Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Π Τ Υ Χ Ι Α Κ Η Ε Ρ Γ Α Σ Ι Α

Καθοδήγηση ενός ρομπότ NXT Lego Mindstorm με χρήση σαρωτή λέιζερ.

ΜΑΤΣΟΥΚΗΣ ΠΑΝ. ΕΥΑΓΓΕΛΟΣ

ΑΡΙΘΜΟΣ ΜΗΤΡΩΟΥ: **1206**

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ:

ΠΑΛΑΜΑΣ ΓΕΩΡΓΙΟΣ

ΗΡΑΚΛΕΙΟ 2009

Περιγραφή

Σκοπός της έρευνας είναι να κατασκευαστεί ένας αισθητήρας ο οποίος θα καθοδηγεί ένα ρομπότ (lego mindstorm NXT) σε πραγματικό χρόνο καθώς επίσης και σε πραγματικό περιβάλλον χρησιμοποιώντας μία κάμερα σε συνδυασμό με ένα σαρωτή λέιζερ.
Θα πρέπει το ρομπότ να είναι ικανό να αποφεύγει εμπόδια προκαθορισμένου ύψους, διαφορετικού σχήματος, μεγέθους, υλικού και υφής.

Εδώ βρίσκεται και η πρωτοτυπία μας, καθώς σε αυτόν τον τομέα οι άλλοι αισθητήρες υστερούν ή απλά το κόστος απόκτησης και χρήσης τους είναι υπέρογκο.
Η επεξεργασία της εικόνας έγινε με τη βοήθεια του Matlab, γεγονός που δυσχέραινε την ταχύτητα εκτέλεσης. Αυτό οδήγησε στην απαραίτητη δημιουργία τεχνικών και αλγορίθμων οι οποίοι θα ενεργούσαν γρηγορότερα.

Το αποτέλεσμα επιτυγχάνεται με τη χρήση μιας κάμερας, η οποία εστιάζει στο μπροστινό μέρος του ρομπότ (σημείο όπου εστιάζει και ο σαρωτής λέιζερ).
Από την εικόνα (που λαμβάνεται από την κάμερα) γίνεται διαχωρισμός του λέιζερ από το υπόλοιπο φόντο, με χρήση μιας μεθόδου προσαρμοστικής κατάτμησης την οποία και δημιουργήσαμε.
Τα σημεία πάνω στα οποία προσπίπτει η ακτίνα του λέιζερ είναι και αυτά τα οποία προσδιορίζουν τα εμπόδια ως προς το ύψος και το μήκος.

Λέξεις κλειδιά: σαρωτής λέιζερ, κάμερα, κατάτμηση, Lego Mindstorm NXT.

1 Δομή της πτυχιακής εργασίας και του γραπτού λόγου.

Η πτυχιακή εργασία εκπονήθηκε κατά κύριο λόγο στο εργαστήριο ευφυών συστημάτων του Α.Τ.Ε.Ι Ηρακλείου.

Κατά την εξέλιξη της διαπιστώθηκε ότι χρειαζότανε πολύ έρευνα διότι δεν υπάρχει κάτι παρόμοιο ή κάτι έτοιμο το οποίο να μπορούσε να χρησιμοποιηθεί (όπως διαπιστώνετε και στα παρακάτω κεφάλαια).

Γι αυτό τον λόγο θεώρησα ότι η εξέλιξη του γραπτού λόγου θα έπρεπε να έχει μορφή ιστορικής αναδρομής. Να εξελίσσεται, δηλαδή, από τον αρχικό προβληματισμό μέχρι το τελικό αποτέλεσμα, ώστε να αναδείξω τα σημαντικότερα κομμάτια – πειράματα της έρευνας μου. Με αυτό τον τρόπο μπορώ να αποδείξω τα αποτελέσματα που προέκυψαν από την έρευνα μου και να οδηγήσω τον αναγνώστη στην κατανόηση των προβλημάτων που αντιμετώπισα και των λύσεων που προτείνω.

2 Εισαγωγή

Η λέξη ρομπότ προέρχεται από το σλαβικό robota που σημαίνει εργασία. Καθιερώθηκε ως όρος με την σημερινή του έννοια το 1920 από τον Τσέχο θεατρικό συγγραφέα Karel Čapek στο έργο του "R.U.R." (Rossum's Universal Robots), όπου σατιρίζει την εξάρτηση της κοινωνίας από τους μηχανικούς εργάτες (ρομπότ) της τεχνολογικής εξέλιξης και που τελικά εξοντώνουν τους δημιουργούς τους.

Το ρομπότ είναι μια μηχανική συσκευή η οποία μπορεί να υποκαθιστά τον άνθρωπο σε διάφορες εργασίες. Ένα ρομπότ μπορεί να δράσει κάτω από τον απευθείας έλεγχο ενός ανθρώπου ή αυτόνομα κάτω από τον έλεγχο ενός προγραμματισμένου υπολογιστή. Τα ρομπότ μπορούν να χρησιμοποιηθούν ώστε να κάνουν εργασίες οι οποίες είναι δύσκολες ή επικίνδυνες για να γίνουν απ' ευθείας από τον άνθρωπο. Σε άλλες περιπτώσεις, χρησιμοποιούνται για να εκτελέσουν εργασίες ταχύτερα ή φθηνότερα απ' ότι ο άνθρωπος. Έτσι, μπορούν να χρησιμοποιηθούν στην αυτόματη παραγωγή μεγάλων ποσοτήτων κάποιου προϊόντος γρηγορότερα καθώς και με χαμηλότερο κόστος (για παράδειγμα, στις αλυσίδες παραγωγής).

Από τα πρώτα ρομπότ που αναφέρονται στη λογοτεχνία είναι ο Τάλως από την ελληνική μυθολογία και οι 20 τρίποδες λέβητες του Ηφαίστου θεωρούμενοι "θαύμα ήδεσαι" κ.α.

2.1 Lego Mindstorms

Τα Lego Mindstorms είναι μια γραμμή παραγωγής της Lego που συνδυάζει προγραμματίσημα τούβλα με ηλεκτρικές μηχανές, αισθητήρες, τούβλα Lego, και τεχνικά κομμάτια Lego (όπως εργαλεία, άξονες, ακτίνες, και υδραυλικά μέρη)κατάλληλα για να χτίσει ο χρήστης ρομπότ και άλλα αυτοματοποιημένα ή διαλογικά συστήματα.

Τα Lego Mindstorms μπορούν να χρησιμοποιηθούν για να κατασκευαστεί ένα μοντέλο ενσωματωμένου συστήματος με ηλεκτρομηχανικά μέρη ελεγχόμενα από υπολογιστή . Πολλά είδη πραγματικών ενσωματωμένων συστημάτων, από ελεγκτές ανελκυστήρων έως βιομηχανικά ρομπότ, μπορούν να διαμορφωθούν χρησιμοποιώντας τα Mindstorms.

Η πρώτη λιανική έκδοση Lego Mindstorms κυκλοφόρησε το 1998 και πωλήθηκε εμπορικά με την επωνυμία Robotics Invention System (RIS). Η τρέχουσα έκδοση κυκλοφόρησε το 2006 ως Lego Mindstorms NXT (την οποία και χρησιμοποίησα στην πτυχιακή μου εργασία).

Η αρχική Mindstorms Robotics Invention System περιείχε δύο μηχανές, δύο αισθητήρες αφής, και έναν αισθητήρα φωτός. Η έκδοση NXT έχει τρεις σερβομηχανές και τέσσερις αισθητήρες για την αφή, το φως, τον ήχο, και την απόσταση.

2.2 Lego Mindstorms NXT

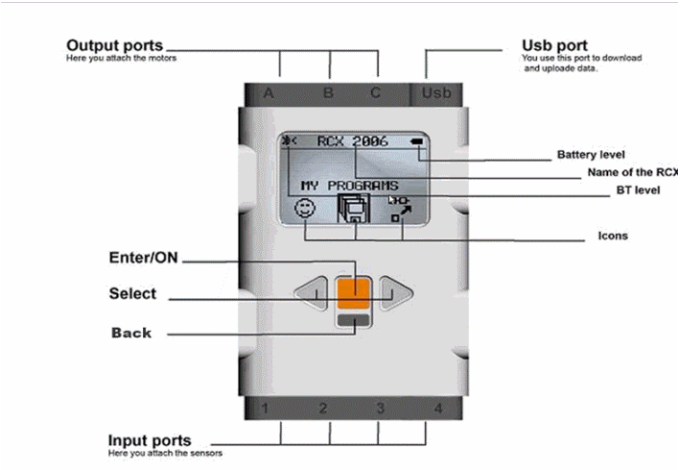
Το NXT βασίζεται στο επιτυχημένο Robotics System Invention της εταιρείας, το οποίο έχει βελτιωθεί με την προσθήκη νέων τεχνολογιών και αισθητήρων αυξημένων ικανοτήτων.

Το «τουβλάκι» NXT που αποτελεί τον εγκέφαλο του ρομπότ είναι ένας αυτόνομος μικροεπεξεργαστής των 32 bit (σε αντίθεση με τα 16 bit της πρώτης γενιάς), ο οποίος μπορεί να προγραμματιστεί μέσω ηλεκτρονικού υπολογιστή PC ή - άλλη καινοτομία - Mac.

Αφού κατασκευάσει το ρομπότ του, ο χρήστης δημιουργεί ένα δικό του πρόγραμμα χρησιμοποιώντας ένα εύχρηστο αλλά πλούσιο σε χαρακτηριστικά λογισμικό LabVIEW, το οποίο έχει σχεδιαστεί από τη National Instruments.

Τεχνικά χαρακτηριστικά

- 32-bit ARM7 microcontroller
- 256 Kbytes FLASH, 64 Kbytes RAM
- 8-bit AVR microcontroller
- 4 Kbytes FLASH, 512 Byte RAM
- Bluetooth wireless communication (Bluetooth Class II V2.0 compliant)
- USB full speed port (12 Mbit/s)
- 4 input ports, 6-wire cable digital platform (One port includes a IEC 61158 Type 4/EN 50 170 compliant expansion port for future use)
- 3 output ports, 6-wire cable digital platform
- 100 x 64 pixel LCD graphical display
- Loudspeaker - 8 kHz sound quality. Sound channel with 8-bit resolution and 2-16 KHz sample rate.
- Power source: 6 AA batteries



Εικόνα 1: Μικροελενκτης NXT, η διεπαφή του και τα βασικά χαρακτηριστικά του.

2.2.1 Απεικόνιση του NXT – Ανάλυση των μερών που το αποτελούν

Το NXT είναι ο εγκέφαλος ενός ρομπότ MINDSTORMS®. Είναι ένα ευφυές, ελεγχόμενο από υπολογιστή τούβλο LEGO® που δίνει τη δυνατότητα σε ένα ρομπότ MINDSTORMS να ζωντανέψει και να εκτελέσει τις διαφορετικές διαδικασίες.

2.2.1.1 Θύρες εισόδου – εξόδου

Το NXT αποτελείται από τις εξής συσκευές εισόδου – εξόδου:

- Τρεις θύρες παραγωγής για την ένωση των κινητήρων – θύρες A, B και C θύρες αισθητήρων
- Τέσσερις εισαγμένες θύρες για την ένωση των αισθητήρων - θύρες 1, 2, 3 και 4.
- Μια θύρα USB για την επικοινωνία του NXT με τον υπολογιστή προκειμένου να φορτώσει τα προγράμματα από τον υπολογιστή στο NXT (ή το αντίστροφο). Μπορεί επίσης να χρησιμοποιήσει την ασύρματη σύνδεση Bluetooth για την επικοινωνία με τον υπολογιστή. Μεγάφωνο για αναπαραγωγή πραγματικών ήχων.
- Οθόνη υγρών κρυστάλλων ανάλυσης 100 x 64 εικονοστοιχείων για απεικόνιση βασικών λειτουργιών και μηνυμάτων.
- Τέσσερα πλήκτρα για τις βασικές λειτουργίες της συσκευής.

2.2.1.2 Αισθητήρας αφής του NXT

Ο αισθητήρας αφής δίνει στο ρομπότ την αίσθηση της αφής. Με τον τρόπο αυτό το ρομπότ έχει την ικανότητα να ανιχνεύει πότε πιέζεται από κάτι και πότε απελευθερώνεται πάλι.

Ένα σενάριο χρήσης του, είναι να χρησιμοποιηθεί ο αισθητήρας αφής για να κάνει το ρομπότ να αισθανθεί ένα αντικείμενο.

Αυτό επιτυγχάνεται με τον εξής τρόπο : ένας ρομποτικός βραχίονας που εξοπλίζεται με έναν αισθητήρα αφής ενημερώνει το ρομπότ εάν υπάρχει ή όχι κάτι στο βραχίονά του προκειμένου να το αρπάξει.

Ο αισθητήρας αφής στην ουσία είναι ένας διακόπτης διπλής κατάστασης (ανοιχτός – κλειστός).

2.2.1.3 Αισθητήρας ήχου του NXT

Ο αισθητήρας ήχου μπορεί να ανιχνεύσει δύο ειδών decibel[1]: decibels [DB] και σταθμισμένο decibel [DBA].

DBA: στην ανίχνευση ρυθμισμένων decibels, όπου η ευαισθησία του αισθητήρα προσαρμόζεται στην ευαισθησία του ανθρώπινου αυτιού.

DB: στην ανίχνευση τυποποιημένων (χωρίς διόρθωση) decibels, όπου όλοι οι ήχοι μετρούνται με ίση ευαισθησία.

Κατά συνέπεια, αυτοί οι ήχοι μπορούν να περιλάβουν μερικούς πάρα πολύ υψηλούς ή πάρα πολύ χαμηλούς για να ακουστούν από το ανθρώπινο αφτί. Ο αισθητήρας ήχου μπορεί να μετρήσει τα επίπεδα υγιούς πίεσης μέχρι 90 DB. Τα επίπεδα υγιούς πίεσης είναι εξαιρετικά περίπλοκα, έτσι ο αισθητήρας στο MINDSTORMS NXT επιδεικνύεται σε ποσοστό [%]. Όσο χαμηλότερα τα τοις εκατό τόσο πιο ήρεμο το περιβάλλον

Για παράδειγμα :

• 4-5% είναι όπως ένα σιωπηλό καθιστικό

• 5-10% θα ήταν σαν κάποιος που μιλά από μακρινή απόσταση

• 10-30% είναι η κανονική συνομιλία κοντά στον αισθητήρα ή μουσική που παίζεται σε κανονικό επίπεδο

• 30-100% είναι σαν να φωνάξει κάποιος ή σαν να υπάρχει δυνατή μουσική.

2.2.1.4 Αισθητήρας φωτός του NXT

Ο αισθητήρας φωτός είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ (ο υπερηχητικός αισθητήρας είναι ο άλλος). Ο αισθητήρας φωτός επιτρέπει στο ρομπότ να διακρίνει μεταξύ του φωτός και του σκοταδιού. Μπορεί να διαβάσει την ελαφριά ένταση σε ένα δωμάτιο και να μετρήσει την ελαφριά ένταση των χρωματισμένων επιφανειών.

Π.χ. μπορεί να χρησιμοποιηθεί για να κάνετε ένα ρομπότ να ταξινομεί αντικείμενα κατά χρώμα.

2.2.1.5 Αισθητήρας Υπέρηχων του NXT

Ο αισθητήρας υπέρηχων είναι ο δεύτερος αισθητήρας που δίνει όραση στο ρομπότ (ο αισθητήρας φωτός είναι ο άλλος). Ο υπερηχητικός αισθητήρας επιτρέπει στο ρομπότ σας να δει και να ανιχνεύσει τα αντικείμενα - εμπόδια. Μπορεί επίσης να το χρησιμοποιηθεί για να

[1] Decibel είναι μια μονάδα μέτρησης της πίεσης.

κάνει το ρομπότ να αποφύγει τα εμπόδια, την απόσταση αίσθησης και μέτρου, και να ανιχνεύσει τη μετακίνηση. Ο υπερηχητικός αισθητήρας μετρά την απόσταση σε εκατοστόμετρα και σε ίντσες. Είναι σε θέση να μετρήσει τις αποστάσεις από 0 έως 255 εκατοστόμετρα με ακρίβεια +/- 3 εκατ.

Ο υπερηχητικός αισθητήρας μετρά την απόσταση με τον υπολογισμό του χρόνου που παίρνει ένα ηχητικό κύμα για να χτυπήσει ένα αντικείμενο και να επιστρέψει - ακριβώς όπως μια ηχώ. Τα μεγάλου μεγέθους αντικείμενα με τις σκληρές επιφάνειες επιστρέφουν τις καλύτερες αναγνώσεις. Τα αντικείμενα φτιαγμένα από μαλακό ύφασμα ή τα κυρτά (όπως μια σφαίρα) ή τα πολύ λεπτά ή τα μικρά μπορεί να είναι δύσκολα για τον αισθητήρα να ανιχνευθούν.

2.2.1.6 Σερβοκινητήρες του NXT

Οι τρεις βηματικοί σερβοκινητήρες δίνουν στο ρομπότ την δυνατότητα να κινηθεί. Υπάρχει δυνατότητα να χρησιμοποιηθεί αυτόματος συγχρονισμός των σερβοκινητήρων έτσι ώστε το ρομπότ να κινηθεί σε μια ευθεία γραμμή.

Κάθε μηχανή έχει ενσωματωμένο έναν αισθητήρα περιστροφής. Αυτό αφήνει τον έλεγχό σε ακριβείς μετακινήσεις του ρομπότ. Ο αισθητήρας περιστροφής μετρά τις περιστροφές του κινητήρα στους βαθμούς ή στις πλήρεις περιστροφές (ακρίβεια +/- μιας μοίρας). Μια περιστροφή είναι ίση με 360 μοίρες, έτσι εάν θέσετε τον έναν κινητήρα σε στροφή 180 μοιρών, ο άξονας παραγωγής του θα κάνει μισή στροφή. Ο ενσωματωμένος αισθητήρας περιστροφής σε κάθε κινητήρα επιτρέπει την προσαρμογή με ακρίβεια στις διαφορετικές ταχύτητες (με τον καθορισμό των διαφορετικών παραμέτρων ταχύτητας στο λογισμικό).

2.2.1.7 Επιπλέων αισθητήρες του NXT

Πέρα από τους βασικούς αισθητήρες που περιλαμβάνονται στο πακέτο Lego Mintstorm NXT υπάρχουν πολλοί ακόμα από τρίτες εταιρίες, ορισμένες εκ των οποίων:

HiTechnic, a division of Dataport Systems, Inc. (http://www.hitechnic.com)

- NXT Touch Sensor Multiplexer (NTX1060)
- Compass Sensor (Model NMC1034) - LEGO Certified Hardware
- Color Sensor (Model NCO1038) - LEGO Certified Hardware
- Acceleration / Tilt Sensor (Model NAC1040) - LEGO Certified Hardware
- IRSeeker Sensor (Model NKS1042) - LEGO Certified Hardware
- Gyro Sensor (Model NGY1044) - LEGO Certified Hardware

- IRLink Sensor (Model NIL1046) - LEGO Certified Hardware
- Prototype Board - Solderable (Model NPB1050)

CODATEX Hainzlmaier GmbH & Co.KG (http://www.codatex.com)

- RFID Sensor for LEGO MINDSTORMS NXT - LEGO Certified Hardware
- RFID Keyfob Transponders - LEGO MINDSTORMS NXT Certified Hardware

Mindsensors.com, USA (http://www.mindsensors.com)

- Vision Subsystem v2 for NXT
- 8 Channel Servo Controller for NXT
- RCX Sensor multiplexer for NXT
- Multi-Sensitivity Acceleration Sensor v3 for NXT - (ACCL-Nx-v3)
- Motor Multiplexer for NXT (MTRMX-Nx)
- RCX to NXT Communication Adapter (NRLink-Nx)
- Sony PlayStation 2 Controller interface for NXT (PSP-Nx)
- Magnetic compass for NXT (CMPS-Nx)
- Pneumatic Pressure Sensor for NXT (PPS35-Nx)
- Realtime Clock for NXT

Εικόνα 2: Βασικά μέρη που αποτελούν το πακέτο Lego Mindstorm NXT: A) σερβοκινητήρες, B) Μικροελενκτής NXT, C) Αισθητήρας αφής, D) Αισθητήρας Ήχου, E) Αισθητήρας φωτός, F) Αισθητήρας υπερήχων.

2.2.2Υποστηριζόμενες Γλώσσες προγραμματισμού

Ο μικροεπεξεργαστής NXT υποστηρίζεται από πληθώρα γλωσσών προγραμματισμού (java, C, C++, C#, python, VB, γλώσσες τύπου scripting κ.α.). Αναφορικά ορισμένες γλώσσες προγραμματισμού του NXT είναι οι ακόλουθες:

- RCX Code (περιέχεται στις Mindstorm εκδόσεις λιανικής)
- ROBOLAB (βασίζεται στο LabVIEW και αναπτύχθηκε στο Tufts University)

Δημοφιλείς Γλώσσες τρίτων κατασκευαστών:

- C and C++ under BrickOS (formerly Lego's)
- Java under leJOS or TinyVM
- NQC ("Not Quite C")
- pbFORTH (επεκτάσεις της Forth γλώσσας προγραμματισμού)
- Visual Basic (μέσω του COM+ interface περιεχόμενο με το CD)
- RobotC (νέα γλώσσα συμβατή με την έκδοση NXT)
- RWTH - Mindstorms NXT βιβλιοθήκη για το MATLAB
- Κ.α.

2.3 Matlab

MATLAB ("matrix laboratory") εφευρέθηκε προς το τέλος της δεκαετίας του '70 από τον Cleve Moler, τότε πρόεδρο του τμήματος πληροφορικής στο πανεπιστήμιο του Νέου Μεξικό. Το σχεδίασε για να δώσει την πρόσβαση στους σπουδαστές του σε LINPACK[2] και EISPACK[3] χωρίς να πρέπει να μάθουν το FORTRAN. Το διέδωσε σύντομα σε άλλα πανεπιστήμια και βρήκε ένα ισχυρό ακροατήριο μέσα στα εφαρμοσμένα μαθηματικά που διαδόθηκαν σε άλλα πανεπιστήμια και βρήκε ένα ισχυρό ακροατήριο εντός της εφαρμοσμένης κοινότητας μαθηματικών. Αναγνωρίζοντας την εμπορική δυνατότητά του, ενώθηκε με το Moler και το Steve Bangert. Ξαναέγραψαν το MATLAB στο C και ίδρυσαν το MathWorks το 1984 για να συνεχίσουν την ανάπτυξή του. Αυτές οι ξαναγραμμένες βιβλιοθήκες ήταν γνωστές ως JACKPAC. Το 2000, MATLAB ξαναγράφηκε για να χρησιμοποιήσει ένα νεότερο σύνολο βιβλιοθηκών για το χειρισμό μιτρών, LAPACK.

[2] LINPACK βιβλιοθήκη προγράμματος για τον υπολογισμό αριθμητικής γραμμικής άλγεβρας σε ψηφιακούς υπολογιστές γραμμένων σε FORTRAN.

[3] EISPACK βιβλιοθήκη λογισμικού για αριθμητικό υπολογισμό των ιδιοτιμών και ιδιοδιανυσμάτων των πινάκων γραμμένων σε FORTRAN.

Το MATLAB υιοθετήθηκε αρχικά από τους μηχανικούς σχεδίου ελέγχου, αλλά γρήγορα κ από πολλές άλλες ειδικότητες. Τώρα, επίσης, χρησιμοποιείται στην εκπαίδευση, και ιδιαίτερα στη διδασκαλία της γραμμικής άλγεβρας και της αριθμητικής ανάλυσης. Είναι δημοφιλές μεταξύ των επιστημών που ασχολούνται με την επεξεργασία εικόνας.

2.3.1 Τι είναι το Matlab

Το MATLAB είναι ένα ολοκληρωμένο λογισμικό, το άνομα του οποίου προέρχεται από την συνένωση των λέξεων Matrix laboratory δηλαδή εργαστήριο μήτρων (συνηθίζεται στη επιστημονική κοινότητα ο όρος πίνακας να καλείται μήτρα). Μία πρώτη περιγραφή του προγράμματος αυτού μας κρύβει άτι θα χειρίζεται πίνακες κατά τους υπολογισμούς του. Πρόκειται για ένα δημοφιλές πακέτο στους ακαδημαϊκούς και δεν είναι τίποτα άλλο, παρά ένα σύγχρονο και εξελιγμένο εργαλείο της πληροφορικής με πολύ καλή υποστήριξη γραφικών. Προορίζεται σε φοιτητές, καθηγητές (κυρίως θετικών επιστημών), επιστήμονες διάφορων βαθμίδων, επαγγελματίες, μηχανικούς, τεχνικούς και σε πάρα πολλούς άλλους ανθρώπους που ασχολούνται σε εξειδικευμένους τομείς της τεχνολογίας
Το MATLAB έχει πολλά πλεονεκτήματα σε σχέση με τις γνωστές γλώσσες προγραμματισμού διότι:

- Ο προγραμματισμός γίνεται σε μια γλώσσα υψηλού επιπέδου γρήγορα και άνετα.
- Υποστηρίζει τον αντικειμενοστραφή προγραμματισμό
- Παρέχει υψηλής ποιότητας γραφικά.
- Έχει πληθώρα διαθέσιμων κωδικών στο internet και πολλά έτοιμα αρχεία M-files.
- Παρέχει στην Βοήθεια του πολλά παραδείγματα.

Όπως προαναφέραμε το όνομα του MATLAB κρύβει και την κύρια λειτουργία αυτού του προγράμματος, δηλαδή την χρήση πινάκων οι οποίοι μπορούν να έχουν στοιχεία είτε πραγματικούς αριθμούς είτε μιγαδικούς. Έτσι κάθε αριθμός δηλαδή κάθε βαθμωτή ποσότητα θα μπορεί να θεωρηθεί σαν ένα πίνακα με ένα μοναδικό στοιχείο.

Το MATLAB από την κατασκευή του έχει προοριστεί για να λύνει τα προβλήματα με αριθμητικές μεθόδους, οπότε θα έχει και αριθμητική πεπερασμένη ακρίβεια, οι δε λύσεις του θα είναι προσεγγιστικές πάρα αναλυτικές. Να τονίσουμε ότι κάθε πρόγραμμα κατασκευάζεται για ένα συγκεκριμένο σκοπό και επομένως δεν μπορούμε να το συγκρίνουμε με άλλα προγράμματα τις ίδιας κατηγορίας όπως είναι το Mathematica και το Maple. Βέβαια η αριθμητική μέθοδος που ακολουθεί το MATLAB δεν πρέπει να μας ανησυχεί καθόλου αφού τα αποτελέσματα του είναι μεγάλης εμπιστοσύνης αξιοπιστίας και ακρίβειας.

2.3.2 Πλεονεκτήματα του Matlab

Μερικά από τα σημαντικότερα πλεονεκτήματα του Matlab είναι τα ακόλουθα:

- Ευκολότερη εκμάθηση από μια γλώσσα προγραμματισμού
- Βελτιστοποιημένος κώδικας για διεξαγωγή υπολογισμών με πίνακες
- Γλώσσα προγραμματισμού για ανάπτυξη εφαρμογών και ταυτόχρονα λογισμικού υλοποίησης επιστημονικών υπολογισμών
- Εύκολος εντοπισμός και διόρθωση λαθών
- Φιλικό περιβάλλον επικοινωνίας με το χρήστη

2.3.3 Μειονεκτήματα

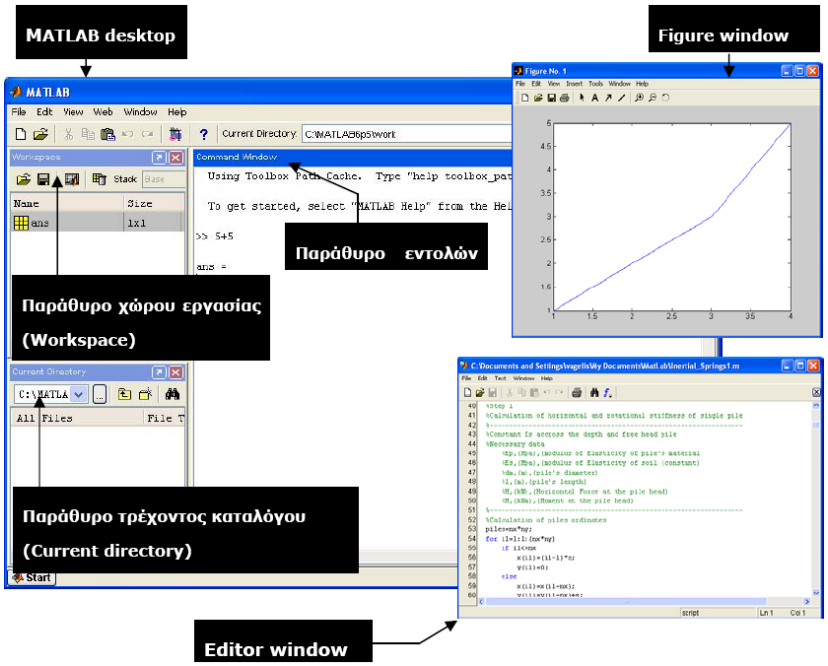
Τα κυριότερα μειονεκτήματα που αντιμετωπίζει ο χρήστης του Matlab είναι τα ακόλουθα:

- Εξειδικευμένη γλώσσα προγραμματισμού
- Το λογισμικό MATLAB αναπτύχθηκε μόνο για διεξαγωγή επιστημονικών υπολογισμών κι έτσι δεν ενδείκνυται ή υποστηρίζει την ανάπτυξη άλλου είδους εφαρμογών, π.χ. επεξεργασία κειμένου
- Οι αναπτυσσόμενες εφαρμογές υστερούν σε απόδοση από την άποψη χρόνου εκτέλεσης σε σχέση με αντίστοιχες που αναπτύσσονται με τις κλασικές γλώσσες προγραμματισμού (C, C++, Fortran)

2.3.4 Το περιβάλλον του MATLAB

Το MATLAB υποστηρίζει σχεδόν όλα τα διατιθέμενα λειτουργικά συστήματα. Εκτός από την πλατφόρμα των Windows, μπορεί να εφαρμοστεί και σε άλλες πλατφόρμες λειτουργικών συστημάτων, όπως το UNIX, Sun Solaris, Linux και MAC OS 10.3.

