refer to the documentation for reasons for
            // possible errors if you ever reach this code
            debug.Invoke(new Action(() => debug.AppendText(String.Format("Acquisition error: {0}.\n", requestNr))));
        }
        //debug.Invoke(new Action(() => debug.Clear()));
    }//End of the LOOP

    /*
    //prevent display of already freed memory
    win.buffer = new mvIMPACT_NET.Image();
    win.update();
    */

    win.Dispose();

    // free the last potential locked request
    if (fi.isRequestNrValid(requestNr))
    {
        fi.imageRequestUnlock(requestNr);
    }
    // clear the request queue
    fi.imageRequestReset(0, 0);
    // extract and unlock all requests that are now returned as 'aborted'
    while ((requestNr = fi.imageRequestWaitFor(0)) >= 0)
    {
        fi.imageRequestUnlock(requestNr);
    }
    dev.close();
}

private void buttonDebug_Click(object sender, EventArgs e)
{

}

private void buttonHelp_Click(object sender, EventArgs e)
{
    MessageBox.Show("TROLLLLLLL");

}

private void buttonBrowse_Click(object sender, EventArgs e)
{
    openFileDialog1.InitialDirectory = ".\\";//default path the path of the .exe
    if (openFileDialog1.ShowDialog() == DialogResult.OK)
    {
        settingsFilename = openFileDialog1.FileName;
        textBoxSettings.Text = settingsFilename;
    }
```

```
}

private void buttonBrowseLogfile_Click(object sender, EventArgs e)
{
   openLogFile.InitialDirectory = ".\\";//default path the path of the .exe
   if (openLogFile.ShowDialog() == DialogResult.OK)
   {
      LogFilename = openLogFile.FileName;
      textBoxLogFile.Text = LogFilename;
   }
   if (System.IO.File.Exists(LogFilename))
   {
      System.IO.File.Delete(LogFilename);
   }
}

private void LogWrite(string str)
{
   Byte[] data = new UTF8Encoding(true).GetBytes(str);
   // Add some information to the file.
   try
   {
      logFileStream.Write(data, 0, data.Length);
   }
   catch
   {
      AppendLine("log file problem");
   }
}

private void checkBoxLogging_CheckedChanged(object sender, EventArgs e)
{
   if (checkBoxLogging.Checked)//just checked
   {
      if (!System.IO.File.Exists(LogFilename))
      {
         logFileStream = System.IO.File.Create(LogFilename);
      }
   }
}

private void buttonServer_Click(object sender, EventArgs e)
{
   mytcpserver = new tcpThreadedServer(12001);
   mytcpserver.Start();

   UDPConnect(25000);
}

delegate void Remote_setup(int exposure,int clock);

public void remote_settings(int exposure,int clock)
{
   if (this.debug.InvokeRequired)
   {
      Remote_setup a = new Remote_setup(remote_settings);
      this.Invoke(a, new object[] {exposure, clock});
```

```
        }
        else
        {
            if (exposure <= 500)
            {
                settingsFilename = ".\\Overlapping_overclocked_500exp.xml";
            }
            else if (exposure >= 5000)
            {
                settingsFilename = ".\\Overlapping_overclocked_5000exp.xml";
            }
            else if (exposure >= 3000)
            {
                settingsFilename = ".\\Overlapping_overclocked_3000exp.xml";
            }

            textBoxSettings.Text = settingsFilename;
        }
    }

/***********************************UDP*************************************/


    private void SendCameraMessage(byte[] _msg)
    {
        Framing frm = new Framing();
        byte[] _newmsg = frm.EscapeBytes(_msg);
        //SendMessageCamera(_newmsg, _newmsg.Length);
    }

    private void UDPConnect(int GroupPort)
    {
        udpClient = new UdpClient();
        udpClient.EnableBroadcast = true;
        UDPgroupEP = new IPEndPoint(IPAddress.Broadcast, GroupPort);
        UDPreceiveEndPoint = new IPEndPoint(IPAddress.Any, GroupPort);

