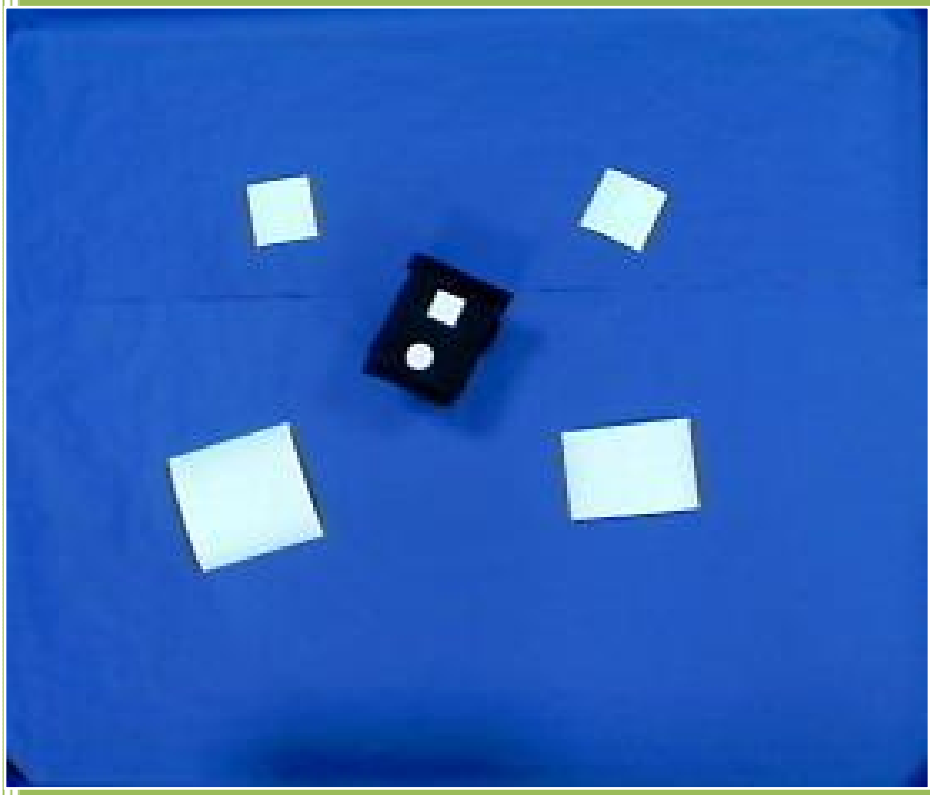
Α.Τ.Ε.Ι ΚΡΗΤΗΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΕΛΕΓΧΟΣ ΡΟΜΠΟΤΙΚΗΣ ΠΛΑΤΦΟΡΜΑΣ ΜΕΣΩ ΚΑΜΕΡΑΣ



Φοιτητής : ΚΑΣΙΩΤΗΣ ΦΙΛΙΠΠΟΣ

Επιβλέπων καθηγητής : ΠΑΛΑΜΑΣ ΓΕΩΡΓΙΟΣ

Έλεγχος ρομποτικής πλατφόρμας μέσω κάμερας

Σκοπός της άσκησης είναι ο έλεγχος της κίνησης ενός ρομποτικού οχήματος μέσω της πληροφορίας που αντλούμε από την επεξεργασμένη εικόνα που λαμβάνουμε από μια web camera η οποία είναι παράλληλα και το μοναδικό μας αισθητήριο .Τα κομμάτια τα οποία απαρτίζουν και υλοποιούν την άσκηση είναι τα εξής:

- *Web camera* : Λαμβάνουμε την εικόνα μας.
- *Bluetooth:* Μέσω αυτού γίνεται η ασύρματη επικοινωνία μεταξύ NXT και υπολογιστή.
- *Υπολογιστής :* Εδώ λαμβάνουμε και επεξεργαζόμαστε την εικόνα και έπειτα στέλνουμε τις επιθυμητές εντολές κίνησης μέσω του Bluetooth.
- *Matlab* : Εδώ χρησιμοποιούμε δύο toolboxes της :

1) ***Image processing toolbox :*** για την επεξεργασία και άντληση των πληροφοριών που λαμβάνουμε από την εικόνα.

2) ***Mindstorms NXT Toolbox :*** για τον έλεγχο του οχήματος μας.

Περιεχόμενα

ΚΕΦΑΛΑΙΟ 1 *<<LEGO MINDSTORMS NXT>>* 6

Εισαγωγικά 6

Ανάλυση Lego Nxt 7

Αισθητήρες Lego Nxt 8

Υποστηριζόμενες γλώσσες προγραμματισμού 10

ΚΕΦΑΛΑΙΟ 2 *<<IMAGE ACQUISITION & PROCESSING TOOLBOX>>* 11

Γενικά 11

Υλοποίηση του image processing & acquisition toolbox 16

Ιδιότητες κάμερας 17

Λήψη εικόνας 19

Επεξεργασία εικόνας 20

Επεξήγηση καπέλου 23

ΚΕΦΑΛΑΙΟ 3 *<<ΕΥΡΕΣΗ ΓΩΝΙΑΣ>>* 24

Εισαγωγικά 24

Εύρεση γωνίας 23

Κώδικας εύρεσης γωνίας 29

Κώδικας περιπτώσεων 27

Ανάλυση συνάρτησης 42

ΚΕΦΑΛΑΙΟ 4 *<<MINDSTORM NXT TOOLBOX>>* 43

Γενικά 43

Σύνδεση NXT με υπολογιστή μέσω Bluetooth 43

PID ελεγκτής 47

Επεξήγηση κώδικα PID 52

Κίνηση NXT με PID 53

Επεξήγηση κώδικα κίνησης με PID 59

ΚΕΦΑΛΑΙΟ 5 <<GNG ΑΛΓΟΡΙΘΜΟΣ>> 61

Γενικά 61

Εφαρμογή αγλορίθμου 62

Κώδικας υλοποίησης 64

Επεξήγηση κώδικα 65
GNG συνάρτηση 67
Distance Nodes συνάρτηση 71
Επεξήγηση Distance Nodes συνάρτηση 72
Κίνηση NXT με GNG 74
Αλλαγές κίνησης 79
Βιβλιογραφία 80

<<ΚΕΦΑΛΑΙΟ 1>>

LEGO MINDSTORMS NXT

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

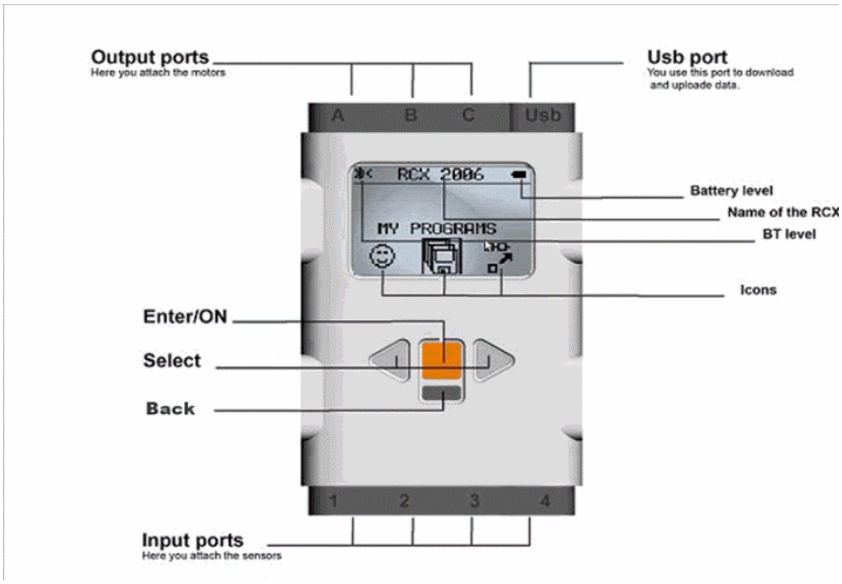
Lego Mindstorms NXT

Με τη νέα γενιά NXT, , η Lego προχωρεί ένα βήμα πιο πέρα από την επανάσταση των «οικιακής κατασκευής» ρομπότ που η ίδια είχε ξεκινήσει πριν από οκτώ χρόνια με τα Lego Mindstorms. Πολύ πιο εύκολα και γρήγορα στην κατασκευή τους, τα κιτ Mindstorms NXT δίνουν τη δυνατότητα στους ερασιτέχνες λάτρεις της ρομποτικής κάθε ηλικίας να φτιάξουν και να προγραμματίσουν το δικό τους μίνι ρομπότ μέσα σε μόλις 30 λεπτά της ώρας. Η αναπροσαρμογή λογισμικού για το δημοφιλές σύστημα εφευρέσεων ρομποτικής LEGO MINDSTORMS NXT απελευθερώνεται. Η νέα έκδοση λογισμικού LEGO MINDSTORMS NXT1.1 τώρα παρέχει την υποστήριξη για Vista και Macintosh Windows. Με τη βελτιωμένη χρήση μνήμης του λογισμικού, το LEGO MINDSTORMS NXT περιλαμβάνει μικρότερα συνταγμένα προγράμματα και συμπιεσμένα αρχεία.

Το NXT βασίζεται στο επιτυχημένο Robotics System Invention της εταιρείας, το οποίο έχει βελτιωθεί με την πρόσθεση 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.

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.



Εικ.(1)

Θύρες μηχανών

Το NXT έχει τρεις θύρες παραγωγής για την ένωση των μηχανών – θύρες A, B και Γ θύρες αισθητήρων. *Εικ.(1)*
Το NXT έχει τέσσερις εισαγμένες θύρες για την ένωση των αισθητήρων - θύρες 1, 2, 3 και 4. Συνδέστε ένα καλώδιο USB με το λιμένα USB και μεταφορτώστε τα προγράμματα από τον υπολογιστή σας στο NXT (ή φορτώστε τα στοιχεία από το ρομπότ στον υπολογιστή σας). Μπορείτε επίσης να χρησιμοποιήσετε την ασύρματη σύνδεση Bluetooth για το φόρτωμα και τη μεταφόρτωση. Κάνετε ένα πρόγραμμα μεγάφωνο με τους πραγματικούς ήχους και ακούστε τους όταν τρέχετε το πρόγραμμα .

Αισθητήρας αφής

Ο αισθητήρας αφής δίνει στο ρομπότ σας μια αίσθηση της αφής. Ο αισθητήρας αφής ανιχνεύει πότε πιέζεται από κάτι και πότε απελευθερώνεται πάλι.

Μπορείτε να χρησιμοποιήσετε τον αισθητήρα αφής για να κάνετε το ρομπότ σας να πάρει τα πράγματα: ένας ρομποτικός βραχίονας που εξοπλίζεται με έναν αισθητήρα αφής ενημερώνει το ρομπότ εάν υπάρχει ή όχι κάτι στο βραχίονά του που μπορεί να αρπάξει. Ή μπορείτε να χρησιμοποιήσετε έναν αισθητήρα αφής ώστε το ρομπότ σας να κάνει μια πράξη. Παραδείγματος χάριν, με τη συμπίεση του αισθητήρα αφής μπορείτε να κάνετε το ρομπότ σας να περπατήσει, να μιλήσει, να κλείσει μια πόρτα, ή να ανοίξει τη TV σας.

Αισθητήρας ήχου

Ο υγιής αισθητήρας μπορεί να ανιχνεύσει και τα δύο decibels [DB] και ρυθμισμένο decibel [DBA]. Decibel είναι μια μέτρηση της υγιούς πίεσης. DBA: στην ανίχνευση ρυθμισμένων decibels, η ευαισθησία του αισθητήρα προσαρμόζεται στην ευαισθησία του ανθρώπινου αυτιού. Με άλλα λόγια, αυτοί είναι οι ήχοι που τα αυτιά σας είναι σε θέση να ακούσουν. DB: στην ανίχνευση τυποποιημένων [χωρίς διόρθωση] decibels, όλοι οι ήχοι μετριούνται με την ίδια ευαισθησία. Κατά συνέπεια, αυτοί οι ήχοι μπορούν να περιλάβουν μερικούς που είναι πάρα πολύ υψηλοί ή πάρα πολύ χαμηλοί για να ακουστούν από το ανθρώπινο αυτί. Ο υγιής αισθητήρας μπορεί να μετρήσει τα επίπεδα υγιούς πίεσης μέχρι 90 DB - για το επίπεδο ενός χαρτοκόπτη. Τα επίπεδα υγιούς πίεσης είναι εξαιρετικά περίπλοκα, έτσι οι υγιείς αναγνώσεις αισθητήρων στο MINDSTORMS NXT επιδεικνύονται σε ποσοστό [%]. Όσο χαμηλότερα τα τοις εκατό τόσο πιο ήρεμο το παράδειγμα soundFor: • 4-5% είναι το ποσοστό σε ένα σιωπηλό καθιστικό • 5-10% θα ήταν κάποιος που μιλά κάποια απόσταση μακριά • 10-30% είναι κανονική συνομιλία κοντά στον αισθητήρα ή τη μουσική που παίζεται σε κανονικό επίπεδο • 30-100% είναι να φωνάξει κάποιος άνθρωπος ή μουσική που παίζεται σε μεγάλη ένταση.

Αισθητήρας φωτός

Ο αισθητήρας φωτός είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ σας [ο υπερηχητικός αισθητήρας είναι άλλος]. Ο ελαφρύς αισθητήρας επιτρέπει στο ρομπότ σας 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.

Μπορείτε να χρησιμοποιήσετε τον ελαφρύ αισθητήρα για να κάνετε ένα ρομπότ συναγερμών διαρρηκτών: όταν ένας εισβολέας ανοίγει το φως στο δωμάτιό σας το ρομπότ μπορεί να αντιδράσει για να υπερασπίσει την ιδιοκτησία σας. Μπορείτε επίσης να χρησιμοποιήσετε τον ελαφρύ αισθητήρα για να κάνετε το ρομπότ να ταξινομήσει πράγματα κατά το χρώμα.

Εκτός αυτού, μπορείτε να εξετάστε τη δυνατότητα του ελαφριού αισθητήρα να διαβάσει το περιβαλλοντικό φως με τη μέτρηση του ελαφρού επιπέδου στις διαφορετικές θέσεις του δωματίου. Παραδείγματος χάριν, κρατήστε αρχικά τον αισθητήρα ενάντια στο παράθυρο. Κατόπιν κρατήστε τον στο πλαίσιο του πίνακα.

Αισθητήρας υπέρηχων

Ο αισθητήρας υπέρηχων είναι ένας από τους δύο αισθητήρες που δίνουν όραση στο ρομπότ σας [ο αισθητήρας φωτός είναι άλλος]. Ο υπερηχητικός αισθητήρας επιτρέπει στο ρομπότ σας να δει και να ανιχνεύσει τα αντικείμενα. Μπορείτε επίσης να το χρησιμοποιήσετε για να κάνετε το ρομπότ σας να αποφύγει τα εμπόδια, να αποκτήσει την αίσθηση της απόστασης 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 μια ακρίβεια +/- 3 εκατ. Ο υπερηχητικός αισθητήρας χρησιμοποιεί την ίδια επιστημονική αρχή με τα ρόπαλα: μετρά την απόσταση υπολογίζοντας τον χρόνο που χρειάζεται ένα κύμα για να χτυπήσει ένα αντικείμενο και να επιστρέψει - ακριβώς όπως μια ηχώ. Τα μεγάλου μεγέθους αντικείμενα με τις σκληρές επιφάνειες επιφέρουν τις καλύτερες αναγνώσεις. Τα αντικείμενα φτιαγμένα από μαλακό ύφασμα ή τα κυρτά [όπως μια σφαίρα] ή τα πολύ λεπτά ή μικρά μπορεί να είναι δύσκολο ανιχνευθούν από τον αισθητήρα.

Σερβομηχανές

Οι τρεις σερβομηχανές δίνουν στο ρομπότ σας τη δυνατότητα να κινηθεί. Εάν χρησιμοποιήσετε το φραγμό κίνησης στο λογισμικό LEGO MINDSTORMS NXT για να προγραμματίσετε τις μηχανές σας, οι δύο μηχανές θα συγχρονιστούν αυτόματα, έτσι ώστε το ρομπότ σας να κινηθεί σε μια ευθεία γραμμή.

Αισθητήρας περιστροφής

Κάθε μηχανή έχει έναν ενσωματωμένο αισθητήρα περιστροφής. Αυτό αφήνει τον ακριβή έλεγχο στις μετακινήσεις του ρομπότ σε σας. Ο αισθητήρας περιστροφής μετρά τις περιστροφές μηχανών στους βαθμούς ή τις πλήρεις περιστροφές [ακρίβεια +/- ενός βαθμού]. Μια περιστροφή είναι ίση με 360 βαθμούς, έτσι εάν θέσετε μια μηχανή στη στροφή 180 βαθμών, ο άξονας παραγωγής του θα κάνει μισή στροφή. Ο ενσωματωμένος αισθητήρας περιστροφής σε κάθε μηχανή αφήνει επίσης να θέσετε τις διαφορετικές ταχύτητες για τις μηχανές σας [με τον καθορισμό των διαφορετικών παραμέτρων δύναμης στο λογισμικό].

Υποστηριζόμενες Γλώσσες προγραμματισμού

- RCX Code (περιέχεται στις Mindstorm εκδόσεις λιανικής[παλαιά έκδοση])
- LEGO MINDSTORM NXT SOFTWARE (περιέχεται στις Mindstorm εκδόσεις λιανικής [νέα έκδοση]).
- ROBOLAB (βασίζεται στο LabVIEW και αναπτύχθηκε στο Tufts University)

Δημοφιλείς Γλώσσες τρίτων κατασκευαστών:

- C and C++ under BrickOS (formerly LegOS)
- Java under leJOS or TinyVM
- NQC ("Not Quite C")
- pbFORTH (επεκτάσεις της Forth γλώσσας προγραμματισμού)
- Visual Basic (μέσω του COM+ interface παρεχόμενο με το CD)
- RobotC (νέα γλώσσα συμβατή με την έκδοση NXT)
- Matlab Toolbox
- Κ.α.

<<ΚΕΦΑΛΑΙΟ 2>>

IMAGE ACQUISITION & PROCESSING TOOLBOX

Το Image Processing Toolbox παρέχει 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 , δίνοντάς σας τη δυνατότητα να επιθεωρήσετε τους αλγόριθμους, να τροποποιήσετε τον πηγαίο κώδικα, ακόμη και να δημιουργήσετε τους δικούς σας 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. Μπορείτε να δείτε, να αναλύσετε, και να επεξεργαστείτε 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.

Εισαγωγή και εξαγωγή εικόνων

Υπάρχουν διάφοροι τρόποι για την εισαγωγή ή εξαγωγή εικόνων μέσα και έξω από το περιβάλλον MATLAB για μεταποίηση. Μπορείτε να χρησιμοποιήσετε το Image Acquisition Toolbox για να λάβετε ζωντανά εικόνες από κάμερες Web, frame grabbers, DCAM-συμβατές κάμερες, και άλλες συσκευές. Με την χρήση Database Toolbox, μπορείτε να έχετε 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. Χαμηλού επιπέδου I / O λειτουργίες σας επιτρέπουν να αναπτύξετε 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.

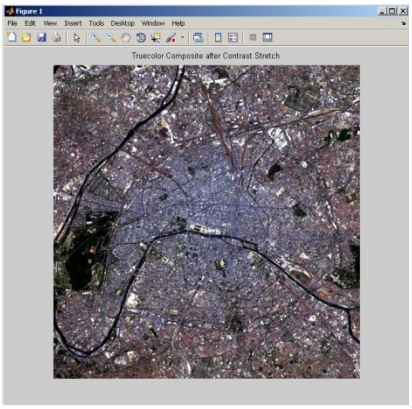
Πριν και μετά την Επεξεργασία Εικόνων

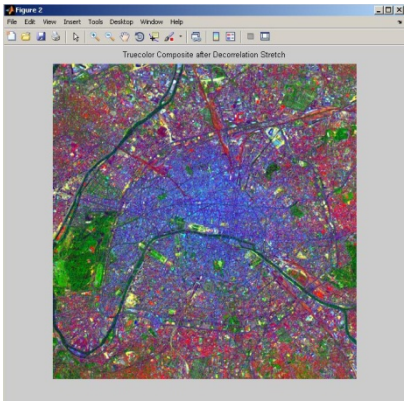
Η εργαλειοθήκη προσφέρει αλγόριθμους για την πριν-και-μετά επεξεργασία, επιλύοντας συχνά προβλήματα του συστήματος, όπως η παρεμβολή του θορύβου, η χαμηλή δυναμική περιοχή, η μη επικεντρωμένη οπτική, και η διαφορά στη χρωματική απεικόνιση μεταξύ συσκευών εισόδου και εξόδου.

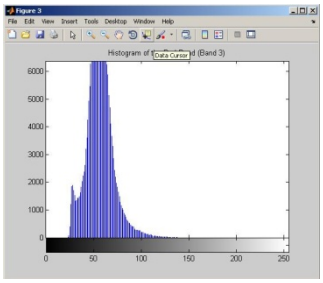
Ενίσχυση Εικόνων

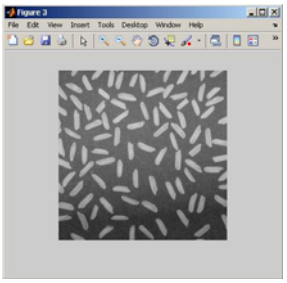
Οι τεχνικές βελτίωσης της εικόνας στο Image Processing Toolbox επιτρέπουν να αυξήσετε το λόγο του σήματος προς τον θόρυβο και να επιτείνετε τα χαρακτηριστικά της εικόνας με την τροποποίηση του χρώματος ή της έντασης της εικόνας. Μπορείτε επίσης να κάνετε τα εξής:

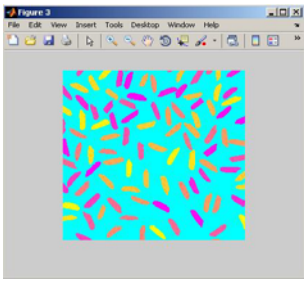
Εκτέλεση εξίσωση ιστογράμματος .
Decorrelation stretching.
Επαναπροσδιορισμός της δυναμικής περιοχής .
Ρύθμιση της αξίας γάμμα.
Εκτέλεση γραμμικών ή προσαρμοστικών φίλτρων.