Σε όλα λοιπόν τα παραπάνω συστήματα, το MATLAB λειτουργεί μέσω τριών βασικών παραθύρων, τα οποία παρουσιάζονται στην Εικ.2 και αναλύονται ευθύς αμέσως.



Εικόνα 3: Περιβάλλον χρήσης του Matlab

2.4 Εργαλειοθήκη του Matlab image acquisition & processing

Οι εργαλειοθήκες Image acquisition και Processing παρέχουν μια πλήρη σειρά από αλγόριθμους και γραφικά εργαλεία για την επεξεργασία ,την ανάλυση, την απεικόνιση, και την ανάπτυξη αλγόριθμων για εικόνες. Μπορεί να επαναφέρει θορυβώδεις ή υποβαθμισμένες εικόνες, να εξάγει τα χαρακτηριστικά τους και να αναλύσει σχήματα και υφές. Οι περισσότερες λειτουργίες της εργαλειοθήκης αυτής είναι γραμμένες, στην γλώσσα MATLAB , δίνοντας τη δυνατότητα να επιθεωρήσει τους αλγόριθμους, να τροποποιήσει τον πηγαίο κώδικα, ακόμη και να δημιουργήσει τους προσαρμοσμένους αλγόριθμους. Το Image Processing Toolbox βοηθάει τους μηχανικούς και τους επιστήμονες σε τομείς όπως μετρήσεις βιομετρικών στοιχείων, τηλεανίχνευση, εποπτεία, γονιδιακή έκφραση, μικροσκοπία, έλεγχο ημιαγωγών, αισθητήριο σχεδιασμό της εικόνα και στην επιστήμη των υλικών. Επίσης, διευκολύνει την εκμάθηση και τη διδασκαλία της τεχνικής επεξεργασίας εικόνας.

Υποστηρίζει εικόνες δημιουργημένες από ένα ευρύ φάσμα συσκευών, όπως είναι οι ψηφιακές φωτογραφικές μηχανές, οι frame grabbers, οι δορυφορικοί και αερομεταφερόμενοι αισθητήρες, οι συσκευές ιατρικής απεικόνισης , τα μικροσκόπια, τα τηλεσκόπια, και άλλα επιστημονικά όργανα. Μπορεί να αναλύσει και να επεξεργαστεί αυτές τις εικόνες σε πολλών τύπων δεδομένα, συμπεριλαμβανομένης της μονής και διπλής ακρίβειας floating-point και signed ή unsigned 8 -, 16 - και 32-bit ακέραιους.

2.4.1 Εισαγωγή και εξαγωγή εικόνων

Υπάρχουν διάφοροι τρόποι για την εισαγωγή ή εξαγωγή εικόνων μέσα και έξω από το περιβάλλον MATLAB για μεταποίηση. Μπορεί να χρησιμοποιηθεί το Image Acquisition Toolbox για να λάβει ζωντανά εικόνες από κάμερες Web, frame grabbers, DCAM[4]-συμβατές κάμερες, και άλλες συσκευές. Με την χρήση Database Toolbox, υπάρχει πρόσβαση σε εικόνες αποθηκευμένες σε ODBC / JDBC συμβατές βάσεις δεδομένων.

Η MATLAB υποστηρίζει διαφόρων τύπων και μορφών δεδομένα και εικόνες, συμπεριλαμβανομένων των: JPEG, TIFF, PNG, HDF, HDF-EOS, FITS, το Microsoft Excel, ASCII, και δυαδικά αρχεία. Υποστηρίζει επίσης multiband τύπους εικόνας, όπως LANDSAT[5]. Χαμηλού επιπέδου I / O λειτουργίες επιτρέπουν να αναπτύξει προσαρμοσμένες ρουτίνες για συνεργασία με οποιαδήποτε μορφή δεδομένα. Υποστηρίζει ένα αριθμό εξειδικευμένων μορφών αρχείων εικόνας. Για ιατρικές εικόνες, υποστηρίζετε η DICOM μορφή αρχείου, η εργαλειοθήκη μπορεί επίσης να διαβάσει γεωχωρικές εικόνες στη μορφή NITF και στο υψηλό δυναμικό εύρος εικόνες του HDR.

2.4.2 Πριν και μετά την Επεξεργασία Εικόνων

Η εργαλειοθήκη προσφέρει αλγόριθμους για την πριν-και-μετά επεξεργασία, επιλύοντας συχνά προβλήματα του συστήματος, όπως η παρεμβολή του θορύβου, η χαμηλή δυναμική περιοχή, η μη επικεντρωμένη οπτική, και η διαφορά στη χρωματική απεικόνιση μεταξύ συσκευών εισόδου και εξόδου.

2.4.3 Διαχείριση χρώματος ανεξαρτήτως συσκευής

Μας επιτρέπει να αντιπροσωπεύονται τα χρώματα με ακρίβεια ανεξάρτητα από τις συσκευές εισόδου και εξόδου. Αυτό είναι χρήσιμο όταν αναλύονται τα χαρακτηριστικά μίας συσκευής, κατά την ακριβή και ποσοτική μέτρηση τους χρώματος, ή κατά την ανάπτυξη αλγορίθμων για πολλές διαφορετικές συσκευές. Με τις εξειδικευμένες λειτουργίες στην εργαλειοθήκη, μπορεί να μετατρέψει τις εικόνες ανεξαρτήτου συσκευής και χρωματικού τύπου , όπως sRGB, XYZ, xyY, L * α * β *, uvL, κ.α.

[4] **Dcam: είναι ένας νέος τύπος ψηφιακής κάμερας παρακολούθησης (βρίσκεται ακόμα σε δοκιμαστικό στάδιο).**

[5] **Landsat: είναι ένα πρόγραμμα το οποίο χρησιμοποιείται για την απόκτηση εικόνων από το διάστημα.**

2.4.4 Μετατροπές Εικόνας

Η απεικόνιση εφαρμογών συχνά απαιτεί μετατροπή μεταξύ των κατηγοριών δεδομένων και του είδους της εικόνας.

Το Image Processing Toolbox παρέχει μια ποικιλία υπηρεσιών κοινής ωφελείας για τη μετατροπή μεταξύ δεδομένων , συμπεριλαμβανομένης της μονής και διπλής ακρίβειας floating-point και signed ή unsigned 8 -, 16 - και 32-bit δεκαδικούς. Η εργαλειοθήκη περιλαμβάνει αλγόριθμους για τη μετατροπή μεταξύ τύπου εικόνας, συμπεριλαμβανομένων των δυαδικών, γκρι, και truecolor.

Συγκεκριμένα για εικόνες με χρώμα, υποστηρίζεται μια ποικιλία τύπου χρωμάτων YIQ, HSV, και YCrCb, Bayer κ.α. κωδικοποιημένα πρότυπα, και υψηλό δυναμικό εύρος εικόνων. Τα χρωματικά μοντέλα αναλύονται στο κεφάλαιο 5.1

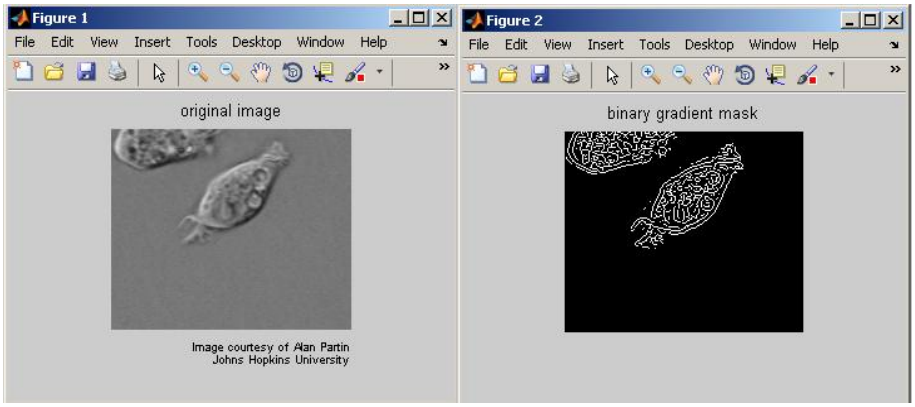
2.4.5 Αναλύοντας Εικόνες

Παρέχεται μία περιεκτική ακολουθία αλγορίθμων και γραφικών εργαλείων για θέματα ανάλυσης εικόνων, όπως είναι η ανάλυση στατιστικών στοιχείων, εξαγωγή χαρακτηριστικών, και μετρήσεις.

Οι *Στατιστικές λειτουργίες* επιτρέπουν την γενική ανάλυση των χαρακτηριστικών μιας εικόνας κατά:

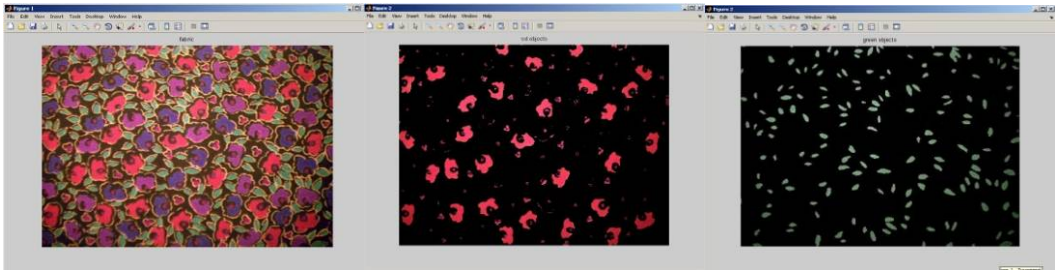
- Τον καθορισμό των τιμών έντασης κατά μήκος ενός τμήματος γραμμής.
- Την εμφάνιση του ιστογράμματος μιας εικόνας
- Την εμφάνιση ενός προφίλ των τιμών έντασης .

Οι αλγόριθμοι εντοπισμού ακμών (Εικόνα 4) μας επιτρέπουν να εντοπίζουμε τα όρια ενός αντικειμένου σε μια εικόνα. Οι αλγόριθμοι εντοπισμού των ακμών περιλαμβάνουν τις εξής μεθόδους: Sobel, Prewitt, Roberts, Canny, Laplacian και Gaussian. Ο ισχυρός Canny αλγόριθμος μπορεί να ανιχνεύσει τις αδύναμες ακμές, χωρίς να ξεγελαστεί από το θόρυβο που μπορεί να υπάρχει στην εικόνα.



Εικόνα 4: Παράδειγμα αλγόριθμου εντοπισμού ακμών

Οι αλγόριθμοι κατάτμησης (Εικόνα 5) μιας εικόνας, καθορίζουν τα όρια σε μία περιοχή της εικόνας. Μπορούμε να ανακαλύψουμε διάφορες προσεγγίσεις για την κατάτμηση μιας εικόνας, όπως τον αυτοματισμό του thresholding(κατωφλιού), της μεθόδου ακμής με βάση, και μορφολογικά βασισμένες μεθόδους όπως είναι η watershed transform.



Εικόνα 5: Παράδειγμα κατάτμησης εικόνας.

Οι μορφολογικές ενέργειες μας επιτρέπουν να ανιχνεύσουμε τις ακμές, να ενισχύσουμε την αντίθεση, να αφαιρέσουμε τον θόρυβο, να κατατμήσουμε μια εικόνα σε περιοχές, και να δημιουργήσουμε λεπτές περιοχές.
Άλλες μορφολογικές λειτουργίες που περιλαμβάνονται στην εργαλειοθήκη είναι:

- Διάβρωση και διαστολή.
- Άνοιγμα και κλείσιμο εικόνων.
- Επισήμανση συστατικών.
- Καμπή τμηματοποίησης.
- Ανασυγκρότηση.
- Μετατροπή απόστασης.
- Κ.α.

Η εργαλειοθήκη επίσης μας προσφέρει κάποιες προηγμένες λειτουργίες για την ανάλυσης της εικόνας που μας επιτρέπουν να μετρήσουμε τις ιδιότητες μιας συγκεκριμένης περιοχής της εικόνας , όπως είναι το εμβαδόν, και το κέντρο της μάζας.

2.5 RWTH - Mindstorms NXT Toolbox for MATLAB

Αυτή η εργαλειοθήκη αναπτύχθηκε για να ελέγξει τα ρομπότ LEGO® MINDSTORMS® NXT με τη χρήση του MATLAB μέσω μιας ασύρματης σύνδεσης Bluetooth ή μέσω USB. Αυτό το λογισμικό είναι δωρεάν ανοικτού κώδικα και υπόκειται στην άδεια χρήσης ευρέος κοινού GNU (GPL). Η ανάπτυξη του RWTH - εργαλειοθήκη Mindstorms NXT παρακινήθηκε από ένα πρόγραμμα φοιτητών του πανεπιστημίου RWTH Aachen «MATLAB meets LEGO Mindstorms» για τους σπουδαστές ηλεκτρικής εφαρμοσμένης μηχανικής και επομένως είναι σχεδιασμένος κυρίως για λόγους εκπαίδευσης.

Οι λειτουργίες εργαλειοθηκών είναι βασισμένες στο πρωτόκολλο επικοινωνίας LEGO MINDSTORMS NXT Bluetooth για να ελέγξουν το ευφυές τούβλο NXT μέσω μιας ασύρματης σύνδεσης Bluetooth ή μέσω USB. Αν και μια σύνδεση Bluetooth δεν συστήνεται για ένα ρομπότ για κίνηση σε πραγματικό χρόνο (λόγω της υψηλής λανθάνουσας κατάστασής του), ελέγχοντας γενικά αυτή η εργαλειοθήκη παρέχει τις λειτουργίες MATLAB για να αλληλεπιδράσει με ένα ρομπότ άμεσα. Οι λανθάνουσες καταστάσεις μέσω των συνδέσεων USB είναι πολύ μικρότερες και επιτρέπουν τις περιπλοκότερες εφαρμογές.

Το κύριο πλεονέκτημα αυτής της έννοιας τηλεχειρισμού επιτρέπει σε σας να συνδυάσετε τις εφαρμογές ρομπότ με τις σύνθετες μαθηματικές διαδικασίες και τις απεικονίσεις μέσα σε MATLAB. Αυτή η εργαλειοθήκη ανοίγει τις απεριόριστες δυνατότητες να παρασχεθεί η τεχνητή νοημοσύνη στο ρομπότ και άλλες βελτιώσεις χρησιμοποιώντας τα πολλαπλά χαρακτηριστικά του MATLAB και τους υπολογισμούς για την επεξεργασία ψηφιακού σήματος.

Το RWTH - η εργαλειοθήκη Mindstorms NXT είναι ελεύθερο λογισμικό: μπορείτε να το ανακατανείμετε ή/και να το τροποποιήσετε υπό τον όρο της άδειας του ευρέος κοινού GNU όπως δημοσιεύεται από τη Free Software Foundation, είτε στην έκδοση 3 της άδειας, είτε οποιαδήποτε πιο πρόσφατη έκδοση κατά επιλογή σας.

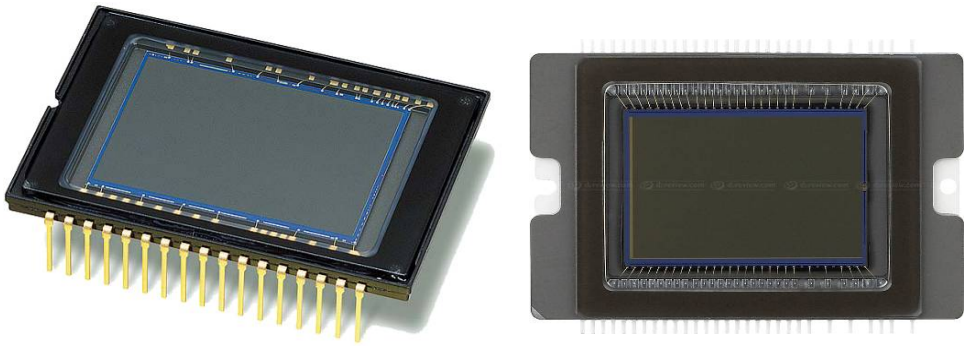
Το RWTH - η εργαλειοθήκη Mindstorms NXT διανέμεται χωρίς οποιαδήποτε εξουσιοδότηση ούτε καν με την υπονοούμενη εξουσιοδότηση MERCHANTABILITY ή την προσαρμογή του για την πραγματοποίηση ενός ιδιαιτέρου σκοπού.

2.6 Αισθητήρες CCD και CMOS

Υπάρχουν δυο κύρια είδη αισθητήρων, οι CCD και CMOS. CCD σημαίνει Charge Coupled Device, ενώ CMOS σημαίνει Complementary Metal Oxide Semiconductor. Ευρύτερα χρησιμοποιούμενοι είναι οι αισθητήρες CCD. Το πλεονέκτημα των αισθητήρων CMOS είναι ότι το κόστος κατασκευής τους είναι πολύ χαμηλότερο από αυτών των

αισθητήρων CCD. Όμως η ποιότητα εικόνας που δίνουν είναι χαμηλότερη. Έτσι χρησιμοποιούνται κυρίως σε φτηνές μηχανές με χαμηλή ανάλυση.

Μερικές εταιρείες χρησιμοποιούν ένα είδος αισθητήρα CCD που είναι διαφορετικός ως προς την κατασκευαστική του δομή από τους άλλους αισθητήρες και τον ονομάζουν Super CCD. Σε αυτόν η διάταξη των εικονοστοιχείων είναι διαφορετική και το σχήμα τους οκταγωνικό. Η διάταξη αυτή προσφέρει μεγαλύτερη ευαισθησία, καλύτερο λόγο σήματος/θορύβου και ευρύτερο δυναμικό πεδίο.



Εικόνα 6: Αισθητήρες CCD και CMOS

2.6.1 Τι είναι το CCD (Charge Coupled Device)

Το CCD στα ελληνικά αποδίδεται ως είδος αισθητήρα. Είναι ένας “μηχανισμός εικόνας” εντός των περισσότερων μοντέρνων καμερών ο οποίος είναι ευαίσθητος στο φως. Είναι ένα μεγάλης κλίμακας ολοκληρωμένο κύκλωμα που περιέχει εκατοντάδες χιλιάδες εικονοστοιχεία (pixels) τα οποία αποτελούνται από φωτοευαίσθητες επιφάνειες (τα φωτοδιόδια) που αναλαμβάνουν την καταγραφή του φωτός το οποίο το μεταφράζει σε ηλεκτρικό ρεύμα, το οποίο μετατρέπεται σε ψηφιακό σήμα από τον αντίστοιχο μετατροπέα.

Το μέγεθος του μετράται διαγώνια και μπορεί να είναι 1/4, 1/3, 1/2, ή 2/3 της ίντσας. Όσο μεγαλύτερο τόσο καλύτερη η ποιότητα της εικόνας.

Ανακαλύφθηκε το 1970 και σκοπός του ήταν να χρησιμοποιηθεί σαν μηχανισμός μνήμης. Χρησιμοποιείται περισσότερο στις κάμερες αλλά επίσης και στα τηλέφωνα, στις συσκευές φαξ, σαρωτές, κλπ.

2.6.2 Πως λειτουργεί ο αισθητήρας CCD

Ένας αισθητήρας αποτελείται από σειρές εικονοστοιχείων (pixels που προέρχεται από το picture elements) τα οποία με τη σειρά τους έχουν ως βασικό συστατικό το πυρίτιο. Τα εικονοστοιχεία είναι μικρότερες μονάδες του αισθητήρα που μπορούν να καταγράψουν πληροφορίες για την ένταση του φωτός και το χρώμα σε μια εικόνα.

Κάθε τέτοιο εικονοστοιχείο είναι ευαίσθητο στο φως, έτσι όταν αυτό πέσει επάνω του παράγει ηλεκτρική τάση. Η ηλεκτρική τάση είναι ανάλογη με την ποσότητα του φωτός που πέφτει στο εικονοστοιχείο. Όσο περισσότερο το φως, τόσο μεγαλύτερη τάση παράγεται.

Υπάρχουν δυο βασικές κατηγορίες αισθητήρων. Οι αισθητήρες διάταξης και οι γραμμικοί.

Στους πρώτους τα εικονοστοιχεία χωρίζονται σε τρεις κατηγορίες:

- σε αυτά που αντιδρούν μόνο στο κόκκινο φως,
- σε αυτά που αντιδρούν μόνο στο πράσινο,
- και τέλος σε αυτά που αντιδρούν μόνο στο μπλε.

Ως γνωστό όλα τα χρώματα σχηματίζονται από το συνδυασμό αυτών των τριών χρωμάτων: κόκκινου, πράσινου και μπλε.
Έτσι λοιπόν, ένα εικονοστοιχείο που αντιδρά στο κόκκινο φως παράγει ηλεκτρική τάση μόνο όταν πέσει πάνω του κόκκινο φως, ενώ μένει ανεπηρέαστο από τα άλλα δύο χρώματα. Αντίστοιχα αντιδρούν και τα άλλα δύο.
Συνεπώς χρειαζόμαστε τρία διαφορετικά εικονοστοιχεία για να αναπαράγουμε ένα οποιοδήποτε χρώμα.

2.7 Λέιζερ

Ο όρος λέιζερ προέρχεται από το αγγλικό ακρωνύμιο Laser: Light Amplification by Stimulated Emission of Radiation που αποδίδεται στα ελληνικά ως ενίσχυση φωτός με εξαναγκασμένη εκπομπή ακτινοβολίας και καλύπτει τόσο τις συσκευές που την παράγουν όσο και την αντίστοιχη ακτινοβολία.

Τα λέιζερ παράγουν σύμφωνο, μονοχρωματικό φως (δηλαδή φως με συγκεκριμένο μήκος κύματος-χρώμα) το οποίο διαδίδεται σε μια συγκεκριμένη κατεύθυνση, σχηματίζοντας στενές δέσμες. Αντίθετα, οι συνηθισμένες πηγές φωτός, όπως οι λαμπτήρες πυρακτώσεως, παράγουν μη-σύμφωνο φως προς όλες τις διευθύνσεις και, επιπλέον, έχουν μεγάλο φασματικό εύρος.

Η λειτουργία των λέιζερ ερμηνεύεται από την θεωρία της κβαντικής μηχανικής και της θερμοδυναμικής. Πολλά υλικά έχουν βρεθεί ότι έχουν τα απαραίτητα χαρακτηριστικά για να αποτελέσουν ενεργό υλικό των λέιζερ, με αποτέλεσμα την δημιουργία πολλών τύπων λέιζερ με διαφορετικά χαρακτηριστικά, που χρησιμοποιούνται σε μεγάλο εύρος εφαρμογών.

Η εφεύρεση των λέιζερ στηρίχθηκε στην κατασκευή των λέιζερ στην δεκαετία του 1950. Το πρώτο λέιζερ κατασκευάστηκε το 1960, από τότε όμως τα λέιζερ βρήκαν εφαρμογή στις θετικές επιστήμες, στην βιομηχανία, στην ιατρική, και στην ηλεκτρονική.

Εικόνα 7: Λέιζερ πανομοιότυπο με αυτό που χρησιμοποιήθηκε στην εργασία

2.7.1 Αρχή λειτουργίας του λέιζερ

Τα λέιζερ αποτελούνται από το ενεργό υλικό, και την οπτική κοιλότητα.

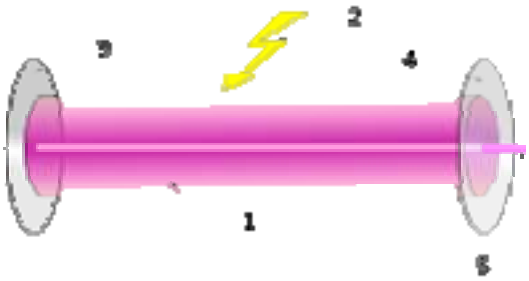
Το ενεργό υλικό μετατρέπει την εξωτερική ενέργεια σε δέσμη φωτός. Συνήθως είναι υλικό με συγκεκριμένο μέγεθος, σύσταση, καθαρότητα και μορφή, που παράγει φως μέσω εξαναγκασμένης εκπομπής, η οποία αποτελεί κβαντομηχανική διαδικασία που προτάθηκε από τον Αλβέρτο Αϊνστάιν για να ερμηνεύσει το φωτοηλεκτρικό φαινόμενο.

Το ενεργό υλικό αντλείται από μία εξωτερική πηγή ενέργειας. Τέτοιες πηγές μπορεί να είναι ηλεκτρικές ή φωτεινές, όπως η λυχνία έκλαμψης (flash lamp) ή κάποια άλλη πηγή λέιζερ. Η ενέργεια που απορροφάτε αποτίθεται στα σωματίδια του ενεργού υλικού έτσι, ώστε αυτά να οδηγηθούν σε μια διεγερμένη κβαντική κατάσταση.

Όταν ο αριθμός των σωματιδίων που βρίσκονται στην διεγερμένη κατάσταση είναι μεγαλύτερος από τον αριθμό των ατόμων που βρίσκεται στην βασική κατάσταση, επιτυγχάνεται αντιστροφή πληθυσμού.

Έτσι λοιπόν, μία δέσμη φωτός που περνάει μέσα από το υλικό έχει μεγαλύτερη πιθανότητα να οδηγήσει σε εξαναγκασμένη εκπομπή φωτονίων από ότι σε εξαναγκασμένη απορρόφηση,

με αποτέλεσμα να επιτυγχάνεται ενίσχυση της δέσμης. Ένα διεγερμένο ενεργό υλικό μπορεί να λειτουργήσει επίσης και σαν οπτικός ενισχυτής.



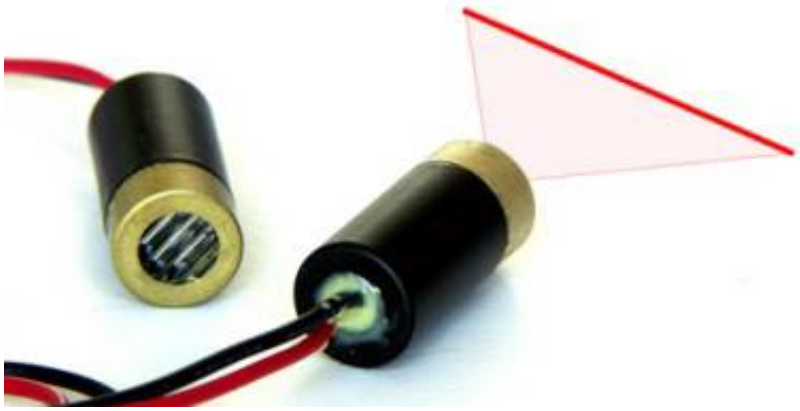
Εικόνα 8: Αρχή λειτουργίας του Λέιζερ 1. Ενεργό υλικό του Λέιζερ, 2. Προσφερόμενη ενέργεια άντλησης, 3. Υψηλής ανακλαστικότητας κάτοπτρο, 4. Διάταξη εξόδου δέσμης, 5. Δέσμη Λέιζερ

Τα χαρακτηριστικά του φωτός που παράγονται από εξαναγκασμένη εκπομπή είναι παρόμοια με αυτά του αρχικού φωτός, ως προς το μήκος κύματος, την πόλωση και την φάση. Έτσι λοιπόν, το φως του λέιζερ που παράγεται είναι σύμφωνο, ενώ η σταθερότητα της πόλωσης και η μονοχρωματικότητα εξαρτώνται από τα χαρακτηριστικά της οπτικής κοιλότητας.

Αν η δέσμη δημιουργείται και διαδίδεται σε ελεύθερο περιβάλλον και όχι μέσα σε κυματοδηγούς (όπως στην περίπτωση των οπτικών ινών), τότε η ένταση του φωτός εμφανίζει κανονική (Γκαουσιανή) κατανομή, κάθετα στην διεύθυνση διάδοσής της. Η δέσμη του λέιζερ είναι σχεδόν απόλυτα ευθυγραμμισμένη, δηλαδή δεν αποκλίνει. Παρόλα αυτά, τέλεια ευθυγραμμισμένη δέσμη δεν μπορεί να υπάρξει λόγω περίθλασης. Για παράδειγμα, μια δέσμη με αρχική διάμετρο 2 mm, που δημιουργείται από ένα μικρό εργαστηριακό λέιζερ (όπως ένα λέιζερ Ηλίου-Νέου), αποκλίνει αποκτώντας διάμετρο 1,6 χιλιόμετρα, όταν διανύσει απόσταση ίση με αυτή της γης-σελήνης.