        //udpClient.BeginReceive(new AsyncCallback(UDPReceive), UDPreceiveEndPoint);
        debug.Invoke(new Action(() => debug.AppendText(String.Format("UDP server broadcast to {0}\n",GroupPort))));
    }

    public static void UDPReceive(IAsyncResult iar)
    {
        byte[] bytes = udpClient.EndReceive(iar, ref UDPreceiveEndPoint);

        // Parse received data

        udpClient.BeginReceive(new AsyncCallback(UDPReceive), UDPreceiveEndPoint);
    }

    private void UDPSend(byte[] _msg)
    {
        udpClient.Send(_msg, _msg.Length, UDPgroupEP);
    }

    private void UDPBroadcastRobotPos(ImageProccessing.Robot[] Rob, double timeDelay)
    {
```

```
    List<byte> buffer =new List<byte>();
    buffer.Capacity = 14;

    for (int i = 0; i < 1; i++)
    {
        if (Rob[i].Exist)
        {
            byte[] x = BitConverter.GetBytes((Int16)(10*Rob[i].x));
            buffer.AddRange(x);
            byte[] y = BitConverter.GetBytes((Int16)(10*Rob[i].y));
            buffer.AddRange(y);
            byte[] theta = BitConverter.GetBytes((Int16)(10000*Rob[i].theta));
            buffer.AddRange(theta);
            byte[] u = BitConverter.GetBytes((Int16)(10*Rob[i].u));
            buffer.AddRange(u);
            byte[] v = BitConverter.GetBytes((Int16)(10*Rob[i].v));
            buffer.AddRange(v);
            byte[] omega = BitConverter.GetBytes((Int16)(10000*Rob[i].omega));
            buffer.AddRange(omega);
            byte[] delay = BitConverter.GetBytes((Int16)(1000*timeDelay));
            buffer.AddRange(delay);
        }
        else
        {
            Int16 nan = 999;

            byte[] x = BitConverter.GetBytes(nan);
            buffer.AddRange(x);
            byte[] y = BitConverter.GetBytes(nan);
            buffer.AddRange(y);
            byte[] theta = BitConverter.GetBytes(nan);
            buffer.AddRange(theta);
            byte[] u = BitConverter.GetBytes(nan);
            buffer.AddRange(u);
            byte[] v = BitConverter.GetBytes(nan);
            buffer.AddRange(v);
            byte[] omega = BitConverter.GetBytes(nan);
            buffer.AddRange(omega);
            byte[] delay = BitConverter.GetBytes(timeDelay);
            buffer.AddRange(delay);
        }
        byte[] _msg = new byte[buffer.Count];
        for (int j = 0; j < buffer.Count; j++)
        {
            _msg[j] = buffer[j];
        }

        udpClient.Send(_msg, _msg.Length, UDPgroupEP);
    }


}

private void UDPBroadcastStream(ImageProccessing.Robot[] Rob, double timeDelay)
{
    List<byte> buffer = new List<byte>();
    buffer.Capacity = (4 * 8 * Rob.Length) + 2;
```

```
//message delimiter
Int32 msgDelimiter = 0x7E;
byte[] msgDelim = BitConverter.GetBytes(msgDelimiter);
buffer.AddRange(msgDelim);

for (int i = 0; i < Rob.Length; i++)
{
   //robot delimiter
   Int32 robDelimiter = 0x7E;
   byte[] robDelim = BitConverter.GetBytes(robDelimiter);
   buffer.AddRange(robDelim);