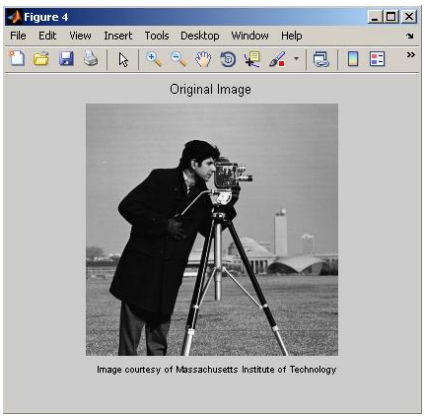


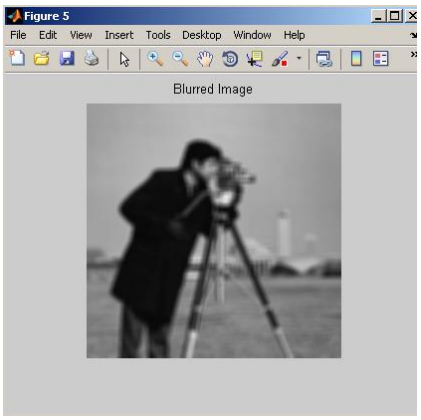


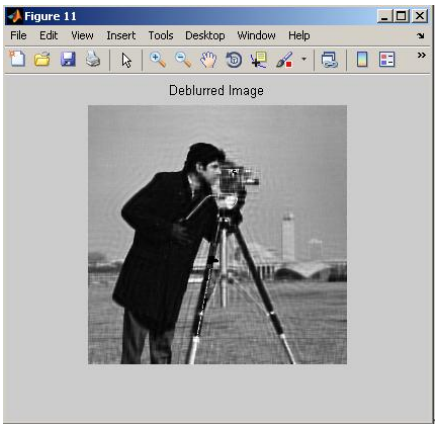
Η εργαλειοθήκη περιλαμβάνει επίσης εξειδικευμένες και γενικευμένες πολυδιάστατες ρουτίνες φιλτραρίσματος που χειρίζονται δεκαδικές εικόνες , καθώς και συνέλιξη και συσχέτιση.

Deblurring Εικόνων

Υποστηρίζει διάφορους θεμελιώδεις αλγορίθμους deblurring (ξεθόλωμα),συμπεριλαμβανομένων των: Lucy-Richardson, Wiener, και φίλτρο deconvolution, καθώς και μετατροπές μεταξύ σημείων εξάπλωσης και λειτουργική μεταφορά οπτικής. Οι εν λόγω λειτουργίες βοηθούν στο blurring που προκαλείται στην εικόνα από την out-of-focus εστίαση, την κίνηση της κάμερας ή του θέματος λήψης , των ατμοσφαιρικών συνθηκών, την σύντομη έκθεση του χρόνου, άλλα και από άλλους παράγοντες κατά τη διάρκεια της λήψης εικόνων . Όλοι οι αλγόριθμοι deblurring δουλεύουν με πολυδιάστατες εικόνες.







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, μπορείτε να μετατρέψετε τις εικόνες ανεξαρτήτου συσκευής και χρωματικού τύπου , όπως sRGB, XYZ, xyY, L * α * β *, uvL,.

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, η Bayer κωδικοποιημένα πρότυπα, και υψηλό δυναμικό εύρος εικόνων.

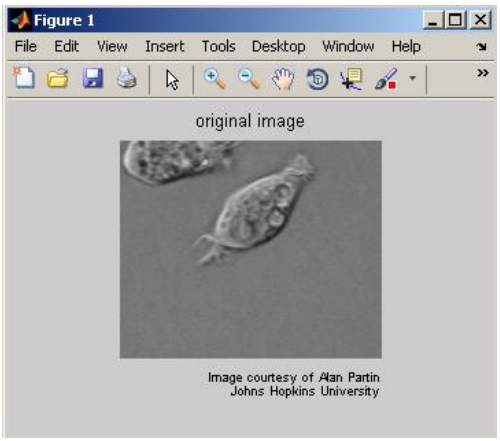
Αναλύοντας Εικόνες

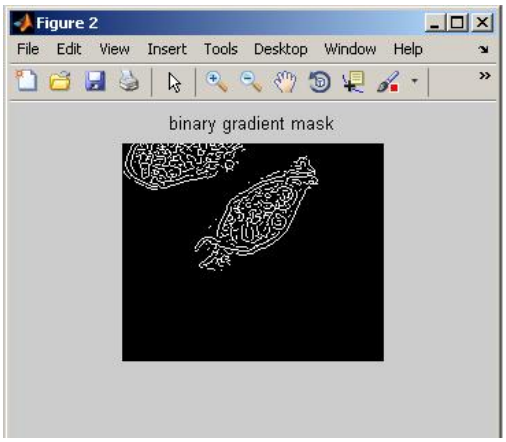
Παρέχει μία περιεκτική ακολουθία αλγορίθμων και γραφικών εργαλείων για θέματα ανάλυσης εικόνων, όπως είναι η ανάλυση στατιστικών στοιχείων, εξαγωγή χαρακτηριστικών, και μετρήσεις.

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

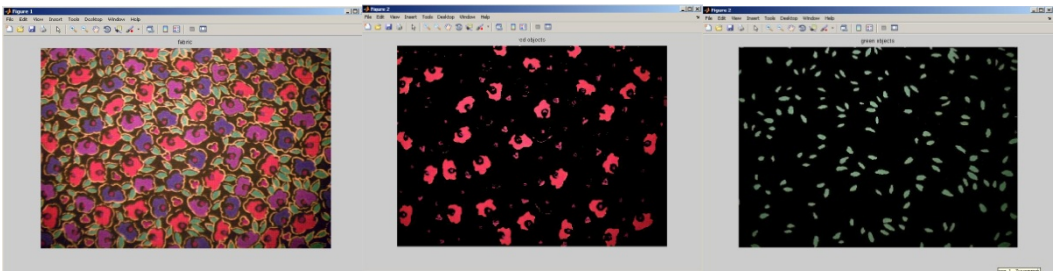
Τον καθορισμό των τιμών έντασης κατά μήκος ενός τμήματος γραμμής.
Την εμφάνιση του ιστογράμματος μιας εικόνας
Την εμφάνιση ενός προφίλ των τιμών έντασης .

*Οι **αλγόριθμοι εντοπισμού ακμών*** 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.





Οι **αλγόριθμοι κατάτμησης** 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.

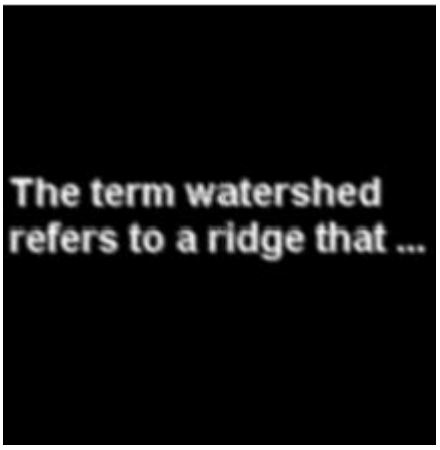


Οι ***μορφολογικές ενέργειες*** μας επιτρέπουν να ανιχνεύσουμε τις ακμές, να ενισχύσουμε την αντίθεση, να αφαιρέσουμε τον θόρυβο, να κατατμήσουμε μια εικόνα σε περιοχές, και να δημιουργήσουμε λεπτές περιοχές. Άλλες μορφολογικές λειτουργίες που περιλαμβάνονται στην εργαλειοθήκη είναι:

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

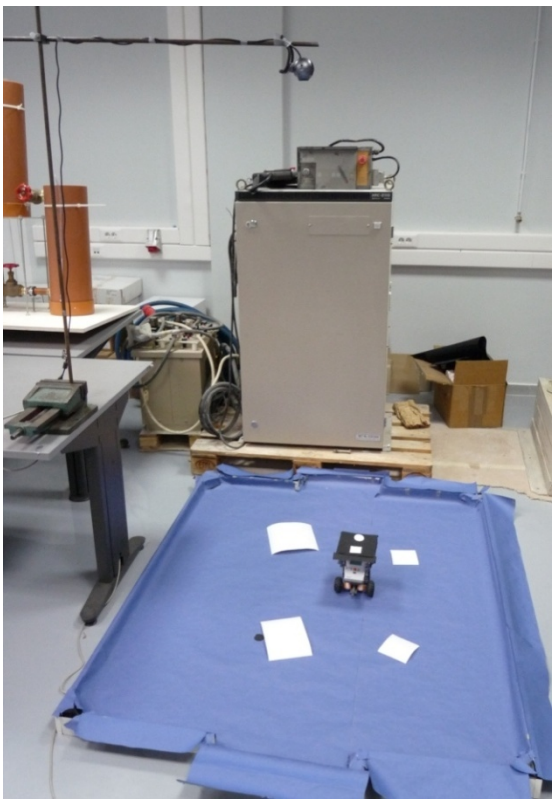
Η εργαλειοθήκη επίσης μας προσφέρει κάποιες προηγμένες λειτουργίες για την ανάλυσης της εικόνας που μας επιτρέπουν να μετρήσουμε τις ιδιότητες μιας συγκεκριμένης περιοχής της εικόνας , όπως είναι το εμβαδόν, και το κέντρο της μάζας.

Ανίχνευση γραμμών και τμήματα γραμμών από μια εικόνα, γίνονται με την χρησιμοποίηση του μετασχηματισμού Hough. Η χρήση του μας επιτρέπει να μετρήσουμε την επιφανειακή τραχύτητα ή την παραλλαγή των χρωμάτων μίας εικόνας.



Υλοποίηση του image processing & acquisition toolbox

Η χρήση της συγκριμένης εργαλειοθήκης είχε καταλυτικό ρόλο στην υλοποίηση της εργασίας έχοντας ως μοναδικό αισθητήριο του ρομπότ την κάμερα. Μέσω της επεξεργασίας της εικόνας λαμβάναμε τα επιθυμητά μας δεδομένα για την επίτευξη του εκάστοτε στόχου μας. Η συσκευή εισόδου εικόνας που χρησιμοποιήσαμε ήταν μια 'Logitech QuickCam Pro 5000' web camera, την οποία και στήσαμε έτσι ώστε να βλέπει από πάνω την πλατφόρμα στην οποία θα κινείτε το ρομπότ. *ΕΙΚ.(2)*



ΕΙΚ.(2)

Ένα άλλο βασικό κομμάτι για την υλοποίηση της εργασίας ήταν η εύρεση μια λύσης για την αναγνώριση του ρομπότ μέσα στην πλατφόρμα. Η λύση που βρήκαμε ήταν να κατασκευάσουμε ένα «καπέλο» για το ρομπότ το οποίο θα είχε πάνω του δύο γεωμετρικά σχήματα, ένα τετράγωνο και έναν κύκλο λίγο μικρότερο από το τετράγωνο, έτσι ώστε να γνωρίζουμε αρχικά τον προσανατολισμό του ρομπότ καθώς και άλλων ιδιοτήτων τις οποίες θα αναλύσουμε λεπτομερώς παρακάτω.

Ιδιότητες Κάμερας

Για να λάβουμε μια εικόνα στην matlab έπρεπε πρώτα να δούμε πώς αναγνωρίζει η matlab την κάμερα και τι δυνατότητες-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 συναρτήσεων να αλλάξουμε τυχόν ανεπιθύμητες ιδιότητες της κάμερας ή να σετάρουμε την κάμερα, για να απεικονίζει καλύτερα την εικόνα που μας επιστρέφει.

Λήψη εικόνας

Κάνοντας κατάλληλες ρυθμίσεις είμαστε έτοιμοι να πάρουμε εικόνα. Για την λήψη της εικόνας κάνουμε χρήση τριών συναρτήσεων. Της *preview* , *getsnapshot* και *imshow*.

- Η *preview* μας εμφανίζει την προεπισκόπηση της κάμεράς μας, δίνοντάς της όρισμα το αντικείμενο που έχουμε δημιουργήσει. (*preview(vid))*

- Η *getsnapshot* επιστρέφει το frame την ώρα που την καλούμε. *(I2 = getsnapshot(vid))*

- Για να εμφανίσουμε το frame που έχουμε λάβει από το κάλεσμα της *getsnapshot* χρησιμοποιούμε την *imshow* η οποία μας εμφανίζει το επιθυμητό μας frame. *(imshow(I2))*

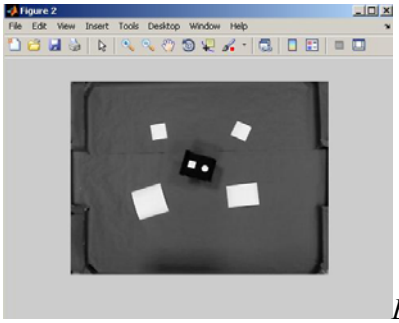
Έχοντας φτιάξει τις ρυθμίσεις και έπειτα λάβει την εικόνα, είμαστε έτοιμοι να δούμε πώς θα επεξεργαστούμε την εικόνα μας ώστε να λάβουμε τις πληροφορίες που θέλουμε.

Επεξεργασία εικόνας

Για την επεξεργασία της εικόνας έχουμε φτιάξει μια συνάρτηση, την *prosnap*, η οποία παίρνει ως όρισμα την εικόνα που θέλουμε να επεξεργαστούμε και επιστρέφει το label της επεξεργασμένης μας εικόνας για να μπορέσουμε να το χρησιμοποιήσουμε σε μία συνάρτηση με το όνομα *regionprops*,την οποία θα αναλύσουμε παρακάτω.

Επεξήγηση *prosnap*

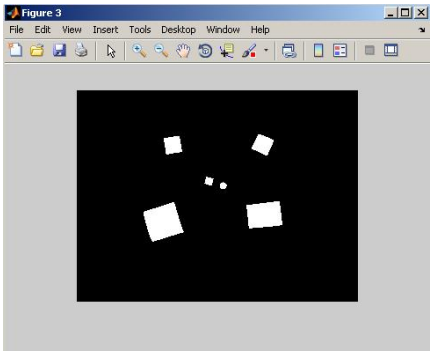
Μετατρέπουμε την αρχική μας εικόνα σε κλίμακα του γκρι για να περιορίσουμε τα χρώματα έτσι ώστε να είναι ευκολότερη η επεξεργασία της. *ΕΙΚ.(3)*



ΕΙΚ.(3)

Εικόνα σε grayscale

Βρίσκουμε το κατάλληλο κατώφλι για να αποκόψουμε κάποιες τιμές χρωμάτων, για να γίνει έπειτα σωστή απεικόνιση της δυαδικής εικόνας. Παίρνοντας το κατώφλι που έχουμε βρει προηγουμένως και την εικόνα που έχουμε μετατρέψει σε κλίμακα του γκρι, μπορούμε πλέον να μετατρέψουμε την εικόνα μας σε δυαδική μορφή. *ΕΙΚ.(4)*



ΕΙΚ.(4)

Εικόνα σε δυαδική μορφή

Αφαιρούμε τυχόν μικρά αντικείμενα, που μπορεί να μην έχουν φύγει με την χρήση του κατωφλιού και έπειτα κλείνουμε τα κενά που μπορεί να υπάρχουν μεταξύ των σχημάτων μας. Στην συνέχεια καθαρίζουμε την περιοχή γύρω από τα όρια των σχημάτων και το τελικό βήμα στην επεξεργασία της εικόνας μας είναι, να λάβουμε την κατάλληλη ετικέτα της εικόνας, για επιστραφούν τα σωστά αποτελέσματα στις συναρτήσεις που χσησιμοποιούμε.

Επεξήγηση *regionprops*

Η συγκεκριμένη συνάρτηση έχει την ικανότητα να μπορεί να κάνει διάφορες μετρήσεις σε μία εικόνα. Οι μετρήσεις της χωρίζονται σε δύο κατηγορίες : στις μετρήσεις σχημάτων και στις μετρήσεις των τιμών έντασης των pixel. Κάποιες ιδιότητες είναι:

- Μετρήσεις σχημάτων:

    1. Εύρεση εμβαδού.
    2. Εύρεση κέντρων.
    3. Εύρεση διαμέτρου.
    4. Εύρεση αριθμό Euler.
    5. Εύρεση συντεταγμένες των pixel.
    6. Εύρεση του προσανατολισμού.

- Μετρήσεις τιμών pixel:

    1. Μεγαλύτερη τιμή στην εικόνα.
    2. Μικρότερη τιμή στην εικόνα.
    3. Μέση τιμή της εικόνας.

Η συνάρτηση μας βοήθησε πολύ στην επίλυση του προβλήματος, για το πώς θα αναγνωρίζουμε το ρομπότ μέσα στην πλατφόρμα. Για την εργασία μας χρησιμοποιήσαμε τέσσερεις ιδιότητες της *regionprops:*

1. `Centroid`: Επιστρέφει έναν πίνακα με τις x, y συντεταγμένες των αντικειμένων.
2. `Area`: Επιστρέφει έναν αριθμό με βάση τον αριθμό των pixel από το οποίο αποτελείται ένα αντικείμενο.
3. `PixelList`: Επιστρέφει τις συντεταγμένες των pixel που βρίσκονται στην εικόνα.
4. `PixelValues`: Επιστρέφει τις τιμές των pixel που βρίσκονται στην εικόνα σε αντιστοιχία με τον πίνακα που μας δίνει η *PixelList.*

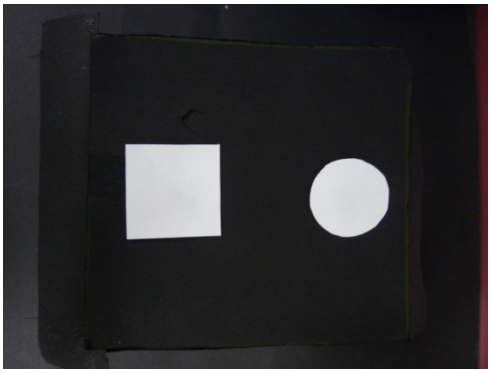
Οι δύο πρώτες ιδιότητες χρησιμοποιούνται για την εύρεση της γωνίας που σχηματίζει το ρομπότ με το εκάστοτε επιθυμητό μας σημείο, βοηθώντας το έτσι να κάνει τις σωστές κινήσεις.

Οι δύο τελευταίες ιδιότητες χρησιμοποιούνται στον αλγόριθμο του GNG , όπου βρίσκουν τα εμπόδια με τιμή ίση με ένα και την υπόλοιπη πλατφόρμας με τιμή ίση με μηδέν.

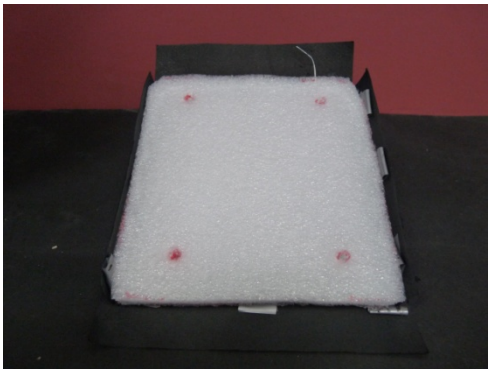
Περισσότερες λεπτομέρειες δίνονται στα παρακάτω κεφάλαια

Επεξήγηση καπέλου

Το καπέλο που κατασκευάσαμε φαίνεται παρακάτω. Στην *ΕΙΚ.(5)* απεικονίζεται το πάνω μέρος, ενώ στην *ΕΙΚ.(6)* το κάτω. Το πάνω μέρος αποτελείται το τετράγωνο και τον κύκλο κομμένα σε άσπρο χαρτόνι, ενώ το κάτω μέρος αποτελείται από ένα κομμάτι φελιζόλ κομμένο έτσι ώστε να καλύπτει όλο το ρομπότ και είναι ντυμένο με μαύρο χαρτόνι.



ΕΙΚ.(5)



ΕΙΚ.(6)

Με την χρήση του καπέλου και σε συνδυασμό με την *regionprops* μπορούσαμε αρχικά να διακρίνουμε πού είναι το μπροστά και πού είναι το πίσω μέρος του ρομπότ. Το καπέλο έχει τοποθετηθεί έτσι ώστε το τετράγωνο δείχνει το μπροστινό μέρος και ο κύκλος το πίσω μέρος του ρομπότ.

Χρησιμοποιώντας λοιπόν την ιδιότητα *Area* της *regionprops* μπορούμε να βρούμε εύκολα και γρήγορα πού είναι το μπροστά και το πίσω μέρος του ρομπότ, συγκρίνοντας απλά το μέγεθος των αντικειμένων μας.