2.7.2 Φακός δημιουργίας γραμμής λέιζερ (σαρωτής λέιζερ)

Ο φακός δημιουργίας γραμμής λέιζερ είναι ένας ειδικά διαμορφωμένος φακός από πλαστικό ή γυαλί, όπου προσαρμόζεται στην έξοδο του λέιζερ. Αυτό έχει ως αποτέλεσμα να παράγετε μια συνεχόμενη γραμμή (αντί μιας κουκίδα όπως τα κοινά λέιζερ). Τα κυριότερα χαρακτηριστικά των φακών είναι η γωνία που διαθλάται η ακτίνα φωτός και το μέγεθος του φακού (προκείμενου να προσαρμόζεται στα διαφορετικά λέιζερ)



Εικόνα 9: Σαρωτής λέιζερ.

Εικόνα 10: Φακός δημιουργίας γραμμής λέιζερ.

2.8 Αισθητήρες μέτρησης απόστασης

Οι αισθητήρες απόστασης είναι απαραίτητοι εάν το ρομπότ σκοπεύει να κάνει οποιαδήποτε είδους χαρτογράφηση, δεδομένου ότι πρέπει να ξέρουμε πόσο μακριά είναι τα αντικείμενα.

Οι αισθητήρες απόστασης είναι συνήθως οι υπέρυθροι και οι υπερηχητικοί.

Και στις δύο περιπτώσεις, ένας παλμός (φωτός ή ήχου) στέλνεται και η αντανάκλασή του χρονομετρείτε προκειμένου να μετατραπεί σε μια αίσθηση της απόστασης.

2.8.1 Αισθητήρες υπερύθρων

Οι κοινοί αισθητήρες υπερύθρων δεν είναι σε θέση να μετρήσουν απόσταση, παρά μόνο να μας ενημερώσουνε για την ύπαρξη ή μη ενός εμποδίου στο οπτικό του πεδίο.

Εξέλιξη των απλών IR αισθητήρων ανακλαστικότητας, αποτελούν τα αισθητήρια της σειράς GP2Dxx της Sharp.

Κύριο χαρακτηριστικό τους είναι η δυνατότητα μέτρησης της απόστασης των αντικειμένων που ανιχνεύονται. Μαζί με μια υπέρυθρη φωτοδίοδο (IR LED), ενσωματώνουν σε μια ολοκληρωμένη συσκευή έναν ανιχνευτή θέσης (PSD - position sensitive detector) συνδυαζόμενο με φακό εστίασης.

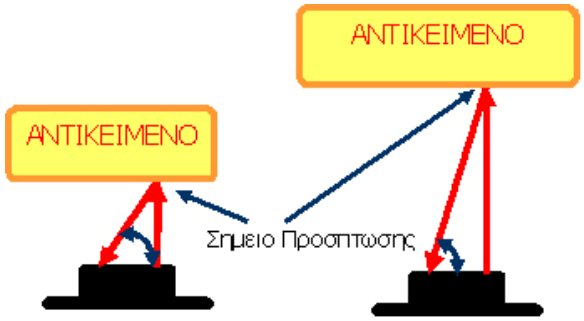
Η έξοδος τους δεν επηρεάζεται από τον περιβάλλοντα φωτισμό, αλλά ούτε και από το χρώμα του αντικειμένου που ανιχνεύεται.

Άλλα πλεονεκτήματα είναι το συμπαγές μέγεθος, η μικρή κατανάλωση ρεύματος και η ποικιλία από διαθέσιμες εξόδους.

2.8.1.1 Αρχή Λειτουργίας

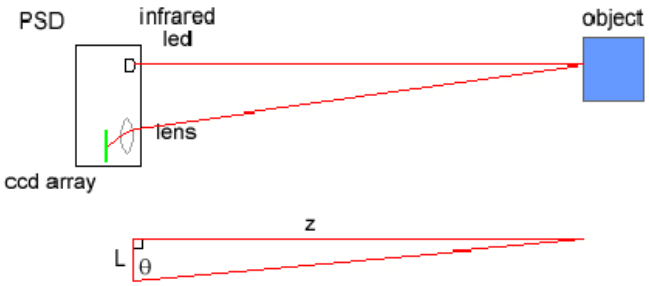
Βασίζεται στον τριγωνισμό (triangulation) και τη χρήση μιας σειράς αισθητήρων τύπου CCD στον ανιχνευτή θέσης PSD Η υπέρυθρη φωτοδίοδος εκπέμπει παλμικό σήμα μέτρησης, όπως στα απλά αισθητήρια υπερύθρων. Η ανάκλαση του σήματος σε κάποιο αντικείμενο, ανιχνεύεται στο PSD του αισθητηρίου, Δημιουργείται έτσι ένα τρίγωνο μεταξύ του πομπού, του σημείου πρόσπτωσης και του ανιχνευτή.

Η γωνία επιστροφής εξαρτάται από την απόσταση μεταξύ του αντικειμένου και του αισθητηρίου



Εικόνα 11: Αρχή λειτουργίας υπέρυθρου αισθητήρα.

Ο ανιχνευτής διαθέτει ένα φακό εστίασης, ο οποίος κατευθύνει το ανακλώμενο σήμα σε μια ενσωματωμένη γραμμική διάταξη αισθητήρων τύπου CCD στο PSD. Ανάλογα με το ποιο τμήμα της διάταξης αυτής ενεργοποιείται, μπορεί να υπολογιστεί η γωνία επιστροφής, άρα και η απόσταση του αντικειμένου



Εικόνα 12: Αρχή λειτουργίας υπέρυθρου αισθητήρα.

Το μοντάρισμα του αισθητήρα IR είναι το σημαντικότερο σημείο.

Ο αισθητήρας IR πρέπει να τοποθετείται σταθερά πάνω στο ρομπότ.

Ο προσανατολισμός του αισθητήρα είναι πολύ σημαντικός, καθώς καθορίζει την σωστή μέτρηση του αισθητήρα.

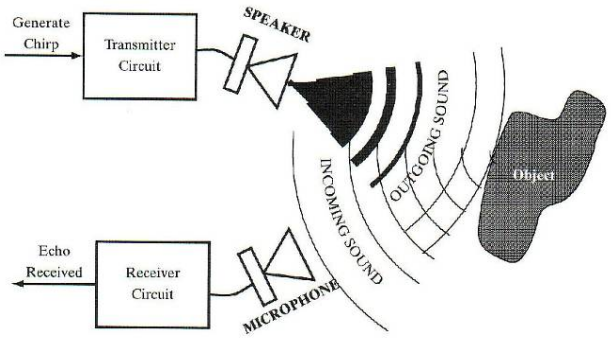
2.8.2 Αισθητήρια Υπερήχων (SONAR)

Εικόνα 13: Αισθητήρας υπερήχων.

Τα αισθητήρια υπερήχων (SONAR: Sound Navigation and Ranging) αναπτύχθηκαν αρχικά για υποβρύχια χρήση.
Η αρχή λειτουργίας τους βασίζεται στην εκπομπή και λήψη υπερήχων.
Απαντάται δε και στη φύση, στον ηχοεντοπισμό που χρησιμοποιούν δελφίνια και νυχτερίδες.
Τα αισθητήρια υπερήχων χρησιμοποιούνται σε σειρά εφαρμογών και στη βιομηχανία (π.χ για μέτρηση αποστάσεων, διαστασιομέτρηση, μέτρηση στάθμης κλπ.)

2.8.2.1 Υπολογισμός Αποστάσεων με Sonar

Τα sonar λειτουργούν με βάση την αρχή υπολογισμού χρόνου πτήσης (time of flight – TOF), η οποία μάλιστα απαντάται και στη φύση (νυχτερίδες και δελφίνια).
Αρχικά, ο ακουστικός μετατροπέας εκπέμπει ένα σύντομο σήμα υπερήχων (στη περιοχή των 50kHz). Το σήμα ανακλάται σε κάποιο εμπόδιο, επιστρέφει πίσω, όπου και ανιχνεύεται από κάποια διάταξη δέκτη.



Εικόνα 14: Αρχή λειτουργίας sonar.

Στη πράξη, τα αισθητήρια υπερήχων παρουσιάζουν αρκετά προβλήματα και περιορισμούς, Αναφορικά έχουμε τα εξής:

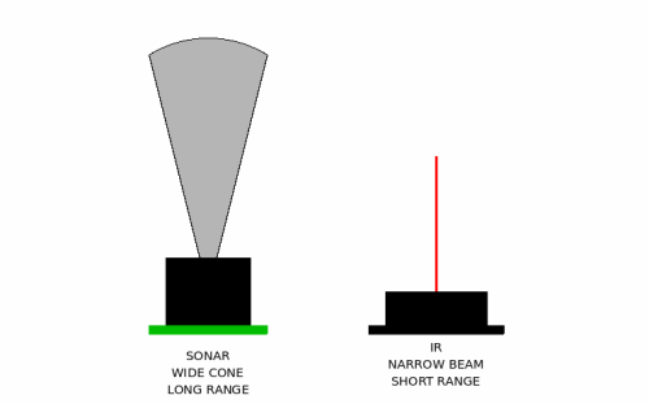
- Κωνικού σχήματος δέσμη του σήματος μέτρησης, με σημαντικό γωνιακό εύρος (10-30°)
- Προβλήματα ανακλάσεων σε κοινές επιφάνειες
- Σχετικά αργή ταχύτητα διάδοσης. Για παράδειγμα απαιτούνται 200 msec για τη διάσχιση 60 (=2x30) μέτρων.

Παρόλα αυτά, η χρήση τους σε εφαρμογές αυτοκινούμενων ρομπότ είναι ιδιαίτερα διαδεδομένη, καθώς σε συνδυασμό με την επαρκή ακρίβεια χαρακτηρίζονται από
χαμηλό κόστος, απλή χρήση, μικρό μέγεθος και αξιόπιστη λειτουργία, ιδιαίτερα σε σχέση με τις διατιθέμενες ολοκληρωμένες μονάδες

2.8.3 Αισθητήρας υπερύθρων εναντίον sonar

Οι αισθητήρες sonar καθώς και οι υπερύθρων, μας επιστρέφουν αναγνώσεις απόστασης.
Μεταξύ τους, όμως, υπάρχουν μερικές διαφορές.

Η αρχική διαφορά τους είναι ότι ο sonar έχει έναν ευρύ κώνο ανίχνευσης. Όσο απομακρύνεται το αντικείμενο από τον αισθητήρα, τόσο μεγαλώνει η γραμμή ανίχνευσης του κώνου.
Αυτό είναι ιδιαίτερα σημαντικό για την ανίχνευση των λεπτών αντικειμένων, όπως π.χ. τα πόδια μιας καρέκλας.
Οι αισθητήρες IR δεν μπορούν να ανιχνεύσουν ένα πόδι καρέκλας, εκτός κι αν είναι μπροστά σε αυτό, ή περνούν πολύ κοντά.



Εικόνα 15: Πεδίο ανειχνεφσης αισθητήρων IR, sonar.

Σε ένα κοντό ρομπότ η ακτίνα του σόναρ είναι λεπτή. Αυτό συνεπάγεται ότι η διάμετρος ανίχνευσης μπροστά από το ρομπότ είναι μικρότερη από το πλάτος του ρομπότ. Συμπεραίνουμε, άρα, πως δεν θα μπορέσει να ανιχνεύσει έγκαιρα το αντικείμενο, καθώς η ακτίνα του θα είναι πολύ μικρή για να το «πιάσει».

Η δεύτερη σημαντική διαφορά τους είναι η απόσταση.
Οι περισσότεροι IR (κυρίως η σειρά της sharp) αισθητήρες μπορούν να ανιχνεύσουν μεγαλύτερες και ακριβέστερες αποστάσεις .
Σημειώστε ότι και οι δυο τύποι αισθητήρων όταν ένα αντικείμενο είναι πάρα πολύ κοντά σε αυτούς μπορεί να εμφανιστεί πολύ μακριά ή να μην ανιχνευτεί καθόλου.

2.8.4 Αισθητήρια λείζερ μέτρησης απόστασης

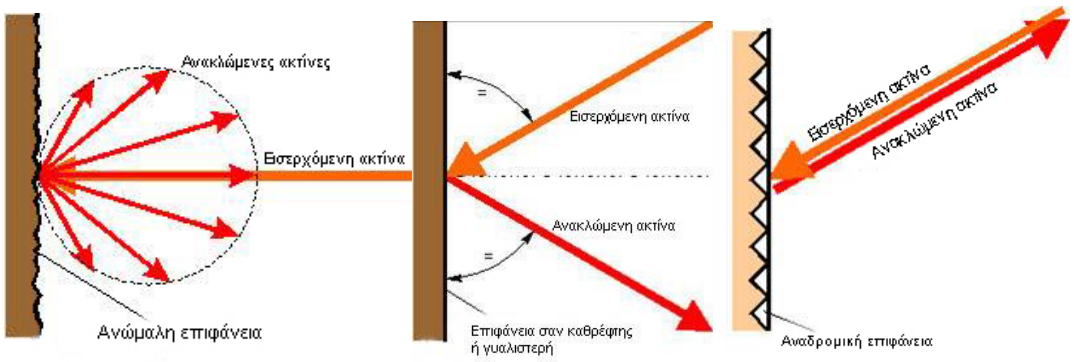
Τα αισθητήρια λείζερ μέτρησης απόστασης (laser range finders) ανήκουν στη κατηγορία των TOF (time of flight) αισθητήρων και χρησιμοποιούνται για την πλοήγηση και αποφυγή εμποδίων από το ρομπότ.
Αντί για υπέρηχους (όπως στα sonar), εκπέμπεται δέσμη ακτίνας laser και ανιχνεύεται η επιστροφή της μετά από ανάκλαση σε κάποιο αντικείμενο. Η μεγάλη ταχύτητα μετάδοσης του φωτός, επιτρέπει την σάρωση μεγάλου γωνιακού εύρους σε μικρό χρονικό διάστημα.

Εικόνα 16: Αισθητήρας λείζερ μέτρησης απόστασης.

Πλεονεκτήματα

- Μεγάλη εμβέλεια (από μέτρα έως και χιλιόμετρα, ανάλογα με την ισχύ του laser και την εφαρμογή)
- Αυξημένη ακρίβεια
- Λαχεία λειτουργία
- Κατάλληλα για μέτρηση αντικειμένων από μεγάλο εύρος υλικών
- Συγκριτικά φθηνή τεχνολογία (σε σχέση με τις παρερχόμενες δυνατότητες)

Τα περισσότερα laser μέτρησης απόστασης χρησιμοποιούν τη μέθοδο υπολογισμού χρόνου πτήσης (TOF). Ένα κύκλωμα παραγωγής ηλεκτρικών παλμών οδηγεί περιοδικά μια διάταξη ημιαγωγών, η οποία αποστέλλει μια ακολουθία από οπτικούς παλμούς (πάνω από το ορατό φάσμα), οι οποίοι συγκεντρώνονται σε μια πολύ εστιασμένη δέσμη, μέσω οπτικών διατάξεων. Η δέσμη μέτρησης ταξιδεύοντας με την ταχύτητα του φωτός, προσκρούει σε κάποιο αντικείμενο που βρίσκεται στην (ευθεία) διαδρομή της και ανακλάται πάνω του. Τμήμα της ανάκλασης συλλέγεται μέσω του φακού λήψης και ενεργοποιεί μια φωτοδιοδο, η οποία δημιουργεί ένα ηλεκτρικό σήμα, το οποίο ειδοποιεί την υπολογιστική μονάδα, προκειμένου να υπολογιστεί το χρονικό διάστημα μεταξύ αποστολής και λήψης.
Η αρχή μέτρησης είναι ανάλογη με αυτή που περιγράφηκε για το sonar.
Και στη περίπτωση του λέιζερ παίζει σημαντικό ρολό το είδος της επιφάνειας πρόσπτωσης.



Εικόνα 17: Είδη επιφανειών και ανάκλαση ακτίνας λέιζερ.

2.8.5 Τρισδιάστατοι Σαρωτές Laser

Οι Τρισδιάστατοι Σαρωτές λέιζερ (3D Laser scanner), είναι αυτή τη στιγμή η αιχμή της τεχνολογίας σε εφαρμογές τρισδιάστατης αποτύπωσης, χρησιμοποιώντας έναν 2D σαρωτή laser ο οποίος τοποθετείται σε βάση, η οποία επιτρέπει μεταβολή της κλίσης του, μέσω σερβομηχανισμού.

Ο τρισδιάστατος σαρωτής βρίσκει εφαρμογή και σε αυτοκινούμενα ρομπότ, για την ακριβή χαρτογράφηση των χώρων στους οποίους κινείται. Επιτρέπει έτσι την υλοποίηση πολύπλοκων αλγόριθμων πλοήγησης σε μη-γνωστά περιβάλλοντα.

Η επεξεργασία της πληθώρας των δεδομένων που παρέχει ο σαρωτής ανεβάζει σημαντικά τις απαιτήσεις σε υπολογιστική ισχύ, ταχύτητα και μνήμη για τον ελεγκτή κίνησης.

Βεβαία όπως κάθε νέα τεχνολογία το κόστος ενός τρισδιάστατου σαρωτή λέιζερ κυμαίνεται περίπου στα 8000€

Εικόνα 18: Τρισδιάστατος σαρωτής λέιζερ.

3 Ανάλυση πλατφόρμας και μηχανικών μερών του ρομπότ

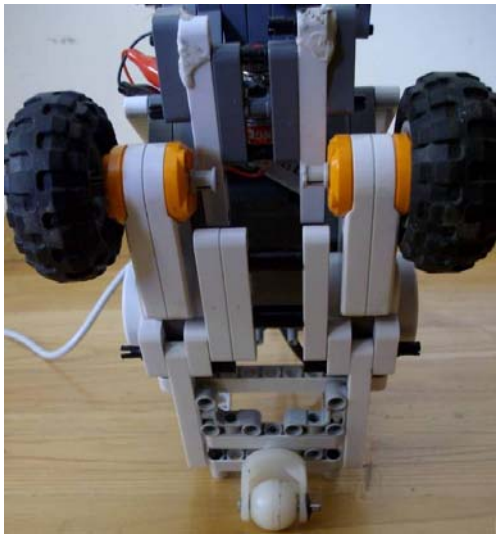
Σαν βασική πλατφόρμα του ρομπότ χρησιμοποιήθηκε το λεγόμενο tribot από το πακέτο Lego Mindstorm NXT στο οποίο γίνανε κάποιες κατάλληλες προσαρμογές προκείμενου να στηριχθεί η κάμερα, ο σαρωτής laser και μια κατασκευή στήριξης του καλωδίου της κάμερας. Η πλατφόρμα διαθέτει τρεις τροχούς, στους δύο εκ των οποίων είναι προσαρμοσμένοι δυο ανεξάρτητοι σερβοκινητήρες. Ο τρίτος τροχός κινείται ελεύθερα.

Με αυτή την κατασκευή το ρομπότ είναι σε θέση να κινείται προς οποιαδήποτε κατεύθυνση και να εκτελεί με ευκολία επιτόπου περιστροφή ή να ακολουθεί πορεία με απόκλιση αριστερά η δεξιά. Λόγω του μικρού μεγέθους του είναι κατάλληλο για στενούς χώρους.

Εικόνα 19: πλατφόρμα ρομπότ.

Η βάση στήριξης του καλωδίου εξυπηρετεί στο να κρατάει σε απόσταση το καλώδιο ώστε να εξαλειφτεί το ενδεχόμενο να μπλεχτεί με του τροχούς του ρομπότ κατά της επιτόπου περιστροφές.

Εικόνα 20: βάση στήριξης καλωδίου.



Εικόνα 21: Βάση του ρομπότ με τους τροχούς κίνησης.

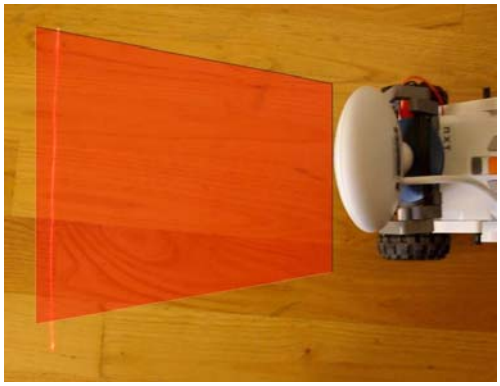
3.1 Βάση στήριξης κάμερας

Η κάμερα αποτελεί το βασικότερο κομμάτι του ρομπότ.

Γιαυτό χρειάστηκε μια κατασκευή η οποία να είναι σταθερή αλλά και να έχει τη δυνατότητα να ρυθμίζεται στην επιθυμητή θέση. Η βάση είναι τοποθετημένη στο πάνω μπροστινό μέρος του ρομπότ. Το οπτικό πεδίο της κάμερας καλύπτει σε μάκρος περίπου 25 εκατοστά μπροστά από το ρομπότ και πλάτος λίγο μεγαλύτερο από αυτό του ρομπότ, δηλαδή περίπου 15 εκατοστά.

Εικόνα 22: Βάση στήριξης κάμερας.

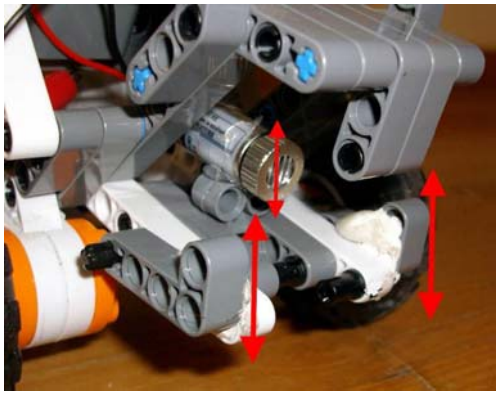
Με αυτή την κατασκευή διασφαλίστηκε ότι δεν θα αλλάζει η θέση της κάμερας και κατά συνέπεια το οπτικό της πεδίο.



Εικόνα 23: οπτικό πεδίο κάμερας.

3.2 Βάση στήριξης σαρωτή λέιζερ

Ένα άλλο κρίσιμο κομμάτι της κατασκευής είναι η βάση για τον σαρωτή Laser. Ο σαρωτής πρέπει να στοχεύει μπροστά από το ρομπότ με μία μικρή κλίση προς τα κάτω. Επίσης πρέπει να υπάρχει η δυνατότητα προσαρμογής της κλίσης στόχευσης προς όλες τις κατευθύνσεις (πάνω - κάτω καθώς και περιστρεφόμενα διαγώνια).



Εικόνα 24: Βάση στήριξης του σαρωτή λέιζερ.

4 Ανάλυση βασικών ιδεών

Σκοπός της έρευνας είναι να κατασκευαστεί ένα ρομπότ το οποίο να καθοδηγείται από τον υπολογιστή χρησιμοποιώντας ως αισθητήρα μία κάμερα σε συνδυασμό με ένα σαρωτή laser.

Θα πρέπει να είναι ικανό να αποφεύγει εμπόδια προκαθορισμένου ύψους πχ. 2 εκατοστών καθώς επίσης να αναγνωρίζει τη θέση της γραμμής στην οποία βρίσκονται τα αντικείμενα/εμπόδια στο χώρο που θα κινηθεί. Να αναγνωρίζει δηλαδή αν βρίσκονται αν

βρίσκονται δεξιά ή αριστερά έτσι ώστε να στρίψει για να αποφευχθεί η σύγκρουση του με αυτά.

Επιδιώκουμε τη λειτουργία του ρομπότ σε πραγματικό περιβάλλον. Να είναι ικανό δηλαδή να αποφεύγει αντικείμενα διαφορετικού σχήματος, μεγέθους, υλικού και υφής. Έτσι θα επιτύχουμε για παράδειγμα την αποφυγή του από γυάλινα αντικείμενα καθώς επίσης κι από ένα μάλλινο ύφασμα. Αυτός είναι ένας τομέας στον οποίο υστερούν ο αισθητήρας υπερήχων και ο υπερύθρων.

Τέλος θα πρέπει η επεξεργασία της εικόνας από τον αλγόριθμο να γίνεται σε πραγματικό χρόνο έτσι ώστε το ρομπότ να κινείται έγκαιρα. Να προλαβαίνει δηλαδή να αντιδράσει πριν συγκρουστεί με το εμπόδιο.

Ως ρομποτική πλατφόρμα χρησιμοποιήθηκε ένα τρίτροχο ρομπότ, το οποίο κατασκευάστηκε με τουβλάκια lego από το πακέτο lego mindstorms nxt. (αποτελεί μια παραλλαγή του tribot).

Σαν μικροελεγκτής χρησιμοποιήθηκε ένα lego mindstroms nxt.

Στο μπροστινό και πάνω μέρος του ρομπότ προσαρμόστηκε μια webcamera της οποίας το οπτικό πεδίο είναι περίπου 25εκατ. Σε μάκρος και 15εκατ. Σε πλάτος, μπροστά από το ρομπότ.

Υπάρχει επίσης ένας σαρωτής laser ο οποίος στοχεύει μέσα στο οπτικό πεδίο της κάμερας σε οριζόντια κατεύθυνση. Χρησιμοποιήθηκε το λογισμικό πρόγραμμα Matlab και ιδιαίτερα οι βιβλιοθήκες Image acquisition και Processing αυτού.

Η επεξεργασία γίνεται μέσω του υπολογιστή (όπου είναι συνδεδεμένη κι η κάμερα), με τον εξής τρόπο :

Από την κάμερα λαμβάνονται εικόνες οι οποίες στη συνέχεια αναλύονται επιδιώκοντας την αφαίρεση του backgroynd, έτσι ώστε να έχουμε ως μοναδική πληροφορία μόνο τα σημεία στα οποία προσπίπτει ο σαρωτής laser.

Με αυτό τον τρόπο επιτυγχάνεται μια τρισδιάστατη αναπαράσταση του χώρου μπροστά από το ρομπότ

Τα σημεία που προσπίπτει ο σαρωτής λέιζερ αυτά θεωρούνται ως εμπόδιο κάθε φορά που μετακινούνται κάτω ή πάνω από ένα προκαθορισμένο όριο pixel.

Ο υπολογιστής αποφασίζει για τις κινήσεις του ρομπότ και αποστέλλει τις κατάλληλες εντολές κίνησής του μέσω μιας ασύρματης επικοινωνίας bluetooth.

Κατά την εξέλιξη της έρευνας δοκιμάστηκαν διάφοροι τρόποι επίλυσης των προβλημάτων. Προσεγγίστηκαν οι στόχοι ύστερα από πολλές αλλαγές στους πιθανούς τρόπους επίλυσης των προβλημάτων.

Χρειάστηκε μάλιστα, κάποιες φορές, τόσο ο αλγόριθμος όσο και τα μηχανικά μέρη της κατασκευής να απέλθουν σε ριζικές αλλαγές.

Ο κάθε τρόπος αναφέρεται αναλυτικότερα στο κεφάλαιο 6

4.1 Κύρια προβλήματα

Στον παρακάτω πίνακα απεικονίζονται τα πιθανά προβλήματα και υποπροβλήματα που θα αντιμετώπιζα στην πορεία της έρευνας μου.

Έπαιξαν καθοριστικό ρόλο καθώς αποκτήθηκε η πρώτη αντίληψη ως προς τον τρόπο που έπρεπε να ακολουθηθεί για την επίτευξη του επιθυμητού αποτελέσματος.

Όσο εξελίσσεται η έρευνα επέρχεται λύση κάποιων καθώς και δημιουργία νέων προβλημάτων και υποπροβλημάτων, τα οποία αναλύονται στα συμπεράσματα κάθε κεφαλαίου και λύνονται διαδοχικά σε κάθε επόμενο κεφάλαιο.

- Διαχωρισμός των χρωμάτων και background με σκοπό την διατήρηση μόνο της κόκκινης γραμμής του σαρωτή λέιζερ.
    - Φωτισμό στο χώρο και αντανακλάσεις.
    - Κατά πόσο τα άλλα χρώματα έχουνε τιμές του κόκκινου χρώματος και απομόνωση αυτών.
    - Πως θα μπορεί να λειτουργεί ικανοποιητικά σε διάφορες επιφάνειες και υλικά (π.χ. γυαλί, μεταλλικά αντικείμενα, πορώδες υλικά κ.α.)
- Επαλήθευση αποτελεσμάτων μετά το διαχωρισμό.

4.2 Αρχική προσέγγιση λύσεων του προβλήματος

Σαν πρώτο και κύριο πρόβλημα αντιμετωπίζουμε τον διαχωρισμό της γραμμής laser από το background.

Σαν δεδομένα έχουμε:
Αρχικά ότι το χρώμα της γραμμής έχει έντονο κόκκινο χρώμα.
Από αυτό προκύπτει ότι ο διαχωρισμός θα είναι δύσκολος, καθώς πολλά χρώματα έχουν σαν βάση το χρώμα αυτό.
Το θετικό είναι ότι, τις περισσότερες φορές, η φωτεινότητα της γραμμής είναι εντονότερη συγκριτικά με αυτή των άλλων αντικειμένων.
Η λύση λοιπόν μπορεί να δοθεί με την διατήρηση μόνο των εντονότερων και φωτεινότερων κόκκινων σημείων. Αυτό συνεπάγεται πως σχεδόν πάντα θα διατηρείτε η γραμμή του λέιζερ, καθώς ήδη γνωρίζουμε πως αυτή διαθέτει εντονότερα τα παραπάνω χαρακτηριστικά.