   if (Rob[i].Exist)
   {
      //robot id
      byte[] id = BitConverter.GetBytes((Rob[i].ID));
      buffer.AddRange(id);
      byte[] x = BitConverter.GetBytes((Int32)(10*Rob[i].x));
      buffer.AddRange(x);
      byte[] y = BitConverter.GetBytes((Int32)(10*Rob[i].y));
      buffer.AddRange(y);
      byte[] theta = BitConverter.GetBytes((Int32)(1000*Rob[i].theta));
      buffer.AddRange(theta);
      byte[] u = BitConverter.GetBytes((Int32)(10*Rob[i].u));
      buffer.AddRange(u);
      byte[] v = BitConverter.GetBytes((Int32)(10*Rob[i].v));
      buffer.AddRange(v);
      byte[] omega = BitConverter.GetBytes((Int32)(1000*Rob[i].omega));
      buffer.AddRange(omega);
      byte[] delay = BitConverter.GetBytes((Int32)(1000*timeDelay));
      buffer.AddRange(delay);
   }
   else
   {
      //robot id
      byte[] id = BitConverter.GetBytes((Rob[i].ID));
      buffer.AddRange(id);
      //didnt found
      Int32 nan = -9999;
      byte[] x = BitConverter.GetBytes(nan);
      buffer.AddRange(x);
      byte[] y = BitConverter.GetBytes(nan);
      buffer.AddRange(y);
      byte[] theta = BitConverter.GetBytes(nan);
      buffer.AddRange(theta);
      byte[] u = BitConverter.GetBytes(nan);
      buffer.AddRange(u);
      byte[] v = BitConverter.GetBytes(nan);
      buffer.AddRange(v);
      byte[] omega = BitConverter.GetBytes(nan);
      buffer.AddRange(omega);
      byte[] delay = BitConverter.GetBytes(timeDelay);
      buffer.AddRange(delay);
   }
   //robot delimiter
   buffer.AddRange(robDelim);
}
```

```
    //message delimiter
    buffer.AddRange(msgDelim);

    byte[] _msg = new byte[buffer.Count];
    for (int j = 0; j < buffer.Count; j++)
    {
        _msg[j] = buffer[j];
    }

    udpClient.Send(_msg, _msg.Length, UDPgroupEP);
}

private void UDPBroadcastRobots(ImageProccessing.Robot[] Rob, double timeDelay)
{
    for (int i = 0; i < Rob.Length; i++)
    {
        //initialize stream
        List<byte> buffer = new List<byte>();
        buffer.Capacity = 17;

        //message starter delimiter
        byte delimiter = (byte)0x7E;
        buffer.Add(delimiter);

        //robot ID
        byte robID = (byte)Rob[i].ID;
        buffer.Add(robID);

        if (Rob[i].Exist)
        {
            byte[] x = BitConverter.GetBytes((Int16)(10 * Rob[i].x));
            buffer.AddRange(x);
            byte[] y = BitConverter.GetBytes((Int16)(10 * Rob[i].y));
            buffer.AddRange(y);
            byte[] theta = BitConverter.GetBytes((Int16)(10000 * Rob[i].theta));
            buffer.AddRange(theta);
            byte[] u = BitConverter.GetBytes((Int16)(10 * Rob[i].u));
            buffer.AddRange(u);
            byte[] v = BitConverter.GetBytes((Int16)(10 * Rob[i].v));
            buffer.AddRange(v);
            byte[] omega = BitConverter.GetBytes((Int16)(10000 * Rob[i].omega));
            buffer.AddRange(omega);
            byte[] delay = BitConverter.GetBytes((Int16)(1000 * timeDelay));
            buffer.AddRange(delay);
        }
        else
        {
            Int16 nan = -999;

            byte[] x = BitConverter.GetBytes(nan);
            buffer.AddRange(x);
            byte[] y = BitConverter.GetBytes(nan);
            buffer.AddRange(y);
            byte[] theta = BitConverter.GetBytes(nan);
            buffer.AddRange(theta);
            byte[] u = BitConverter.GetBytes(nan);
            buffer.AddRange(u);
```

```
            byte[] v = BitConverter.GetBytes(nan);
            buffer.AddRange(v);
            byte[] omega = BitConverter.GetBytes(nan);
            buffer.AddRange(omega);
            byte[] delay = BitConverter.GetBytes(timeDelay);
            buffer.AddRange(delay);
        }

        //message end delimiter
        buffer.Add(delimiter);

        //convert to byte stream
        byte[] _msg = new byte[buffer.Count];
        for (int j = 0; j < buffer.Count; j++)
        {
            _msg[j] = buffer[j];
        }

        //send byte stream
        udpClient.Send(_msg, _msg.Length, UDPgroupEP);
        Debug.WriteLine("Robot coords sended");
      }
    }

    private void UDPDisconnect(int GroupPort)
    {
      udpClient.Close();
    }
  }
}
```