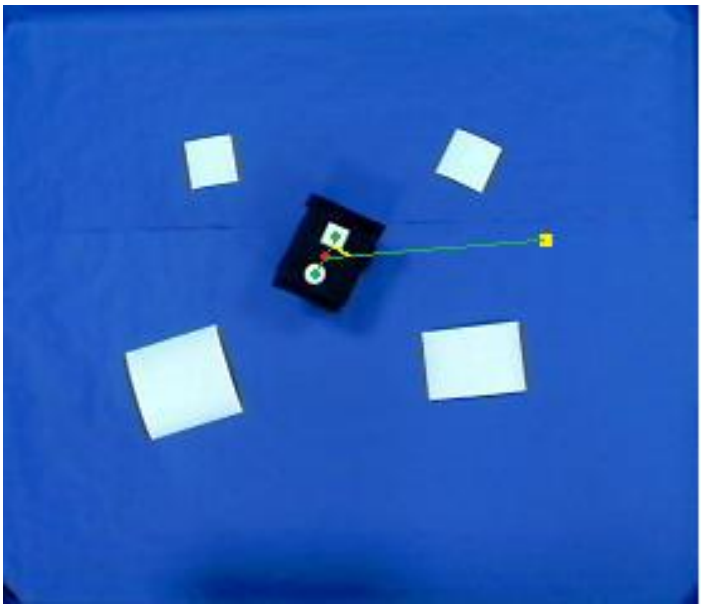
Αντιθέτως χρησιμοποιώντας την ιδιότητα *Centroid* μπορούμε να βρούμε τα κέντρα των γεωμετρικών μας σχημάτων. Βρίσκοντας τα κέντρα των σχημάτων, μπορούμε να βρούμε την ευθεία που ενώνει τα κέντρα, άρα μπορούμε να βρούμε και την κλίση της ευθείας αυτής καθώς και την κλίση που έχει το ρομπότ άρα και την γωνία που σχηματίζει με τον άξονά x.

Καλύτερη κατανόηση των παραπάνω γίνεται στο επόμενο κεφάλαιο, όπου θα αναλύσουμε πλήρως πώς χρησιμοποιούμε τις πληροφορίες που λαμβάνουμε με την χρήση της *regionprops* για την εύρεση της κλίσης και της γωνίας κίνησης του ρομπότ.

Η χρήση των δεδομένων δεν είναι πάντα η ίδια. Αλλιώς χρησιμοποιούμε τα δεδομένα για την κίνηση του ρομπότ με κίνηση PID και αλλιώς με κίνηση του GNG αλγορίθμου. Στα εκάστοτε κεφάλαια αναφέρεται ο τρόπος με τον οποίο γίνεται η χρήση των δεδομένων.

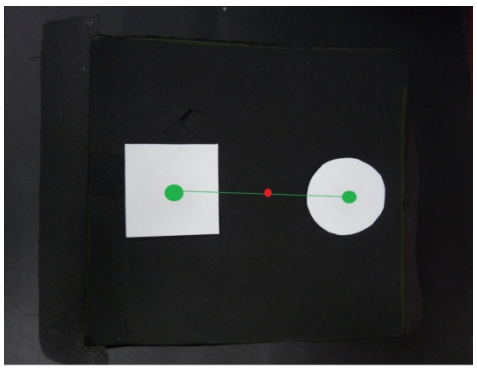
<<ΚΕΦΑΛΑΙΟ 3>>

ΕΥΡΕΣΗ ΓΩΝΙΑΣ ΜΕ ΣΗΜΕΙΟ

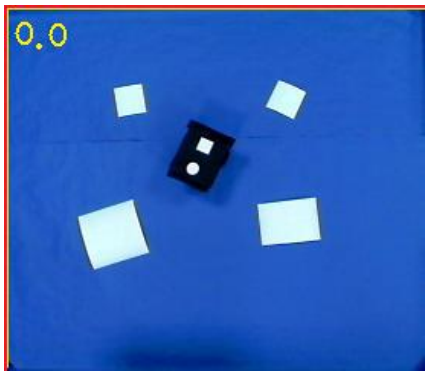


Στο κεφάλαιο αυτό θα αναλύσουμε τον τρόπο με τον οποίο καταφέραμε να ευθυγραμμίσουμε το ρομπότ με τα επιθυμητά μας σημεία. Θα δούμε αναλυτικότερα τις περιπτώσεις που πήραμε, τους μαθηματικούς τύπους που χρησιμοποιήσαμε για την εύρεση της κλίσης του ρομπότ και των γωνιών που σχηματίζει με τα εκάστοτε επιθυμητά σημεία.

Ξεκινώντας λοιπόν πρέπει να έχουμε ένα σταθερό σημείο αναφοράς και ένα σύστημα αξόνων για να μπορούμε να βγάζουμε σωστά αποτελέσματα. Ως σημείο αναφοράς πήραμε την μέση τιμή της ευθείας *ΕΙΚ.(7)* που περνάει από τα δυο *Centroid* και ως σύστημα αξόνων πήραμε αυτό της κάμερας. *ΕΙΚ.(8)*



ΕΙΚ.(7)



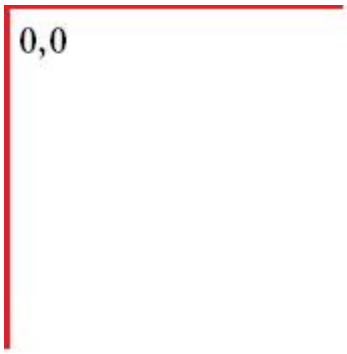
ΕΙΚ.(8)

Έχοντας κάνει λοιπόν τα παραπάνω ήρθαμε αντιμέτωποι με το εξής πρόβλημα: πώς θα γνωρίζουμε πού θα βρίσκεται το επιθυμητό μας σημείο σε σχέση με το ρομπότ, ούτως ώστε να ξέρει προς ποια κατεύθυνση πρέπει να στρίψει για να ευθυγραμμιστεί;

Έπειτα λοιπόν από διάφορους πειραματισμούς καταλήξαμε στην εξής λύση: να θεωρήσουμε ως αρχή των αξόνων την μέση τιμή της ευθείας που αναφέραμε παραπάνω. Κάνοντάς το, μπορούμε πλέον να γνωρίζουμε που βρίσκεται το κάθε σημείο σε σχέση με το ρομπότ συγκρίνοντας τις συντεταγμένες του σημείου με αυτές της μέσης τιμής. Πρέπει όμως να δώσουμε ιδιαίτερη προσοχή στο σύστημα συντεταγμένων που επιλέγουμε να δουλέψουμε.

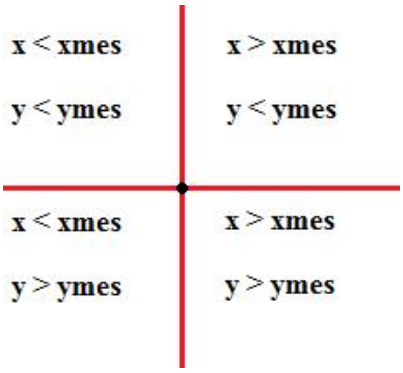
Δουλεύοντας τους πειραματισμούς μας και τις περιπτώσεις μας στο πρόγραμμα *Geogebra,* που παρέχεται δωρεάν, με τον γνωστό άξονα συντεταγμένων που χρησιμοποιείται σε όλες τις μαθηματικές παραστάσεις, υπήρχαν κάποιες διαφορές κατά την εφαρμογή τους στο matlab. Οι διαφορές αυτές ήταν αρκετές για να μην λειτουργεί σωστά το πρόγραμμά μας. Παρατηρήσαμε λοιπόν ότι η αιτία των διαφορών αυτών ,ήταν ότι το σύστημα συντεταγμένων ήταν ανάστροφο. Ο κανονικός *ΕΙΚ.(9)* και ο ανάστροφος *ΕΙΚ.(10)* άξονας φαίνεται παρακάτω.

ΕΙΚ.(9)

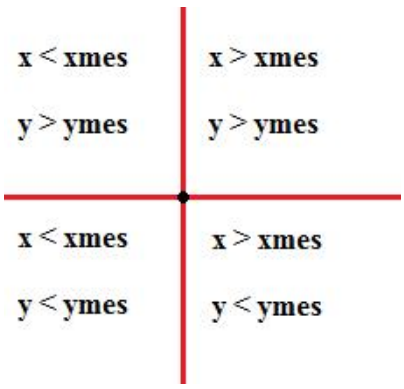


ΕΙΚ.(10)

Βρίσκοντας λύση στο παραπάνω πρόβλημα μπορούσαμε τώρα να δούμε την τελική διαμόρφωση *ΕΙΚ.(11)* των συγκρίσιμων τιμών μεταξύ των σημείων και της μέσης τιμής (που θεωρούμε ως αρχή των αξόνων) σε σχέση με τα αποτελέσματα που πήραμε στον κανονικό άξονα συντεταγμένων. *ΕΙΚ.(12)*



ΕΙΚ.(11)



ΕΙΚ.(12)

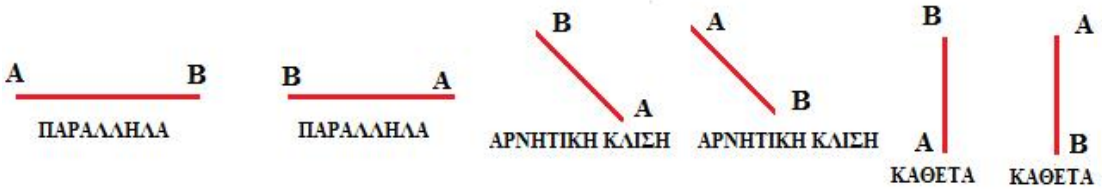
Όπου x, y είναι οι συντεταγμένες των σημείων και xmes, ymes οι συντεταγμένες της μέσης τιμής.

Εύρεση γωνίας

Έχοντας δει τα παραπάνω είμαστε έτοιμοι να δούμε με ποια μεθοδολογία βρίσκουμε την γωνία που μας ενδιαφέρει. Για την εύρεση της γωνίας έχουμε δημιουργήσει μια συνάρτηση με το όνομα *Euresi_gonias* στην οποία εμπεριέχονται όλες οι περιπτώσεις που μπορεί να πάρει η κλίση του ρομπότ. Οι περιπτώσεις είναι έξι όταν η κλίση είναι θετική και έξι όταν η κλίση είναι αρνητική. Στο πρόγραμμά μας αντί για κύκλο και τετράγωνο έχουμε βάλει Α και Β αντίστοιχα, δηλαδή όταν θα λέμε Α θα εννοούμε τον κύκλο και όταν θα λέμε Β θα εννοούμε το τετράγωνο. Οι περιπτώσεις με βάση την κλίση φαίνονται παρακάτω:

Για κλίση θετική:

Για κλίση αρνητική:



Η εύρεση της γωνίας γίνεται με βάση το πρόσημο της κλίσης και της θέσης που βρίσκεται το επιθυμητό σημείο με βάση το ρομπότ. Βρίσκοντας έπειτα ποια συνθήκη ικανοποιεί τα δεδομένα μας καλούμε την συνάρτηση που της αντιστοιχεί. Οι συναρτήσεις για την κάθε περίπτωση φαίνονται παρακάτω, από τις οποίες θα αναλύσουμε μια, για καλύτερη κατανόηση του τρόπου εύρεσης της γωνίας, μιας και οι υπόλοιπες συναρτήσεις ακολουθούν την ίδια μεθοδολογία.

Μεθοδολογία εύρεσης γωνίας

Για να βρούμε την κλίση της ευθείας χρησιμοποιούμε τον τύπο *λ= (y2-y1) / (x2-x1)* όπου αντί για y1,y2 και x1,x2 βάζουμε τις συντεταγμένες της ευθείας και της μέσης τιμής. Έπειτα επειδή το αποτέλεσμά μας είναι σε ακτίνια, και εμείς το θέλουμε σε μοίρες, το μετατρέπουμε με την χρήση κατάλληλης συνάρτησης. Στην συνέχεια για να πάρουμε την κατάλληλη γωνία με βάση την θέση του ρομπότ παίρνουμε περιπτώσεις συγκρίνοντας τις συντεταγμένες των Α και Β (του κύκλου και του τετραγώνου) σε σχέση με την κλίση που έχουμε βρει προηγουμένως. Τέλος για την εύρεση της γωνίας καλούμε την κατάλληλη συνάρτηση, έχοντας ικανοποιήσει πρώτα την αντίστοιχη συνθήκη.

Ανάλυση BpanoapoAmeKthet

Η παρούσα συνάρτηση, όπως και όλες οι υπόλοιπες, παίρνει ως όρισμα όλα τα στοιχεία εκείνα τα οποία σχετίζονται με τα σημεία και το ρομπότ για την σωστή εύρεση της γωνίας. Έχοντας βάλει διαφορετικές ονομασίες στα ορίσματα της συνάρτησης για να αποφύγουμε τυχόν προβλήματα με μεταβλητές ίδιου ονόματος έχουμε φτιάξει τον παρακάτω πίνακα αντιστοίχισης των μεταβλητών *ΕΙΚ.(13)*:

x , y	xmes , ymes	x1 , y1	x2 , y2
xm , ym	xs , ys	xss , yss	xss2 , yss2

ΕΙΚ.(13)

Πίνακας αντιστοίχισης μεταβλητών

Έχοντας βρει την σωστή θέση του ρομπότ, μας μένει μόνο να βρούμε την θέση του σημείου που θέλουμε σε σχέση με το ρομπότ. Για τον λόγο αυτό, παίρνουμε πάλι περιπτώσεις συγκρίνοντας αυτήν την φορά τις συντεταγμένες της μέση τιμής με εκείνες του σημείου, βρίσκοντας έτσι σε ποιο τεταρτημόριο είναι το επιθυμητό σημείο. Έπειτα, χρησιμοποιούμε τον τύπο $f = \frac{||y_{\mu\varepsilon\tau}\beta\lambda - y_{\sigma\tau}\alpha\theta||}{\sqrt{(x_{\mu\varepsilon\tau}\beta\lambda - x_{\sigma\tau}\alpha\theta)^2 + (y_{\mu\varepsilon\tau}\beta\lambda - y_{\sigma\tau}\alpha\theta)^2}}$ για να βρούμε την γωνία που κυμαίνεται μεταξύ 90 και -90 μοίρες. Μετατρέπουμε την γωνία πάλι σε μοίρες και έπειτα αφαιρούμε ή προσθέτουμε αναλόγως την θέση που βρίσκεται το σημείο, την κλίση ή την παραπληρωματική της γωνία, τοποθετώντας παράλληλα και το σωστό πρόσημο επιστροφής της γωνίας.

Έχουμε δημιουργήσει έτσι το πρόγραμμά μας ώστε αν το σημείο είναι στην δεξιά πλευρά του ρομπότ (βλέποντάς το από πίσω) να μας επιστρέφει αρνητική γωνία, και έτσι το nxt καταλαβαίνει ότι πρέπει να κινηθεί δεξιά, αντίθετα αν η γωνία μας είναι θετική το nxt γνωρίζει ότι πρέπει να κινηθεί αριστερά.

<<ΚΕΦΑΛΑΙΟ 4>>

MINDSTORM NXT TOOLBOX

Η Mindstorms NXT εργαλειοθήκη για MATLAB παρέχει επικοινωνία με το LEGO Mindstorms NXT ρομπότ μέσω Bluetooth ή USB. Η εργαλειοθήκη περιλαμβάνει ρουτίνες για αλληλεπίδραση μεταξύ ρομπότ και MATLAB. Κάποιες από τις ιδιότητές της είναι:

* Άνοιγμα και κλείσιμο της σύνδεσης Bluetooth ή USB .
* Αποστολή και λήψη δεδομένων μεταξύ του ρομπότ και MATLAB .
* Υψηλό επίπεδο μηχανοκίνητου ελέγχου του NXT.
* Υψηλό επίπεδο ανάγνωσης αισθητήρων NXT.
* Σύστημα εντολών NXT.
* Άμεσες εντολές NXT.
* Χαμηλού επιπέδου εντολές και βοήθειες.

Οι απαιτήσεις του συστήματος για να λειτουργήσει η εργαλειοθήκη είναι:

* Λειτουργικό σύστημα: Windows ή Linux
* MATLAB Version 7.3 (R2006b) ή υψηλότερη έκδοση για τις βασικές λειτουργίες (κλασσικός έλεγχος κινητήρων).
* MATLAB Version 7.6 (R2008a) ή υψηλότερη έκδοση για βελτιωμένο έλεγχο κινητήρων (πιο ακριβή).
* LEGO Mindstorms NXT κιτ (π.χ. Εκπαιδευτικό κιτ)
* LEGO Mindstorms NXT firmware v1.05 (συνιστάται)
* Προσαρμογέας Bluetooth 2.0 μοντέλο συνιστάται από LEGO ® (π.χ. AVM BlueFRITZ! USB) με υποστήριξη σειριακής θύρας (SPP)
* Καλώδιο USB και LEGO Mindstorms NXT Fantom USB οδηγό για τα Windows (http://mindstorms.lego.com/support/updates/) ή libusb βιβλιοθήκη για Linux.

Πλεονεκτήματα του Toolbox.

* Αύξηση της διαθέσιμης ισχύος CPU και της μνήμης .
* Απεριόριστα μεγέθη προγραμμάτων (σε σύγκριση με τα κλασικά προγράμματα NXT) .
* Έλεγχος πολλαπλών ρομπότ μέσα από ένα ενιαίο πρόγραμμα (με τον μοναδικό περιορισμό τον αριθμό των ταυτόχρονα εγκατασταθέντων προσαρμογέων Bluetooth)
* Χρησιμοποίηση επιπλέον εξοπλισμού, π.χ. webcamera, χειριστήρια, κλπ.
* Προχωρημένα χαρακτηριστικά debugging υποστηρίζονται πλήρως, όπως για παράδειγμα , βήμα προς βήμα εκτέλεση ή λειτουργία μεταβλητού ελέγχου.
* Πλεονεκτήματα της δικτύωσης και του διαδικτύου στη ρομποτικές σας εφαρμογές.
* Εντυπωσιακή και καλά τεκμηριωμένες 2D και 3D αναπαραστάσεις των δεδομένων.
* Χρήση ήδη υπάρχοντα προγράμματα, εργαλειοθήκες ή βιβλιοθήκες, π.χ. νευρωνικά δίκτυα, επεξεργασία εικόνας, βάσεις δεδομένων.

Την κίνηση του NXT την έχουμε ελέγξει με δύο τρόπους. Ο ένας είναι η κίνηση με PID έλεγχο και ο άλλος με τον αλγόριθμο GNG. Στο παρόν κεφάλαιο θα αναλύσουμε την κίνηση με PID, ενώ με κίνηση GNG αναλύεται στο αντίστοιχο κεφάλαιο.

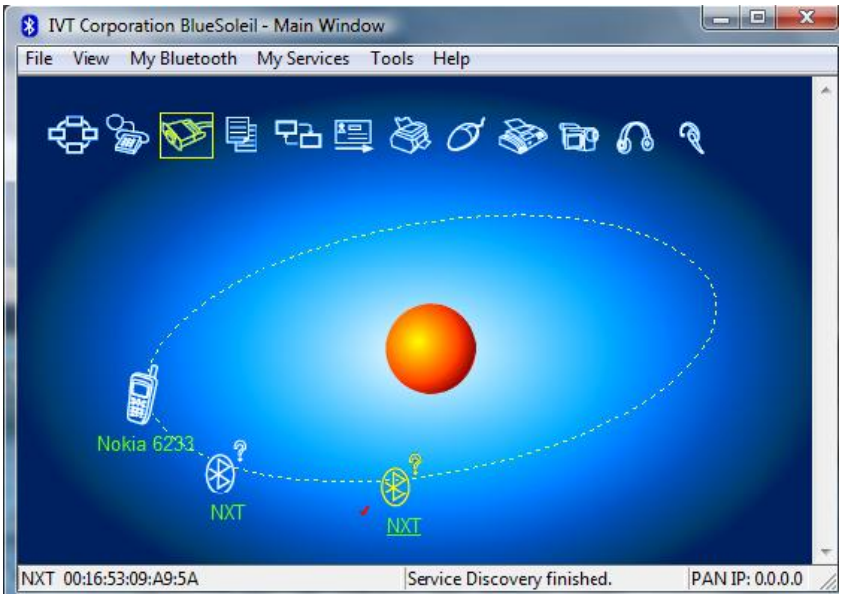
Πριν ξεκινήσουμε με την ανάλυση της κίνησης πρέπει πρώτα να δούμε πώς συνδέουμε τον NXT με τον ηλεκτρονικό υπολογιστή ώστε να μπορούμε να του στέλνουμε εντολές, καθώς και τί είναι ο PID ελεγκτής.

Σύνδεση NXT με υπολογιστή μέσω Bluetooth.

Για την σύνδεση του NXT χρησιμοποιήσαμε ένα Bluetooth που συνίσταται από την lego, το πρόγραμμα BlueSoleil version 2.6.09 και τον κώδικα της Mindstorm Nxt εργαλειοθήκης.