4.2.1 Εξοικείωση του χρήστη με τις βασικές εντολές Matlab

Για να λάβουμε μια εικόνα στο Matlab πρέπει αρχικά να δούμε πώς αναγνωρίζεται από το Matlab η κάμερα και τι δυνατότητες κι επιλογές μας δίνει έπειτα από την αναγνώρισή της.

Στην συνέχεια έχοντας αναγνωρίσει την κάμερα κατασκευάζουμε ένα video-αντικείμενο, με το οποίο θα μπορούμε εύκολα και γρήγορα (με την χρήση των κατάλληλων εντολών) να αλλάξουμε τις ρυθμίσεις της κάμερας έτσι ώστε να απεικονίζεται με τον τρόπο που επιθυμούμε η εικόνα που μας επιστρέφεται.

```
imaqhwinfo
      ans =
            InstalledAdaptors: {'coreco'  'winvideo'}
            MATLABVersion: '7.7 (R2008b)'
            ToolboxName: 'Image Acquisition Toolbox'
            ToolboxVersion: '3.2 (R2008b)'
```

Με την εντολή imaqhwinfo επιστρέφονται οι συσκευές εικόνας που είναι εγκατεστημένες στον υπολογιστή.

```
vid = videoinput('winvideo',1,'RGB24_320x240');
```
δημιουργία ενός αντικειμένου με όνομα vid στο οποίο συνδέουμε την κάμερα winvideo με χρωματικό μοντέλο και ανάλυση επιστροφής RGB24_320x240.

```
preview(vid)
```
Προβολή σε ένα νέο παράθυρο το βίντεο που λαμβάνει η κάμερα.

```
Image=getsnapshot(vid);
```
Λήψη εικόνας από το vid και αποθήκευση αυτής στο πίνακα image.

```
imshow(image)
```
Προβολή της εικόνας σε ένα νέο παράθυρο.

Έχοντας φτιάξει τις ρυθμίσεις και έχοντας λάβει την εικόνα, είμαστε έτοιμοι να δούμε πώς θα την επεξεργαστούμε ώστε να λάβουμε τις πληροφορίες που θέλουμε.

4.2.2 Πειραματικές δοκιμές επίλυσης αρχικού προβλήματος

Βασιζόμενοι σε γνωστούς αλγόριθμους και τεχνικές θα προσπαθήσουμε να επιτύχουμε το επιθυμητό αποτέλεσμα.

Οι αλγόριθμοι και οι τεχνικές που θα πειραματιστούμε είναι οι εξής :

- Threshold
- Αλγόριθμος K-means
- Τεχνικές διαχωρισμού χρωμάτων.

4.2.2.1 Threshold

Το Threshold θεωρείται η απλούστερη μέθοδος κατάτμησης από μια grayscale[6]. Χρησιμοποιείται για τη δημιουργία δυαδικών εικόνων.

Μέθοδος

Κατά τη διάρκεια της διαδικασίας threshold, τα μεμονωμένα εικονοστοιχεία[7] σε μια εικόνα μπορεί να είναι χαρακτηρισμένα ως εικονοστοιχεία αντικείμενα αν η τιμή (ένταση) τους είναι μεγαλύτερη από κάποιο κατώτερο όριο ή ως εικονοστοιχεια υποβάθρου αν είναι κατώτερη. Αυτός ο τρόπος είναι γνωστός ως threshold ανώτερου ορίου.

Χαρακτηριστικά, σε ένα εικονοστοιχείο αντικειμένου δίνεται η τιμή "1" , ενώ σε ένα εικονοστοιχείο υποβάθρου δίνεται η τιμή "0".

Το αποτέλεσμα από το threshold, είναι μια δυαδική εικόνα με το χρωματισμό κάθε λευκού ή μαύρου εικονοστοιχείου, ανάλογα με την ετικέτα ενός εικονοστοιχείου.

Ξεκινώντας κάνουμε σύνδεση της εικόνας με το Matlab, ακολουθώντας την διαδικασία που αναφέραμε παραπάνω. (κεφάλαιο 4.2.1).

Παρακάτω παρουσιάζεται ο κώδικας του πειράματος καθώς και φωτογραφίες με τα αποτελέσματα.

Αφού δημιουργήσουμε το αντικείμενο, πάνω στο οποίο συνδέεται η κάμερα, ανοίγουμε νέο παράθυρο για την προεπισκόπηση της εικόνας.

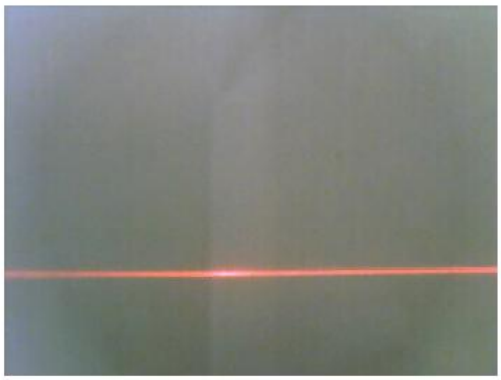
```
vid = videoinput('winvideo',1,'RGB24_320x240');
preview(vid)
```

λαμβάνουμε μια εικόνα από την κάμερα και την αποθηκεύουμε στην image. (εικόνα 25)

[6] Grayscale είναι μια εικόνα που έχει αποχρώσεις μόνο μεταξύ δυο χρωμάτων.

[7] εικονοστοιχειο (pixel) είναι το μικρότερο αντικείμενο πληροφορίας σε μια εικόνα.

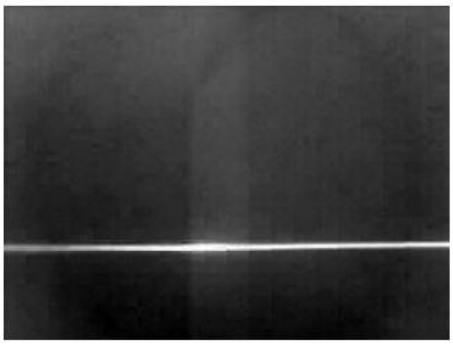
```
image=getsnapshot(vid);
```



Εικόνα 25: εικόνα που λάβαμε από την κάμερα.

```
r=image(:,:,1);
```

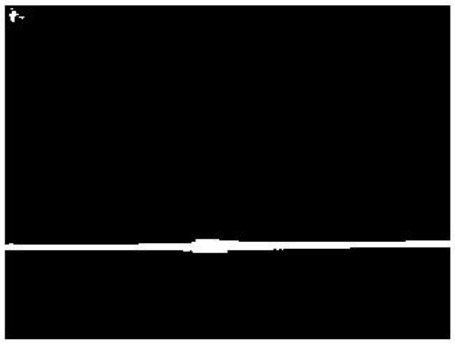
από την εικόνα (Image) λαμβάνουμε μόνο τον πρώτο πίνακα, στον οποίο βρίσκονται όλα τα στοιχεία των αντικειμένων που περιέχουν το κόκκινο χρώμα (εικόνα 26)



Εικόνα 26: μόνο το κανάλι του κόκκινου από την εικόνα image.

```
I = im2bw(r,0.6);
```

Με την εντολή im2bw εφαρμόζουμε threshold στην εικόνα r με κατώφλι 0.6. Δηλαδή εικονοστοιχεία από τιμές 0.6 και κάτω θα μηδενιστούν. Το αποτέλεσμα επιστρέφεται στο I. Το I είναι μια δυαδική εικόνα. (εικόνα 27)

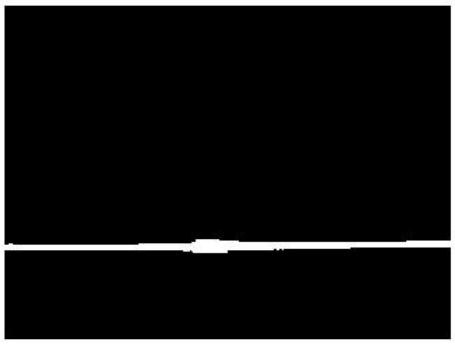


Εικόνα 27: αποτέλεσμα του threshold.

Παρατηρούμε, ενίοτε, ότι μετά το φιλτράρισμα της εικόνας με threshold, εκτός από τη γραμμή του λέιζερ, εμφανίζονται και άλλα σημεία τα οποία δεν θέλουμε.
Το πρόβλημα αυτό λύνεται εάν προσαρμόσουμε την τιμή του κατωφλιού, Θα πρέπει επίσης να χρησιμοποιηθεί ένας άλλος αλγόριθμος καθαρισμού ομάδων εικονοστοιχείων, όπως είναι ο bwareaopen.

Από την εικόνα I διαγράφουμε όλα τα αντικείμενα που αποτελούνται με λιγότερα από 60 εικονοστοιχεία [εικόνα 28].

```
I1 = bwareaopen(I,60);
```



Εικόνα 28: Εικόνα μετά την απομάκρυνση μικρών αντικειμένων.

Το τελικό αποτέλεσμα είναι ικανοποιητικό.
Παρά ταύτα, όπως διαπιστώνουμε παρακάτω, η συγκεκριμένη τεχνική δεν έχει τα ίδια αποτελέσματα σε όλες τις εικόνες.

Εικόνα 29α: Επανάληψη του πειράματος με διαφορετική εικόνα

Τα αποτελέσματα στην παραπάνω εικόνα δεν είναι ικανοποιητικά.
Αυτό συμβαίνει διότι υπάρχει μια ομάδα εικονοστοιχείων (στο πάνω αριστερό μέρος). Έχει διαγραφτεί, επίσης, πληροφορία από την γραμμή του λέιζερ (κάτω δεξιά). Παρακάτω [εικόνα **] δοκιμάζεται η ίδια εικόνα με διαφορετικές τιμές κατωφλιού του threshold.

Εικόνα 29β: Threshold με τιμή κατωφλιού 0.63.

Με τιμή κατωφλιού 0.63 απομακρύνεται ο θόρυβος (ο οποίος προκαλείτο από την αντανάκλαση του φωτός) από την εικόνα. Έχει, όμως, επίσης χαθεί πληροφορία από την γραμμή λέιζερ.

Εικόνα 30α: Threshold με τιμή κατωφλιού 0.6.

Τα αποτελέσματα δεν είναι καθόλου καλά. Αυτό οφείλεται στις αντανακλάσεις του φωτός. Στην παρακάτω εικόνα [εικόνα **] έχουμε την βέλτιστη ρύθμιση της τιμής του κατωφλιού κι άρα κατά συνέπεια τα βέλτιστα αποτελέσματα.

Εικόνα 30β: Threshold με τιμή κατωφλιού 0.68

Όπως διαπιστώνεται από την παραπάνω εικόνα, δεν υπάρχει δυνατότητα να επιτευχθεί καλύτερο αποτέλεσμα χρησιμοποιώντας τη συγκεκριμένη τεχνική.

4.2.2.2 Αλγόριθμος k-means

Ο αλγόριθμος k-means χρησιμοποιείτε συνήθως στην τεχνική όραση ως μορφή κατάτμησης εικόνας. Τα αποτελέσματα της κατάτμησης χρησιμοποιούνται στην ανίχνευση συνόρων και την αναγνώριση αντικειμένων. Σε αυτό το πλαίσιο, η τυποποιημένη Ευκλίδια απόσταση είναι συνήθως ανεπαρκής στη διαμόρφωση των συστάδων (κοινές ομάδες στοιχείων).
Αντ' αυτού, ένα σταθμισμένο μέτρο απόστασης χρησιμοποιεί τις συντεταγμένες εικονοστοιχείου, το RGB χρώμα ή/και την ένταση εικονοστοιχείου.

Εφαρμογές του αλγορίθμου

Ο αλγόριθμος K-means είναι μια επαναληπτική τεχνική που χρησιμοποιείται για να χωρίσει μια εικόνα στις συστάδες K.

Ο βασικός αλγόριθμος είναι:

1. Επιλέγονται K συστάδες, τυχαία ή σύμφωνα με άλλες τεχνικές.
2. Ορίζετε κάθε εικονοστοιχείο στην εικόνα στη συστάδα που ελαχιστοποιεί τη διαφορά μεταξύ του εικονοστοιχείου και του κέντρου συστάδων
3. Επαν-υπολογίζονται τα κέντρα συστάδων με τον υπολογισμό του μέσου όρου όλων των εικονοστοιχείων στη συστάδα
4. Επαναλαμβάνονται τα βήματα 2 και 3 έως ότου επιτυγχάνεται η σύγκλιση (π.χ. κανένα εικονοστοιχείο δεν αλλάζει συστάδες)

Μετά από πειραματισμούς διαπίστωσα ότι έχουμε καλύτερα αποτελέσματα όταν ορίζουμε τον k-means να μας επιστρέφει τέσσερις συστάδες.

Η διαδικασία ξεκινάει, όπως συνήθως, με την σύνδεση της κάμερας, με το Matlab. (με την διαδικασία που αναλύσαμε παραπάνω) (κεφάλαιο 4.2.1).

Παρακάτω ακολουθεί ο κώδικας του πειράματος καθώς και φωτογραφίες των αποτελεσμάτων.

Αφού δημιουργήσουμε το αντικείμενο, πάνω στο οποίο συνδέεται η κάμερα, ανοίγουμε νέο παράθυρο για την προεπισκόπηση της εικόνας

```
vid = videoinput('winvideo',1,'RGB24_320x240');
preview(vid)
```

Λαμβάνουμε μια εικόνα από την κάμερα και την αποθηκεύουμε στην image. (εικόνα 31)

```
image=getsnapshot(vid);
```

Είμαστε έτοιμοι να περάσουμε την εικόνα στον αλγόριθμο k-means.

Παρακάτω ακολουθεί ο κώδικας του k-means, στον οποίο θα εξηγήσουμε μόνο τα σημεία τα οποία μας αφορούν.

```
%% Color-Based Segmentation Using K-Means Clustering
% Your goal is to segment colors in an automated fashion using the
L*a*b* color
% space and K-means clustering.
% This demo requires Statistics Toolbox(TM).
% Copyright 1993-2007 The MathWorks, Inc.

%% Step 1: Read Image
```

```
tic
```

Με την εντολή tic ξεκινάμε έναν μετρητή, ο οποίος τερματίζεται με την εντολή toc, η οποία θα μπει στο τέλος του αλγόριθμου, προκειμένου να μας επιστρέψει τον συνολικό χρόνο που χρειάστηκε για να εκτελεστεί ο αλγόριθμος.

```
he = image;
```

```
%% Step 2: Convert Image from RGB Color Space to L*a*b* Color Space
cform = makecform('srgb2lab');
lab_he = applycform(he,cform);

%% Step 3: Classify the Colors in 'a*b*' Space Using K-Means
Clustering
ab = double(lab_he(:,:,2:3));
nrows = size(ab,1);
ncols = size(ab,2);
ab = reshape(ab,nrows*ncols,2);

nColors = 4;
```

Στη μεταβλητή nColors αποδίδουμε τον ακέραιο αριθμό των ομάδων / συστάδων που θέλουμε να μας επιστραφεί από τον αλγόριθμο.

```
% repeat the clustering 3 times to avoid local minima
[cluster_idx                    cluster_center]                    =
kmeans(ab,nColors,'distance','sqEuclidean', ...
                                    'Replicates',3);

%% Step 4: Label Every Pixel in the Image Using the Results from
KMEANS
pixel_labels = reshape(cluster_idx,nrows,ncols);

%% Step 5: Create Images that Segment the H&E Image by Color.
segmented_images = cell(1,3);
rgb_label = repmat(pixel_labels,[1 1 3]);

for k = 1:nColors
    color = he;
    color(rgb_label ~= k) = 0;
    segmented_images{k} = color;
```

```
end

s1=segmented_images{1};
s2=segmented_images{2};
s3=segmented_images{3};
s4=segmented_images{4};
```

Στις μεταβλητές s1, s2, s3 και s4 αποθηκεύουμε την κάθε ξεχωριστή ομάδα / συστάδα.

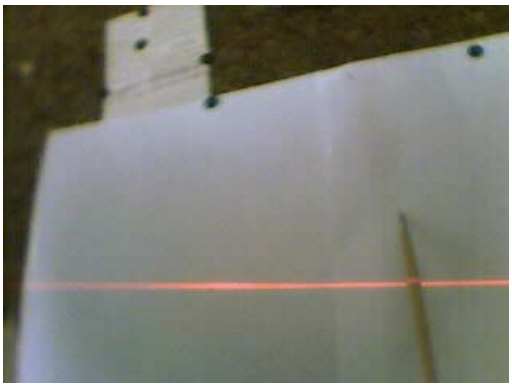
```
figure; imshow(s1)
figure; imshow(s2)
figure; imshow(s3)
figure; imshow(s4)
```

Προβάλουμε κάθε εικόνα σε νέο παράθυρο.

```
x=toc
```

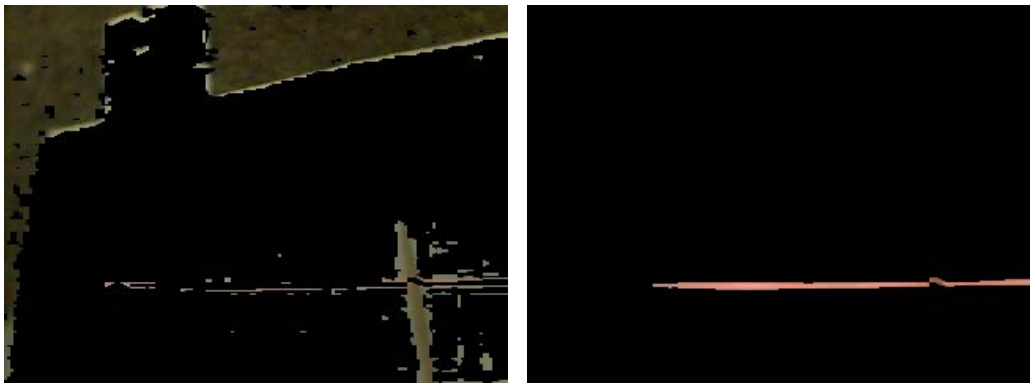
Η toc μας επιστρέφει τον συνολικό χρόνο που χρειάστηκε ο αλγόριθμος.

Τα αποτελέσματα του αλγόριθμου παρουσιάζονται στις παρακάτω εικόνες.



Εικόνα 31: Αρχική εικόνα

Εικόνα 32: Πρώτη και δεύτερη ομάδα μετά τον διαχωρισμό.



Εικόνα 33: Τρίτη και τέταρτη ομάδα μετά τον διαχωρισμό

Τα αποτελέσματα είναι πολύ θετικά.

Το πρόβλημα είναι, βέβαια, ότι ο αλγόριθμος χρειάζεται κατά μέσο όρο **4.6 δευτερόλεπτα.** Χρόνος υπερβολικά μεγάλος για μία εικόνα και για να μπορέσει να γίνει η επεξεργασία και να λειτουργήσει το ρομπότ σε πραγματικό χρόνο.

4.2.2.3 Άλλοι αλγόριθμοι-τεχνικές.

Πέραν των προαναφερθέντων έγιναν πολλά πειράματα με διαφορετικές τεχνικές και αλγόριθμους, οι οποίοι είτε θέλανε πολύ χρόνο είτε τα αποτελέσματα τους δεν ήτανε ικανοποιητικά ή και τα δυο.

Αναφορικά:

- Επεξεργασία της εικόνας πριν την χρήση αλγορίθμου threshold.
- Προσπάθεια επαλήθευσης και διόρθωσης της εικόνας από τυχόν προβλήματα μετά το threshold.

- Αλγόριθμος *colorDetectHSV* ο οποίος είναι δημοσιευμένος στην σελίδα του Matlab και δημιουργήθηκε από τον Θεόδωρο Γιαννακόπουλο.
- Αλγόριθμος *Colour Filter* ο οποίος είναι δημοσιευμένος στην σελίδα του Matlab και δημιουργήθηκε από τον *K.Kannan & Jeny Rajan, Medical Imaging Research Group (MIRG), NeST, Trivandrum.*

4.3 Συμπεράσματα.

Τα αποτελέσματα δεν μας ικανοποιούν απόλυτα.
Η μοναδική τεχνική που πλησιάζει στους στόχους μας (κεφάλαιο 4.2.2.1) είναι αυτή του threshold.
Σύμφωνα με τα παραπάνω διαπιστώνουμε ότι θα πρέπει να κατασκευαστεί ένας αλγόριθμος ο οποίος να κάνει threshold και να λύσει τα προβλήματα που παρουσιάστηκαν.
Θα πρέπει δηλαδή να ικανοποιεί τα εξής ζητούμενα:

- Να λειτουργεί σε ικανοποιητικό χρόνο επεξεργασίας (λιγότερο από 1 δευτερόλεπτο κάθε εικόνα).
- Τα αποτελέσματα να μην μεταβάλλονται εάν μεταβάλλεται το περιβάλλον.
- Να έχει δυνατότητες περεταίρω βελτίωσης.
- Να έχει δυνατότητες παραμετροποίησης, δυναμικής και μη.

5 Αλγόριθμος HSVCLEAR

Από τα συμπεράσματα που προέκυψαν ύστερα από πειράματα (κεφάλαιο 4.2.2) , δημιουργήθηκαν κάποια ζητούμενα τα οποία πρέπει να ικανοποιεί ο αλγόριθμος HSVCLEAR. Ο αλγόριθμος θα τροφοδοτείτε με διαδοχικές εικόνες από την κάμερα.
Έχει ως βάση την τεχνική threshold, με τη διαφορά ότι θα πρέπει να προσαρμόζει την τιμή του κατώτερου ορίου. Με αυτό τον τρόπο θα έχουμε το σωστό threshold σε κάθε εικόνα που λαμβάνουμε ενώ κινείται το ρομπότ. Για παράδειγμα όταν εισέρχεται από ένα σκοτεινό σε ένα φωτεινό περιβάλλον ή όταν ενώ στρίβει έρχεται σε οπτική επαφή με αντανακλάσεις φωτός. Άρα δεν θα μεταβάλλονται τα αποτελέσματα εάν μεταβάλλεται το περιβάλλον του.

Η διαδικασία επιλογής της τιμής του ορίου βασίστηκε στο δεδομένο ότι η γραμμή του λείζερ έχει την υψηλότερη τιμή έντασης (φωτεινότητα), καθώς επίσης και στο ότι η γραμμή είναι κόκκινη.

Με αυτά τα δεδομένα μπορούμε να αποκλίσουμε ένα μεγάλο μέρος πληροφορίας από την εικόνα. Στα αρχικά πειράματα (κεφάλεο 4.2.2) χρησιμοποιήθηκε σαν χρωματικό

μοντέλο το RGB, το οποίο δεν είναι κατάλληλο εφόσον πρέπει να ασχοληθούμε με συγκεκριμένο χρώμα. Έτσι επιλέξαμε την χρήση του χρωματικού μοντέλου HSV ο οποίος και θεωρείται η καλύτερη λύση όπως θα εξηγήσουμε και στο παρακάτω (κεφαλαίο 5.1).

Επισημαίνεται ότι δεν χρησιμοποιήθηκε η εντολή im2bw του Matlab για threshold, αλλά κατασκευάστηκε κώδικας, προκειμένου να έχουμε τον πλήρη έλεγχο και την δυνατότητα προσαρμογής του αλγόριθμου.

Παρακάτω γίνεται μια θεωρητική αναφορά στους χρωματικούς χώρους ή μοντέλα και ακολουθεί η ανάλυση του αλγορίθμου HSVCLEAR.

5.1 Χρωματικοί χώροι ή μοντέλα

Στην αυτόματη ανάλυση εικόνας, όπως στην περίπτωσή μας, το χρώμα αποτελεί ένα πολύ ισχυρό χαρακτηριστικό όσον αφορά την περιγραφή, και απλοποιεί την αναγνώριση αντικειμένων και την εξαγωγή τους από μια σκηνή που περιέχει διαφόρων ειδών αντικείμενα. Η χρωματική πληροφορία είναι ένα απ' τα βασικότερα στοιχεία που θα αξιοποιήσουμε, οπότε είμαστε υποχρεωμένοι να χρησιμοποιήσουμε έγχρωμες εικόνες, έστω και αν αυτό αυξάνει την πολυπλοκότητα του προβλήματος μας. Στο υποκεφάλαιο αυτό θα εξετάσουμε τον τρόπο αναπαράστασης έγχρωμων ψηφιακών εικόνων, ο οποίος βασίζεται στην χρήση χρωματικών χώρων.

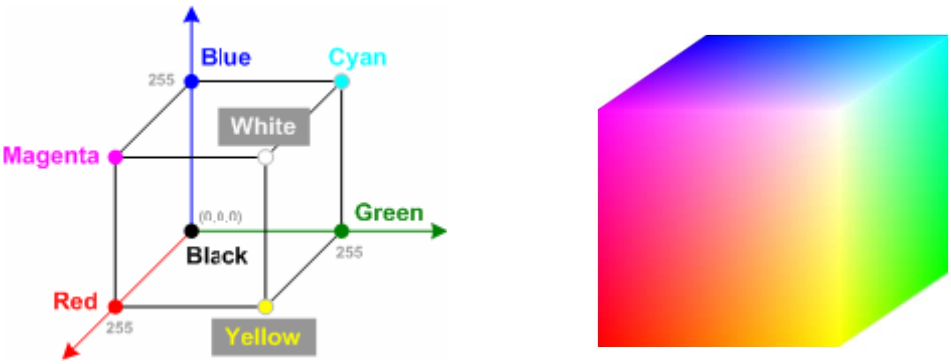
Με τον όρο χρωματικός χώρος εννοούμε ένα μοντέλο που χρησιμοποιείται για την αναπαράσταση του χρώματος με τη μορφή τιμών έντασης. Ο χρωματικός χώρος ορίζει τον τρόπο με τον οποίο αναπαρίσταται η χρωματική πληροφορία. Σκοπός του είναι να διευκολύνεται ο ορισμός των χρωμάτων στα πλαίσια της τυποποίησης. Στην ουσία, ένα χρωματικό μοντέλο είναι ένα τρισδιάστατο σύστημα συντεταγμένων και ένα υποσύστημα μέσα σε αυτό όπου κάθε χρώμα αναπαρίσταται από ένα σημείο. Σήμερα, τα περισσότερα μοντέλα είναι προσαρμοσμένα στα φυσικά εξαρτήματα υπολογιστικών συστημάτων (π.χ. οθόνες και εκτυπωτές) ή σε εφαρμογές όπου είναι επιθυμητή η διαχείριση των χρωμάτων (π.χ. graphic animation).

Στην πρώτη κατηγορία ανήκουν το μοντέλο RGB (Red, Green,Blue) για έγχρωμες οθόνες και κάμερες, το μοντέλο CMY (Cyan, Magenta, Yellow) για έγχρωμους εκτυπωτές και το μοντέλο YIQ (luminance, Inphase, Quadrature) που είναι το πρότυπο για την τηλεοπτική μετάδοση. Στη δεύτερη κατηγορία έχουμε το μοντέλο HSI (Hue, Saturation, Intensity) και το HSV (Hue, Saturation, Value). Στη συνέχεια παρουσιάζουμε τους
ευρύτερα χρησιμοποιούμενους χρωματικούς χώρους και τα χαρακτηριστικά τους.

5.1.1 Χρωματικός χώρος RGB

Στον χρωματικό χώρο RGB το κάθε χρώμα εκφράζεται συναρτήσει των τριών βασικών χρωμάτων, του κόκκινου, του πράσινου και του μπλε και βασίζεται σε ένα καρτεσιανό σύστημα συντεταγμένων. Κάθε χρώμα μπορεί να αναπαρασταθεί από μια μίξη συγκεκριμένων τιμών έντασης σε καθεμία από τις παραπάνω χρωματικές συνιστώσες. Οι διάφορες χρωματικές αποχρώσεις μπορούν να αναπαρασταθούν σε σύστημα τριών διαστάσεων αν φανταστούμε ότι κάθε άξονας αντιστοιχεί σε μια χρωματική συνιστώσα.