Βήμα 1ο: *Σύνδεση NXT με BlueSoleil.*

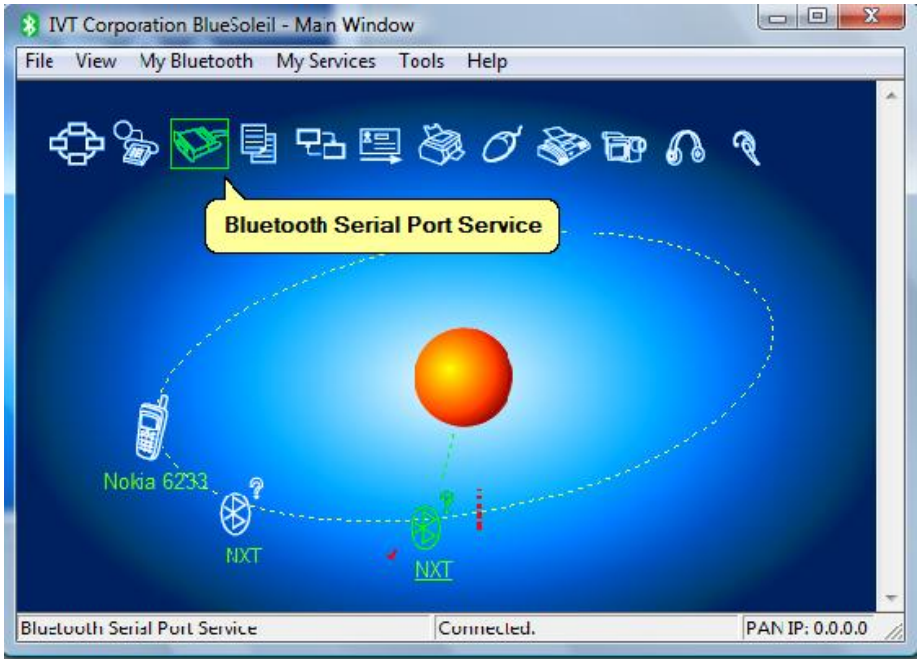
Ανοίγουμε το BlueSoleil πρόγραμμα και πατάμε πάνω στον πορτοκαλί κύκλο για να ξεκινήσουμε το σκανάρισμα και την ανίχνευση των Bluetooth συσκευών. *ΕΙΚ.(14)*



ΕΙΚ.(14)

Βήμα 2ο: *Επιλογή Συσκευής*

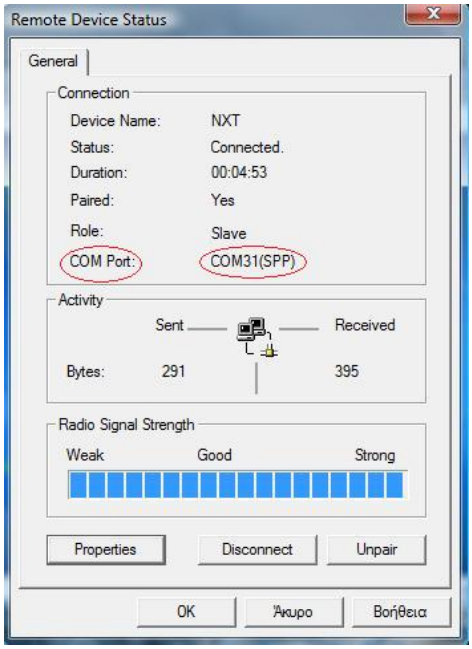
Έχοντας σκανάρει και ανιχνεύσει για συσκευές επιλέγουμε αυτήν που επιθυμούμαι, δηλαδή την NXT, και έπειτα πατάμε πάνω στο κουμπί *Bluetooth Serial Port Service* για να ξεκινήσουμε την σύνδεση μεταξύ του NXT και του ηλεκτρονικού υπολογιστή. *ΕΙΚ.(15)*



ΕΙΚ.(15)

Βήμα 3ο: *Εύρεση αριθμού πόρτας*

Έπειτα κάνουμε δεξί κλικ πάνω στην συσκευή μας και πατάμε *Properties* για να δούμε σε ποια πόρτα COM έχει συνδεθεί, έτσι ώστε να μπορέσουμε να φτιάξουμε το αρχείο bluetooth.ini. *ΕΙΚ.(16)*



ΕΙΚ.(16)

Βήμα 4ο: *Δημιουργία συνάρτησης σύνδεσης NXT με Matlab.*

Για την σύνδεση του matlab με το nxt έχουμε φτιάξει την συνάρτηση με όνομα SindesiPcMeNxt.

PID ελεγκτής

Αποτελείται από τρείς όρους:

Τον αναλογικό P – Proportional

Τον ολοκληρωτικό I – Integral

Τον διαφορικό D – Derivative

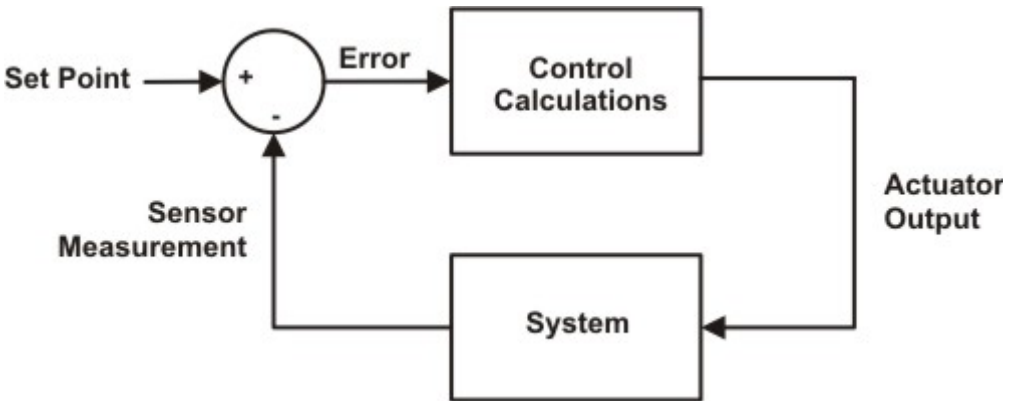
Σε αυτό το σημείο θα παρουσιάσουμε τα χαρακτηριστικά τους καθώς και τον τρόπο με τον οποίο θα πρέπει να τους χρησιμοποιήσουμε για να πετύχουμε την επιθυμητή απόκριση στο σύστημα μας.

Λίγα θεωρητικά:

Η συνάρτηση μεταφοράς ενός PID ελεγκτή είναι:

$$\frac{Y(s)}{R(s)} = K_p + \frac{K_i}{s} + K_d s = \frac{K_d s^2 + K_p s + K_i}{s}$$

όπου Kp = Αναλογικό κέρδος
Ki = Ολοκληρωτικό κέρδος
Kd = Διαφορικό κέρδος
Ας ρίξουμε μια ματιά στον τρόπο που συμπεριφέρεται ένας PID Controller, ο οποίος λειτουργεί σε σύστημα κλειστού βρόγχου



Η μεταβλητή Error αντιπροσωπεύει το σφάλμα παρακολούθησης, δηλαδή την διαφορά ανάμεσα στην επιθυμητή τιμή της επιθυμητής εισόδου Set Point και στην τιμή της πραγματικής εξόδου Sensor Measurement που μετρήθηκε από το αισθητήριο.
Η τιμή του σφάλματος Error θα σταλεί στον PID Controller, ο οποίος θα υπολογίσει τόσο την παράγωγο όσο και το ολοκλήρωμα αυτού του σήματος. Το σήμα u ,που δίνει ο ελεγκτής σαν είσοδο στο σύστημα, θα είναι ίσο με το αναλογικό κέρδος Kp επί την τιμή του σφάλματος, συν το ολοκληρωτικό κέρδος Ki επί το ολοκλήρωμα του σφάλματος, συν το διαφορικό κέρδος Kd επί την παράγωγο του σφάλματος.

$$U = K_p e + K_i \int e dt + K_d \frac{de}{dt}$$

Το σήμα αυτό θα είναι η είσοδος του συστήματος που θέλουμε να ελέγξουμε, το οποίο θα ανταποκριθεί ανάλογα και θα λάβουμε μια νέα έξοδο Y, η οποία με τη σειρά της θα σταλεί ξανά πίσω στο αισθητήριο για να ανιχνεύσει το νέο σήμα σφάλματος Error. Ο ελεγκτής θα πάρει αυτό το νέο σήμα και θα υπολογίσει ξανά την παράγωγο και το ολοκλήρωμα και η ίδια διαδικασία θα επαναλαμβάνεται συνέχεια.

Χαρακτηριστικά των ελεγκτών P-I-D:

Η χρησιμοποίηση ενός αναλογικού ελεγκτή (P) έχει ως αποτέλεσμα την ελάττωση του χρόνου ανύψωσης (κάνει το σύστημα πιο γρήγορο) αλλά δεν μπορεί ποτέ να εξαλείψει το μόνιμο σφάλμα.

Ο ολοκληρωτικός όρος (I) θα εξαλείψει το μόνιμο σφάλμα αλλά θα χειροτερέψει την μεταβατική απόκριση (ο αριθμός των ταλαντώσεων μέχρι την τελική ισορροπία του συστήματος) .

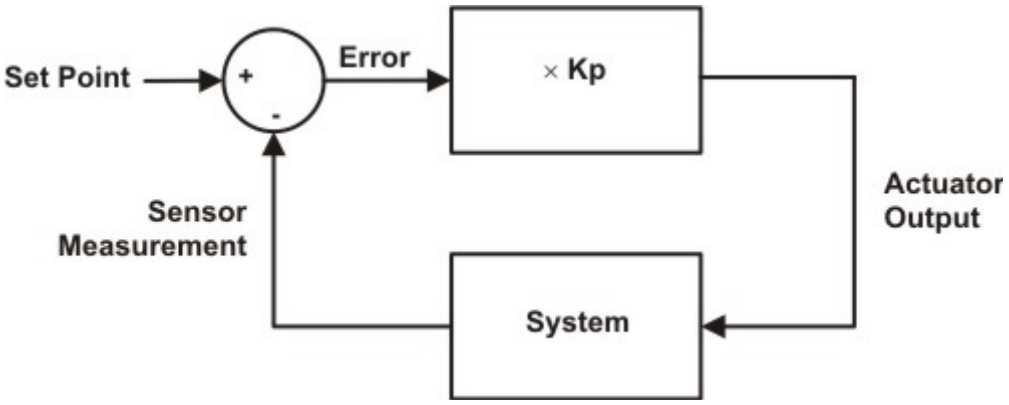
Ο διαφορικός έλεγχος (D) θα έχει ως αποτέλεσμα την αύξηση της σταθερότητας του συστήματος, μειώνοντας την υπερύψωση και βελτιώνοντας την μεταβατική απόκριση.
Τα αποτελέσματα της επίδρασης κάθε όρου σε ένα σύστημα κλειστού βρόχου συνοψίζονται παρακάτω:

	Χρόνος Ανύψωσης	Υπερύψωση	Χρόνος αποκατάστασης	Μόνιμο σφάλμα
K_p	-	+	Μικρή αλλαγή	-
K_i	-	+	+	→0
K_d	Μικρή αλλαγή	-	-	Μικρή αλλαγή

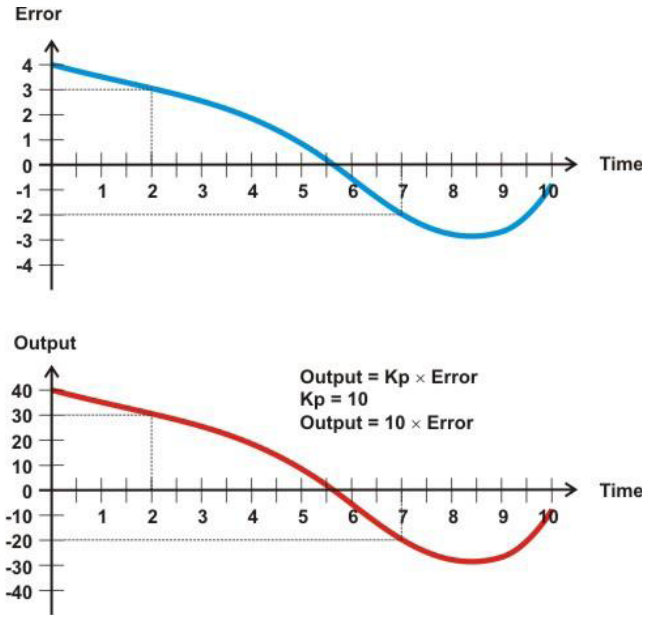
Ας σημειωθεί ότι αυτοί οι συσχετισμοί μπορεί να μην είναι πολύ ακριβείς επειδή στην πραγματικότητα τα Kp , Ki , Kd αλληλοεξαρτώνται. Η αλλαγή μιας από αυτές τις μεταβλητές αλλάζει την επίδραση και των άλλων δύο μεταβλητών.
Για τον λόγο αυτό ο παραπάνω πίνακας πρέπει να χρησιμοποιείται μόνο ως σημείο αναφοράς όταν προσδιορίζουμε τις τιμές των Kp , Ki , Kd.

ΑΝΑΛΟΓΙΚΟΣ ΕΛΕΓΧΟΣ (P)

Ο Αναλογικός έλεγχος είναι πολύ διαδεδομένος τόσο στην βιομηχανία όσο και στην ρομποτική.



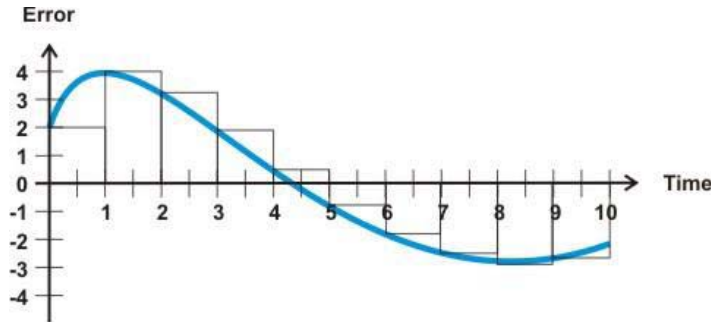
Το μπλοκ διάγραμμα του αναλογικού ελέγχου φαίνεται στο παραπάνω σχήμα. Το κυκλάκι στην αρχή ονομάζεται συγκριτής και συγκρίνει την επιθυμητή τιμή με την πραγματική τιμή (αυτή που μετρήθηκε από το αισθητήριο), το αποτέλεσμα που παράγει ονομάζεται σφάλμα και είναι αυτό που πρέπει να διορθώσει ο ελεγκτής. Ο αναλογικός ελεγκτής P προσπαθεί να εξαλείψει το σφάλμα πολλαπλασιάζοντάς το με κάποια σταθερά (Kp).
Ας υποθέσουμε ότι έχουμε ένα ρομπότ που διατηρεί συγκεκριμένη απόσταση από ένα αντικείμενο, αλλά το αντικείμενο αλλάζει θέση συνεχώς. Ο ελεγκτής P θα εντοπίσει το σφάλμα και θα δώσει ώθηση στους κινητήρες ανάλογη του σφάλματος.
Στο παρακάτω σχήμα φαίνεται καθαρά ότι η έξοδος του ελεγκτή είναι πάντα ανάλογη του σφάλματος



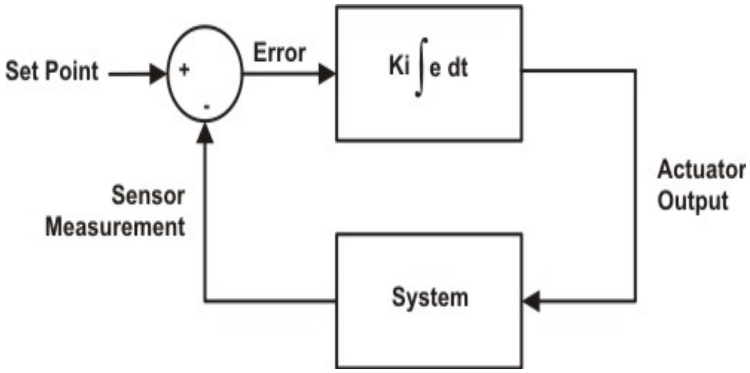
ΟΛΟΚΛΗΡΩΤΙΚΟΣ ΕΛΕΓΧΟΣ (I)

Ο ολοκληρωτικός έλεγχος έχει το προσόν ότι μπορεί να εξαλείψει το σφάλμα , κάτι που δεν μπορεί να κάνει ο ελεγκτής P.
Με την προσθήκη του ολοκληρωτικού όρου έχουμε τη δυνατότητα κατά κάποιο τρόπο να παρακολουθήσουμε την πορεία του σφάλματος σε σχέση με το χρόνο και να το μηδενίσουμε. Αυτό επιτυγχάνεται με την αριθμητική ολοκλήρωση. Είναι λογικό ότι από τη στιγμή που μπαίνει και ο χρόνος στο παιχνίδι οι μαθηματικές πράξεις και οι υπολογισμοί δυσκολεύουν λίγο.

Στο παρακάτω γράφημα βλέπουμε πως μπορούμε να προσομοιάσουμε την πράξη της ολοκλήρωσης, υπολογίζοντας τα εμβαδά των ορθογωνίων και προσθέτοντας τα. Αν τα δείγματα είναι κάθε 1 μονάδα χρόνου αυτό κάνει τους υπολογισμούς ευκολότερους, αφού πολλαπλασιάζουμε κάθε τιμή σφάλματος επί 1 και μετά τις προσθέτουμε.

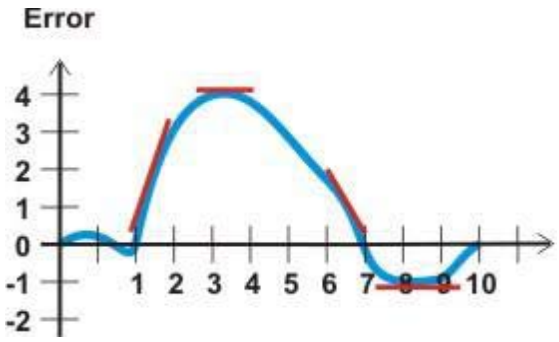


Στο παρακάτω σχήμα φαίνεται το μπλοκ διάγραμμα του ολοκληρωτικού ελέγχου. Ο όρος Ki ∫edt σημαίνει ότι το κέρδος – σταθερά Ki πολλαπλασιάζεται με το ολοκλήρωμα του σφάλματος e ως προς τον χρόνο.

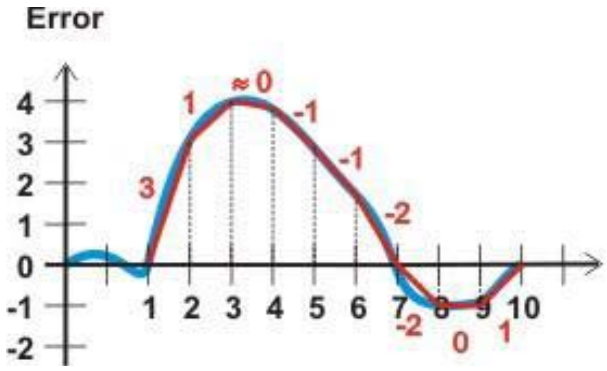


ΔΙΑΦΟΡΙΚΟΣ ΕΛΕΓΧΟΣ (D)

Το κύριο χαρακτηριστικό του διαφορικού ελέγχου είναι ότι μπορεί να αντιληφθεί τις απότομες αλλαγές του σφάλματος, οι οποίες συνήθως προέρχονται από εξωτερικές πηγές (διαταραχές, θόρυβος κλπ) και προκαλούν αστάθεια στο σύστημα μας.
Η πράξη που κρύβεται πίσω από την λειτουργία του είναι η πράξη της παραγώγισης. Η παράγωγος ή ο ρυθμός μεταβολής, μας δείχνει τον ρυθμό δηλαδή το πόσο γρήγορα μεταβάλλεται ένα σήμα σε σχέση με το χρόνο.

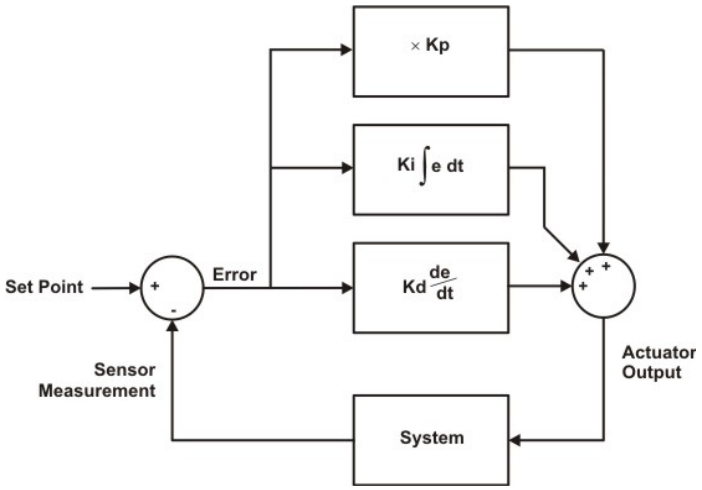


Έτσι λοιπόν, κύριος σκοπός του είναι να παρακολουθεί τον ρυθμό μεταβολής του σφάλματος και να υπολογίζει την έξοδό του με βάση αυτήν την τιμή, με αποτέλεσμα την αύξηση της σταθερότητας του συστήματος.



PID ΕΛΕΓΧΟΣ

Ας εξετάσουμε τώρα την συμπεριφορά ενός PID ελεγκτή ο οποίος περιέχει και τους τρεις όρους P-I-D.
Ρυθμίζοντας το κέρδος του κάθε όρου Kp, Ki, και Kd και παίζοντας με τις αναλογίες μπορούμε και αλλάζουμε τη συμπεριφορά του ελεγκτή. Οι βέλτιστες τιμές που θα μας δώσουν την καλύτερη απόκριση έρχονται μετά από πολλές δοκιμές και εξαρτώνται από το σύστημα που θέλουμε να ελέγξουμε κάθε φορά. Π.χ. ένα σύστημα μπορεί να θέλει πολύ λίγο I, λίγο P και δυνατότερο D, ενώ άλλο σύστημα να ανταποκρίνεται καλύτερα σε P-I έλεγχο και καθόλου D, και άλλο να θέλει και τους τρείς όρους P-I-D.
Στο σχήμα βλέπουμε το μπλοκ διάγραμμα του συστήματος που θα εξομοιώσουμε στο παρακάτω πρόγραμμα. Και εδώ Kp=Ki=Kd=10.



Επεξήγηση κώδικα PID

```
gonia_e=abs(gonia) % angle in absolute value for PID use

P= gonia_e*Kp;

Int=(Int+gonia_e)*Ki;

D=(gonia_e-gonia_old)*Kd;

gonia_old=gonia_e;

PID=abs(P+Int+D);
PID=round(PID)
```

Βήμα 1ο: *Αναλογικός έλεγχος*

Για να βρούμε τον αναλογικό έλεγχο, παίρνουμε την απόλυτη τιμή την γωνίας που έχουμε βρει χρησιμοποιώντας την συνάρτηση *abs* που παίρνει ως όρισμα την μεταβλητή που θέλουμε να μας επιστρέψει το απόλυτό της και την πολλαπλασιάζουμε με την *Kp* σταθερά.

```
P= gonia_e*Kp;
```

Βήμα 2ο: *Ολοκληρωτικός έλεγχος*

Προσθέτουμε την γωνία στην *Int* μεταβλητή και έπειτα το πολλαπλασιάζουμε με την *Ki* σταθερά.

Βήμα 3ο: *Διαφορικός έλεγχος*

Αφαιρούμε την παλιά τιμή της γωνίας με την καινούργια που βρήκαμε και την πολλαπλασιάζουμε επί την *Kd* σταθερά. Η αρχική τιμή της *gonia_old* είναι μηδέν αν το πρόγραμμα εκτελεστεί για πρώτη φορά.

Βήμα 3ο: *Παλιά γωνία*

Στο σημείο αυτό κρατάμε την τιμή της γωνίας που βρίσκουμε σε μία μεταβλητή με το όνομα *gonia_old*. Για να την χρησιμοποιήσουμε στο παραπάνω βήμα.

Βήμα 4ο: *Εύρεση PID – στρογγυλοποίηση*

Ως τελευταίο βήμα προσθέτουμε τον αναλογικό, ολοκληρωτικό και διαφορικό έλεγχο για να βρούμε τον PID ελεγκτή και τον στρογγυλοποιούμε με την χρήση της συνάρτησης *round*.

Κίνηση NXT με PID

Στο κομμάτι αυτό θα αναλύσουμε την μεθοδολογία μου ακολουθήσαμε για να καταφέρει το ρομπότ μας να πάει στα σημεία που του έχουμε υποδείξει. Θα δούμε τί συναρτήσεις χρησιμοποιήσαμε για την υπόδειξη των επιθυμητών σημείων, καθώς και για την κίνηση του nxt.

Βήμα 1ο: *PID Parameters*

Σετάρουμε τις σταθερές του PID ανάλογα με τα αποτελέσματα που θέλουμε να λάβουμε.

Βήμα 2ο: *NXT Movement Parameters*

Δημιουργούμε τα αντικείμενα για την κίνηση του ρομπότ. Ξεκινάμε με την δημιουργία της μεταβλητής *Ports* η οποία αποτελείται από τις πόρτες που θέλουμε να χρησιμοποιήσουμε για την κίνησή. Στην περίπτωση επιλέξαμε την πόρτα A και B ή αλλιώς την πόρτα 1 και 2 του nxt. Έπειτα σετάρουμε την ταχύτητα που θέλουμε για την κίνηση του ρομπότ και δημιουργούμε τα αντικείμενα της κίνησής μας. Για την δημιουργία των αντικειμένων στραφήκαμε στην χρήση της *Nxtmotor* συνάρτησης, η οποία παίρνει ως όρισμα την πόρτα που θέλουμε να χρησιμοποιήσουμε για τον αντίστοιχο κινητήρα, δίνοντάς μας έπειτα την δυνατότητα να χειριστούμε τις ιδιότητες του κινητήρα που επιλέξαμε. Τα αντικείμενα που δημιουργήσαμε εμείς είναι τρία, ένα για ευθεία και δύο για αριστερά και δεξιά.

Βήμα 3ο: *Setting Coordinates For Desired Point*

Εδώ προσπαθούμε να δώσουμε στο nxt να καταλάβει ποιες είναι οι συντεταγμένες του σημείου που θέλουμε να πάει. Για να του δώσουμε τα σημεία που θέλουμε να πάει χρησιμοποιούμε την *ginput* συνάρτηση η οποία παίρνει ως όρισμα πόσα σημεία θέλουμε και φτιάχνει έναν πίνακα με τις συντεταγμένες των σημείων αυτών. Τα σημεία τα επιλέγουμε κάνοντας κλικ με το ποντίκι στην εικόνα της πλατφόρμας, που έχουμε προηγούμενος εμφανίσει. Έπειτα κάνουμε χρήση της *size* συνάρτησης για να βρούμε το μέγεθος του πίνακα και χρησιμοποιούμε μια for με το αποτέλεσμα της συνάρτησης size για να μπορέσουμε να προσπελάσουμε τον πίνακα, παίρνοντας σε κάθε βήμα της for τις συντεταγμένες των σημείων που επιλέξαμε.

Αφού λοιπόν έχουμε πάρει τις πρώτες συντεταγμένες δίνουμε αρχικά στο ρομπότ την εντολή να πάει ευθεία. Αναφερόμενοι στο αντικείμενό που είναι υπεύθυνο ώστε το nxt μας να πάει ευθεία δηλαδή το *Straight* , σετάρουμε την ταχύτητά του βάζοντας την τιμή που θέλουμε. Αυτό γίνεται με την αναφορά που αντικειμένου και τις ιδιότητάς του όπως φαίνεται παρακάτω.

```
Straight.Power              = DrivingSpeed;
```

Τέλος με την χρήση της `Straight.SendToNXT()` στέλνουμε την παραπάνω εντολή στο nxt για να μπορέσει να κινηθεί, τοποθετώντας όμως και μια παύση έτσι ώστε να μην κινείται επ' αόριστον.

Βήμα 4ο: *Finding The Bigger Area-Angle and Setting NXT Movement*

Κάνοντας την επεξεργασία της εικόνας όπως έχουμε αναφέρει σε προηγούμενο κεφάλαιο και σετάροντας τον PID έτσι ώστε να ικανοποιεί τα επιθυμητά αποτελέσματα, η κίνηση πλέον περιορίζεται σε δυο περιπτώσεις, την κίνηση αριστερά και την κίνηση δεξιά.
Το προς τα πού θα κινηθεί το ρομπότ εξαρτάται από το τι γωνία παίρνουμε. Αν η γωνία μας είναι θετική και πάνω από δέκα μοίρες τότε το ρομπότ μας κινείται προς τα δεξιά, αντίθετα αν η γωνία είναι αρνητική και πάνω από δέκα μοίρες τότε κινείται προς τα αριστερά. Ο παραπάνω τρόπος σκέψης υλοποιείται με το να προσθαφαιρούμε την ίδια ποσότητα, στην περίπτωσή μας το PID, στην ταχύτητα των κινητήρων μας, αναλόγως με την κίνηση θέλουμε να κάνει το ρομπότ.

Βήμα 5ο: *Τερματισμός κίνησης*

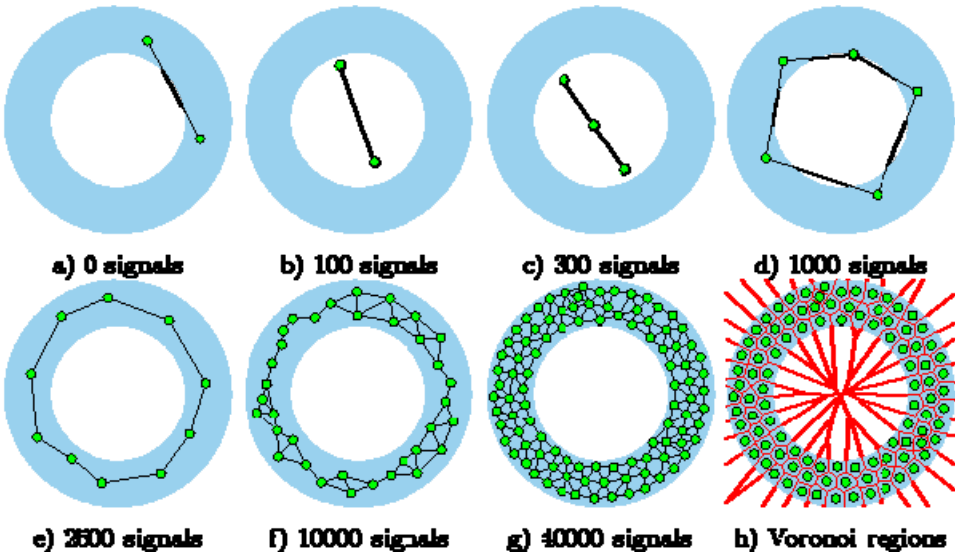
Όταν το ρομπότ ευθυγραμμιστεί με το επιθυμητό σημείο τότε τερματίζουμε τον έλεγχο της κίνησης. Όταν δηλαδή πρέπει να πάει δεξιά ή αριστερά, και στην συνέχεια του στέλνουμε πάλι τις κατάλληλες εντολές για να πάει ευθεία ελέγχοντας όμως τώρα την απόστασή του από το επιθυμητό σημείο , μέχρι να γίνει μικρότερη ή ίση από αυτήν που θέλουμε. Όταν το ρομπότ μας φτάσει στο επιθυμητό σημείο, σταματάει και παίρνουμε τις συντεταγμένες του επόμενου σημείου. Το nxt σταματάει να κινείται όταν έχει περάσει απ' όλα τα σημεία που του έχουμε δώσει ,στέλνοντάς του την εντολή: `StopMotor('all' , 'brake');`

<<ΚΕΦΑΛΑΙΟ 5>>

GROWING NEURAL GAS (GNG)

Ένας γνωστός και αποδοτικός αυτο-οργανόμενος νευρωνικός ταξινομητής με ιδιαίτερα χαρακτηριστικά είναι το αυτο-οργανόμενο νευρωνικό αέριο (Growing Neural Gas – GNG). Σε αντίθεση με τον ταξινομητή Kohonen SOFM, ο GNG ξεκινάει αρχικά με δύο μόνο νευρώνες και κατά την διάρκεια της εκπαίδευσης παρεμβάλλονται νέοι μεταξύ του ζεύγους των νευρώνων που χαρακτηρίζονται από το μεγαλύτερο σφάλμα μέτρησης. Το δίκτυο σταματάει να μεγαλώνει όταν ο αριθμός των κλάσεων φτάσει τον μέγιστο προκαθορισμένο αριθμό.

Ιδιαίτερα χαρακτηριστικά του ταξινομητή αυτού είναι ότι οι σχέσεις γειτνίασης μεταξύ των κλάσεων που καθορίζονται δυναμικά και περιγράφονται με πλευρικές συνδέσεις μεταξύ των νευρώνων, τα κέντρα των κλάσεων και οι πλευρικές συνδέσεις αποτελούν σε όλη την διάρκεια της εκπαίδευσης μια καλή περιγραφή του χώρου των δεδομένων, ενώ περιγράφουν με βέλτιστο τρόπο την τοπολογία αυτών, τέλος η πλαστικότητα των κλάσεων (δηλαδή η ικανότητα τους να προσαρμόζονται στα δεδομένα) είναι ίδια σε όλη την διάρκεια της εκπαίδευσης. Το τελευταίο χαρακτηριστικό μπορεί να θεωρηθεί και ως μειονέκτημα καθώς η υπέρμετρη πλαστικότητα εμποδίζει την σύγκλιση των κλάσεων. Όμοια με τον ταξινομητή Kohonen SOFM για κάθε διάνυσμα εισόδου αναπροσαρμόζεται η κοντινότερη κλάση και οι γειτονικές της, καθιστώντας τα όρια αυτών είναι ασαφή. Σε αντίθεση με τον Kohonen SOFM αυτό συμβαίνει σε όλη την διάρκεια της εκπαίδευσης. Επίσης ο ταξινομητής Kohonen SOFM είναι πολύ πιο αργός από τον GNG καθώς (α) σε όλη την διάρκεια της εκπαίδευσης ο αριθμός των νευρώνων είναι ίσος με τον μέγιστο προκαθορισμένο αριθμό και κάθε φορά που εισάγεται ένα διάνυσμα πρέπει να ελεγχθούν όλοι οι νευρώνες, και (β) το εύρος της γειτονιάς στον Kohonen SOFM είναι αρχικά πολύ μεγάλο οπότε για κάθε διάνυσμα εισόδου εκτός από την πλησιέστερη κλάση προσαρμόζεται και μεγάλο πλήθος γειτονικών, ενώ στον GNG γειτονικοί νευρώνες είναι μόνο οι άμεσα συνδεδεμένοι.

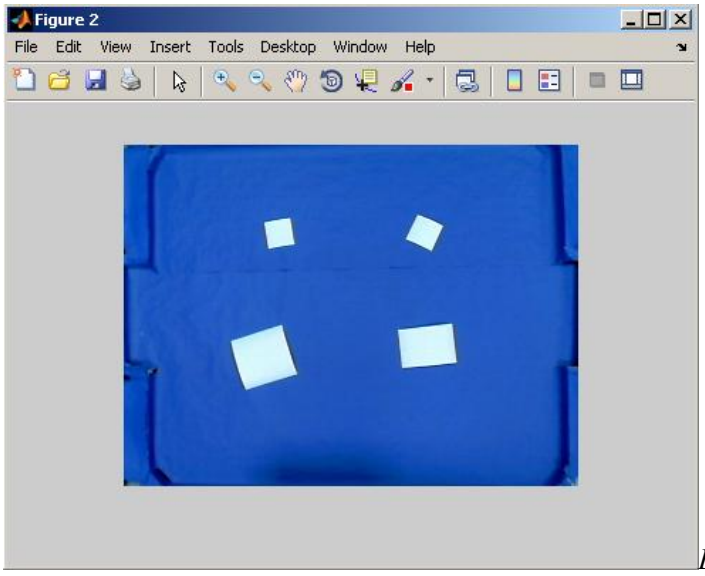


Συνοπτικά

Σε σύγκριση με το φυσικό νευρωνικό αέριο(*Neural Gas*), GNG έχει τις ακόλουθες διακρίσεις:
1. Το σύστημα έχει τη δυνατότητα να προσθέτει και να διαγράφει κόμβους.
2. Λάθος τοπικές μετρήσεις σημειώνονται σε κάθε βήμα, που βοηθά τοπικά να εισαγάγετε / διαγράφεται ο κάθε κόμβος.
3. Οι ακμές είναι συνδεδεμένες μεταξύ των κόμβων, έτσι μια αρκετά παλιά ακμή διαγράφεται. Αυτές οι ακμές προορίζονται για εντοπισμένες θέσεις δεδομένων διανομής.
4. Αυτές οι ακμές μπορούν επίσης να βοηθήσουν στον εντοπισμό διακριμένων συσπειρώσεων(clusters) (οι συσπειρώσεις αυτές δεν συνδέονται με τις ακμές).

Εφαρμογή του GNG αλγόριθμου.

Στο τελευταίο στάδιο της πτυχιακής εργασίας θέλαμε να βρούμε έναν τρόπο ώστε το NXT μας να μπορεί να καταλάβει αν υπάρχουν τοποθετημένα εμπόδια στον δρόμο του και πώς μπορεί να τα αποφύγει. Το ρομπότ θα μπορεί στο τέλος να κινείται μέσα στον χώρο αποφεύγοντας τα εμπόδια, παίρνοντας το συντομότερο μονοπάτι του γράφου που θα δημιουργείται, δίνοντάς του το σημείο εκκίνησης και τερματισμού. Τα εμπόδια όπως φαίνεται στην *ΕΙΚ.(18)* είναι τετράγωνα από άσπρο χαρτόνι κομμένα σε διάφορα μεγέθη. Προσπαθώντας στις πρώτες μας δοκιμές να φτιάξουμε έναν γράφο χρησιμοποιώντας τις συναρτήσεις της matlab όπως είναι η *triplot* ή η *voronoi* παρατηρήσαμε, ότι κάποιες ακμές του δημιουργηθέντα γράφου περνούσαν πάνω από τα εμπόδια. Λαμβάνοντας λοιπόν αυτό το αποτέλεσμα στραφήκαμε στην χρήση του GNG αλγορίθμου.



ΕΙΚ.(18)

Εμφάνιση εμποδίων

Προβλήματα

Στρεφόμενοι στην χρήση του GNG τα πράγματα δεν ήταν τόσο ρόδινα όσο τα περιμέναμε. Ήρθαμε αντιμέτωποι με τα εξής προβλήματα:

1. *Πόσα nodes πρέπει να έχουμε ώστε να καλύπτεται ικανοποιητικά όλη η πλατφόρμα;*

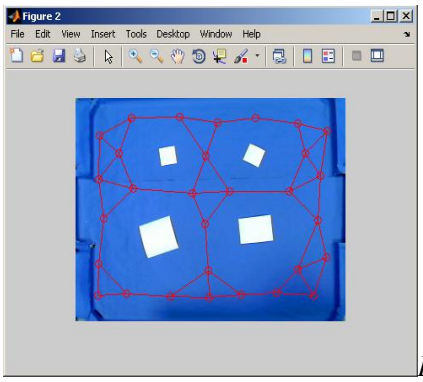
   Έπειτα από πειραματισμούς καταλήξαμε ότι ο βέλτιστος αριθμός nodes που πρέπει να έχουμε είναι 30.Βάζοντας παραπάνω nodes παρατηρήσαμε ότι ναι μεν ο αλγόριθμος κάλυπτε καλύτερα τον χώρο αλλά θα είχαμε αργότερα πρόβλημα με την κίνηση του ρομπότ, λόγω του ότι τα nodes είναι πολύ κοντά τα ένα με το άλλο. Βάζοντας λιγότερα nodes ο αλγόριθμος δεν δούλευε σωστά γιατί τα nodes ήταν πολύ λίγα για να καλύψουν τον χώρο της πλατφόρμας με αποτέλεσμα κάποιες ακμές να περνάνε πάνω από τα εμπόδια.

2. *Πόσα δεδομένα πρέπει να περαστούν στον αλγόριθμο ώστε να προλάβει να μάθει;*

   Λύνοντας το παραπάνω πρόβλημα ο αλγόριθμος παρόλα αυτά δεν μας έβγαζε το επιθυμητό αποτέλεσμα. Διαπιστώσαμε λοιπόν ότι το πρόβλημα οφειλόταν στον αριθμό των δεδομένων που δίναμε στον αλγόριθμο για να μάθει πού να τοποθετήσει τα εκάστοτε nodes.Τα δεδομένα μας αποτελούνται από τις x,y συντεταγμένες των pixel που έχουν τιμή ίση με μηδέν (τα pixel των εμποδίων έχουν τιμή ίση με ένα).Για να μπορέσει ο αλγόριθμος να μάθει καλύτερα τον χώρο, αυξήσαμε τα δεδομένα μας διπλασιάζοντάς τα.

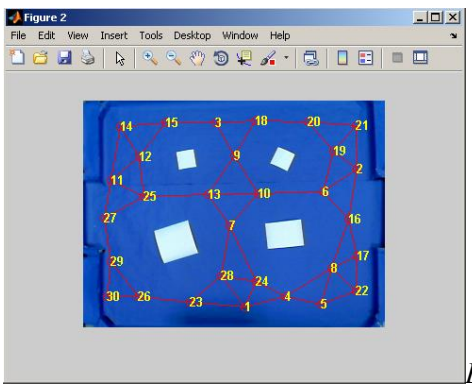
3. *Ποιό node είναι ποιό για να χρησιμοποιήσουμε την graphshortestpath;*

   Έχοντας κάνει τους σωστούς συνδυασμούς παίρνουμε το εξής αποτέλεσμα *ΕΙΚ.(19)*.Βλέπουμε όμως ότι δεν μπορούμε να καταλάβουμε ποιος κόμβος είναι ποιος, έτσι ώστε να καλέσουμε την *graphshortestpath* που παίρνει ως όρισμα έναν πίνακα με το βάρος της κάθε ακμής του γράφου και τον αρχικό και τελικό κόμβο του επιθυμητού μονοπατιού. Εκτυπώνοντας όμως το i του κάθε κόμβου στην αρχική εικόνα, γνωρίζουμε τώρα ποιος κόμβος είναι ποιος .*ΕΙΚ.(20)*



ΕΙΚ.(19)

GNG χωρίς nodes



ΕΙΚ.(20)

GNG με nodes

GNG

Ο αλγόριθμος GNG υποθέτει ότι κάθε κόμβος κ αποτελείται από τα ακόλουθα:

- $\bar{w}_k$ - φορέας αναφοράς, στο R^n.
- $error_k$ - μια τοπική συσσωρευμένη μεταβλητή σφάλματος.
- Ένα σύνολο ακμών που καθορίζουν τοπογραφικά τους γείτονες του κόμβου k.

Ο φορέας αναφοράς μπορεί να ερμηνευθεί ως η θέση ενός κόμβου στον δοθέν χώρο.
Το τοπικό συσσωρευμένο σφάλμα είναι ένα στατιστικό μέτρο, το οποίο χρησιμοποιείται για τον προσδιορισμό κατάλληλων σημείων, για την προσθήκη νέων κόμβων. Περαιτέρω, κάθε ακμή έχει μια μεταβλητή ηλικία, που χρησιμοποιείται για να αποφασιστεί πότε να γίνει η απομάκρυνση παλαιών ακμών, ώστε να διατηρείτε ενημερωμένη η τοπολογία.

Ψευδοκώδικας GNG

Δημιουργία δύο τυχαίων τοποθετημένων κόμβων, οι οποίοι συνδέονται με μια ακμή μηδενικής ηλικίας και τα λάθη τους έχουν επίσης την τιμή μηδέν.

Δημιουργήστε ένα διάνυσμα εισόδου $\bar{x}$ σύμφωνα με κάποια διανομή.