Επομένως ο RGB χώρος μπορεί να αναπαρασταθεί από ένα κύβο, όπως φαίνεται στην εικόνα 34

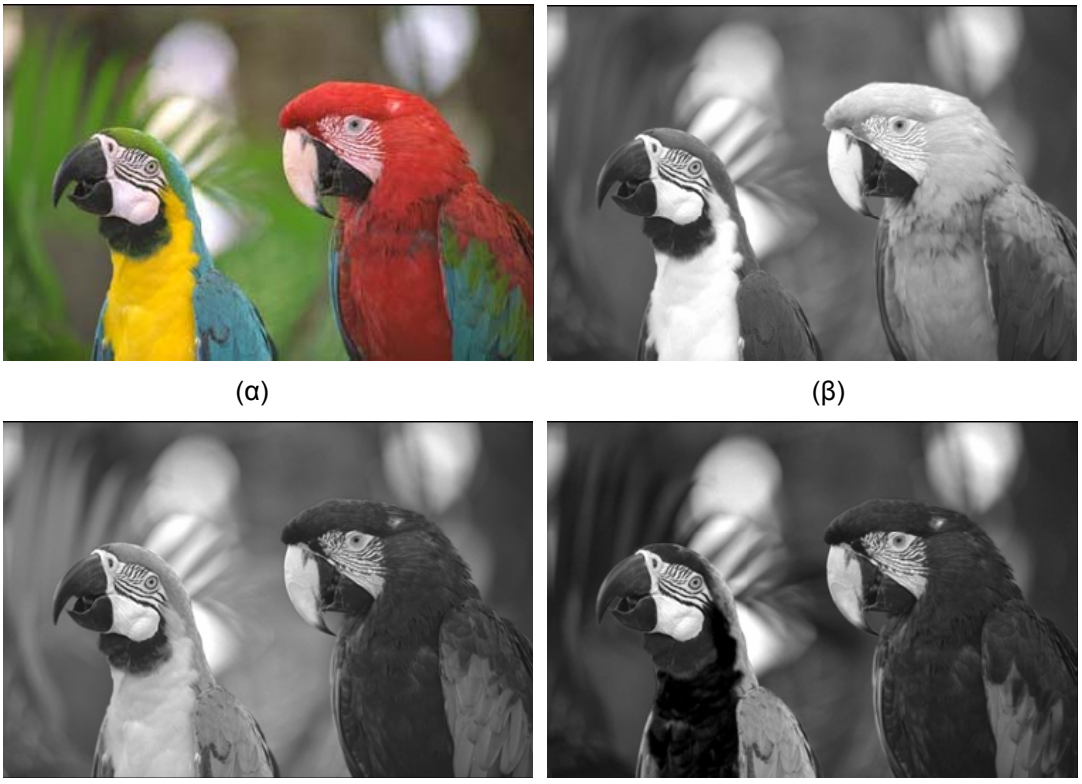


Εικόνα 34: Σχηματική απεικόνιση του RGB χρωματικού κύβου.

Στην εικόνα 34 φαίνονται τα κύρια χρώματα (κόκκινο, πράσινο, μπλέ) που βρίσκονται πάνω στους άξονες. Το κυανό, το ματζέντα και το κίτρινο βρίσκονται στις άλλες τρείς κορυφές, ενώ το μαύρο είναι στην αρχή των αξόνων και το άσπρο στην απώτατη κορυφή απ' την αρχή. Τα επίπεδα του γκρί είναι οι τιμές που εκφράζει το ευθύγραμμο τμήμα απο την αρχή των αξόνων (μαύρο) μέχρι το άσπρο. Ένα οποιοδήποτε άλλο σημείο μέσα ή πάνω στον κύβο εκφράζει κάποιο χρώμα.

Στο Matlab η αναπαράσταση μιας RGB εικόνας διαστάσεων M x N γίνεται με έναν πίνακα τριών διαστάσεων M x N x 3 που περιέχει pixel χρώματος.

Κάθε pixel είναι μια τριπλέτα χρωμάτων που αντιστοιχεί στις συνιστώσες του κόκκινου, του πράσινου και του μπλέ για το συγκεκριμένο σημείο. Μια RGB εικόνα, δηλαδή, μπορεί να αναπαρασταθεί ως μια "στοίβα" απο τρείς gray-scale εικόνες οι οποίες όταν τροφοδοτηθούν στις εισόδους για κόκκινο, πράσινο και μπλέ μιας έγχρωμης οθόνης παράγουν μια έγχρωμη εικόνα. Οι τρείς επιμέρους εικόνες που αποτελούν την RGB εικόνα αναφέρονται ως χρωματικές συνιστώσες.



(α) (β)

(γ) (δ)

Εικόνα 35: Οι χρωματικές συνιστώσες του μοντέλου RGB (α) Αρχική εικόνα, (β) Κόκκινη συνιστώσα, (γ) Πράσινη συνιστώσα, (δ) Μπλε συνιστώσα.

5.1.2 Χρωματικός χώρος HSV

Το χρωματικό μοντέλο HSV (Hue, Saturation, Value) εκμεταλλεύεται τον τρόπο που οι άνθρωποι αντιλαμβάνονται το χρώμα.

Συγκεκριμένα συνηθίζεται να περιγράφουμε τις διάφορες σκηνές, όχι σε συνθήκες κόκκινου, πράσινου και μπλε, αλλά ως απόχρωση, καθαρότητα και ένταση. Βλέπουμε τα πράγματα ως χρώματα ή αποχρώσεις, οι οποίες έχουν είτε μια “ξεπλυμένη” όψη, είτε βαθύ και έντονο χαρακτήρα.

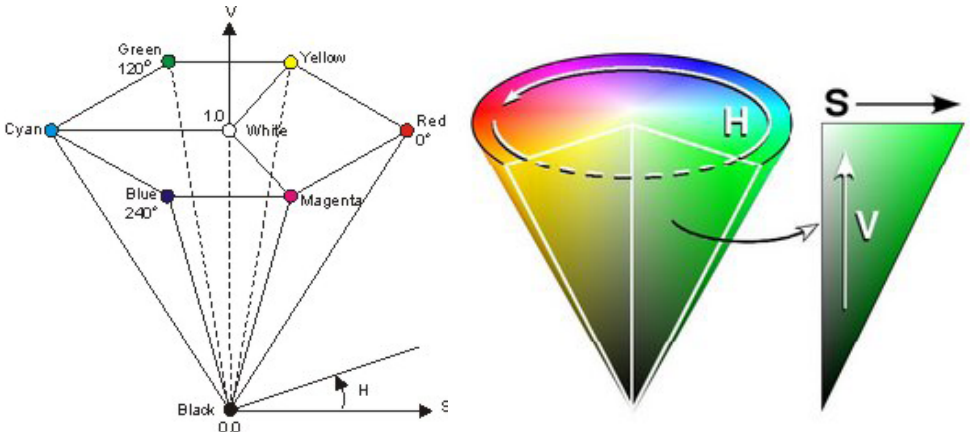
Το Hue (απόχρωση) είναι το χρώμα που γίνεται αντιληπτό λόγω του μήκους κύματος. Το Saturation (καθαρότητα) είναι ο βαθμός καθαρότητας του χρώματος, δηλαδή το κατά πόσο το χρώμα έχει πρόσμιξη λευκού μέσα. Το Value (τιμή) αναφέρεται στο βαθμό μίξης ενός καθαρού χρώματος με το μαύρο.

Το σύνολο των τριών αυτών ιδιοτήτων μπορεί να παράγει οποιοδήποτε χρώμα βρίσκεται στη φύση. Η τρισδιάστατη αναπαράσταση του HSV προκύπτει από τον κύβο RGB.

Αν κοιτάξουμε στον RGB κύβο κατά μήκος της διαγωνίου του γκρί, μπορούμε να δούμε ένα εξάγωνο, το οποίο είναι το HSV εξάγωνο. Η απόχρωση δίνεται από τη γωνία με τον οριζόντιο άξονα με το κόκκινο στις 0°, το κίτρινο στις 60°, το πράσινο στις 120°, το κυανό στις 180°, το μπλε στις 240° και το ματζέντα στις 300°.

Να σημειωθεί ότι τα συμπληρωματικά χρώματα έχουν 180° διαφορά. Η χρωματική καθαρότητα κυμαίνεται μεταξύ 0.0 ≤ S ≤ 1.0 και είναι ο λόγος της καθαρότητας μιας συγκεκριμένης απόχρωσης προς τη μέγιστη καθαρότητα (S=1). Όταν S=0 βρισκόμαστε στην κλίμακα του γκρί, δηλαδή στη διαγώνιο του RGB κύβου.

Για την επιλογή ενός χρώματος διαλέγουμε αρχικά μια καθαρή απόχρωση (καθορίζουμε δηλαδή την τιμή του H και θέτουμε S=V=1). Στη συνέχεια προσθέτοντας μαύρο μειώνουμε την τιμή του V και προσθέτοντας άσπρο μειώνουμε το S.



(α) (β)

Εικόνα 36: Σχηματική αναπαράσταση του χρωματικού χώρου HSV (α) HSV εξάγωνο, (β) Χρωματική άποψη του HSV εξαγώνου.

Ο μετασχηματισμός από RGB σε HSV είναι μη γραμμικός, είναι όμως αντιστρεπτός.

Ο ψευδοκώδικας μετασχηματισμού από RGB σε HSV και αντίστροφα παρουσιάζεται παρακάτω:

(α)

```
Max = max(R,G,B);
Min = min(R,G,B);

Delta = Max - Min;
V = Max;

If (Max==0)
      S = 0;
else
      S = (Max - Min)/Max;

if (Max == Min)
      H = 0;
else if (R==Max & G>=B)
      H = 60 × (G-B)/Delta;
else if (R(i,j)==Max & G(i,j)<B(i,j))
      H = 360 + 60 × (G-B)/Delta;
else if (G==Max)
      H = 60 × (2 + (B-R)/Delta);
else
      H = 60 × (4 + (R-G)/Delta);
```

(β)

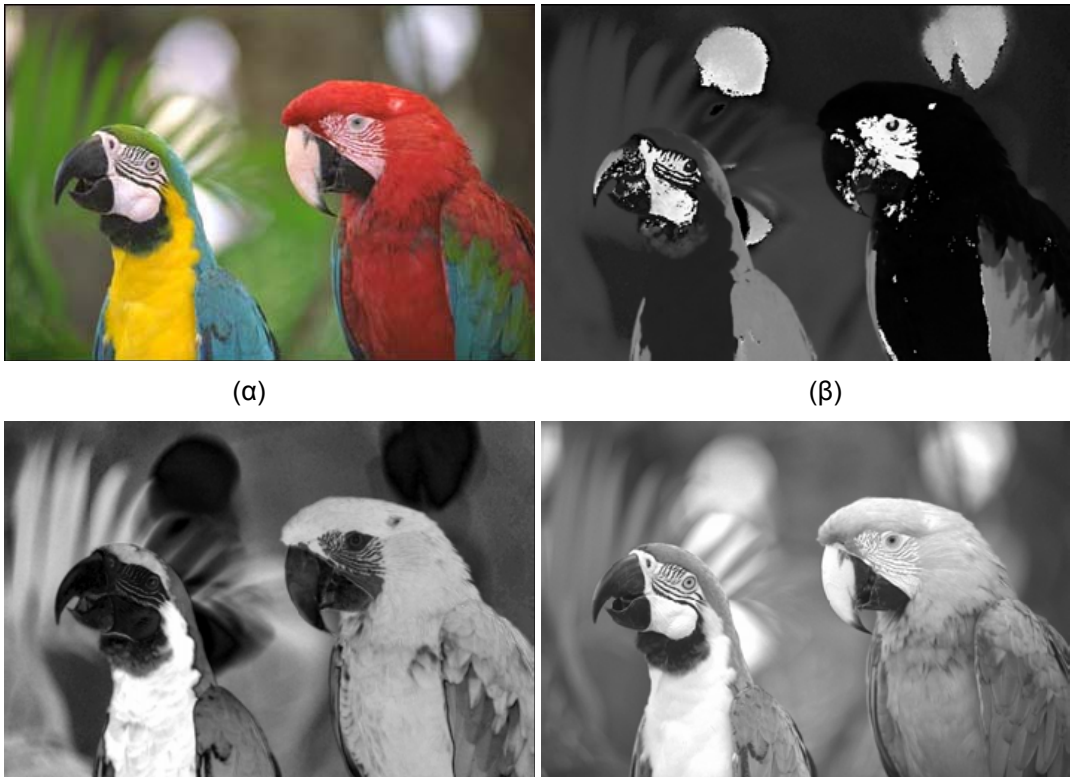
```
Hi = mod6( floor(H/60) )
f = H/60 – floor(H/60)
p = V × (1 - S)
q = V × (1 - f×S)
t = V × (1 - (1-f) ×S)

(R, G, B) = (V, t, p) if Hi =0

(q, V, p) if Hi =1

(p, V, t) if Hi =2

(p, q, V) if Hi =3

(t, p, V) if Hi =4

(V, p, q) if Hi =5
```

A) Ψευδοκώδικας μετασχηματισμού RGB→HSV

B) Ψευδοκώδικας μετασχηματισμού HSV→RGB

Το HSV παρουσιάζει δυο βασικά πλεονεκτήματα. Πρώτον η τιμή V είναι ανεξάρτητη από το χρώμα και δεύτερον η απόχρωση H και η χρωματική καθαρότητα S είναι στενά συσχετισμένες με τον τρόπο αντίληψης του χρώματος από το ανθρώπινο μάτι.

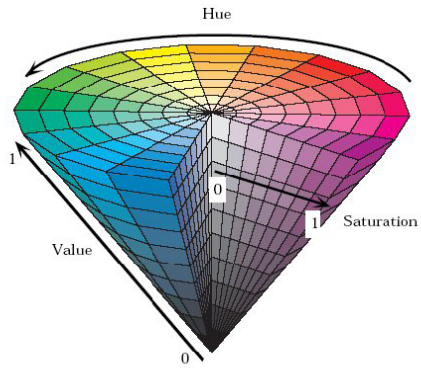
Αυτά τα χαρακτηριστικά καθιστούν το μοντέλο HSV ιδανικό εργαλείο για την ανάπτυξη αλγορίθμων επεξεργασίας εικόνας βασισμένων στην αίσθηση χρώματος από το ανθρώπινο οπτικό σύστημα.



(α) (β)

(γ) (δ)

Εικόνα 37: Οι χρωματικές συνιστώσες του μοντέλου HSV, (α) Αρχική εικόνα, (β) Απόχρωση (hue), (γ) Καθαρότητα (Saturation), (δ) Τιμή (Value)



Εικόνα 38: Αναλυτική αναπαράσταση του εξαγώνου HSV και χρωματική αποψή του.

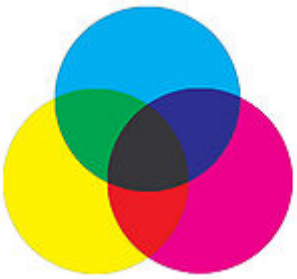
5.1.3 Αλλοι χρωματικοί χώροι.

Χρωματικό μοντέλο CMYK

Στην εκτύπωση των εντύπων χρησιμοποιείται ευρέως το σύστημα CMYK.

Τα τρία βασικά χρώματα στο CMY είναι: Γαλάζιο (Cyan) – Ματζέντα (Magenta) – Κίτρινο (Yellow). Με τα τρία αυτά χρώματα δημιουργούνται τα δευτερογενή Κόκκινο – Πράσινο – Μπλε ως εξής:

- Κόκκινο: Ματζέντα + Κίτρινο
- Πράσινο: Κίτρινο + Γαλάζιο
- Μπλε: Γαλάζιο + Ματζέντα



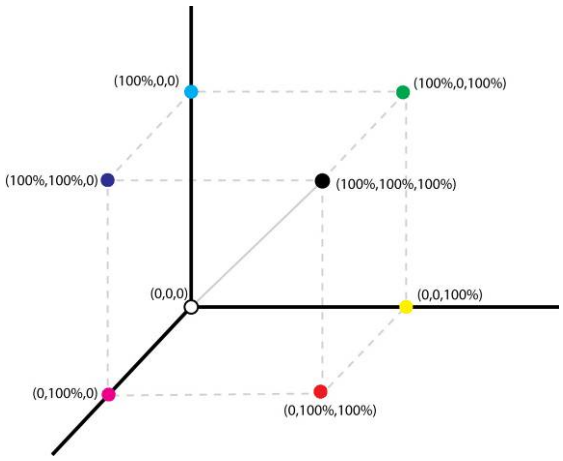
Εικόνα 39: Βασικά και δευτερογενή χρώματα.

Το μοντέλο αυτό βασίζεται στο γεγονός ότι το υπόβαθρο της εκτύπωσης είναι το λευκό χαρτί που ανακλά όλα τα χρώματα (μήκη κύματος).

Κάθε βασικό χρώμα που προστίθεται με ένα μελάνι απορροφά ορισμένα χρώματα και αποδίδει τα υπόλοιπα. Για παράδειγμα το κίτρινο μελάνι απορροφά το μπλε χρώμα και αφήνει το πράσινο και το κόκκινο να ανακλαστεί. Εδώ ο συνδυασμός των τριών βασικών χρωμάτων δίνει το μαύρο χρώμα (πλήρης απορρόφηση των ακτινοβολιών).

Για το λόγο αυτό το μοντέλο CMY χαρακτηρίζεται και ως "αφαιρετικό".

Μαύρο χρώμα, επίσης, προκύπτει από το συνδυασμό ενός βασικού και του αντίθετου δευτερογενούς:



Εικόνα 40: Γραφική απεικόνιση του χρωματικού μοντέλου CMY

- Μαύρο: Γαλάζιο + Ματζέντα + Κίτρινο

- Μαύρο: Γαλάζιο + Κόκκινο
- Μαύρο: Ματζέντα + Πράσινο
- Μαύρο: Κίτρινο + Μπλε

Τα μελάνια, όμως, από τη φύση τους δεν μπορούν να αποδώσουν συγκεκριμένα μήκη κύματος – χρώματα (όπως τα εικονοστοιχεία (pixels) μίας οθόνης) αλλά, μία πιο ευρεία περιοχή του χρωματικού φάσματος.
Το αποτέλεσμα είναι ο συνδυασμός των τριών βασικών χρωμάτων να δίνει ένα καφετί χρώμα αντί για το μαύρο. Για το λόγο αυτό προστέθηκε στο μοντέλο CMY και το μαύρο μελάνι, με αποτέλεσμα να προκύψει το χρωματικό μοντέλο CMYK (Cyan – Magenta – Yellow – Black). Πρακτικά στην εκτύπωση δεν χρησιμοποιείται σήμερα το CMY μοντέλο αλλά το CMYK. Το μοντέλο CMY μπορεί να παρασταθεί όπως και το RGB με ένα κύβο σε ένα καρτεσιανό σύστημα αξόνων με το λευκό χρώμα στην αρχή των αξόνων και τα βασικά χρώματα επάνω στους άξονες.

Χρωματικό μοντέλο Lab

Το μοντέλο αυτό αναλύει το χρώμα του κάθε εικονοστοιχείου σε τρεις παραμέτρους L,a,b:

- Ένταση φωτεινότητας (Luminocity).
- Χρωματική θέση ανάμεσα στο κόκκινο και το συμπληρωματικό του πράσινο (a).
- Χρωματική θέση ανάμεσα μπλε και το συμπληρωματικό του κίτρινο (b).

Στην πραγματικότητα το μοντέλο αποτελείται από δυο επιμέρους μοντέλα, το CIE 1976 και το Hunter. Το πρώτο χρησιμοποιεί συντεταγμένες "κυβικής ρίζας" ($\sqrt[3]{x}$), ενώ το δεύτερο "τετραγωνικής ρίζας" ($\sqrt{x}$). Πιο διαδεδομένο είναι το μοντέλο CIE. Η διαβάθμιση του είναι από τις πλησιέστερες προς την ανθρώπινη αντίληψη.

Χρωματικό μοντέλο RYB

Όπως το RGB αλλά με χρήση Κίτρινου αντί για Πράσινο.

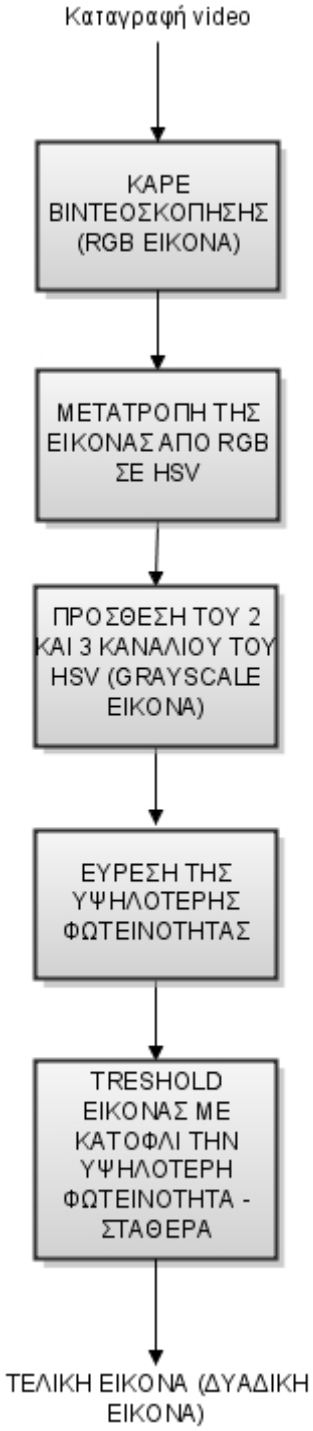
Χρωματικό μοντέλο HSL

Όπως το HSV αλλά με διαφορετικό ορισμό του κορεσμού και της έντασης. Έτσι εδώ χρησιμοποιείται η φωτεινότητα (Luminocity), αντί της λαμπρότητας. Επίσης οι τιμές αναφοράς για μαύρο γκρι κλπ είναι διαφορετικές.

5.2 Ανάλυση βασικού αλγόριθμου HSVCLEAR

Ένα πρωτότυπο του αλγόριθμου HSVCLEAR παρουσιάζεται παρακάτω.
Ο αλγόριθμος μετατρέπει τις εικόνες που εισέρχονται από RGB σε HSV, με χρήση κάποιων ποσοστών συμμετοχής των δύο πινάκων της HSV που μας ενδιαφέρουν κι έτσι παράγουμε μια Grayscale εικόνα, την οποία και επεξεργαζόμαστε. Στην εικόνα αυτή είναι πιο ξεκάθαρη η γραμμή του λείζερ.

Στη συνέχεια βρίσκουμε την μεγαλύτερη τιμή έντασης της εικόνας. Η τιμή αυτή αφαιρείτε με μια σταθερά και αποτελεί την τιμή του κατωφλιού για το threshold. Οποιεσδήποτε άλλες τιμές μικρότερες από την τιμή του κατωφλιού μηδενίζονται ενώ οι τιμές μεγαλύτερες από αυτό γίνονται ίσες με ένα. Σαν αποτέλεσμα έχουμε μια δυαδική εικόνα.



Εικόνα 41: Διάγραμμα ροής πρωτότυπου αλγόριθμου HSVCLEAR

Κώδικας Matlab

```
function [imageout] = hsvClear(imagein)
```

Ο αλγόριθμος δέχεται ως είσοδο μια RGB εικόνα (imagein) και μας επιστρέφει μια binary εικόνα (imageout).

```
hsvOrig=rgb2hsv(imagein);
```
Η εικόνα imagein μετατρέπετε από RGB σε HSV και αποθηκεύετε στην hsvOrig

```
hsv3=hsvOrig(:,:,3);
```
Από την εικόνα hsvOrig παίρνουμε μόνο τον 3^{ο} πίνακα (δηλαδή την ένταση) και τον αποθηκεύουμε στην hsv3

```
hsv2=hsvOrig(:,:,2);
```
Από την εικόνα hsvOrig παίρνουμε μόνο τον 2^{ο} πίνακα (δηλαδή την καθαρότητα) και τον αποθηκεύουμε στην hsv2

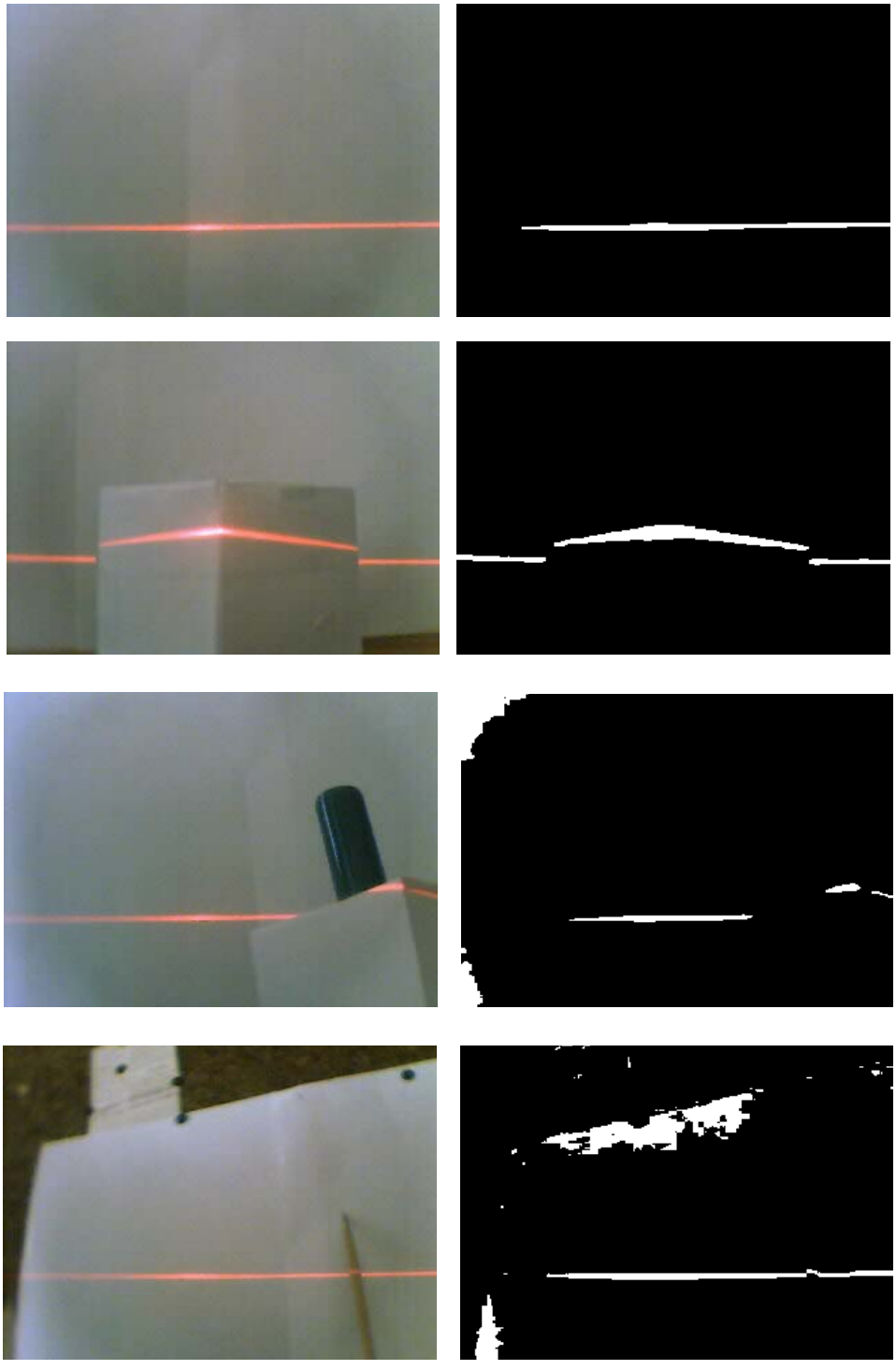
```
hsv=(hsv2 * 0.4)+(hsv3*0.8);
```
Πολλαπλασιάζουμε την κάθε εικόνα (hsv2, hsv3) με ένα ποσοστό και στη συνέχεια προσθέτουμε τα δύο γινόμενα. Η εικόνα hsv2 έχει λιγότερη σημασία από τον hsv3, γιαυτό το λόγο την πολλαπλασιάζουμε με το ανάλογο ποσοστό. Ως αποτέλεσμα έχουμε μια grayscale εικόνα που αποτελείται από έναν συνδυασμό δυο πινάκων της HSV εικόνας.

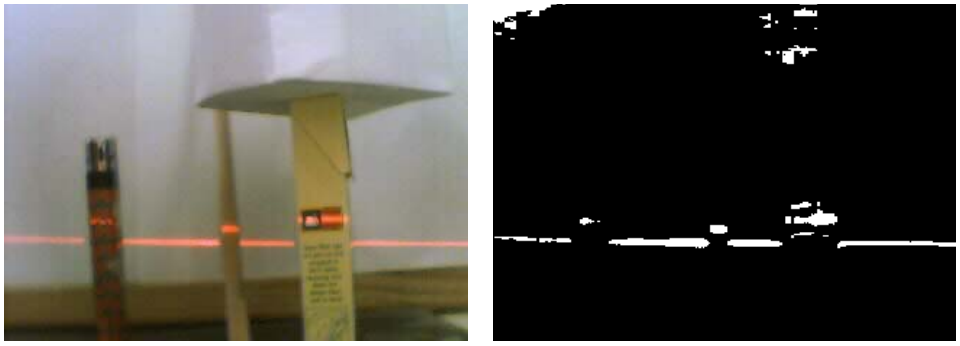
```
maxPix=max(max(hsv(:,80:110)));
```
Σε όλο τον άξονα του Y και από την τιμή 80 μέχρι 110 του άξονα X ερευνούμε για την μεγαλύτερη τιμή έντασης. Την τιμή την αποθηκεύουμε στην μεταβλητή maxPix, η τιμή αυτή είναι μια τιμή double από 0 έως 1.

```
out=(hsv > maxPix - 0.26);
```
Ερευνάτε όλη η εικόνα hsv (εικονοστοιχείο προς εικονοστοιχείο). Όποια τιμή είναι μικρότερη από την τιμή του maxPix όταν αυτό αφαιρείται από μια σταθερά (0.26) μηδενίζετε, όποια είναι μεγαλύτερη γίνεται ίση με ένα.

```
imageout=out;
```
Επιστρέφουμε την δυαδική εικόνα out στην εικόνα imageout και τερματίζει ο αλγόριθμος μας.