Εντοπίστε τους δύο κόμβους S και T που είναι πλησιέστεροι ως προς το $\bar{x}$, δηλαδή, οι δύο κόμβοι με διανύσματα αναφοράς ήταν $\bar{w}_s$ και $\bar{w}_t$ τέτοια ώστε $|\bar{w}_s - \bar{x}|^2$ είναι η μικρότερη τιμή και $|\bar{w}_t - \bar{x}|^2$ είναι η δεύτερη μικρότερη τιμή, για όλους τους κόμβους k

Ο νικητής-κόμβος s πρέπει να ενημερώσει την τοπική μεταβλητή σφάλματος έτσι ώστε να μπορούμε να προσθέσουμε την
τετραγωνισμένη απόσταση μεταξύ των $\bar{w}_s$ και $\bar{x}$, στο σφάλμα $error_s$

$$error_s \leftarrow error_s + |\bar{w}_s - \bar{x}|^2$$

Μετακίνηση του s και της τοπολογία των γειτόνων του (δηλαδή όλους τους κόμβους που συνδέονται με τον s μέσω μιας ακμής) προς το $\bar{x}$,με κλάσματα e_w και e_n της απόστασης. e_w , $e_n \in [0,1]$

$$\bar{w}_s \leftarrow \bar{w}_s + e_w(\bar{x} - \bar{w}_s)$$
$$\bar{w}_n \leftarrow \bar{w}_n + e_n(\bar{x} - \bar{w}_n), \forall n \in Neighbour(s)$$

Προσαύξηση της ηλικίας όλων των ακμών από τον κόμβο s, προς την τοπολογία των γειτόνων του.

Αν ο s και ο t κόμβος συνδέονται με μια ακμή, τότε θέτουμε την τιμή της ακμής αυτής με μηδέν. Εάν δεν είναι συνδεδεμένοι ,τότε δημιουργούμε μια ακμή μεταξύ τους.

Αν υπάρχουν ακμές με ηλικίες μεγαλύτερες από α_{max} τις αφαιρούμε. Εάν, μετά από αυτό, υπάρχουν κόμβοι που δεν έχουν ακμές τότε τους αφαιρούμε και αυτούς.

Εάν η τρέχουσα επανάληψη είναι ένας ακέραιος πολλαπλάσιος του λ και η μέγιστη αριθμός κόμβων δεν έχει επιτευχθεί, τότε εισαγάγουμε ένα νέο κόμβο. Η εισαγωγή ενός νέου κόμβου r γίνεται ως εξής:

ο Βρείτε τον κόμβο u με το μεγαλύτερο σφάλμα.
ο Μεταξύ των γειτόνων του u, βρείτε τον κόμβο v με το μεγαλύτερο σφάλμα.
ο Εισαγάγετε τον νέο κόμβο r μεταξύ των κόμβων u και v ως εξής:

$$\overline{w}_r \leftarrow \frac{(\overline{w}_u + \overline{w}_v)}{2}$$

Δημιουργήστε τις ακμές μεταξύ των u και r, καθώς και των v και r, και στη συνέχεια αφαιρέστε την ακμή μεταξύ των u και v κόμβων.

Μειώστε το μεταβλητό σφάλμα του u και v και να ρυθμίσετε το σφάλμα του κόμβου r

$$error_u \leftarrow \alpha \times error_u$$
$$error_v \leftarrow \alpha \times error_v$$
$$error_r \leftarrow error_u$$

Μειώστε όλα τα μεταβλητά σφάλματα των κόμβων j με παράγοντα β.

$$error_j \leftarrow error_j - \beta \times error_j$$

Εάν το κριτήριο διακοπής δεν έχει επιτευχθεί, τότε επαναλαμβάνουμε την διαδικασία. Το κριτήριο διακοπής μπορεί να είναι για
παράδειγμα η απόδοση σε ένα σύνολο δοκιμής για το αν είναι αρκετά καλή, ή αν το ανώτερο όριο των κόμβων έχει επιτευχθεί, κλπ.

Οι αλλαγές που κάναμε είναι οι εξής:

Αλλαγή 1η: *Μείωση ταχύτητας*

Για να μπορέσει το nxt να κινηθεί σωστά πάνω στον γράφο έπρεπε να του αφαιρέσουμε τον PID έλεγχο και να του μειώσουμε την ταχύτητα. Ο PID δημιουργούσε πρόβλημα, διότι μέχρι να μπορέσει να ευθυγραμμιστεί με το σημείο, κάποιες φορές έχανε το σημείο ή πατούσε το εμπόδιο, πράγμα το οποίο δεν θέλουμε.

Αλλαγή 2η: *Graphshortestpath*

Η συνάρτηση *Graphshortestpath* μπορεί να βρει το συντομότερο μονοπάτι μέσα σε ένα γράφο. Τα ορίσματα που παίρνει είναι έναν πίνακα με τις αποστάσεις των κόμβων(*data_dist_nodes)* καθώς και τους κόμβους που θέλουμε να βρούμε το συντομότερο μονοπάτι μεταξύ τους. Λαμβάνοντας λοιπόν έναν πίνακα με το ελάχιστο μονοπάτι, προσπελαύνουμε τον πίνακα αυτόν και παίρνουμε τις αντίστοιχες συντεταγμένες των κόμβων που μας εμφανίζει.

Αλλαγή 3η: *Ταξινόμηση πίνακα εμβαδών*

Η επεξεργασία της εικόνας γίνετε κανονικά με την διαφορά ότι κάθε φορά θα καθαρίζουμε τον πίνακα *min* ο οποίος εμπεριέχει τις τιμές των εμβαδών που είναι μικρότερα από 90, ο λόγος που φτιάχνουμε τον πίνακα αυτόν είναι για αποκλείσουμε τα κέντρα (*Centroids)* των εμποδίων , και τον σβήσουμε γιατί αλλιώς κρατάει τα παλιά δεδομένα τα οποία μπορεί να διαφέρουν από τα καινούργια και έτσι μπερδεύεται το ρομπότ. Έπειτα καλούμε την συνάρτηση *sort* με την οποία ταξινομούμε τον πίνακα *min* κατά φθίνουσα σειρά για να έχουμε πάντα τους μικρότερους αριθμούς στις πρώτες θέσεις του πίνακα, οι οποίοι είναι ο κύκλος και το τετράγωνο, για την διευκόλυνση της αναγνώρισης του ρομπότ και εύρεσης της γωνίας.

Αλλαγή 4η: *Κίνηση Nxt*

Η νοοτροπία και ο τρόπος σκέψης είναι ο ίδιος με τον διαφορά ότι όπως είπαμε και παραπάνω δεν χρησιμοποιούμε τον PID. Θέλουμε το ρομπότ μας να κάνει επιτόπου στροφή για να ευθυγραμμιστεί με το σημείο. Αυτό επιτυγχάνεται βάζοντας την ίδια ταχύτητα και στους δυο κινητήρες με την διαφορά ότι η μία τιμή θα είναι αρνητική και η άλλη θετική.

```
RightW.Power=   TurningSpeed
LeftW.Power =  -(DrivingSpeed)
```

Κώδικες

Ιδιότητες κάμερας

Τα βήματα για την αναγνώριση κάμερας καθώς και των δυνατοτήτων-επιλογών της φαίνονται παρακάτω:

Βήμα 1ο: *Πληροφορίες για της διαθέσιμες συσκευές εισόδου εικόνας.*

```
imaqInfo = imaqhwinfo

imaqInfo =

   InstalledAdaptors: {'coreco'  'winvideo'}

      MATLABVersion: '7.6 (R2008a)'

        ToolboxName: 'Image Acquisition Toolbox'

     ToolboxVersion: '3.1 (R2008a)'
```

Βήμα 2ο: *Πληροφορίες για τον winvideo αντάπτωρα.*

- hwInfo = imaqhwinfo('winvideo')

```
hwInfo =

     AdaptorDllName: [1x81 char]

   AdaptorDllVersion: '3.1 (R2008a)'

        AdaptorName: 'winvideo'

          DeviceIDs: {[1]  [2]}

         DeviceInfo: [1x2 struct]
```

- hwInfo.DeviceInfo

```
ans =

1x2 struct array with fields:

   DefaultFormat

  DeviceFileSupported
```

DeviceName

DeviceID

ObjectConstructor

- device1 = hwInfo.DeviceInfo(1)

```
device1 =

    DefaultFormat: 'RGB24_320x240'

    DeviceFileSupported: 0

   DeviceName: 'Logitech QuickCam Pro 5000

   DeviceID: 1

 ObjectConstructor: 'videoinput('winvideo', 1)'

  SupportedFormats: {1x10 cell}
```

Βήμα 3ο: *Υποστηριζόμενες αναλύσεις video*

- device1.SupportedFormats

```
ans =

 Columns 1 through 5

  'I420_160x120'   'I420_176x144'   'I420_320x240'   'I420_352x288'   'I420_640x480'


 Columns 6 through 9

  'RGB24_160x120'   'RGB24_176x144'   'RGB24_320x240'   'RGB24_352x288'


 Column 10

  'RGB24_640x480'
```

Βήμα 4ο: *Δημιουργία αντικειμένου video.*

Αφού έχουμε δει λοιπόν τους υποστηριζόμενους τύπους και τις αναλύσεις που μας προσφέρει το matlab μαζί με την κάμερα, δημιουργούμε το video αντικείμενό μας, χρησιμοποιώντας την *videoinput* συνάρτηση δίνοντας της ως όρισμα ,τον τύπο του αντάπτωρα, τον αριθμό της συσκευής μας και την ανάλυση που θέλουμε.

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

Βήμα 5ο: *Αλλαγή ρυθμίσεων κάμερας*

Επειδή η κάμερά μας έχει την ιδιότητα να κάνει αυτόματο ζουμ, πράγμα το οποίο δεν θέλουμε, χρησιμοποιούμε την συνάρτηση *getselectedsource* για να μπούμε στις ρυθμίσεις της κάμερας βάζοντας ως όρισμα το αντικείμενο που έχουμε δημιουργήσει παραπάνω, δηλαδή *getselectedsource (vid)*. Έπειτα τοποθετούμε το αποτέλεσμά της σε ένα άλλο αντικείμενο έτσι ώστε να μπορούμε να προσπελάσουμε τα αντικείμενα της *getselectedsource* χρησιμοποιώντας το αντικείμενο μας και το όνομα της ιδιότητας που θέλουμε να διαμορφώσουμε.

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

src=getselectedsource(vid);

src.ZoomMode='manual'; %apenergopio to auto zoom
```

Prosnap

```
function L= prosnap(I)


%% Creating Disk with Radius 4

disk=strel('disk',4);

 %% Processing Snapshots

I2=rgb2gray(I); % Converting image to grayscale

level = graythresh(I);
level=level+0.2;
bwi=im2bw(I2,level); % Converting image to binary image, based on
threshold

bwi = bwareaopen(bwi,10);

bwi=imclose(bwi,disk); % Morphologically open image (bwi=binary
image, disk object from strel function)

bwi=imclearborder(bwi);

L = bwlabel(bwi);
```

Βήμα 1ο: Μετατροπή αρχικής εικόνας από RGB σε κλίμακα του γκρι.

Για να μετατρέψουμε την αρχική μας εικόνα σε κλίμακα του γκρι χρησιμοποιούμε την συνάρτηση *rgb2gray* δίνοντάς της όρισμα την εικόνα που θέλουμε να μετατρέψουμε.

Βήμα 2ο: *Εύρεση τιμής για το κατάλληλο κατώφλι.*

Χρησιμοποιούμε την συνάρτηση *graythresh,* η οποία βρίσκει αυτόματα το κατώφλι της εικόνας που δίνουμε ως όρισμα για να μπορέσουμε έπειτα να το χρησιμοποιήσουμε στην δυαδική μετατροπή της εικόνας μας.

Βήμα 3ο: *Μετατροπή εικόνας σε δυαδική μορφή.*

Παίρνοντας το κατώφλι που βρήκαμε στο προηγούμενο βήμα αυξημένο κατά 0.2 και την εικόνα που έχουμε μετατρέψει σε κλίμακα του γκρι, καλούμε την *im2bw* συνάρτηση με ορίσματα τα παραπάνω αντικείμενα για να μας κάνει την δυαδική μετατροπή

Βήμα 4ο: *Αφαίρεση μικρών αντικειμένων.*

Για να αφαιρέσουμε τυχόν μικρά αντικείμενα, που μπορεί να μην έχουν φύγει με την χρήση του κατωφλιού καλούμε την *bwareaopen* συνάρτηση, η οποία αφαιρεί από την εικόνα όλα τα αντικείμενα που συνδέονται μεταξύ τους έχοντας μέγεθος P pixel, δίνοντάς της ορίσματα την εικόνα σε δυαδική μορφή και το μέγεθος P που θέλουμε.

Βήμα 5ο: *Κλείσιμο κενών.*

Έπειτα κλείνουμε τα κενά που μπορεί να υπάρχουν μεταξύ των σχημάτων μας, με την χρήση της συνάρτησης *imclose,* με ορίσματα την εικόνα και το αντικείμενο της συνάρτησης *strel.*Η *strel* κατασκευάζει ένα μορφολογικό αντικείμενο το οποίο μπορεί να πάρει διάφορους γεωμετρικούς σχηματισμούς όπως, διαμάντι-δίσκο-γραμμή-οκτάγωνο, καθώς και διάφορα μεγέθη pixel.Εμείς χρησιμοποιήσαμε ως γεωμετρικό σχήμα τον δίσκο με μέγεθος 4 pixel

Βήμα 6ο: *Καθαρισμός ορίων-Επιστροφή label.*

Για να είναι πιο εμφανή τα σχήματα καθαρίζουμε την περιοχή γύρω από τα όρια τους. Αυτό γίνετε με την χρήση της *imclearborder* συνάρτησης που παίρνει ως όρισμα την δυαδική εικόνα μας. Έπειτα χρησιμοποιούμε την *bwlabel* με όρισμα την τελική επεξεργασμένη μας εικόνα για να μας επιστρέψει την ετικέτα της, την οποία θα χρησιμοποιήσουμε στην *regionprops*

Κώδικας εύρεσης γωνίας

```
function gon=Euresi_Gonias (x,y,xmes,ymes,x1,y1,x2,y2) %Function
finding the angle

klisi_s=round(rad2deg(atan((ymes-y2)/(xmes-x1))));     % Ence of the
line from average value of the line passing
                                                       % through the
                                                       % centers and
the front point.

if klisi_s>0 || klisi_s==0   %an einai theriki i isi me to 0

    if     (x1<x2 && y1<y2) || (x1>x2 && y1<y2)

         gon=BkatoapoAmeKthet(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1>x2 && y1>y2) || ( x1<x2 && y1>y2)

          gon=BpanoapoAmeKthet(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1>x2 && y1==y2)

          gon=BpanoapoAmeParal(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1<x2 && y1==y2 )

          gon=BkatoapoAmeParal(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1==x2 && y1<y2 )

        gon=BkatoapoAmeKatheto(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1==x2 && y1>y2 )

        gon=BpanoapoAmeKatheto(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);
     end

end

if klisi_s<0

     if     (x1<x2 && y1<y2) || (x1>x2 && y1<y2)

         gon=BkatoapoAmeKarn(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1>x2 && y1>y2) || ( x1<x2 && y1>y2)

          gon=BpanoapoAmeKarn(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1>x2 && y1==y2)

          gon=BpanoapoAmeParal(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1<x2 && y1==y2 )
```

```
            gon=BkatoapoAmeParal(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1==x2 && y1<y2 )

        gon=BkatoapoAmeKatheto(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);

    elseif ( x1==x2 && y1>y2 )

        gon=BpanoapoAmeKatheto(x,y,xmes,ymes,x1,y1,x2,y2,klisi_s);


    end

end
```

Κώδικας περιπτώσεων

```
function gon=BkatoapoAmeKarn(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)

if (xm<=xs && ym<=ys)


    x=ys-ym;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    prox=abs(klisi_s)-abs(gontip);


     if prox<0 %gia na ksexorizo pou einai to simeio otan
               %einai konta sto piso simeio
               %pezo me tin "prox" gonia k analogos an einai thetiki
               %i arnitiki brisko tin katalili gonia me to
               %katalalilo prosimo

        gon=180-abs(prox);

    else

        gon=abs(prox)-180;

     end


end
```

```
if ( xm<=xs && ym>=ys )

    if xm>xss

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=klisi_s-abs(gontip);


    else

    x=ym-ys;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    prox=abs(gontip)-klisi_s;
    gon=abs(prox)-180;


    end
end


if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   gon=abs(klisi_s)-abs(gontip);



end

if ( xm>=xs && ym<=ys)

    if xm<xss2

      x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
```

```
        prox=klisi_s-gontip;
        gon=180-abs(prox);



    else

    x=ym-ys;

        x1=(xm-xs)^2;
        x2=(ym-ys)^2;
        y=sqrt(x1+x2);
        f=x/y;
        gontip=rad2deg(asin(f));
        gon=abs(klisi_s)+abs(gontip);


    end
end
end




function gon=BkatoapoAmeKatheto(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys)


    x=ys-ym;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(klisi_s)-gontip;
    gon=-gon;

end

if (xm<=xs && ym>=ys)

    if xm>xss

    x=ym-ys;

        x1=(xm-xs)^2;
        x2=(ym-ys)^2;
        y=sqrt(x1+x2);
        f=x/y;
        gontip=rad2deg(asin(f));
        gon=abs(gontip)-abs(klisi_s);
```

```
    else

    x=ym-ys;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-abs(klisi_s);

end
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   prox=abs(klisi_s)+abs(gontip);
   gon=180-prox;


end

if ( xm>=xs && ym<=ys)

    if xm<xss2

      x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=abs(gontip)-abs(klisi_s);
     gon=prox-180;



    else


    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=klisi_s-abs(gontip);
     gon=180-prox;
```

```
    end
end




function  gon=BkatoapoAmeKthet(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys) % an x_node<xmes && y_node<=ymes
    if xm>=xs

    end

    x=ys-ym;          %arithmitis tou tipou

    x1=(xm-xs)^2;     %paranomastis tou tipou
    x2=(ym-ys)^2;     %paranomastis tou tipou
    y=sqrt(x1+x2);    %    -//-
    f=x/y;            %apotelesma tipou
    gontip=rad2deg(asin(f)); %apotelesma se moires
    prox=klisi_s+gontip;
    gon=-prox;         %bazoume to apotelesma arnitiko gia na
stripsei deksia



end

if (xm<=xs && ym>=ys)

    if xm>xss

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(gontip)-abs(klisi_s);

    else

    x=ym-ys;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-abs(klisi_s);

end
```

```
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   prox=abs(klisi_s)+abs(gontip);
   gon=180-prox;


end

if ( xm>=xs && ym<=ys)

    if xm<xss2

      x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=abs(gontip)-abs(klisi_s);
     gon=prox-180;



    else

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=klisi_s-abs(gontip);
     gon=180-abs(prox);
     gon=-gon;

    end
end
end
```

```
function gon=BkatoapoAmeParal(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys)

    x=ys-ym;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-klisi_s;

end

if (xm<=xs && ym>=ys)

    x=ym-ys;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-klisi_s
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(gontip)-180;
end

if ( xm>=xs && ym<=ys)

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=180-abs(gontip);

end
```

```
function gon=BpanoapoAmeKarn(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys)


    x=ys-ym;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(klisi_s)-gontip;


end

if (xm<=xs && ym>=ys)

    if xm>xss

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=klisi_s-abs(gontip);
     gon=-gon;


    else

    x=ym-ys;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-klisi_s;

    end
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   prox=klisi_s+abs(gontip);
```

```
    if prox>0
    gon=180-abs(prox);

    else
        gon=abs(prox)-180;
    end

end

if ( xm>=xs && ym<=ys)

    if xm<xss2

      x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=klisi_s-gontip;
     gon=abs(prox)-180;



    else

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=abs(klisi_s)-gontip;
     gon=abs(prox)-180;

    end
end
end
```

```
function
gon=BpanoapoAmeKatheto(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys)

    x=ys-ym;
    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(klisi_s)-abs(gontip);
end

if (xm<=xs && ym>=ys)

    x=ym-ys;
    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=gontip+klisi_s;
    gon=-gon;
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;
     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   gon=abs(klisi_s)+abs(gontip);
   gon=-gon;
end

if ( xm>=xs && ym<=ys)

    x=ym-ys;
     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(klisi_s)-abs(gontip);
     gon=-gon;

end
```

```
function gon=BpanoapoAmeParal(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys)


    x=ys-ym;
    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=180-abs(gontip);
end

if (xm<=xs && ym>=ys)

    x=ym-ys;
    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    gon=abs(gontip)-180;
end

if ( xm>=xs && ym>=ys)

    x=ys-ym;
     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
   gon=abs(klisi_s)+abs(gontip);
   gon=-gon;
end

if ( xm>=xs && ym<=ys)


    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(klisi_s)-abs(gontip);
     gon=-gon;

end
end
```

```
function gon=BpanoapoAmeKthet(xm,ym,xs,ys,xss,yss,xss2,yss2,klisi_s)


if (xm<=xs && ym<=ys) %an x_node<=xmes && y_node<=ymes
    if xm>=xs

    end

    x=ys-ym;

    x1=(xm-xs)^2;
    x2=(ym-ys)^2;
    y=sqrt(x1+x2);
    f=x/y;
    gontip=rad2deg(asin(f));
    prox=abs(klisi_s)+abs(gontip);
    gon=180-prox;


end

if (xm<=xs && ym>=ys) %an x_node<=xmes && y_node>=ymes

    if xm>xss2 && ym<yss2 %an x_node>x2 && y_node<y2

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     prox=klisi_s-abs(gontip);
     gon=180-abs(prox);
     gon=-gon;



    else

    x=ym-ys;   %afairo to y_node apo to ymes arithmitis tou tipou

    x1=(xm-xs)^2;    % brisko tin apostasi paranomastis tou tupou
    x2=(ym-ys)^2;    %              -//-
    y=sqrt(x1+x2);   %              -//-
    f=x/y;           %briko to apotelesma tou tipou
    gontip=rad2deg(asin(f)); % metatrepo se moires
    prox=klisi_s-abs(gontip); %afairo tis moires pou brika apo tin
klisi
    gon=180-prox;             %k epeita apo tis 180

    if gon>180                %an einai megaliteri apo 180
        gon=gon-360;          %afairo apo tis 360
    end


end
end
```

```
if ( xm>=xs && ym>=ys)

    x=ys-ym;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(klisi_s)+abs(gontip);
     gon=-gon;


end

if ( xm>=xs && ym<=ys)

    if xm>xss2

      x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=abs(gontip)-abs(klisi_s);




    else

    x=ym-ys;

     x1=(xm-xs)^2;
     x2=(ym-ys)^2;
     y=sqrt(x1+x2);
     f=x/y;
     gontip=rad2deg(asin(f));
     gon=klisi_s-abs(gontip);


    end
end
end
```

Function SindesiPcNxt

```
function SindesiPcMeNxt


%COM_MakeBTconfigFile(); %dimiourgia arxeiou bluetooth.ini
%handle=Com_OpenNXTEx('Bluetooth','00165309A95A','bluetooth.ini',
'check') %anoigma epikoinonias meso epilegmenis  sindesis
%COM_SetDefaultNXT(handle);