Ακολουθούν παραδείγματα του αλγόριθμου πάνω σε εικόνες.
Από αριστερά είναι η αρχική εικόνα και από δεξιά ακολουθεί το αποτέλεσμα του αλγόριθμου.



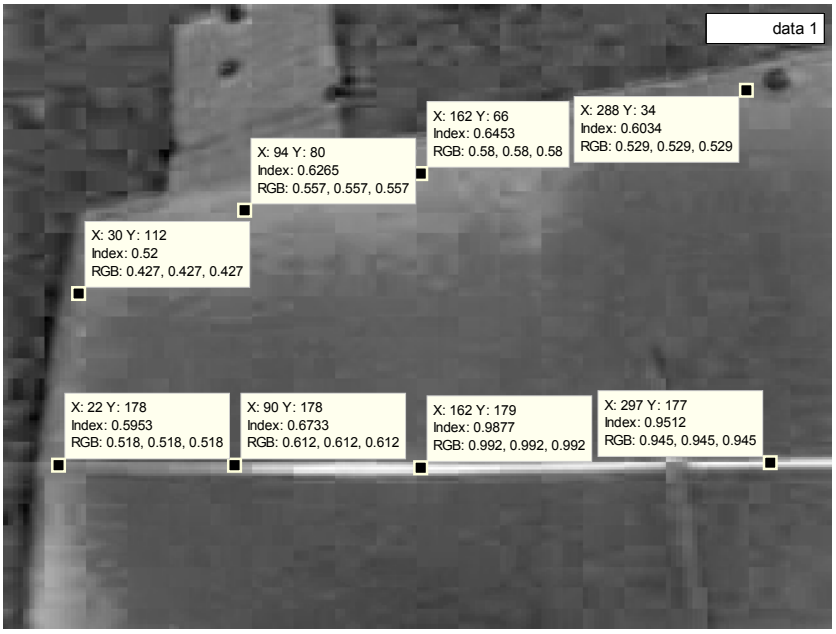


Μετά από τα παραπάνω πειράματα διαπιστώσαμε ότι στις εικόνες, στις οποίες υπήρχαν έντονες διακυμάνσεις στον φωτισμό ο αλγόριθμος έχει πρόβλημα.(όπως είχε και ο threshold με την εντολή im2bw). Βέβαια καταφέραμε να κάνουμε threshold με τον ίδιο αλγόριθμο σε όλες τις εικόνες.

Από την φύση της η γραμμή του λέιζερ είναι φωτεινότερη στο κέντρο, οπότε βάση αυτής της διαπίστωσης ο παραπάνω αλγόριθμος θα μας επιστρέφει σαν μεγαλύτερη τιμή έντασης(κατώφλι), την τιμή στο κέντρο της γραμμής.

Αντίθετα στις περισσότερες εικόνες, οι φωτεινότερες περιοχές διακρίνονται στα άκρα της εικόνας (εικόνα 42). Άρα η τιμή του κατωφλιού δεν είναι κατάλληλη για όλα τα σημεία της εικόνας.

Στην παρακάτω εικόνα παρουσιάζεται μια δειγματοληψία των τιμών έντασης, κι αποτελεί την τελική εικόνα του αλγόριθμου προ threshold.



Εικόνα 42: Δειγματοληψία σημείων φιλτραρισμένης εικόνας HSV.

Από τα αποτελέσματα που λαμβάνονται (παραπάνω εικόνα) προκύπτει ποιες είναι οι τιμές του Index που προέρχονται από την αντανάκλαση(οι οποίες είναι και οι παραπλανητικές) και

ποιες οι αληθινές τιμές που προκύπτουν από την πραγματική πρόσκρουση της ακτίνας του λέισερ πάνω στο αντικείμενο - εμπόδιο. Συγκρίνοντας μία προς μία τις τιμές κάθετα, αντιλαμβανόμαστε ότι κάνοντας threshold ξεχωριστά για κάθε τμήμα εικόνας, τα αποτελέσματά μας είναι καλύτερα.
Αυτό συμβαίνει γιατί η γραμμή του λέιζερ έχει διαφορετικές τιμές index στα άκρα της, όπου κι εξασθενούν.

Από τα παραπάνω καταλήγουμε ότι πρέπει να χωρίσουμε την εικόνα σε τμήματα και να κάνουμε threshold ανεξάρτητα για κάθε τμήμα εικόνας.

Επίσης ο αλγόριθμος θα πρέπει με κάποιο τρόπο να ελέγχει τα αποτελέσματα, για να αποτρέψει τυχόν εμφάνιση άλλων σημείων εκτός της γραμμής λέιζερ,
Για παράδειγμα στην εικόνα 42 Συγκρίνοντας τη διαφορά του index (φωτεινότητα) της γραμμής λέιζερ με το background στην αριστερή πλευρά, είναι μικρότερη από αυτή στην μέση της εικόνας. Κατά συνέπεια, η τιμή του κατωφλιού θα πρέπει να είναι ανάλογα προσαρμοσμένη έτσι ώστε να περιορίζει την εμφάνιση άλλων σημείων.
Αυτό μπορεί να γίνει, αθροίζοντας τον αριθμό των εικονοστοιχείων που είναι ίσα με “1”, σε κάθε τμήμα (μετά το threshold). Αν το άθροισμα είναι μεγαλύτερο από ένα προκαθορισμένο όριο, η τιμή του κατωφλιού θα αυξάνετε και θα ξαναγίνετε η διαδικασία του threshold. Αν είναι μικρότερο, τότε η τιμή του κατωφλιού μειώνεται. Η διαδικασία αυτή επαναλαμβάνεται μέχρι τα αποτελέσματα να είναι μες στα όρια.
Στο παρακάτω κεφάλαιο παρουσιάζουμε μια βελτιωμένη έκδοση του HSVCLEAR.

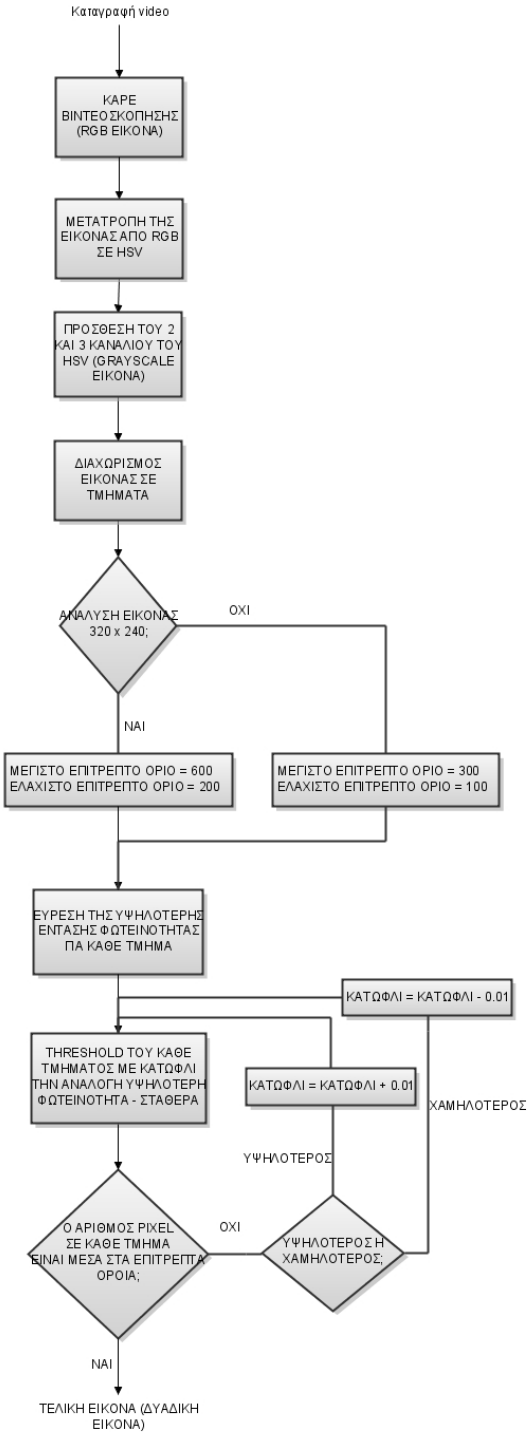
5.3 Προσαρμοστικός HSVCLEAR

Στο κεφάλαιο 5 παρουσιάσαμε μια πρώτη προσέγγιση του αλγόριθμου HSVCLEAR.
Σε αυτό το κεφάλαιο θα παρουσιάσουμε μια πιο εξελιγμένη έκδοση του αλγόριθμου, σε πραγματικό περιβάλλον, και θα πειραματιστούμε σε αυτόν.

Προσαρμοστικό threshold καλείται το Threshold όταν χρησιμοποιείται ένα διαφορετικό κατψφλί (κατώτατο όριο) για κάθε διαφορετικές περιοχές στην εικόνα. επίσης ονομάζετε και ως τοπικό ή δυναμικό threshold.
Η βασική παράμετρος στη διαδικασία είναι η επιλογή της τιμής του κατωφλιού (κατώτερου ορίου).

Η έκδοση του συγκεκριμένου αλγόριθμου εξελισσότανε σε όλη την πορεία της έρευνας και ιδιαίτερα όταν αυτός δοκιμάστηκε σε συνεργασία με το ρομπότ και με την κίνηση αυτού.
Οι κυριότερες βελτιώσεις που επήλθαν για τη σωστή λειτουργία του HSVCLEAR είναι οι εξής:

- Δοκιμάστηκαν κάμερες διαφόρων τύπων, ώστε να επιλεχθεί η καλύτερη για την κάλυψη των αναγκών μας. Η ανάγκη αυτή δημιουργήθηκε διότι η κάμερα που χρησιμοποιήσαμε αρχικά αντιμετώπισε μερικά προβλήματα, κυρίως στις αντανακλάσεις καθώς επίσης και στις μειωμένες δυνατότητες ρύθμισής της.
- Με τη χρήση του Matlab επιτύχαμε να αυξήσουμε τον κορεσμό, να μειώσουμε την φωτεινότητα και να θέσουμε στο χειροκίνητο την αυτόματη διόρθωση φωτεινότητας της κάμερας. Με αυτό τον τρόπο καταφέραμε οι εικόνες που επιστρέφονται από την κάμερα να έχουν εντονότερη τιμή φωτεινότητας στη γραμμή του λέιζερ και χαμηλότερη στον υπόλοιπο φωτισμό.
- Οι εικόνες που εισέρχονται στον αλγόριθμο έχουνε υποστεί προεπεξεργασία, Συγκεκριμένα, έχουν αφαιρεθεί περιοχές που δεν ήταν αξιοποιήσιμες και η επεξεργασία τους καθυστερούσε τον αλγόριθμο.
- Προσαρμοστικό threshold. Η εικόνα χωρίζεται σε τμήματα (ο αριθμός των τμημάτων ορίζεται από τον χρήστη). Σε κάθε τμήμα υπολογίζεται ανεξάρτητα η τιμή του κατωφλιού και κατά συνέπεια το threshold. Έτσι επιτεύχθηκε η ακρίβεια των αποτελεσμάτων.
- Επαλήθευση των αποτελεσμάτων και διόρθωση όπου απαιτείται.
- Οι εκάστοτε τιμές του κατωφλιού (για το κάθε τμήμα), αποθηκεύονται και χρησιμοποιούνται για την επόμενη εικόνα. Έτσι γλιτώνουμε χρόνο για επόμενο υπολογισμό της τιμής του κατωφλιού. Επίσης ο αλγόριθμος προσαρμόζεται από περιβάλλον σε περιβάλλον σταδιακά.
- Αναπροσαρμογή του κώδικα με τέτοιο τρόπο, ώστε να εκτελείται γρηγορότερα (vectorization)



Εικόνα 43: Διάγραμμα ροής εξελιγμένου HSVCLEAR.

Κώδικας Matlab

Προετοίμασα κάμερας

```
vid = videoinput('winvideo',2,'RGB24_320x240');
src = getselectedsource(vid);
src.Brightness=3;
src.Saturation=3;
```

```
set(src,'WhiteBalanceMode','manual');
```
Αύξηση του κορεσμού, μείωση φωτεινότητας και θέσει στο χειροκίνητο την αυτόματη διόρθωση φωτεινότητας της κάμερας.

Εικόνα 44: Εικόνα μετα από τις ρυθμίσεις της κάμερας.

Προεπεξεργασία εικόνας

```
image=imcrop(image, [0 0 Res(1) 150]);
```
Αποκοπή 150 εικονοστοιχείων από το κάτω μέρος της εικόνας.

Εικόνα 45: Μετά την αποκοπή 150 εικονοστοιχείων.

```
image(:,:,2)=zeros;
image(:,:,3)=zeros;
image = double(image)/255;
```
Ο πίνακας 2 και 3 από την RGB εικόνα “image” μηδενίζονται με αποτέλεσμα να παραμένει μόνο ο πίνακας 1 (πίνακας του κόκκινου). Έπειτα η εικόνα “image” μετατρέπεται σε double.

```
[imageout, apokS]=hsvClearSmart(image, apokS, 5);
```
Εντολή για το κάλεσμα του αλγόριθμου HSVCLEAR. Τα ορίσματα του είναι: η RGB εικόνα εισόδου (image), ο πίνακας από τον οποίο τροφοδοτείται ο αλγόριθμος, με τιμές των κατωφλιών για κάθε τμήμα (apokS), καθώς και ο αριθμός των τμημάτων (5). Ο αλγόριθμος επιστρέφει την τελική δυαδική εικόνα (imageout) και τον νέο πίνακα των τιμών των κατωφλιών (apokS).

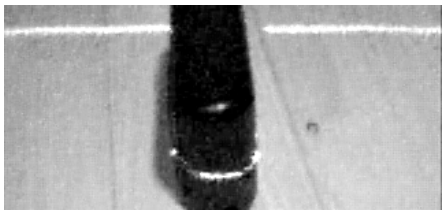
Αλγόριθμος HSVCLEAR

```
function [imageout, apokS] = hsvClearSmart(imagein, aS, segments)
```
Ορισμός του αλγόριθμου και των ονομάτων των μεταβλητών του.

```
hsvOrig=rgb2hsv(imagein);
```
Μετατροπή της εικόνας “imagein” από RGB σε HSV και αποθήκευση αυτής στην “hsvOrig”.

```
hsv3=hsvOrig(:,:,3);
hsv2=hsvOrig(:,:,2);
```
Αποθήκευση του πίνακα 2 και 3 από την εικόνα “hsvOrig” στις μεταβλητές “hsv2” και “hsv3”.

```
hsv=(hsv2 * 0.2)+hsv3*0.7;
```
Πολλαπλασιάζουμε την κάθε εικόνα “hsv2” και “hsv3” με ένα ποσοστό και προσθέτουμε τα γινόμενα τους. Η εικόνα “hsv2” έχει λιγότερη σημασία από την “hsv3” γι’ αυτό το λόγο την πολλαπλασιάζουμε με το ανάλογο ποσοστό, ως αποτέλεσμα έχουμε μια grayscale εικόνα που αποτελείται από τον συνδυασμό των δυο πινάκων της HSV εικόνας.



Εικόνα 46: Τελική εικόνα HSV.

```
[y,xx,e] = size(imagein);
```
Η “size” μας επιστρέφει τις διαστάσεις της εικόνας. Το “y” είναι η διάσταση στον άξονα Y, το “xx” είναι η διάσταση στον άξονα X και το “e” είναι ο αριθμός των πινάκων που αποτελούν την “imagein”.

```
seg=xx/segments;
```
Διαιρούμε τον άξονα X “xx” με των αριθμό των τμημάτων “segments”. Από το πηλίκο τους προκύπτει η απόσταση του κάθε τμημάτος.

```
xMax=1:y;
x=xMax;
```
Δημιουργούμε ένα πίνακα από “1” μέχρι την τιμή του άξονα Y “y” και τον εναποθέτουμε στην μεταβλητή “xMax”.

```
if xx==160
    PanoOrio=300;
    KatoOrio=100;
else
    PanoOrio=600;
    KatoOrio=200;
End
```

Ορίζουμε τα επιτρεπόμενα όρια του αθροίσματος των εικονοστοιχείων ανάλογα με το μέγεθος της εικόνας.

```
maxPix(segments)=zeros;
s(segments)=zeros;
```
Μηδενίζουμε τους πίνακες “maxPix” και “s”.

```
for i=0:segments-1
    maxPix(i+1)=max(max(hsv(xMax,(seg*i)+1:seg*(i+1))));
end
```
Βρίσκουμε την φωτεινότερη τιμή (index) για κάθε τμήμα. Οι τιμές αποθηκεύονται στον πίνακα “maxPix”.

```
apoklisi=0.15;
```
Σταθερά απόκλισης.

```
out=zeros(y, xx);
```
Δημιουργούμε ένα πίνακα “out” με της διαστάσεις της αρχικής εικόνας. Ο “out” είναι και η τελική δυαδική εικόνα.

```
for f=0:segments-1
```
Επανάληψη “segments” φορές (δηλαδή όσα είναι και τα τμήματα), προκειμένου να υπολογιστή το threshold και επαλήθευση αυτού.

```
    out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))           >
maxPix(f+1) - (apoklisi + aS(f+1)) );
```
Έλεγχος εικονοστοιχείο προς εικονοστοιχείο για κάθε τμήματος της εικόνας “hsv”. ‘Οταν η τιμή του εικονοστοιχείου είναι μεγαλύτερη από το “maxPix” (δηλαδή την τιμή κατωφλιού) του εκάστοτε τμήματος τότε ισούται με “1” αλλιώς ισούται με “0”. Τα αποτελέσματα αποθηκεύονται στην ανάλογη θέση στην εικόνα “out”.

Εικόνα 47 Τελική δυαδική εικόνα (μετά από όλες τις επαναλήψεις) χωρίς να έχει επαληθευθεί.

```
    s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
```
Αθροίζετε ο αριθμός των εικονοστοιχείων για κάθε τμήμα του πίνακα “out” και αποθηκεύεται στον πίνακα “s”.

```
    while(s(f+1) > PanoOrio)
        aS(f+1) = aS(f+1) - 0.01;
        out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))       >
maxPix(f+1) - (apoklisi + aS(f+1)) );
        s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
```

```
    end
```

Ελέγχεται το άθροισμα των εικονοστοιχείων (του κάθε τμήματος). Εάν είναι μεγαλύτερο από το “PanoOrio” τότε στο κατώφλι “aS” αφαιρείται 0.01 και επαναυπολογίζεται το threshold όπως επίσης και το νέο άθροισμα των εικονοστοιχείων. Η διαδικασία επαναλαμβάνεται αν ισχύει το παραπάνω.

```
    while(s(f+1) < KatoOrio)
        aS(f+1) = aS(f+1) + 0.01;
        out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))       >
maxPix(f+1) - (apoklisi + aS(f+1)) );
        s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
    end
```

Ελέγχεται το άθροισμα των εικονοστοιχείων (του κάθε τμήματος). Εάν είναι μικρότερο από το “KatoOrio” τότε στο κατώφλι “aS” προστίθεται 0.01 και επαναυπολογίζεται το threshold όπως επίσης και το νέο άθροισμα των εικονοστοιχείων. Η διαδικασία επαναλαμβάνεται αν ισχύει το παραπάνω

```
End
```

Τέλος της επανάληψης “for”.

```
imageout=out;
apokS=aS();
```

Επιστρέφετε η τελική εικόνα “out” όπως και ο πίνακας των κατωφλιών “apokS”.

Εικόνα 48: Τελική δυαδική εικόνα “out” μετά την επαλήθευση.

Τα αποτελέσματα (εικόνα 48) είναι καλά.

Τα επιπλέων σημεία που εμφανίζονται μπορούν να απομακρυνθούν εύκολα με την χρήση της εντολής bwareaopen (καθαρισμός συγκεκριμένων ομάδων εικονοστοιχείων).

Εφόσον το πρόβλημα της εξαγωγής της γραμμής του λείζερ λύθηκε, σειρά έχει να ασχοληθούμε με το χρόνο που χρειάζεται ο αλγόριθμος προκειμένου να επεξεργαστεί μια εικόνα. Ο προσαρμοστικός HSVCLEAR χρειάζεται περίπου 0.3 δευτερόλεπτα ή αλλιώς 300 ms. (χρόνος που επιτρέπει την επεξεργασία τριών, περίπου, εικόνων το δευτερόλεπτο).

Ο συγκεκριμένος χρόνος αφορά αποκλειστικά την επεξεργασία των εικόνων. Θα πρέπει να λάβουμε υπόψη μας τον επιπλέον χρόνο που θα χρειαστούν οι υπόλοιποι αλγόριθμοι για την κίνηση, την επεξεργασία σημείων κτλ.

Ο χρόνος που χρειαζόταν ήταν λόγος δημιουργίας προβληματισμών, οι οποίοι εν συνεχεία οδήγησαν στην αναζήτηση εναλλακτικών τεχνικών (κεφαλαίο 6) ή ακόμη και στη χρήση άλλης γλώσσας προγραμματισμού και βιβλιοθήκης όπως η OpenCV.

5.4 Συμπεράσματα

Ύστερα από διεξοδική έρευνα και μελέτη, κατασκευάστηκε ο προσαρμοστικής αλγόριθμος HSVCLEAR.
Αποτελεί την τελική μορφή της μεθόδου threshold, την οποία κι ακολουθούμε καθώς έχει τα καλύτερα αποτελέσματα όσο αναφορά την ποιότητα των εικόνων που εξάγει και στον χρόνο που τις επεξεργάζεται.

6 Εναλλακτικές τεχνικές

Κατά την διάρκεια βελτίωσες του αλγόριθμου HSVCLEAR δοκιμάστηκαν ποικίλες τεχνικές απομόνωσης της γραμμής του λείζερ. Οι τεχνικές αυτές είτε διευκόλυναν το threshold της εικόνας, είτε εστίαζαν στη μερική ή ολική απομάκρυνση της τεχνικής threshold προκείμενου να εξοικονομήσουμε χρόνο.
Μερικές από τις πιο αξιόλογες τεχνικές αναλύονται στο παρακάτω.

6.1 Φίλτρα κάμερας

Μία μέθοδος που δοκιμάστηκε με σκοπό τη διευκόλυνση του threshold της εικόνας, ήταν η χρήση φίλτρων gel ή colour filter.
Τα φίλτρα αυτά είναι στην ουσία διάφανες χρωματισμένες μεμβράνες, οι οποίες κατασκευάζονται από πολυαθρακικό και υπάρχουν σε διάφορα χρώματα.
Έχουν την ιδιότητα να τονίζουν κάποια χρώματα, όταν τοποθετηθούν μπροστά από τον φακό της κάμερας. Χρησιμοποιούνται για παράδειγμα σε ασπρόμαυρες φωτογραφίες για να χειραγωγούν την φωτεινότητα. Ένα κίτρινο φίλτρο παραδείγματος χάρη θα ενισχύσει την φωτεινότητα ανάμεσα στα σύννεφα και στον ουρανό, σκουραίνοντας τον ουρανό. Αντίθετα, ένα βαθύ πράσινο φίλτρο θα σκούραινε τον ουρανό, τονίζοντας όμως τα πράσινα φύλα.

Εικόνα 49: Φίλτρα gel ή colour filter.

Αγοράστηκαν έξι διαφορετικών χρωμάτων φίλτρα για τους απαραίτητους πειραματισμούς.

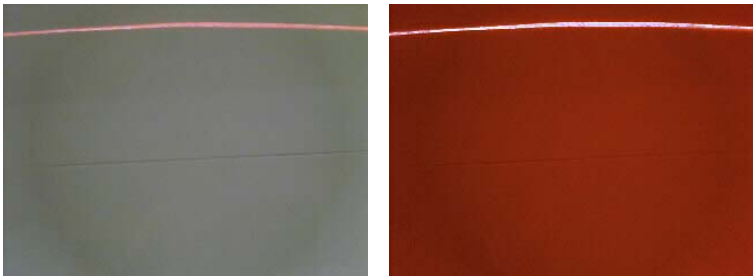
Εικόνα 50: μεμβρανοειδή φίλτρα gel ή colour filter

Ξεκινώντας, χρησιμοποιήσαμε το κόκκινο φίλτρο κι αυτό διότι μας ενδιαφέρει να γίνει ο διαχωρισμός της κόκκινης γραμμής του λέιζερ και να τονιστούν τα κόκκινα χρώματα. Τοποθετούμε λοιπόν το φίλτρο μπροστά από το φακό της κάμερας.

Εικόνα 51: Τοποθέτηση του φίλτρου στην κάμερα.

6.1.1 Συμπεράσματα

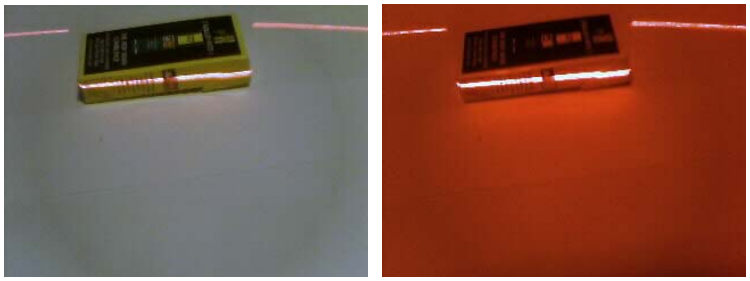
Σε εργαστηριακό περιβάλλον, δηλαδή σε άσπρο φόντο και κανονικό φωτισμό, τα αποτελέσματα ήταν πολύ καλά κι αυτό διότι η φωτεινότητα της γραμμής του λέιζερ ήταν πολύ έντονη κι έτσι μπορούσαμε πολύ εύκολα να κάνουμε threshold



(α) (β)

Εικόνα 52: α) εικόνα χωρίς φίλτρο, β) εικόνα με το κόκκινο φίλτρο.

Κατά τη διάρκεια του πειράματος σε άλλο περιβάλλον, στο οποίο ήταν τοποθετημένα διάφορα αντικείμενα διαφορετικών χρωμάτων και υλικών (εικόνα 53β), διαπιστώνεται ότι πέρα από το λέιζερ τονίζονται και τα άλλα αντικείμενα. Αυτό είχε σαν συνέπεια, μερικές φορές τα αποτελέσματα που παίρναμε από το threshold χρησιμοποιώντας το φίλτρο να είναι ίδια και χωρίς αυτό. (αν όχι και χειρότερα, ενίοτε)



(α) (β)

Εικόνα 53: α) εικόνα χωρίς φίλτρο, β) εικόνα με κόκκινο φίλτρο, τονίστηκε ιδιαιτέρα η σκιά (αρνητικό)

Δοκιμάζοντας κι άλλα χρώματα φίλτρων διαπιστώθηκε ότι το κίτρινο, δημιουργεί καλό αποτέλεσμα στις αντανακλάσεις του φωτός.
Εξομαλύνει τη φωτεινότητα της εικόνας με αποτέλεσμα να αποφεύγουμε τις έντονες διακυμάνσεις του φωτός κι άρα, κατά συνέπεια, με τη χρήση του threshold δεν παρουσιάζονται παραπλανητικά σημεία.

6.2 Κύκλωμα διακόπτη λέιζερ

Μια ανατρεπτική ιδέα διαχωρισμού της γραμμής λέιζερ από την εικόνα εφαρμόζεται με την εξής μέθοδο :

- Λαμβάνω μια εικόνα με το λέιζερ όταν αυτό είναι κλειστό
- Πραγματοποιώ την ίδια διαδικασία με το λέιζερ ανοιχτό αυτή τη φορά

- Αυτές τις δύο εικόνες τις αφαιρούμε, με αποτέλεσμα να έχουμε σαν μόνη διαφορά τους το λέιζερ όπου υπάρχει στην μία εικόνα και όχι στην άλλη (όπου το λέιζερ είναι κλειστό).

6.2.1 Πειράματα

(α) (β) (γ)

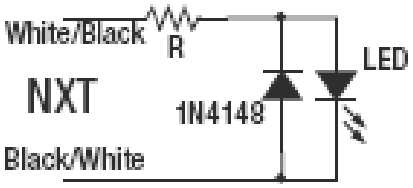
Εικόνα 54: α) εικόνα με ανοιχτό το λέιζερ, β) εικόνα με κλειστό το λέιζερ, γ) υπόλοιπο αφαίρεσης γ=α-β. Παρατηρούμε πόσο καλά είναι τα αποτελέσματα.

Διαπιστώνουμε ότι η παραπάνω μέθοδος λειτουργεί τέλεια.

Σαν επόμενο βήμα έχουμε τη δημιουργία αυτόματου τρόπου κατά τον οποίο να αναβοσβήνει το λέιζερ μέσω μιας εξόδου του μικροελεγκτή του ρομπότ.

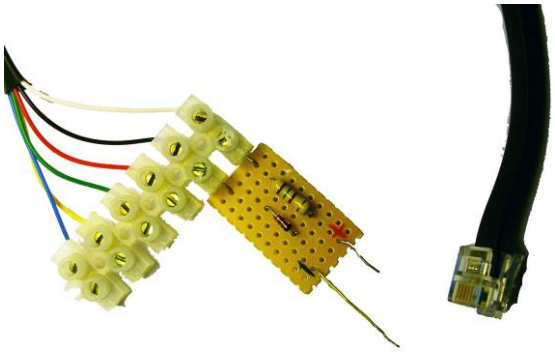
Για να μπορέσει αυτό να επιτευχθεί θα έπρεπε να δημιουργηθεί ένα κύκλωμα το οποίο να τροφοδοτεί το λέιζερ με τάση περίπου 4 volt & 7mw.