COM_CloseNXT('all')
handle=COM_OpenNXT('bluetooth.ini','check');
COM_SetDefaultNXT(handle);
```

Όταν προσπαθήσουμε να συνδέσουμε για πρώτη φορά το matlab με το nxt θα πρέπει να τρέξουμε και τις γραμμές του κώδικα που φαίνονται παραπάνω σε σχόλια.

Επεξήγηση SindesiPcMeNxt

Πρώτα θα πρέπει να δημιουργήσουμε ένα αρχείο τύπου ini. Καλώντας λοιπόν την συνάρτηση *COM_MakeBTconfigFile()* μας εμφανίζει ένα παράθυρο όπου θα πρέπει να βάλουμε τις ρυθμίσεις της σύνδεσης Bluetooth, για την δημιουργία του αρχείου, βάζοντας ως SeirialPort την πόρτα που μας δίνει το BlueSoleil. *ΕΙΚ.(17)*

C:\Users\PHIL\Documents\MATLAB\ptixiaki\NXT_m_Files\bluetooth.ini

```
1  [Bluetooth]
2
3  SerialPort=COM31
4  BaudRate=57600
5  DataBits=8
6
7  SendSendPause=0
8  SendReceivePause=0
9
10 Timeout=2
11
12
```

ΕΙΚ.(17)

Έπειτα πρέπει να ανοίξουμε μια σύνδεση με το nxt καλώντας την *Com_OpenNXTEx* με ορίσματα τον τύπο της σύνδεσης που θέλουμε, την mac διεύθυνση της συσκευής μας, το αρχείο ini που δημιουργήσαμε και τον έλεγχο που θέλουμε να κάνει. Τοποθετούμε το αποτέλεσμα της συνάρτησης αυτής σε ένα αντικείμενο και το θέτουμε default, καλώντας την *COM_SetDefaultNXT* , έτσι ώστε να μην χρειάζεται κάθε φορά να κάνουμε την ίδια διαδικασία.

Έχοντας κάνει ήδη μια φορά την διαδικασία αυτή μπορούμε να κάνουμε σε σχόλια τον παραπάνω κώδικα και κάθε φορά που καλούμε την συνάρτηση να κλείνουμε τυχόν ανοιχτές συνδέσεις με την *COM_CloseNXT* και έπειτα να τσεκάρουμε μόνο το αρχείο ini και να το θέτουμε ως default.

Κίνηση NXT με PID

```
%% PID Parameters

Kp=0.2;
Ki=0.01;
Kd=0.02; %0.02
gonia_old=0;
Int=0;

%% NXT Movement Parameters

Ports = [MOTOR_A; MOTOR_B]; % A gia deksi B gia aristera

DrivingSpeed = 22;
TurningSpeed = 22;

DrivingSpeed2=50;

Straight                    = NXTmotor(Ports);

RightW                       = NXTmotor(Ports(1)); %deksis
LeftW                       = NXTmotor(Ports(2));  %aristeros


%% Getting Snapshot

I2 = getsnapshot(vid);
imshow(I2)

%% Setting Coordinates For Desired Point

[x_epith,y_epith]=ginput(1);  % Setiing deserted points on figure

[m,n]=size(x_epith);

for t=1:1:m                   % Going over all data from ginput


        x=x_epith(t);
        y=y_epith(t);



while tim>0


    Straight.Power              = DrivingSpeed;
    Straight.SendToNXT();

   pause(0.05)


I = getsnapshot(vid);
%% Calling Function Processing Snapshots
```

```
L= prosnap(I); % Calling "prosnap" function

%% Finding Centroids

s = regionprops(L,'Centroid','Area');

%% Showing Line On Original Image Between The Circles



xmes=(s(1).Centroid(1,1) + s(2).Centroid(1,1))/2; % half x coordinate
values of "Circle Line"
ymes=(s(1).Centroid(1,2) + s(2).Centroid(1,2))/2; % half y coordinate
values of "Circle Line"



distance=sqrt((xmes-x)^2+(ymes-y)^2) %distance between desired point
(x-y) and average value (xmes-ymes)

%% Finding The Bigger Area-Angle and Setting NXT Movement

if (s(1,1).Area<s(2,1).Area) % comparing Areas


x1=s(1).Centroid(1,1);
y1=s(1).Centroid(1,2);

x2=s(2).Centroid(1,1);
y2=s(2).Centroid(1,2);



gonia=Euresi_Gonias(x,y,xmes,ymes,x1,y1,x2,y2) %angle between average
value and desired point
gonia_e=abs(gonia) % angle in absolute value for PID use


P= gonia_e*Kp;

Int=(Int+gonia_e)*Ki;

D=(gonia_e-gonia_old)*Kd;

gonia_old=gonia_e;

PID=abs(P+Int+D);
PID=round(PID)


    if gonia>0 && gonia>10 % kouna aristero troxo..opos to blepeis
apo piso
```

```
     RightW.Power = TurningSpeed+PID;
     LeftW.Power =  DrivingSpeed-PID;


     RightW.SendToNXT();
     LeftW.SendToNXT();

    pause(1.14) %% auto 1,4



    elseif gonia>0 && abs(gonia)<10 || distance<25
cc=cc+1


        break


     end


   if gonia<0 && abs(gonia)>10 %&& distance>60% kouna deksi
troxo..opos to blepeis apo piso



        RightW.Power = DrivingSpeed-PID ;
        LeftW.Power = TurningSpeed+PID;

         LeftW.SendToNXT();
         RightW.SendToNXT();


        pause(1.14) %% auto 1.4







   elseif gonia<0 && abs(gonia)<10 || distance<25
ff=ff+1

            break


   end
```

```
else

x1=s(2).Centroid(1,1);
y1=s(2).Centroid(1,2);

x2=s(1).Centroid(1,1);
y2=s(1).Centroid(1,2);



gonia=Euresi_Gonias(x,y,xmes,ymes,x1,y1,x2,y2) %angle between average
value and desired point
gonia_e=abs(gonia) % angle in absolute value for PID use




%% PID Controller

P= gonia_e*Kp;

Int=(Int+gonia_e)*Ki;

D=(gonia_e-gonia_old)*Kd;

gonia_old=gonia_e;

PID=abs(P+Int+D);
PID=round(PID)



     if gonia>0 && abs(gonia)>10 %&& distance>60% kouna aristero
troxo..opos to blepeis apo piso



     RightW.Power = TurningSpeed+PID;
     LeftW.Power =  DrivingSpeed-PID ;


     RightW.SendToNXT();
     LeftW.SendToNXT();

    pause(1.14) %% 1.4



    elseif gonia>0 && abs(gonia)<10 || distance<25


       break
```

```
     end


     if gonia<0 && abs(gonia)>10 % kouna deksi troxo..opos to blepeis
apo piso


        RightW.Power = DrivingSpeed-PID;
        LeftW.Power = TurningSpeed+PID;

         LeftW.SendToNXT();
         RightW.SendToNXT();


        pause(1.14)  %% 1.4



        elseif gonia<0 && abs(gonia)<10 || distance<25

      break

      end


end


end



Straight.Power              = DrivingSpeed+20;  % go straight
Straight.SendToNXT();

%if the distance gets smaller than 20 then break the while loop and
go to
%the next point.

while distance>35

    I4 = getsnapshot(vid);
%% Calling Function Processing Snapshots

L= prosnap(I4);

%% Finding Centroids

st = regionprops(L,'Centroid','Area');

%% Finding Distance
```

```
 try
xmes=(st(1).Centroid(1,1) + st(2).Centroid(1,1))/2; % half x
coordinate values of "Circle Line"
ymes=(st(1).Centroid(1,2) + st(2).Centroid(1,2))/2; % half y
coordinate values of "Circle Line"
 end


    distance=sqrt((xmes-x)^2+(ymes-y)^2)







end
end
%when all point have finished stop the nxt.
StopMotor('all' , 'brake');
```

Κώδικας υλοποίησης GNG

```
%% GNG GRAPH ALGORITHM

I_nodes = getsnapshot(vid);
 figure(1),imshow(I_nodes)
 hold on

%% Creating Strel Properties
disk=strel('square',8);
se_dilate = strel('square',31);

 %% Processing Snapshots
I2=rgb2gray(I_nodes);
dilatedI = imdilate(I2,se_dilate); %psifiaki diastoli simeion
level = graythresh(dilatedI);
level=level+0.2;
bwi=im2bw(dilatedI,level);         % Converting image to binary
bwi = bwareaopen(bwi,18);
bwi=imclose(bwi,disk);           % Morphologically open image
bwi=imclearborder(bwi);
L = bwlabel(bwi);
bwi2=~bwi;                         % kano ta 0 1 k ta 1 0
L2=bwlabel(bwi2);                  % brisko to label to bwi2
s_bw=regionprops(L2,bwi,'PixelList','PixelValues'); % k edo sindiazo
to L2 tou bwi2 me to bwi opote brisko ta coordinates ton pixel me
value 0

%% Duplicating The Data

repmat_matrix=repmat(s_bw.PixelList,1); %ftiaxno enan pinaka idio me
ton PixelList
```

```
matrix1=cat(1,s_bw.PixelList(:,1),repmat_matrix(:,1)); %perno ta x
tis protis stilis tou martix1 k ta bazo stin sinexeia tou PixelList
matrix1(:,2)=cat(1,s_bw.PixelList(:,2),repmat_matrix(:,2)); % perno k
ta y

data_points=matrix1;   % kai ta data mou tou matrix1 ta bazo ston
data_points.

%% Calling GNG Function
[nodes,ad,nn]=GNG(data_points);

%% Calling Distance_Nodes Function
data_dist_nodes=distance_nodes(ad,nodes); %euresi barous kathe akmis
tou  kombou

%% Drawing i Of Every Node On The Original Image

[m,n]=size(nodes);
for i=1:m

text(nodes(i,1),nodes(i,2),sprintf('%d',i),'Color','Y','FontWeight','
bold')
end
```

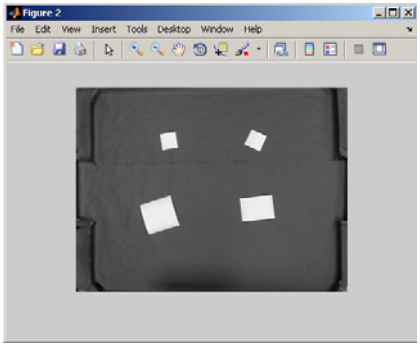
Επεξήγηση κώδικα

Στο κομμάτι του κώδικα αυτού ακολουθούμε την ίδια νοοτροπία που ακολουθήσαμε και τις προηγούμενες φορές προσθέτοντας κάποια επιπλέον χαρακτηριστικά. Πιο αναλυτικά.

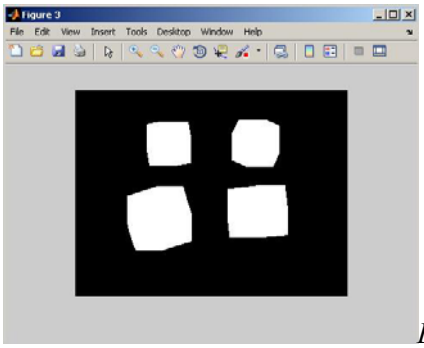
Βήμα 1ο: Παίρνουμε snapshot από την κάμερα, το εμφανίζουμε σε ένα figure και διατηρούμε το συγκεκριμένο snapshot στο figure, χρησιμοποιώντας την *hold on*, ώστε να μπορέσουμε έπειτα να προσθέσουμε και τον γράφο πάνω στην αρχική εικόνα.

Βήμα 2ο: Φτιάχνουμε τις ρυθμίσεις της *strel* συνάρτησης, ένα τετράγωνο με μέγεθος 8 pixel (μεταβλητή disk) και άλλο ένα τετράγωνο με μέγεθος 31 pixel (μεταβλητή se_dilate) για να το χρησιμοποιήσουμε στην *imdilate* συνάρτηση.

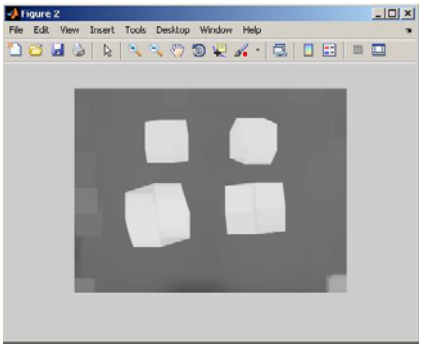
Βήμα 3ο: Μετατρέπουμε την αρχική εικόνα σε grayscale κλίμακα *ΕΙΚ.(21)*. Έπειτα χρησιμοποιούμε την *imdilate* συνάρτηση η οποία μπορεί να διαστέλλει μια εικόνα παίρνοντας ως όρισμα την εικόνα που θέλουμε να διαστέλλουμε και το αντικείμενο της *strel* συνάρτησης *ΕΙΚ.(22)*. Βρίσκουμε το threshold της διασταλμένης εικόνας, την μετατρέπουμε σε binary (δηλαδή σε 0 και 1) , συνεχίζουμε κάνοντας την επεξεργασία της εικόνας όπως έχουμε αναφέρει στα προηγούμενα κεφάλαια με την μόνη διαφορά ότι , τώρα παίρνουμε το label της επεξεργασμένης εικόνας *ΕΙΚ.(23)*, στην συνέχεια μετατρέπουμε τα 0 και 1 σε 1 και 0 της εικόνας αντίστοιχα (`bwi2=~bwi`), φτιάχνοντας έτσι μια καινούργια εικόνα την `bwi2` *ΕΙΚ.(24)*.Στο τελικό στάδιο της επεξεργασίας μας χρησιμοποιούμε την *regionprops* με όρισμα το label της bwi2 (L2) και την εικόνα bwi, βάζοντας ως επιλογές το PixelList και PixelValues .Με αυτόν τον τρόπο βρίσκουμε τις συντεταγμένες που έχουν PixelValue=0 δηλαδή όλης της πλατφόρμας εκτός των εμποδίων.



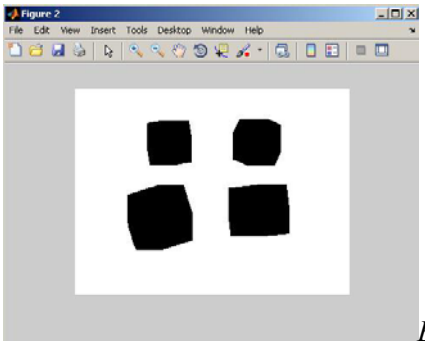
ΕΙΚ.(21)



ΕΙΚ.(23)



ΕΙΚ.(22)



ΕΙΚ.(24)

Βήμα 4ο : Στο παρόν βήμα διπλασιάζουμε τα δεδομένα μας έτσι ώστε να μπορέσει ο αλγόριθμος να προλάβει να μάθει τον χώρο. Φτιάχνουμε λοιπόν έναν ίδιο πίνακα (τον `repmat_matrix`) με τον *s_bw.PixelList* (αυτόν που μας επέστρεψε η *regionprops*) χρησιμοποιώντας το *repmat* δίνοντάς του ως όρισμα τον πίνακα που θέλουμε να ξαναφτιάξουμε και το πόσες φορές τον θέλουμε. Έπειτα χρησιμοποιούμε την *cat* συνάρτηση, η οποία συγχωνεύει τον δεξί πίνακα-όρισμα στον αριστερό πίνακα-όρισμα.Με αυτόν τον τρόπο βάζουμε λοιπόν τις x συντεταγμένες του πίνακα PixelList, που είναι στην πρώτη του στήλη , και τις x συντεταγμένες του repmat_matrix, που είναι επίσης στην πρώτη του στήλη , στην πρώτη στήλη του matrix1. Την ίδια ακολουθία κάνουμε και για τις y συντεταγμένες και στο τέλος βάζουμε τα διπλά μας δεδομένα στον πίνακα `data_points.`

Βήμα 5ο : Καλούμε τον αλγόριθμο GNG δίνοντας του ως όρισμα τον πίνακα data_points.Το μόνο που έχουμε πειράξει από τον αλγόριθμο GNG είναι ο αριθμός των nodes που επιθυμούμε να έχουμε. Μετά το τέλος της εκτέλεσης του GNG μας επιστρέφει έναν πίνακα με τις συντεταγμένες των nodes, έναν πίνακα που μας δείχνει ποιο node συνδέεται με ποιο (πίνακας *ad*) έχοντας ένα 1 δίπλα σε κάθε κόμβο και την *nn* μεταβλητή που μας δείχνει πόσα nodes έχουμε.

Βήμα 6ο: Καλούμε την `distance_nodes` συνάρτηση (την οποία θα αναλύσουμε εκτενώς παρακάτω) για να βρούμε το βάρος της κάθε ακμής του γράφου έτσι ώστε να μπορούμε να καλέσουμε μετέπειτα την *graphshortestpath.*

Βήμα 7ο: Έχοντας μια *for* προσπελαύνουμε τον πίνακα nodes και για κάθε x,y συντεταγμένη του εκάστοτε κόμβου εκτυπώνουμε το i που αντιστοιχεί στις x,y συντεταγμένες.

Παραδείγματος χάριν αν θεωρήσουμε ότι το i=2 και οι συντεταγμένες του nodes(2,1)=3 και nodes(2,2)=5 τότε εκτυπώνουμε στις συντεταγμένες (3,5) το 2.Κάποιες επιλογές που μας δίνει η συνάρτηση *text* είναι ότι μπορούμε να αλλάξουμε το χρώμα, το μέγεθος και το στυλ της εκτύπωσης. Για περισσότερες επιλογές μπορεί κάποιος να ανατρέξει στο help του matlab.

GNG FUNCTION

```
function [nodes, adjacency, nn] = GNG(data)


% GNG   Growing Neural Gas model
%
%   [NODES ADJACENCY NN] = GNG(DATA)
%
%   This function trains the Growing Neural Gas model with DATA,
which is a
%   2-column array, with each row for a data point (x, y).
%
%   Returns 3 variables: NODES is a 3-column data array with NN rows,
with
%   each row for a node vector; and ADJACENCY is the node adjacency
matrix.
%
%   Reference:
%       Growing Neural Gas, Experiments with GNG,GNG with Utility and
Supervised GNG
%       Jim Holmstr?m <jimh@csd.uu.se>
%       Uppsala University, Department of Information Technology
Computer Systems
%       Box 337, SE-751 05 Uppsala, Sweden
%
%   Coded by Ming Pan <mpan@princeton.edu>, Ming Pan
<mingpan@mit.edu>
%   Jan 8, 2007


if (nargin ~= 1)
    error('You must supply one array of training data points.');
end

datlen = length(data);
```

```
if (datlen < 10)
    error('Training data must contain >=10 points.');
end



% some params
MAXNODES = 30;       % 50
EW       = 0.07;     % 0.07
EN       = 0.008;    % 0.008
AMAX     = 50;       % 50
LAMBDA   = 200;      % 200
BETA     = 0.0005;   % 0.0005



% create and initialize code vector data structure
nodes     = zeros(MAXNODES, 3);  % each node: (x, y, error)
adjacency = speye(MAXNODES)*0;   % adjacency/age matrix, 0: no edge,
1: new edge, n: age n
nn        = 0;                    % number of nodes

% initialize the graph
for i=1:2
    [nodes, nn] = add_node(nodes, nn, data(i, 1), data(i, 2), 0);
end
adjacency(1, 2) = 1;
adjacency(2, 1) = 1;



% start training

for i=3:datlen

    ii = ceil(rand*datlen);
    input = data(ii, :);

    % look for two nearest nodes
    distance = nodes(1:nn, 1:2) - repmat(input, nn, 1);
    distance = sum(distance.^2, 2);

    [d1, s1] = min(distance);  distance(s1) = 100000000;
    [d2, s2] = min(distance);

    % update winner node error
    nodes(s1, 3) = nodes(s1, 3) + d1;

    % move the 1st winner and its neighbors

    neighbors = adjacency(:, s1)>0;

    nodes(neighbors, 1:2) = nodes(neighbors, 1:2) + ( repmat(input,
sum(neighbors), 1) - nodes(neighbors, 1:2) ) * EN;

    nodes(s1, 1:2) = nodes(s1, 1:2) + (input - nodes(s1, 1:2)) * (EW-
EN);
```

```
    % increment the neighbors' age
    adjacency(neighbors, s1) = adjacency(neighbors, s1) + 1;
    adjacency(s1, neighbors) = adjacency(s1, neighbors) + 1;

    % create/reset the edge between 1st and 2nd winners
    adjacency(s1, s2) = 1;
    adjacency(s2, s1) = 1;

    % remove edges that are too old
    adjacency(adjacency>AMAX) = 0;

    % remove nodes if not edges to connect
    for j=1:nn
        if ( sum(adjacency(j, :)) == 0 )
            if (j < nn)
                nodes(j, :) = nodes(nn, :);
                adjacency(j, :) = adjacency(nn, :);
                adjacency(:, j) = adjacency(:, nn);
            end

            nodes(nn, :) = 0;
            adjacency(nn, :) = 0;
            adjacency(:, nn) = 0;

            nn = nn - 1;
        end
    end



    % insert a new node every lambda-th iteration while maxnodes
unreached
    if ( mod(i, LAMBDA) == 0 & nn< MAXNODES )


        % find the node with largest error
        [e1 s1] = max(nodes(1:nn, 3));


        % and its neighbor with largest error
        neighbors = adjacency(:, s1)>0;
        [e2 s2] = max(nodes(:, 3).*neighbors);


        % insert the new node
        new_node = (nodes(s1, :) + nodes(s2, :)) / 2;
        new_node(3) = nodes(s1, 3);
        nn = nn + 1;
        nodes(nn, :) = new_node;


        % break the old edge
        adjacency(s1, s2) = 0;
        adjacency(s2, s1) = 0;


        % create edges to the new node
```

```
        adjacency(s1, nn) = 1;
        adjacency(nn, s1) = 1;
        adjacency(s2, nn) = 1;
        adjacency(nn, s2) = 1;


    end


    % decrease the errors for all nodes
    nodes(1:nn, 3) = nodes(1:nn, 3)*(1-BETA);


end




nodes = nodes(1:nn, 1:2);
adjacency = spones(adjacency(1:nn, 1:nn));

gplot(adjacency, nodes, 'ro-');

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% supporting functions from here on, they are all too
% trivial and won't used. For learning purposes only.




% function to add a node
function [nodes, nn] = add_node(nodes, nn, x, y, error0)

nn = nn + 1;
nodes(nn, :) = [x y error0];



% function to add an edge
function adjacency = add_edge(adjacency, node1, node2, age0)

adjacency(node1, node2) = age0;
adjacency(node2, node1) = age0;


% function to delete a node

function [nodes, adjacency, nn] = delete_node(nodes, adjacency, nn,
node1)

if (node1 < nn)
    nodes(node1, :) = nodes(nn, :);
    adjacency(node1, :) = adjacency(nn, :);
    adjacency(:, node1) = adjacency(:, nn);
end

nodes(nn, :) = 0;
```

```
adjacency(nn, :) = 0;
adjacency(:, nn) = 0;

nn = nn - 1;



% function to delete an edge
function adjacency = delete_edge(adjacency, node1, node2)

adjacency(node1, node2) = 0;
adjacency(node2, node1) = 0;
```

Distance Nodes Function

```
function data_dist_nodes=distance_nodes(ad,nodes)

%% Fixing Data

full_ad=full(ad); % kano ton pinaka ad full gia na mporo na ton
prospelaso
dupli_ad=full_ad; % ftiaxno ton dupli outos oste na allakso ta 1 tou
ad stis
                  % antistoixes apostaseis.


%% Finding Distance for every node

[m_fad,n_fad]=size(full_ad);


for i=1:m_fad        %prospelauno to pinaka full_ad
    for j=1:n_fad

       if  full_ad(i,j)==1  %tsekaro an einai i

           x_stand=nodes(i,1); % kratao ta x tou kombou mou

           y_stand=nodes(i,2); % kratao ta y tou kombou mou

           x_change=nodes(j,1); % kratao ta x ton giro kombon pou
briskontai apo ton kombo mou

           y_change=nodes(j,2); % kratao ta y ton giro kombon pou
briskontai apo ton kombo mou


  dist_nodes=sqrt((x_stand-x_change)^2+(y_stand-y_change)^2); %brisko
tin ekastote apostasi
```

```
  dupli_ad(i,j)=dist_nodes; %bazo tin apostasi sto simeio pou brika
to 1


        end

    end
end


%% Making Our Data a Sparse Matrix

data_dist_nodes=sparse(dupli_ad); %kano ta apotelesmata mou stin
arxiki domiki morfi tou ad
```

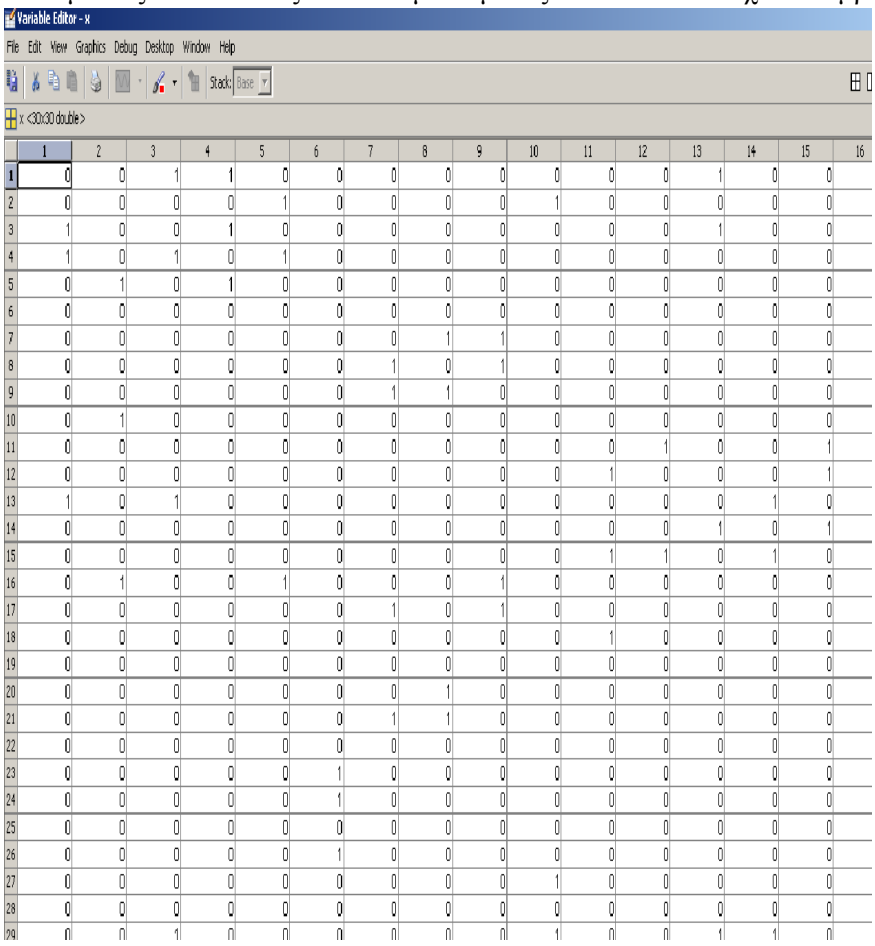
Επεξήγηση Distance Nodes Function

Βήμα 1ο: Ο πίνακας *ad* που επιστρέφεται από την εκτέλεση του GNG είναι τύπου sparse *ΕΙΚ.(25)* .Sparse είναι ο τύπος του πίνακα που είναι μη μηδενικός και εμπεριέχει διαφόρων τύπων δεδομένα. Για να μπορέσουμε να τον προσπελάσουμε θα πρέπει να τον φέρουμε στην γνωστή μορφή πίνακα, για τον σκοπό αυτό χρησιμοποιούμε την συνάρτηση *foul ΕΙΚ.(26)* . Παράλληλα φτιάχνουμε έναν ίδιο πίνακα με τον *foul_ad* (τον dupli_ad) για να μπορέσουμε στο τέλος να βάλουμε τις αποστάσεις των ακμών μεταξύ των αντίστοιχων κόμβων.

```
ad =

   (3,1)        1
   (4,1)        1
   (13,1)       1
   (5,2)        1
   (10,2)       1
   (16,2)       1
   (30,2)       1
   (1,3)        1
   (4,3)        1
   (13,3)       1
   (29,3)       1
   (1,4)        1
   (3,4)        1
   (5,4)        1
   (2,5)        1
   (4,5)        1
   (16,5)       1
   (23,6)       1
   (24,6)       1
   (26,6)       1
   (8,7)        1
   (9,7)        1
   (17,7)       1
   (21,7)       1
   (7,8)        1
   (9,8)        1
   (20,8)       1
   (21,8)       1
   (30,8)       1
```

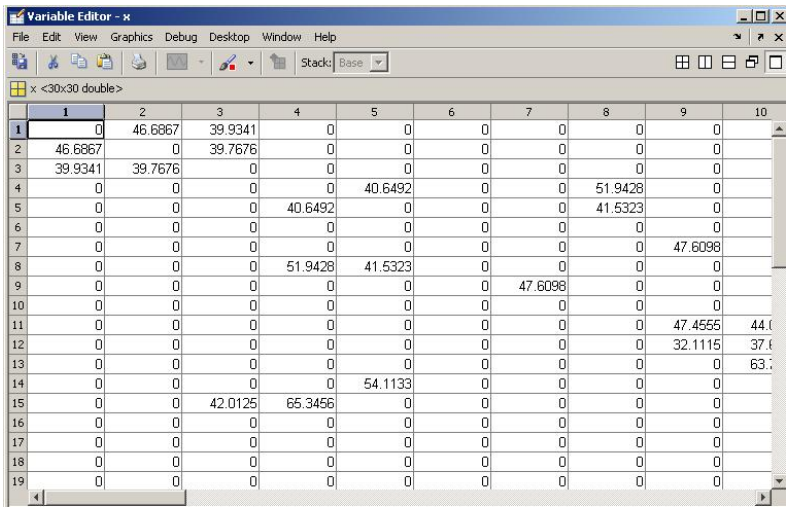
ΕΙΚ.(25)



ΕΙΚ.(26)

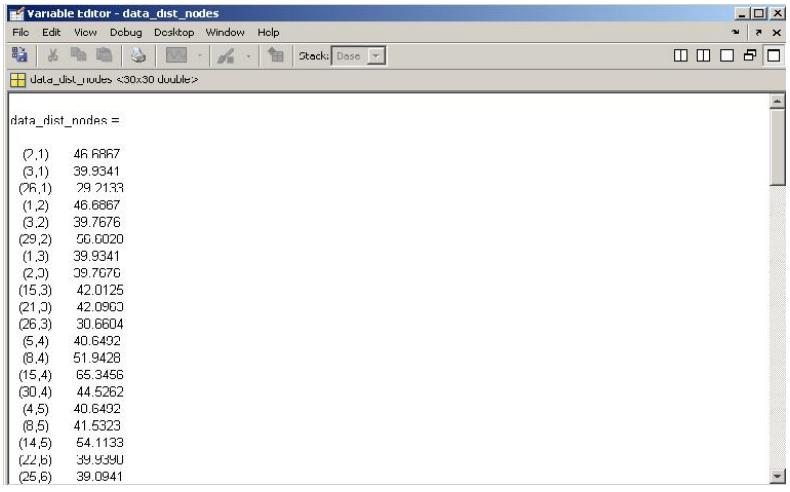
Βήμα 2ο: Βρίσκουμε μέσω της συνάρτησης *size* ,η οποία επιστρέφει τον μεγαλύτερο αριθμό γραμμών και στηλών του πίνακα που έχουμε δώσει για όρισμα, το μέγεθος του πίνακα για να τον προσπελάσουμε σωστά. Έπειτα ελέγχουμε κάθε φορά αν το περιεχόμενο του πίνακα είναι ίσο με 1 , που σημαίνει ότι υπάρχει σύνδεση. Αν είναι τότε κρατάμε τις συντεταγμένες του i ως τον κόμβο που θέλουμε να τσεκάρουμε τις αποστάσεις των ακμών του με τους άλλους κόμβους που συνδέεται. Για να δούμε με ποιους άλλους κόμβους συνδέεται παίρνουμε το j.

Παραδείγματος χάριν αν θεωρήσουμε ότι το *ad(1,2)==1* τότε για *x_stand* παίρνουμε το *node(1,1)=3* οπότε έχουμε *x_stand =3* και για *y_stand* παίρνουμε *node(1,2)=6* οπότε έχουμε *y_stand=6,* αυτός θα είναι ο κόμβος που θα ελέγξουμε για τις άλλες του ακμές. Από εδώ επίσης καταλαβαίνουμε ότι η πρώτη στήλη του *node* είναι τα x και η δεύτερη τα y. Το ίδιο ισχύει και για το *x_change* και *y_change* μόνο που αντί για i βάζουμε j. Έχοντας λοιπόν τα x και y του σταθερού και μεταβλητών μας κόμβων βρίσκουμε την απόσταση μεταξύ τους μέσω του τύπου της απόστασης. Το αποτέλεσμα το βάζουμε στον πίνακα *dupli_ad* που αναφέραμε παραπάνω και τέλος επειδή ο πίνακας μας είναι *full EIK.(27)* τον μετατρέπουμε πάλι σε *sparse EIK.(28)* για να μπορέσουμε να τον χρησιμοποιήσουμε στην *graphshortestpath*.



	1	2	3	4	5	6	7	8	9	10
1	0	46.6867	39.9341	0	0	0	0	0	0	
2	46.6867	0	39.7676	0	0	0	0	0	0	
3	39.9341	39.7676	0	0	0	0	0	0	0	
4	0	0	0	0	40.6492	0	0	51.9428	0	
5	0	0	0	40.6492	0	0	0	41.5323	0	
6	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	0	0	0	47.6098	
8	0	0	0	51.9428	41.5323	0	0	0	0	
9	0	0	0	0	0	0	47.6098	0	0	
10	0	0	0	0	0	0	0	0	0	
11	0	0	0	0	0	0	0	0	47.4555	44.(
12	0	0	0	0	0	0	0	0	32.1115	37.(
13	0	0	0	0	0	0	0	0	0	63.7
14	0	0	0	0	54.1133	0	0	0	0	
15	0	0	42.0125	65.3456	0	0	0	0	0	
16	0	0	0	0	0	0	0	0	0	
17	0	0	0	0	0	0	0	0	0	
18	0	0	0	0	0	0	0	0	0	
19	0	0	0	0	0	0	0	0	0	

EIK.(27)



EIK.(28)

Κίνηση NXT με GNG

Στο κομμάτι αυτό θα δούμε τις αλλαγές που κάναμε στο κομμάτι της κίνησης σε σχέση με τη κίνηση με PID.

```
%% KINISI STON GRAFO


tim=1;
gonia_stop=8;

%% NXT Movement Parameters

Ports = [MOTOR_A; MOTOR_B]; % A gia deksi B gia aristera

DrivingSpeed = 9;
TurningSpeed = 9;


Straight                    = NXTmotor(Ports);

RightW                       = NXTmotor(Ports(1)); %deksis
LeftW                       = NXTmotor(Ports(2));  %aristeros



%% Setting Coordinates For Desired Point

[dist,path] = graphshortestpath(data_dist_nodes,6,24);

[m_path,n_path]=size(path);

for t=1:1:n_path                    % Going over all data from ginput


        x_node=nodes(path(t),1);
        y_node=nodes(path(t),2);



while tim>0


I = getsnapshot(vid);
%% Calling Function Processing Snapshots

L= prosnap(I); % Calling "prosnap" function

%% Finding Centroids

s = regionprops(L,'Centroid','Area');

[m,n]=size(s);
```

```
clear min    %kano clear to min giati allios krataei k ta palia
dedomena k meta mpredeute to robot-dn leitourgei sosta to programma
for i=1:m    %ftiaxno pinaka me ola ta Area.
    if s(i,1).Area<90
min(i)=s(i,1).Area;
    end
end
clear apotel %kano clear to min giati allios krataei k ta palia
dedomena k meta mpredeute to robot-dn leitourgei sosta to programma
[apotel,Seir]=sort(min,'descend');
               %taksinomo ton pinaka me ta Area kata auksonta arithmo
               % apotel einai to apotelesma tou pinaka
               % taksinomimenou kai Seir einai i seira pou
               % emfanizontai opos einai dld sto s.Area


%% Showing Line On Original Image Between The Circles

try
xmes=(s(Seir(1,1)).Centroid(1,1) + s(Seir(1,2)).Centroid(1,1))/2; %
half x coordinate values of "Circle Line"
ymes=(s(Seir(1,1)).Centroid(1,2) + s(Seir(1,2)).Centroid(1,2))/2; %
half y coordinate values of "Circle Line"
end



distance=sqrt((xmes-x_node)^2+(ymes-y_node)^2) %distance between
desired point (x-y) and average value (xmes-ymes)

%% Finding The Bigger Area-Angle and Setting NXT Movement

if (s(Seir(1,1)).Area<s(Seir(1,2)).Area) % comparing Areas


x1=s(Seir(1,1)).Centroid(1,1);
y1=s(Seir(1,1)).Centroid(1,2);

x2=s(Seir(1,2)).Centroid(1,1);
y2=s(Seir(1,2)).Centroid(1,2);



gonia=Euresi_Gonias(x_node,y_node,xmes,ymes,x1,y1,x2,y2) %angle
between average value and desired point

    if gonia>0 && gonia>gonia_stop % %kouna aristero troxo..opos to
blepeis apo piso



     RightW.Power= TurningSpeed;
     LeftW.Power =  -(DrivingSpeed);


     RightW.SendToNXT();
     LeftW.SendToNXT();
```

```
    elseif gonia>0 && abs(gonia)<gonia_stop


        break

     end


   if gonia<0 && abs(gonia)>gonia_stop


        RightW.Power= -(DrivingSpeed);
        LeftW.Power = TurningSpeed;

         LeftW.SendToNXT();
         RightW.SendToNXT();



   elseif gonia<0 && abs(gonia)<gonia_stop


            break


   end


else

x1=s(Seir(1,2)).Centroid(1,1); %to x tou mikroterou giati to deutero
keli tis seir einai mikrotero apo to proto kiklos
y1=s(Seir(1,2)).Centroid(1,2); %to idio alla to y kiklos

x2=s(Seir(1,1)).Centroid(1,1); %to x tou megaliterou tetragono
y2=s(Seir(1,1)).Centroid(1,2); %to y you megaliterou tetragono



gonia=Euresi_Gonias(x_node,y_node,xmes,ymes,x1,y1,x2,y2) %angle
between average value and desired point


     if gonia>0 && abs(gonia)>gonia_stop %&& distance>60% kouna
aristero troxo..opos to blepeis apo piso
```

```
        RightW.Power = TurningSpeed;
        LeftW.Power =  -(DrivingSpeed);


        RightW.SendToNXT();
        LeftW.SendToNXT();



      elseif gonia>0 && abs(gonia)<gonia_stop


         break


        end


        if gonia<0 && abs(gonia)>gonia_stop



           RightW.Power = -(DrivingSpeed);
            LeftW.Power= TurningSpeed;

            LeftW.SendToNXT();
            RightW.SendToNXT();






           elseif gonia<0 && abs(gonia)<gonia_stop


         break

         end

  end



  end
```

```
Straight.Power              = DrivingSpeed+19;
Straight.SendToNXT();

%if the distance gets smaller than 20 then break the while loop and
go to
%the next point.

while distance>20


    I4 = getsnapshot(vid);
%% Calling Function Processing Snapshots

L= prosnap(I4);

%% Finding Centroids

st = regionprops(L,'Centroid','Area');
[m_st,n_st]=size(st);

clear min_st             %kano clear to min giati allios krataei k ta
palia dedomena k meta mpredeute to robot-dn leitourgei sosta to
programma
for j=1:m_st             %ftiaxno pinaka me ola ta Area.
    if st(j,1).Area<90
min_st(j)=st(j,1).Area;
    end
end
clear apotel_st          %kano clear to min giati allios krataei k ta
palia dedomena k meta mpredeute to robot-dn leitourgei sosta to
programma
[apotel_st,Seir_st]=sort(min_st,'descend');
               %taksinomo ton pinaka me ta Area kata fninonta arithmo
               % apotel einai to apotelesma tou pinaka
               % taksinomimenou kai Seir einai i seira pou
               % emfanizontai opos einai dld sto s.Area



 try
xmes=(st(Seir_st(1,1)).Centroid(1,1) +
st(Seir_st(1,2)).Centroid(1,1))/2; % half x coordinate values of
"Circle Line"
ymes=(st(Seir_st(1,1)).Centroid(1,2) +
st(Seir_st(1,2)).Centroid(1,2))/2; % half y coordinate values of
"Circle Line"
 end

distance=sqrt((xmes-x_node)^2+(ymes-y_node)^2)

end
StopMotor('all','brake');
end
%when all point have finished stop the nxt.
StopMotor('all' , 'brake');
```

ΒΙΒΛΙΟΓΡΑΦΙΑ

Βιβλία

- *Image Processing, Analysis & and Machine Vision - A MATLAB Companion. by Tomas Svoboda (Author), Jan Kybic (Author), Vaclav Hlavac (Author)*
- *Digital Image Processing Using MATLAB. by Rafael C. Gonzalez , Steven L. Eddins (Author)*

Ιστοσελίδες

- *http://www.mathworks.com/*
- *http://www.sciencedirect.com/*
- *http://www.springerlink.com/home/main.mpx*
- *http://www.codeproject.com/KB/recipes/PID.aspx*
- *http://web.cecs.pdx.edu/~gerry/MATLAB/programming/basics.html*

Papers

- A Fuzzy Mechanism for Action Selection of Soccer Robots, CHIA-JU WU and TSONG-LI LEE.
- The CAS Robot Navigation Toolbox.
- A Real-Time Role Assignment Mechanism for Five-on-Five Robot Soccer Competition, Chi-Yang Chen and Tzuu-Hseng *S.* Li.
- Design and Implementation of a Prototype Vision-guided Golf-ball Collecting Mobile Robot, Sun-Li Wu and Ming-Yang Cheng.
- System design and strategy integration for five-on-five robot soccer competition, Ming-Yuan Shieh, Juing-Shian Chiou, Tien-Lung You, Ke-Hao Chang and Shen-Pao Cheng.
- A MATHEMATICAL MODEL OF A LEGO DIFFERENTIAL DRIVE ROBOT, Kevin J Worrall+ and Euan W McGookino