Για την κατασκευή μελετήθηκαν τα χαρακτηριστικά των εξόδων του lego mindstrom nxt κ έτσι διαπιστώθηκε από ποια pins θα μπορούσαμε να τροφοδοτήσουμε αυτή τη κατασκευή



Εικόνα 55: Σχεδιάγραμμα κυκλώματος για τροφοδοσία του λέιζερ μέσω του NXT

Το κύκλωμα προσαρμόστηκε πάνω σε ένα καλώδιο RJ12 προκειμένου να συνδέεται με το lego mindstrom nxt.



Εικόνα 56: Προσαρμογή κυκλώματος πάνω σε ένα καλώδιο RJ12 του NXT.

Πείραμα χρήσης της τεχνικής με το ρομπότ να βρίσκεται σε κίνηση.

(α) (β) (γ)

Εικόνα 57: α) εικόνα με ανοιχτό το λέιζερ, β) εικόνα με κλειστό το λέιζερ (το ρομπότ έχει προχωρήσει εμπρός), γ) υπόλοιπο αφαίρεσης γ=α-β.

6.2.2 Συμπεράσματα

Όπως διαπιστώνεται από τα αποτελέσματα, που προκύπτουν από τις πειραματικές δοκιμές, λόγω της κίνησης του ρομπότ εμφανίζονται κι άλλα σημεία εκτός από αυτά της γραμμής λέιζερ. Αυτό έχει ως συνέπεια την απαραίτητη εφαρμογή ενός αλγορίθμου κίνησης για την διόρθωση των διαφορετικών σημείων που δημιουργήθηκαν κατά τη κίνηση.

Ένα σοβαρό πρόβλημα ήταν οι καθυστερήσεις της ασύρματης επικοινωνίας bluetooth. Αυτές οι καθυστερήσεις είχαν σαν αποτέλεσμα να δημιουργούν σύγχυση ως προς το σε ποια εικόνα είναι αναμμένο το λέιζερ και σε ποια όχι. Αυτό συνέβαινε επειδή ενώ έπρεπε να κλείσει το λέιζερ και μετά να πάρουμε μια εικόνα, αρκετές φορές αυτό δεν συνέβαινε κ έτσι στην εικόνα εμφανιζόταν το λέιζερ.

Επισημαίνεται ότι το σύστημα αυτό δεν ήταν σταθερό.

Για να λυθεί το συγκεκριμένο πρόβλημα, έπρεπε να βάλουμε καθυστέρηση μετά από κάθε εντολή ανοίγματος ή κλεισίματος του λέιζερ. Διαπιστώθηκε ότι οι καθυστερήσεις έπρεπε να είναι περίπου 300ms, μετά από κάθε εντολή άρα σύνολο περίπου 600ms. Αν

υπολογίσουμε και τον χρόνο που θα χρειαζότανε ο αλγόριθμος κίνησης τα νούμερα είναι αποθαρρυντικά.

Η μέθοδος αυτή απορρίφτηκε λόγω μεγάλης καθυστέρησης και αστάθειας των αποτελεσμάτων.

7 Υλοποίηση αλγορίθμων που οδήγησαν στον τελικό αποτέλεσμα

Σύμφωνα με όσα έχουμε διαπιστώσει με τις έως τώρα έρευνές και πειράματά μας, η κάθε τεχνική παρουσιάζει μειονεκτήματα καθώς και πλεονεκτήματα.
Καλύτερα ποιοτικά αποτελέσματα έφεραν ο αλγόριθμος k-means, η τεχνική διακόπτη του λέιζερ (με την αφαίρεση εικόνων) και ο προσαρμοστικός αλγόριθμος threshold HSVCLEAR.

Ο χρόνος που χρειάζεται ο k-means για να επεξεργαστεί την κάθε εικόνα είναι υπερβολικά μεγάλος για να χρησιμοποιηθεί στην πράξη για την κίνηση του ρομπότ.
Η τεχνική με διακόπτη για το λέιζερ είχε πολύ καλά αποτελέσματα σε πειράματα που λάμβαναν χώρα σε εργαστηριακό περιβάλλον.
Στην πράξη, όμως, παρουσίαζε πολλά προβλήματα. Επίσης λόγω των μεγάλων καθυστερήσεων και της αστάθειας που παρουσίαζε, αποδείχτηκε μη λειτουργική για τις απαιτήσεις μας.
Ο αλγόριθμος HSVCLEAR βασίζεται σε μια συνήθη τεχνική κατάτμησης εικόνων και αυτό τον καθιστά μια πολύ καλή επιλογή για την λύση του προβλήματος μας.

Κατά την εξέλιξη του HSVCLEAR δημιουργήθηκε ένας νέος προσαρμοστικός HSVCLEAR, ο οποίος βασίζεται σε τεχνικές προσαρμοστικού threshold. Έτσι λοιπόν δημιουργήσαμε έναν νέο αλγόριθμο, με πολλές δυνατότητες προσαρμογής, σε τέτοια μορφή ώστε να μπορούμε να εκτελούμε διαφορετικά πειράματα, αλλάζοντας απλά κάποιες μεταβλητές στον αλγόριθμο. Ένας λόγος προβληματισμού απέναντι στο νέο προσαρμοστικό HSVCLEAR είναι ο χρόνος που χρειάζεται για την επεξεργασία της κάθε εικόνας. (περίπου 300ms ανά εικόνα). Παρότι ο συγκεκριμένος χρόνος δεν είναι ο καλύτερος δυνατός, δεν παρουσιάζει όμως ιδιαίτερα προβλήματα στην πράξη.

Στην προσπάθειά μας για περεταίρω βελτίωση του προσαρμοστικού HSVCLEAR αποφασίστηκε να γίνει μελέτη των βασικότερων αλγόριθμων που διανέμονται με το Matlab. Το πρόβλημα αυτό οδήγησε στην ανάγκη λήψης περισσότερων πληροφοριών, μέσα από τις οποίες θα αποκτούσαμε μία πλήρη εικόνα των τελικών χρόνων εκτέλεσης εντολών. Οι πληροφορίες αντλήθηκαν μέσα από τον "profiler" του Matlab.

Ο profiler μας δίνει αναλυτικές πληροφορίες για τον χρόνο, τον αριθμό επαναλήψεων κτλ που αφορούν την κάθε εντολή που εκτελείται.
Η παρακάτω εικόνα παρουσιάζει τα αποτελέσματα του profiler μετά από τετρακόσιες επαναλήψεις. Επισημαίνετε ότι η εκτέλεση του profiler έγινε στην τελική εκδοχή του προγράμματος μας, με το ρομπότ να κινείται, να υπολογίζονται τα ύψη των αντικειμένων κλπ, Οι συγκεκριμένοι αλγόριθμοι αναλύονται παρακάτω σε αυτό το κεφάλαιο.

Profile Summary

Generated 18-Oct-2009 16:38:49 using real time.

Function Name	Calls	Total Time	Πληροφορίες
LineTracker	1	130.284 s	Κύριος αλγόριθμος από τον οποίο καλούνται όλοι οι άλλοι αλγόριθμοι. Στον "LineTracker" περιλαμβάνονται κι όλες οι εντολές κίνησης του ρομπότ.
hsvClearSmart	400	79.809 s	Αλγόριθμος threshold επεξεργασίας εικόνας.
rgb2hsv	400	67.920 s	**Εντολή μετατροπής εικόνας από RGB χρωματικό μοντέλο σε HSV.**
bwareaopen	400	18.458 s	Εντολή καθαρισμού μικρών ομάδων εικονοστοιχείων.
regionprops	400	9.535 s	Εντολή εύρεσης πληροφοριών για ομάδες εικονοστοιχείων.
Xscanner	400	6.529 s	Αλγόριθμος εύρεσης συντεταγμένων των εικονοστοιχείων.
NXTMotor.SendToNXT	46	5.267 s	Εντολή αποστολής μιας ενεργείας στο ρομπότ.
NXTMotor.WaitFor	46	5.028 s	Εντολή αναμονής για επιτυχή αποστολή ενέργειας στο ρομπότ.

Ο συνολικός χρόνος εκτέλεσης των τετρακοσίων επαναλήψεων είναι περίπου 130 δευτερόλεπτα.

Συνεπώς, για να υπολογίσουμε τον χρόνο που χρειάζεται όλο το πρόγραμμα για κάθε μία εικόνα, διαιρούμε τον συνολικό χρόνο με τις επαναλήψεις. (δηλαδή 130.284 / 400 = 0.325 δευτερόλεπτα)

Η έρευνα κατευθύνθηκε στην εντολή rgb2hsv, η οποία μετατρέπει μια εικόνα από το χρωματικό μοντέλο RGB στο HSV.

Κατά τη διάρκεια της έρευνας βρέθηκε ένας εναλλακτικός αλγόριθμος για μετατροπή εικόνας από το ένα χρωματικό μοντέλο στο άλλο, ο οποίος λειτουργούσε γρηγορότερα και με εξίσου καλά αποτελέσματα με αυτόν του Matlab (εάν χρησιμοποιήσουμε μόνο τον 3^{o} πίνακα αυτού) Ο αλγόριθμος αυτός είναι ο λεγόμενος “colorspace” του *Pascal Getreuer 2005-2006.*

Για εξοικονόμηση χρόνου, ο αλγόριθμος τροποποιήθηκε έτσι ώστε να μην υπολογίζει άσκοπα τον 1^{o} και 2^{o} πίνακα (παρά μόνο τον 3^{o}).

Παρακάτω παρουσιάζεται ο πίνακας του profiler με τη χρήση του νέο αλγόριθμου μετατροπής ανάμεσα σε χρωματικά μοντέλα. (Το πείραμα εκτελέστηκε με τα ίδια δεδομένα όπως το παραπάνω πίνακα του Profiler)

Profile Summary

Generated 18-Oct-2009 16:59:17 using real time.

Function Name	Calls	Total Time	Πληροφορίες
LineTracker	1	44.397 s	Κύριος αλγόριθμος από τον οποίο καλούνται όλοι οι άλλοι αλγόριθμοι. Στον “LineTracker” περιλαμβάνονται όλες οι εντολές κίνησης του ρομπότ.
bwareaopen	400	14.425 s	Εντολή καθαρισμού μικρών ομάδων εικονοστοιχείων.
regionprops	400	12.081 s	Εντολή εύρεσης πληροφοριών για ομάδες εικονοστοιχείων.
rmfield	400	6.745 s	Εντολή που εκτελείται μέσω της regionprops
Xscanner	400	6.592 s	Αλγόριθμος εύρεσης συντεταγμένων των εικονοστοιχείων.
hsvClearSmart	400	6.487 s	Αλγόριθμος threshold επεξεργασίας εικόνας.
cell.strmatch	9200	5.975 s	
NXTMotor.SendToNXT	37	4.187 s	Εντολή αποστολής μιας ενεργείας στο ρομπότ.

NXTMotor.WaitFor	37	4.021 s	Εντολή αναμονής για επιτυχή αποστολή ενέργειας στο ρομπότ.
NXT_GetOutputState	37	3.973 s	Εντολή επικοινωνίας με το ρομπότ.
NXTMotor.ReadFromNXT	37	3.973 s	Εντολή επικοινωνίας με το ρομπότ.
NXC_ResetErrorCorrection	37	3.513 s	Εντολή επικοινωνίας με το ρομπότ.
...GetOutputState>NXT_CollectOutputState	37	3.409 s	Εντολή επικοινωνίας με το ρομπότ.
COM_CollectPacket	37	3.409 s	Εντολή επικοινωνίας με το ρομπότ.
COM_CollectPacket>BT_CollectPacket	37	3.409 s	Εντολή επικοινωνίας με το ρομπότ.
MY_colorspace	400	3.386 s	**Εντολή μετατροπής εικόνας από RGB χρωματικό μοντέλο σε HSV.**

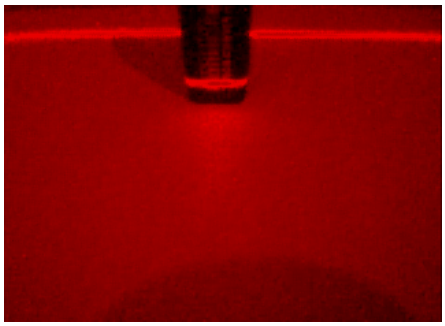
Όπως διαπιστώνεται παραπάνω, ο χρόνος ανά εικόνα έχει βελτιωθεί.
Συγκεκριμένα χρειάζεται 44.397 δευτερόλεπτα για τετρακόσιες επαναλήψεις (δηλαδή τετρακόσιες εικόνες), άρα 44.397 / 400 = 0.110 δευτερόλεπτα / εικόνα.

7.1 Κίνηση του ρομπότ

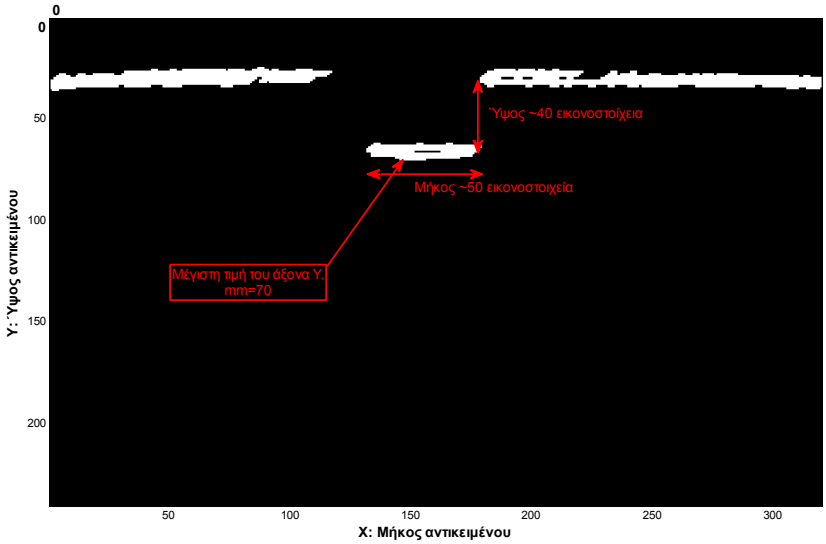
Οι εντολές για την κίνηση του ρομπότ δίνονται μέσα από το Matlab, με τη βοήθεια της βιβλιοθήκης “RWTH - εργαλειοθήκη Mindstorms NXT”, μέσω μίας ασύρματης επικοινωνίας Bluetooth.
Το ρομπότ πρέπει να είναι σε θέση να κινείται μέσα στο χώρο, αποφεύγοντας οποιοδήποτε εμπόδιο μεγαλύτερο ενός προκαθορισμένου ύψους.
Το ρομπότ αντιλαμβάνεται το αντικείμενο/εμπόδιο με τον εξής τρόπο:

- Αρχικά, σαν δεδομένο έχουμε τη δυαδική εικόνα που προήλθε από τον αλγόριθμο κατάτμησης.
- Στην εικόνα αυτή πραγματοποιείται μία δειγματοληψία (μέσω του αλγορίθμου “Xscanner”) από την οποία μας επιστρέφεται ένας πίνακας που περιέχει τις συντεταγμένες των εικονοστοιχείων.
  (Οι συντεταγμένες του άξονα Y απεικονίζουν το ύψος του αντικειμένου, ενώ οι X το μήκος του. Όσο αυξάνεται η τιμή του Y, τόσο αυξάνεται και το ύψος του αντικειμένου. Ομοίως, όσο αυξάνεται η τιμή του X, τόσο αυξάνεται και το μήκος του). (Εικόνα 59)
- Ύστερα από την ανάλυση του πίνακα των συντεταγμένων, μας επιστρέφεται η μέγιστη τιμή του άξονα Y, η οποία και υποδηλώνει το ψηλότερο αντικείμενο. Χάριν ευκολίας η μέγιστη αυτή τιμή θα ονομαστεί “mm”.

- Εάν η τιμή της “mm” είναι μεγαλύτερη από ένα προκαθορισμένο όριο, τότε συμπεραίνεται πως υπάρχει εμπόδιο.



Εικόνα 58: Αρχική εικόνα



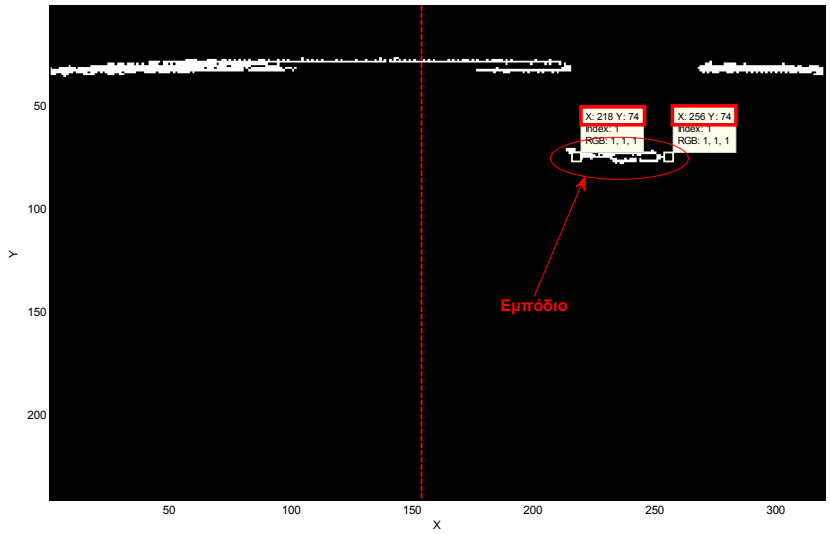
Εικόνα 59: Επεξήγηση τρόπου εύρεσης εμποδίου, «αν mm > 60 τότε εμπόδιο».

Το ρομπότ κινείται σε ευθεία πορεία. Σε περίπτωση που αντικρύσει εμπόδιο αποφασίζει κατά ποια κατεύθυνση θα πρέπει να κάνει στροφή για να το αποφύγει.
Κατά συνέπεια, όταν αντικρύσει εμπόδιο από τη δεξιά μεριά του, πρέπει να στρίψει αριστερά για να μην προσκρούσει με το εμπόδιο, και το αντίθετο.
Σε περίπτωση, όμως, που υπάρχουν εμπόδια και στις δύο πλευρές του πρέπει να αποφασίσει, έχοντας σαν κριτήριο την μικρότερη απόσταση περιστροφής, από ποια μεριά θα πρέπει να στρίψει.

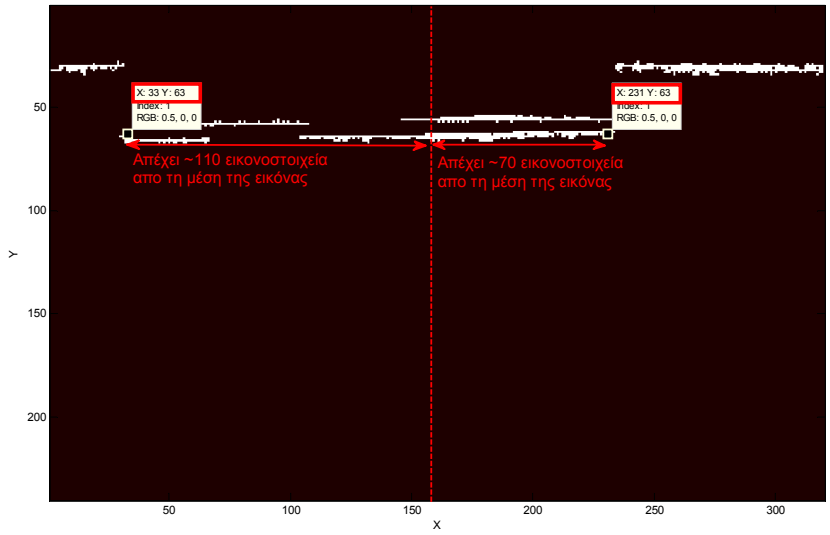
Η απόφαση λαμβάνεται ύστερα από τους εξής υπολογισμούς:

- Η ανάλυση της εικόνας στον άξονα X διαιρείται με το δύο, δίνοντας σαν αποτέλεσμα τη μέση της εικόνας. (Εικόνα 60)

- Από την υψηλότερη τιμή που βρέθηκε στον άξονα Y (εμπόδιο) , βρίσκουμε τις συντεταγμένες που αντιπροσωπεύουν τις τιμές αυτές στον άξονα X. Αυτό επιτυγχάνεται ψάχνοντας στον πίνακα από αριστερά στα δεξιά και το αντίστροφο.
  Με αυτό τον τρόπο βρίσκεται το μήκος του εμποδίου στην εικόνα. (Εικόνα 61)
- Εάν οι συντεταγμένες του μήκους την εικόνας (άξονα X) εκτείνονται πάνω από τη μέση της εικόνας, τότε το εμπόδιο βρίσκεται στη δεξιά πλευρά κι έτσι εκτελείται αριστερή στροφή. (Εικόνα 61)
- Εάν οι συντεταγμένες του μήκους την εικόνας (άξονα X) εκτείνονται κάτω από τη μέση της εικόνας, τότε το εμπόδιο βρίσκεται στην αριστερή πλευρά κι έτσι εκτελείται δεξιά στροφή.
- Εάν οι συντεταγμένες του μήκους την εικόνας (άξονα X) εκτείνονται και αριστερά και δεξιά από τη μέση της εικόνας, τότε υπολογίζονται τα δύο άκρα (αριστερό και δεξί) των συντεταγμένων. Έτσι:
  - Εάν το αριστερό άκρο απέχει λιγότερο από την μέση της εικόνας συγκριτικά με το δεξιό άκρο, τότε στρίβουμε αριστερά. (Εικόνα 61)
  - Εάν το αριστερό άκρο απέχει περισσότερο ή ίσο από την μέση της εικόνας συγκριτικά με το δεξιό άκρο, τότε στρίβουμε δεξιά.



Εικόνα 60: Εμπόδιο από τη μία πλευρά, πρέπει να στρίψει αριστερά.



Εικόνα 61: Το εμπόδιο εκτείνεται και από τις δυο πλευρές, υπολογίζονται οι αποστάσεις σε σχέση με την μέση της εικόνας και παίρνετε η απόφαση για το που θα στρίψει. Στην προκειμένη περίπτωση πρέπει να στρίψουμε αριστερά

Ο αλγόριθμος κίνησης δημιουργήθηκε με τέτοιο τρόπο, ώστε να στέλνεται μία εντολή κίνησης στο ρομπότ μόνο εάν πρέπει να εκτελεστεί μία διαφορετική εντολή από αυτήν στην οποία ήδη βρίσκεται. Με αυτό τον τρόπο μειώθηκε σημαντικά ο χρόνος εκτέλεσης του αλγορίθμου. Αυτό συνέβη επειδή στέλνονται μόνο οι απαραίτητες εντολές (μέσω της ασύρματης επικοινωνίας).

7.2 Κώδικας προγράμματος

Το πρόγραμμα που δημιουργήσαμε αποτελείται από πέντε αλγόριθμους. Παρακάτω ακολουθούν οι αλγόριθμοι με σειρά εκτέλεσης:

- Prepare
- LineTracker
- hsvClearSmart
- MY_colorspace
- Xscanner

Prepare

Ο αλγόριθμος “Prepare” περιέχει την δημιουργία των βασικότερων αντικειμένων και μεταβλητών.

Συγκεκριμένα:

- Δημιουργεί το αντικείμενο σύνδεσης του Matlab με την κάμερα.

- Προετοιμάζει κατάλληλα την κάμερα, ρυθμίζοντας βασικές παραμέτρους της.
- Δημιουργεί κάποιες μεταβλητές και τους εναποθέτει τιμές (αριθμός τμημάτων για τον HSVCLEAR, ταχύτητα κίνησης του ρομπότ κ.α.)
- Δημιουργεί τη σύνδεση του Matlab με το ρομπότ
- Δημιουργεί τα αντικείμενα – εντολές κινήσεις του ρομπότ
- Τέλος, εκτελεί τον “LineTracker.

LineTracker

Ο “LineTracker” είναι ο κυριότερος αλγόριθμος. Διαχειρίζεται τις εικόνες, το ρομπότ και όλους τους άλλους αλγόριθμους.

Οι κυριότερες λειτουργίες του “LineTracker” με σειρά εκτέλεσης είναι οι εξής:

- Λαμβάνει και προεπεξεργάζεται τις εικόνες από την κάμερα.
- Εκτελεί τον “hsvClearSmart” με τα κατάλληλα ορίσματα.
- Λαμβάνει την τελική εικόνα από τον “hsvClearSmart” και τη μορφοποιεί κατάλληλα.
- Εκτελεί τον “Xscanner” με όρισμα την μορφοποιημένη εικόνα, προκειμένου να επιστραφεί από αυτόν ο πίνακας με της συντεταγμένες των εικονοστοιχείων.
- Από τον πίνακα των συντεταγμένων του “Xscanner” υπολογίζει τα υψηλότερα αντικείμενα (εμπόδια).
- Σύμφωνα με τους υπολογισμούς των αντικειμένων εκτελεί την κατάλληλη ενέργεια στο ρομπότ. (στροφή αριστερά ή δεξιά και πορεία εμπρός)

hsvClearSmart

Ο “hsvClearSmart” είναι ο προσαρμοστικός αλγόριθμος κατάτμησης. Ο αλγόριθμος εξάγει την γραμμή του λέιζερ από την εικόνα.

Συγκεκριμένα:

- Μετατρέπει την εικόνα από RGB σε HSV εκτελώντας τον “MY_colorspace”.
- Χωρίζει την εικόνα σε τμήματα και θέτει τις τιμές των επιτρεπτών ορίων εικονοστοιχείων.
- Υπολογίζει το κατώφλι για κάθε τμήμα.
- Διαχωρίζει την γραμμή του λέιζερ από κάθε τμήμα.
- Επαληθεύει την ορθότητα των αποτελεσμάτων.
- Επιστρέφει την φιλτραρισμένη εικόνα στον “LineTracker”.

Xscanner

Ο “Xscanner” κάνει μια δειγματοληψία στη εικόνα και δημιουργεί ένα πίνακα με τις συντεταγμένες της γραμμής λέιζερ.

Συγκεκριμένα:

- Ορίζει τη μεταβλητή με τον ρυθμό δειγματοληψίας.

- Υπολογίζει με τον ανάλογο ρυθμό τα δείγματα και τα επιστρέφει σε έναν πίνακα.
- Ο πίνακας επιστρέφεται στον “LineTracker”.

Ακολουθεί ο κώδικας του κάθε αλγόριθμου με σειρά εκτέλεσης. Για καλύτερη ανάγνωση οι κλήσεις άλλων αλγόριθμων είναι με κόκκινο φόντο.

Prepare (Προετοιμασία για την εκτέλεση των υπολοίπων αλγορίθμων)

```
vid = videoinput('winvideo',2,'RGB24_320x240');
```
Δημιουργία αντικείμενου και σύνδεση κάμερας με το Matlab.

```
src = getselectedsource(vid);
```
Δημιουργία αντικειμένου “src” προκειμένου να μπορούμε να ρυθμίσουμε την κάμερα.

```
src.Brightness=3;
src.Saturation=3;
set(src,'Exposure',double(0.001))
set(src,'WhiteBalanceMode','manual')
triggerconfig(vid, 'manual');
vid.FramesPerTrigger=1;
vid.LoggingMode= 'Memory';
vid.TimerPeriod=0.01;
vid.TriggerFrameDelay=0;
vid.TriggerRepeat=0;
vid.FrameGrabInterval=1;
vid.ReturnedColorSpace='rgb';
vid.Timeout=12;
```
Αλλάζουμε της ρυθμίσεις της κάμερας. Σημαντικότερες ρυθμίσεις είναι : η μείωση της φωτεινότητας, η αύξηση της αντίθεσης και το “whiteBalnceMode” στο manual.

```
start(vid);
pause(1);
```

```
% calibrate(vid);
```
Προαιρετικός αλγόριθμος για να ρυθμίσουμε το οπτικό πεδίο της κάμερας σε σχέση με το λέιζερ.

```
segments=2;
```
Στη μεταβλητή αυτή ορίζουμε τον αριθμό των τμημάτων που θέλουμε να χωριστεί η εικόνα κατά την κατάτμηση

```
apokS(segments)=zeros;
flag=true;
```
Αρχικοποίηση τον μεταβλητών που προορίζονται για τον “HSVCLEARSMART”

```
DrivingSpeed = 18;
TurningSpeed = 27;
```
Μεταβλητές με τις οποίες ορίζουμε την ταχύτητα των κινητήρων.

```
handle = COM_OpenNXT('bluetooth.ini');
COM_SetDefaultNXT(handle);
```
Δημιουργούμε την σύνδεση Matlab – Ρομπότ μέσω ασύρματης επικοινωνίας Bluetooth.

```
straight                          = NXTMotor('B':'C');
straight.SpeedRegulation          = false;
straight.Power                    = DrivingSpeed;
straight.TachoLimit               = 0;
```
Δημιουργία αντικειμένου κίνησης του ρομπότ σε ευθεία πορεία. θέτουμε τις εξόδους όπου είναι συνδεδεμένοι οι σερβοκινητήρες, τον τρόπο όπου θα ξεκινάει το ρομπότ και την ταχύτητα του κινητήρα.

```
StrofiDeksia1                     =                 NXTMotor('B');
StrofiDeksia1.SpeedRegulation     = false;
StrofiDeksia1.Power               = -TurningSpeed;

StrofiDeksia2                     = StrofiDeksia1;
StrofiDeksia2.Port                = 'C';
StrofiDeksia2.Power               =  TurningSpeed;

StrofiAristera1                                 =  NXTMotor('B');
StrofiAristera1.SpeedRegulation   = false;
StrofiAristera1.Power             = TurningSpeed;

StrofiAristera2                   = StrofiAristera1;
StrofiAristera2.Port              = 'C';
StrofiAristera2.Power             =  -TurningSpeed;
```
Δημιουργία αντικειμένων κίνησης του ρομπότ για αριστερή και δεξιά στροφή.

```
run LineTracker;
```
Εκτελούμε τον αλγόριθμο “LineTracker”.

```
try
    StopMotor('all', 'brake');
    stop(vid);
    COM_CloseNXT('all');
end
```
Όταν τελειώσει η εκτέλεση του “LineTracker” σταματάμε όλους τους σερβοκινητήρες καθώς και το αντικείμενο “vid”. Στη συνέχεια απενεργοποιούμε την ασύρματη σύνδεση επικοινωνίας με το ρομπότ.

LineTracker (Κύριος αλγόριθμος)

```
flag=true;
goFlag=false;
oldDeksia=false;
Deksia=false;
o=0;
CutY=150;
OrioEmpodioy=60;
```

Αρχικοποίηση μεταβλητών.

```
Res=vid.VideoResolution;
```
Αποθηκεύουμε την ανάλυση του “vid” (κατά συνέπια της εικόνας που επιστρέφεται από την κάμερα) στην μεταβλητή “Res” π.χ. 320 x 240

```
SendToNXT(straight, handle);
```
Αποστέλλουμε το αντικείμενο “straight” στο ρομπότ. Το αντικείμενο αυτό δημιουργήθηκε από τον αλγόριθμο “Prepare” και πρόκειται για ένα αντικείμενο τύπου NXTMOTOR, με εντολές να κατευθύνει εμπρός το ρομπότ.

```
straight.WaitFor();
```
Αναμένουμε μέχρι την ολοκλήρωση της αποστολής του αντικείμενου “straight” στο ρομπότ.

```
while(flag==true)
```
Επανάληψη όσο το “flag” είναι true. Η επανάληψη αυτή συμβαίνει συνέχεια, αφού η μεταβλητή “flag” δεν αλλάζει πουθενά.

Από εδώ και κάτω ξεκινάει ο κύριος αλγόριθμος επεξεργασίας της εικόνας και καθοδήγησης του ρομπότ.

```
    image = peekdata(vid,1);
```
Λαμβάνουμε ένα καρέ (εικόνα) από το βίντεο της κάμερας (αντικείμενο “vid”) κι εν συνεχεία αποθηκεύεται στην μεταβλητή “image”

Η “peekdata” είναι παρόμοια αλλά λίγο γρηγορότερη από την “getsnapshot”, όπως διαπιστώθηκε μετά από δοκιμές.

```
    image=imcrop(image, [0 0 Res(1) CutY]);
```
Αποκόπτουμε το κάτω μέρος της εικόνας κατά “CutY” εικονοστοιχεία (στην προκειμένη περίπτωση κατά 150 εικονοστοιχεία). Η αποκοπή γίνετε μιας και το κάτω μέρος της εικόνας δεν μας είναι χρήσιμο κατά της επεξεργασία, οπότε γλιτώνουμε περιττό χρόνο επεξεργασίας.

```
    image(:,:,2)=zeros;
    image(:,:,3)=zeros;
    image = double(image)/255;
```
Από την “image” μηδενίζουμε των πίνακα δυο και τρία με αποτέλεσμα σαν μοναδική πληροφορία να έχουμε τον πίνακα ένα (πίνακας του κόκκινου). Στη συνέχεια μετατρέπουμε την “image” σε double.

```
    [imageout, apokS]=hsvClearSmart(image, apokS, segments);
```
Εκτελούμε τον αλγόριθμο “hsvClearSmart” με ορίσματα : την εικόνα “image”, τον πίνακα με τις τιμές των κατωφλιών και τον αριθμό τον τμημάτων όπου θα χωριστεί η εικόνα. Ο αλγόριθμος μας επιστρέφει την κατατμημένη εικόνα “imageout” και τον νέο πίνακα των κατωφλιών.

```
r = imageout;
```
Αποδίδουμε την “imageout” στο “r”.

```
I = im2bw(r);
```
Μετατρέπουμε την εικόνα “r” σε δυαδική εικόνα και την αποθηκεύουμε στην “I”.

```
I = bwareaopen(I,50);
```
Απομακρύνουμε από την εικόνα “I” αντικείμενα αποτελούμενα με λιγότερα από “50” εικονοστοιχεία.

```
dat=Xscanner(I);
```
Εκτελούμε τον αλγόριθμο “Xscanner” με όρισμα την εικόνα “I”. Ο αλγόριθμος μας επιστρέφει έναν πίνακα “dat” ο οποίος περιέχει μια περιγραφή των συντεταγμένων των εικονοστοιχείων της εικόνας “I”.

```
[Times,Thesi] = max(dat);
```
Η “max” μας επιστρέφει τη μεγαλύτερη τιμή του πίνακα “dat” (δηλαδή την θέση του πιο ψήλου εμποδίου) όπως και την θέση στην οποία βρίσκεται αυτό το αντικείμενο (στον άξονα Χ), προκειμένου να αποφασιστεί αν θα στρίψει αριστερά ή δεξιά.

```
mm=Times(1);
```
Αποθηκεύεται στο “mm” η τιμή της δεύτερης στήλης του πίνακα “Times”.

```
if mm>OrioEmpodioy
```
Εάν υπάρχει εμπόδιο μεγαλύτερο από το “OrioEmpodioy”.

```
    Andat=flipdim(dat, 1);
    [AnTimes,AnThesi] = max(Andat);
```
Αντιστρέφουμε τον πίνακα “dat” και ξαναυπολογίζουμε την μεγαλύτερη τιμή όπως και την θέση του εμποδίου. Αυτό γίνεται ώστε να διαπιστώσουμε αν υπάρχει εμπόδιο από αριστερά ή δεξιά ή και στις δυο πλευρές. Στην περίπτωση κατά την οποία υπάρχει και στις δυο πλευρές εμπόδιο, τότε το ρομπότ θα στρίψει κατά την πλευρά στην οποία υπάρχει ο περισσότερος κενός χώρος.

```
    XPos=dat(Thesi(1),2);
    XPosAn=Andat(AnThesi(1),2);
```
Αποθηκεύουμε στην “XPos” και στην “XPosAn” την θέση ως προς τον άξονα Χ του εμπόδιου. Το “XPos” έχει την θέση από αριστερά προς τα δεξιά και το “XPosAn” την θέση από δεξιά προς αριστερά.

```
    Mesi=Res(1)/2;
```
Η ανάλυση της εικόνας ως προς τον άξονα Χ διαιρείται διά δυο, προκείμενου να αποφασιστεί παρακάτω που θα στρίψει το ρομπότ. Αν υπάρχει εμπόδιο από το δεύτερο μισό της εικόνας θα στρίψουμε αριστερά. Αν υπάρχει από το πρώτο μισό θα στρίψουμε δεξιά.

```
        Apostasi1=abs(Mesi-XPos);
        Apostasi2=abs(Mesi-XPosAn);
```

Υπολογίζουμε την απόσταση του εμπόδιου από την μέση της εικόνας, από αριστερά προς δεξιά “Apostasi1” αλλά και από δεξιά προς αριστερά “Apostasi2”.

```
        if XPos<Mesi && XPosAn>Mesi
```

Η περίπτωση αυτή υφίσταται όταν υπάρχει εμπόδιο και από αριστερά και από δεξιά από την μέση της εικόνας. Άρα παρακάτω θα αποφασιστεί κατά ποια κατεύθυνση θα στρίψει το ρομπότ.

```
            if Apostasi1>Apostasi2
                Deksia=true;
```

Εάν η τιμή της “Apostasi1” είναι μεγαλύτερη από αυτή του “Apostasi2” τότε θέτουμε την μεταβλητή “Deksia” true, προκειμένου να στρίψει το ρομπότ δεξιά.

```
            else
                Deksia=false;
```

Εάν δεν ισχύει το παραπάνω τότε θέτουμε την “Deksia” σαν false, προκειμένου να στρίψει το ρομπότ αριστερά.

```
            end
        end
```

```
        if XPos<Mesi && XPosAn<Mesi
            Deksia=true;
        end
```

Εάν υπάρχει εμπόδιο μόνο από αριστερά τότε θέτουμε την μεταβλητή “Deksia” true, προκειμένου να στρηψη το ρομπότ δεξια.

```
        if XPos>Mesi && XPosAn>Mesi
            Deksia=false;
        end
```

Εάν υπάρχει εμπόδιο μόνο από δεξιά τότε θέτουμε την μεταβλητή “Deksia” false, προκειμένου να στρίψει το ρομπότ αριστερά.

```
        oldDeksia=Deksia;
```

Στην μεταβλητή “oldDeksia” κρατάμε την τελευταία τιμή της μεταβλητής “Deksia”.

```
    end
```

```
    if (mm>OrioEmpodioy) && ((goFlag==true) | (oldDeksia~=Deksia) )
```

Εάν ισχύει ότι το “mm” είναι μεγαλύτερο από το “OrioEmpodioy” (δηλαδή υπάρχει εμπόδιο) όπως επίσης και, το “goFlag” είναι true (δηλαδή το ρομπότ πηγαίνει εμπρός) ή το “oldDeksia” είναι διάφορο από το “Deksia” (δηλαδή το ρομπότ ήδη στρίβει.

```
        goFlag=false;
```
Θέτουμε την “goFlag” false, ώστε να μην ξαναδοθεί εντολή να στρίψει το ρομπότ.

```
        StopMotor('all', 'brake');
```
Σταματάμε όλους τους σερβοκινητήρες.

```
        if (Deksia==true)
            %
            StrofiDeksia1.SendToNXT();
            StrofiDeksia1.WaitFor();
            %
            StrofiDeksia2.SendToNXT();
            StrofiDeksia2.WaitFor();
```
Εάν η “Deksia” είναι true (άρα στροφή δεξιά) τότε στέλνουμε στο ρομπότ τα αντικείμενα “StrofiDeksia1” και “StrofiDeksia2” και αναμένουμε την αποστολή τους. Το κάθε αντικείμενο αναφέρετε σε ένα σερβοκινητήρα.

```
        else
            StrofiAristera1.SendToNXT();
            StrofiAristera1.WaitFor();
            %
            StrofiAristera2.SendToNXT();
            StrofiAristera2.WaitFor();
        End
```
Αλλιώς (εάν δηλαδή το “Deksia” είναι false (άρα στροφή αριστερά)) τότε στέλνουμε στο ρομπότ τα αντικείμενα “StrofiAristera1” και “StrofiAristera2” και αναμένουμε την αποστολή τους. Το κάθε αντικείμενο αναφέρετε σε ένα σερβοκινητήρα.

```
    end
```

```
    if (mm<=60) && (goFlag==false)
```
Εάν το “mm” είναι μικρότερο ή ίσο με το “OrioEmpodioy” (δηλαδή δεν υπάρχει εμπόδιο) και το “goFlag” είναι false.

Δηλαδή το ρομπότ πρέπει να πάει εμπρός εφόσον δεν υπάρχει εμπόδιο.

```
        goFlag=true;
```
Θέτουμε την μεταβλητή “goFlag” true προκειμένου να μην ξανασταλθεί η εντολή να πάει εμπρός το ρομπότ. Παρά μόνο να σταλθεί εντολή στροφής.

```
        StopMotor('all', 'brake');
        SendToNXT(straight, handle);
        straight.WaitFor();

    end
```
Σταματάμε όλους τους σερβοκινήτηρες, στέλνουμε το αντικείμενο “straight” στο ρομπότ και αναμένουμε την αποστολή του.

```
end
```

hsvClearSmart (κατάτμηση εικόνας)

```
function [imageout, apokS] = hsvClearSmart(imagein, aS, segments)
```
Ορισμός του αλγόριθμου και των ονομάτων των μεταβλητών του.

```
hsvOrig = MY_colorspace('rgb->hsv',imagein);
hsv=hsvOrig(:,:,3);
```
Μετατρέπουμε την εικόνα “imagein” από RGB σε HSV χρωματικό μοντέλο και την αποθηκεύουμε στο “hsvOrig”. Στη συνέχεια αποθηκεύουμε τον τρίτο πίνακα αυτής στην μεταβλητή “hsv”.

```
[y,xx,e] = size(imagein);
```
Η “size” μας επιστρέφει τις διαστάσεις της εικόνας. Το “y” είναι η διάσταση στον άξονα Υ, το “xx” είναι η διάσταση στον άξονα Χ και το “e” είναι ο αριθμός των πινάκων που αποτελούν την “imagein”.

```
seg=xx/segments;
```
Διαιρούμε τον άξονα Χ “xx” με των αριθμό των τμημάτων “segments”. Από το πηλίκο τους προκύπτει η απόσταση του κάθε τμήματος.

```
xMax=1:y;
x=xMax;
```
Δημιουργούμε ένα πίνακα από “1” μέχρι την τιμή του άξονα Υ “y” και τον εναποθέτουμε στην μεταβλητή “xMax”.

```
if xx==160
    PanoOrio=300;
    KatoOrio=100;
else
    PanoOrio=600;
    KatoOrio=200;
End
```
Ορίζουμε τα επιτρεπόμενα όρια του αθροίσματος των εικονοστοιχείων ανάλογα με το μέγεθος της εικόνας.

```
maxPix(segments)=zeros;
s(segments)=zeros;
```
Μηδενίζουμε τους πίνακες “maxPix” και “s”.

```
for i=0:segments-1
    maxPix(i+1)=max(max(hsv(xMax,(seg*i)+1:seg*(i+1))));
end
```
Βρίσκουμε την φωτεινότερη τιμή (index) για κάθε τμήμα. Οι τιμές αποθηκεύονται στον πίνακα “maxPix”.

```
apoklisi=0.15;
```
Σταθερά απόκλισης.

```
out=zeros(y, xx);
```

Δημιουργούμε ένα πίνακα “out” με της διαστάσεις της αρχικής εικόνας. Ο “out” είναι και η τελική δυαδική εικόνα.

```
for f=0:segments-1
```
Επανάληψη “segments” φορές (δηλαδή όσα είναι και τα τμήματα), προκειμένου να υπολογιστή το threshold και επαλήθευση αυτού.

```
    out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))          >
maxPix(f+1) - (apoklisi + aS(f+1)) );
```
Έλεγχος εικονοστοιχείο προς εικονοστοιχείο για κάθε τμήματος της εικόνας “hsv”. ‘Οταν η τιμή του εικονοστοιχείου είναι μεγαλύτερη από το “maxPix” (δηλαδή την τιμή κατωφλιού) του εκάστοτε τμήματος τότε ισούται με “1” αλλιώς ισούται με “0”. Τα αποτελέσματα αποθηκεύονται στην ανάλογη θέση στην εικόνα “out”.

```
    s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
```
Αθροίζεται ο αριθμός των εικονοστοιχείων για κάθε τμήμα του πίνακα “out” και αποθηκεύεται στον πίνακα “s”.

```
    while(s(f+1) > PanoOrio)
        aS(f+1) = aS(f+1) - 0.01;
        out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))      >
maxPix(f+1) - (apoklisi + aS(f+1)) );
        s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
    end
```
Ελέγχεται το άθροισμα των εικονοστοιχείων (του κάθε τμήματος). Εάν είναι μεγαλύτερο από το “PanoOrio” τότε στο κατώφλι “aS” αφαιρείται 0.01 και επαναυπολογίζεται το threshold όπως επίσης και το νέο άθροισμα των εικονοστοιχείων. Η διαδικασία επαναλαμβάνεται αν ισχύει το παραπάνω.

```
    while(s(f+1) < KatoOrio)
        aS(f+1) = aS(f+1) + 0.01;
        out(x,(seg*f)+1:seg*(f+1))=(hsv(x,(seg*f)+1:seg*(f+1))      >
maxPix(f+1) - (apoklisi + aS(f+1)) );
        s(f+1)=sum(sum(out(x,(seg*f)+1:seg*(f+1) )));
    end
```
Ελέγχεται το άθροισμα των εικονοστοιχείων (του κάθε τμήματος). Εάν είναι μικρότερο από το “KatoOrio” τότε στο κατώφλι “aS” προστίθεται 0.01 και επαναυπολογίζεται το threshold όπως επίσης και το νέο άθροισμα των εικονοστοιχείων. Η διαδικασία επαναλαμβάνεται αν ισχύει το παραπάνω

```
End
```
Τέλος της επανάληψης “for”.

```
imageout=out;
apokS=aS();
```
Επιστρέφεται η τελική εικόνα “out” όπως και ο πίνακας των κατωφλιών “apokS”.

Xscanner (δειγματοληψία δυαδικής εικόνας)

```
function [data] = Xscanner(image)
```
Ο Xscanner δέχεται σαν είσοδο μία δυαδική εικόνα “image” και σαν έξοδο επιστρέφει ένα πίνακα με τις συντεταγμένες του δειγματοληπτικού ελέγχου

```
i=1;
bima=5;
apoY=5;
pin(1000,2)=zeros;
```
Αρχικοποίηση μεταβλητών.

```
s=size(image);
```
Υπολογισμός της ανάλυσης εικόνας και επιστροφή των τιμών στη μεταβλητή “s”.

```
for x=1:bima:s(2)
```
Εκτέλεση του παρακάτω κώδικα από “1” μέχρι “s(2)” φορές με “βήμα” κι εναπόθεση της τιμής της κάθε επανάληψης στο “x”

```
    for y=apoY:s(1)
```
Εκτέλεση του παρακάτω κώδικα από “apoY” μέχρι “s(1)” φορές κι εναπόθεση της τιμής της κάθε επανάληψης στο “y”

```
        if image(y,x) == 1
```
Εάν στις συντεταγμένες “y”,”x” της εικόνας “image” υπάρχει ενεργό εικονοστοιχείο, δηλαδή 1 τότε

```
            pin(i,:)=[y x];
```
Στον πίνακα “pin” αποθηκεύουμε τις συντεταγμένες “y”,”x”

```
            i=i+1;
```
Αυξάνουμε το “i” κατά ένα

```
        end
    end
end
```

```
data=pin;
```
Εναποθέτουμε τον “pin” στη μεταβλητή “data” στην οποία κι επιστρέφεται.

MY_ColorSpace (Μετατροπή από RGB σε HSV χρωματικό μοντέλο).

Ο παρακάτω αλγόριθμος αναφέρεται και δεν αναλύεται. Αποτελεί μία τροποποίηση του colourspace με σκοπό να μας επιστρέφει μόνο τον τρίτο πίνακα.

```
function varargout = colorspace(Conversion,varargin)

if nargin < 2, error('Not enough input arguments.'); end
[SrcSpace,DestSpace] = parse(Conversion);

if nargin == 2
   Image = varargin{1};
elseif nargin >= 3
   Image = cat(3,varargin{:});
else
   error('Invalid number of input arguments.');
end

FlipDims = (size(Image,3) == 1);

if FlipDims, Image = permute(Image,[1,3,2]); end
if ~isa(Image,'double'), Image = double(Image)/255; end
if size(Image,3) ~= 3, error('Invalid input size.'); end

SrcT = gettransform(SrcSpace);
DestT = gettransform(DestSpace);

if ~ischar(SrcT) & ~ischar(DestT)
   % Both source and destination transforms are affine, so they
   % can be composed into one affine operation
   T = [DestT(:,1:3)*SrcT(:,1:3),DestT(:,1:3)*SrcT(:,4)+DestT(:,4)];
   Temp = zeros(size(Image));
   Temp(:,:,1)    =    T(1)*Image(:,:,1)    +    T(4)*Image(:,:,2)    +
T(7)*Image(:,:,3) + T(10);
   Temp(:,:,2)    =    T(2)*Image(:,:,1)    +    T(5)*Image(:,:,2)    +
T(8)*Image(:,:,3) + T(11);
   Temp(:,:,3)    =    T(3)*Image(:,:,1)    +    T(6)*Image(:,:,2)    +
T(9)*Image(:,:,3) + T(12);
   Image = Temp;
elseif ~ischar(DestT)
   Image = rgb(Image,SrcSpace);
   Temp = zeros(size(Image));
   Temp(:,:,1)  =  DestT(1)*Image(:,:,1)  +  DestT(4)*Image(:,:,2)  +
DestT(7)*Image(:,:,3) + DestT(10);
   Temp(:,:,2)  =  DestT(2)*Image(:,:,1)  +  DestT(5)*Image(:,:,2)  +
DestT(8)*Image(:,:,3) + DestT(11);
   Temp(:,:,3)  =  DestT(3)*Image(:,:,1)  +  DestT(6)*Image(:,:,2)  +
DestT(9)*Image(:,:,3) + DestT(12);
   Image = Temp;
else
   Image = feval(DestT,Image,SrcSpace);
end

%%% Output format %%%
if nargout > 1
   varargout = {Image(:,:,1),Image(:,:,2),Image(:,:,3)};
else
   if FlipDims, Image = permute(Image,[1,3,2]); end
```

```
   varargout = {Image};
end
return;

function [SrcSpace,DestSpace] = parse(Str)
% Parse conversion argument

if isstr(Str)
   Str = lower(strrep(strrep(Str,'-',''),' ',''));
   k = find(Str == '>');

   if length(k) == 1         % Interpret the form 'src->dest'
      SrcSpace = Str(1:k-1);
      DestSpace = Str(k+1:end);
   else
      k = find(Str == '<');

      if length(k) == 1      % Interpret the form 'dest<-src'
         DestSpace = Str(1:k-1);
         SrcSpace = Str(k+1:end);
      else
         error(['Invalid conversion, ''',Str,'''.']);
      end
   end
   SrcSpace = alias(SrcSpace);
   DestSpace = alias(DestSpace);
else
   SrcSpace = 1;             % No source pre-transform
   DestSpace = Conversion;
   if any(size(Conversion) ~= 3), error('Transformation matrix must
be 3x3.'); end
end
return;

function Space = alias(Space)
Space = strrep(Space,'cie','');

if isempty(Space)
   Space = 'rgb';
end

switch Space
case {'hsv','hsb'}
   Space = 'hsv';
   return;
end
return;

function T = gettransform(Space)
% Get a colorspace transform: either a matrix describing an affine
transform,
% or a string referring to a conversion subroutine
switch Space
case {'rgb','xyz','hsv','hsl','lab','luv','lch'}
   T = Space;
otherwise
   error(['Unknown color space, ''',Space,'''.']);
end
return;
```

```
function Image = hsv(Image,SrcSpace)
V = max(Image,[],3);
Image(:,:,3) = V;
return;

function Image = rgb(Image,SrcSpace)
% Convert to Rec. 709 R'G'B' from 'SrcSpace'
switch SrcSpace
case 'rgb'
   return;
end

% Clip to [0,1]
Image = min(max(Image,0),1);
return;
```

8 Τελικά αποτελέσματα - συμπεράσματα

Τα τελικά αποτελέσματα ήταν πολύ καλύτερα από τα αναμενόμενα καθώς καταφέραμε να δημιουργήσουμε έναν αλγόριθμο κατάτμησης εικόνων ο οποίος δουλεύει στα ~14fps.
Όλη η διαδικασία χρειάζεται ~11ms ανά εικόνα, δηλαδή μπορεί να βγάλει ~9fps.
Έτσι καταφέραμε να κάνουμε το ρομπότ να ελίσσεται εύκολα ανάμεσα σε εμπόδια διαφορετικών υλικών και διαστάσεων, αποφασίζοντας δυναμικά για το προς ποια κατεύθυνση θα στρίψει.
Τέλος, ύστερα από συμπεράσματα που προέκυψαν από όλη τη διάρκεια της έρευνας, διαπιστώνουμε ότι με τη συγκεκριμένη τεχνική επιτυγχάνεται καλύτερη λειτουργία από ότι με τους αισθητήρες υπερήχων και υπερύθρων, καθώς μπορεί να αναγνωρίσει περισσότερα είδη και μεγέθη αντικειμένων. Αυτό κατά συνέπεια καθιστά ακριβέστερη την μέτρηση.

Μελλοντικές βελτιώσεις :

- Να γίνεται καλύτερος έλεγχος του αποτελέσματος για τη διόρθωση πιθανών λαθών, μέσω αλγορίθμων αναγνώρισης σχημάτων ή και με τη χρήση πιθανοτήτων.
- Να λειτουργεί σε συνεργασία με έναν αισθητήρα υπερήχων ή υπερύθρων.
- Να κάνει ανάγνωση της έντασης του φωτός μέσω ενός αισθητήρα φωτός ώστε να ρυθμίζει κατάλληλα τον αλγόριθμο κατάτμησης.
- Χρήση ασύρματης κάμερας ώστε να γίνει πλήρη αποδέσμευση από τον σταθμό διαχείρισης του.
- Μέσω των πληροφοριών των εμποδίων (ύψος και μήκος) και με τη χρήση οδομετρίας να μπορεί να δημιουργεί ένα χάρτη του χώρου που βρίσκεται, ώστε να επανατοποθετείται μέσα σε αυτόν. (localization)
- Να μπορεί να αναγνωρίζει συγκεκριμένα αντικείμενα μέσω των εικόνων που ήδη λαμβάνει.
- Χρήση σαρωτή λέιζερ με υπέρυθρο φως, ώστε να μην είναι ορατή σε εμάς η γραμμή του λέιζερ.

Για να μπορούν να υφίστανται τα παραπάνω, θα πρέπει να επαναπρογραμματιστούν οι αλγόριθμοι σε διαφορετική γλώσσα-βιβλιοθήκη, όπως η OpenCV / EmguCV, ώστε να επιτευχθούν καλύτεροι χρόνοι επεξεργασίας.

Βιβλιογραφία

[1] Wikipedia [Matlab] http://en.wikipedia.org/wiki/MATLAB

[2] RWTH - Mindstorms NXT Toolbox for MATLAB http://www.mindstorms.rwth-aachen.de/

[3] Wikipedia [Charge-coupled device] http://el.wikipedia.org/wiki/CCD

[4] Εργαστήριο Αυτοματικής – Ρομποτικής του ΤΕΙ Κρήτης, Τεχνολογία Αισθητήρων [Σημειώσεις (.ppt) για μέτρηση απόστασης (Range)] http://www.tm.teicrete.gr/erobot/index.files/Page318.htm

[5] Eric Frew, Andreas Huster, Edward LeMaster. [Image Thresholding for Object Detection] http://sun-valley.stanford.edu/projects/helicopters/final.html

[6] Extreme NXT: Extending the LEGO MINDSTORMS NXT to the Next Level By Michael Gasperi Philippe E. Hurbain Isabelle L. Hurbain [Deriving power from NXT® motor port A] http://www.philohome.com/nxtpwr/pwr.htm

[7] Lego Mindstorm NXT Third Party Hardware and Accessories (update 22/09/2008) http://forum.mindstormsnxt.gr/jforum/posts/list/15.page

[8] Pramuk Boonsieng April-August, 2007. [The Orange Objects Classification by MATLAB] http://www.robocup.hs-eingarten.de/dokumente/report_Boonsieng.pdf

[9] RWTH Aachen University – UMIC Mobile multimedia processing [Computer Vision WS 08/09] http://www.vision.ee.ethz.ch/~bleibe/multimedia/teaching/cv-ws08/index.html

[10] The mathworks [Image Acquisition Toolbox - For Use with MATLAB] http://www.mathworks.co.uk/access/helpdesk_r13/help/pdf_doc/imaq/imaq_online.pdf

[11] H.J.C. Luijten DCT 2005.87 – [Basics of color based computer vision implemented in Matlab] http://alexandria.tue.nl/repository/books/612499.pdf

[12] Pascal Getreuer 28 May 2005 (Updated 16 Aug 2006) [Color Space Converter] http://www.mathworks.com/matlabcentral/fileexchange/7744

[13] Ying Wu, Qiong Liu, Thomas S. Huang Beckman Institute University of Illinois at Urbana-Champaign Urbana, IL 61801, U.S.A. [An Adaptive Self-Organizing Color Segmentation Algorithm with Application to Robust Real-time Human Hand Localization]

[14] A. Mandow, A. García-Cerezo, M. J. López-Baldán [RECOGNITION OF CONTEXTS WITH SCANNER-LASER FOR MOBILE ROBOT NAVIGATION]

[15] Wuihan xiong peking university 1994 [Separating illumination from reflectance in Jens Christian Andersen, Nils A. Andersen, and Ole Ravn Automation DTU, Technical University of Denmark, DTU-build. 326, DK-2800 Kgs. Lyngby, Denmark [Vision Assisted Laser Scanner Navigation for Autonomous Robots]

[16] Wikipedia, [APA style] http://en.wikipedia.org/wiki/APA_style