ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Αναφορά προόδου
Guard Dog Project

Υπεύθυνος καθηγητής: Γ. Παπαϊωάννου

Αμμάρ Γκαμάζ

11/07/2008

ΠΕΡΙΕΧΟΜΕΝΑ

Το project βρίσκεται πλέον σε αρκετά ώριμο στάδιο με τα βασικά του συστήματα πλήρως λειτουργικά . Η κατασκευή του ρομπότ διαιρέθηκε σε 5 υποεργασίες , οι οποίες διαφέρουν πάρα πολύ η μια από την άλλη , αλλά είναι ολες εξίσου σημαντικές για την σωστή λειτουργία του τελικού αποτελέσματος. Η δε δημιουργία του προγραμματιστικού κομματιού του project έχει επίσης διαστρωματωμένη λειτουργία και έχει διαιρεθεί σε βιβλιοθήκες , οι οποίες θα μπορούσαν να χρησιμοποιηθούν και από άλλους φοιτητές του τμήματος , αλλά και απ' την ευρύτερη κοινότητα προγραμματιστών και απλοποιούν πολύ κάποιο παρόμοιο μελλοντικό εγχείρημα.

Επίσης για την εγκατάσταση του συστήματος απαιτείται μόνο το λειτουργικό σύστημα (Windows XP) , DirectX 8.1 + , Microsoft Speech API Libraries και NXT Fantom Drivers , ενώ ο πηγαίος κώδικας για όλα τα υπόλοιπα προγράμματα είναι διαθέσιμος.

Πιο συγκεκριμένα το project διαιρέθηκε στα παρακάτω :

Α) Επιλογή όλων των low level υλικών που θα απαρτίζουν το hardware αλλά και του αντίστοιχου λειτουργικού , βιβλιοθηκών και υπολοίπου προγραμματιστικού υποβάθρου.

Β) Φυσική κατασκευή μιας ανθεκτικής κινούμενης πλατφόρμας με μεγάλη έμφαση στο απαραίτητο βάρος , στην διατήρηση χαμηλής θερμοκρασίας , την υψηλή σταθερότητα αλλά και την εξωτερική εμφάνιση του ρομπότ.

Γ) Το πρόβλημα της ενέργειας καθώς η αυτόνομη λειτουργία σημαίνει πως θα έπρεπε να κατασκευαστεί ένα κύκλωμα φόρτισης , για να είναι δυνατή η αλλαγή μεταξύ AC ρεύματος και των μπαταριών , με όσο το δυνατόν μικρότερο κόστος βάρους και όσο το δυνατόν περισσότερη ώρα μεταξύ φορτήσεων του robot.

Δ) Η ενορχήστρωση του προγραμματιστικού μέρους της εργασίας. Το ρομπότ θα πρέπει να διαθέτει πολλούς μηχανισμούς προώθησης μηνυμάτων , σήμανσης συναγερμού , remote control , web interfaces , αλλά επίσης και την δυνατότητα για φωνητική και οπτική συνεννόηση με τους ιδιοκτήτες του.

Ε) Το πρόγραμμα , το οποίο θα αποτελεί τον εγκέφαλο του Guard Dog και θα αναλαμβάνει την ανάλυση δεδομένων , είτε αυτά είναι οπτικά , ηχητικά , υπέρηχοι , ανάδραση στις κίνησεις του με στόχο την σωστή περιδιάβαση του χώρου που θα φρουρεί και την σήμανση συναγερμού και καταδίωξη του πιθανού εισβολέα.

Όπως είναι προφανές κάθε σφάλμα σε κάποιο από τα παραπάνω έχει καταστροφικές συνέπειες και η αστοχία του καθιστά άχρηστα όλα τα υπόλοιπα μέρη του project. Για παράδειγμα σε περίπτωση που η πηγή ενέργειας αδειάσει , αυτό αυτόματα θέτει τα πάντα εκτός λειτουργίας , ενώ αν για παράδειγμα η φυσική κατασκευή καταρρεύσει λόγω του βάρους θα καθιστούσε το robot ακίνητο. Σε κάθε περίπτωση όμως το ρομπότ παρέχει έναν μηχανισμό για network polling , ο οποίος σε περίπτωση που σταματήσει την εκπομπή , ή διαγνωθεί κάποιο μηχανικό error state (πχ χαμηλό voltage της μπαταρίας) , θέτει το σύστημα σε κατάσταση πανικού έτσι ώστε να μην τεθεί σε κίνδυνο ο στόχος της όλης εφαρμογής και στην συνέχεια περιμένει ανθρώπινη επέμβαση για την διόρθωση του προβλήματος..

Επίσης για λόγους συντομίας παραλείπονται στο παρόν Report ολα τα τεχνικά σημεία της κατασκευής (Β κομμάτι του project) καθώς είναι μάλλον τετριμμένο πρόβλημα το οποίο περιγράφεται με μια ματιά στις φωτογραφίες που έχουν προστεθεί στο τέλος .

Παρακάτω ακολουθεί μια συνοπτική περίληψη των συστατικών του project.

Hardware :

Ο στόχος της αγοράς hardware ήταν μια υλοποίηση ελαφρία σε βάρος , με πολυ χαμηλή ενεργειακή κατανάλωση , καλή θερμική συμπεριφορά , δυνατότητα ασύρματης δικτύωσης και όσο το δυνατόν μεγαλύτερη επεξεργαστική ισχύ.

Με δεδομένο το διαθέσιμο budget το οποίο ευτυχώς ήταν αρκετά μεγάλο μετά από έρευνα αγοράς επέλεξα για motherboard την Intel D201GLY2 , με 1.2Ghz Celeron CPU , 512MB RAM DDR 2 , Atheros based PCI 802.11 b/g wifi , 2GB CF Card που θα έπαιρνε τον ρόλο σκληρού δίσκου , Passive ηχεία με μεγάλους μαγνήτες έτσι ώστε να μην είναι απαραίτητη η τροφοδοσία τους παραμόνο από την έξοδο της onboard κάρτας ήχου , PicoPSU τροφοδοτικό και ένα 120mm ανεμιστήρα για την ψύξη του επεξεργαστή.

Δυστυχώς σχεδόν ένα χρόνο μετά την επιλογή του παραπάνω συστήματος η Intel (ακολουθώντας τον νόμο του Moore) διαθέτει πλέον επεξεργαστές (Intel Atom) κατασκευασμένους σε κλίμακα 45nm εφάμιλης απόδοσης οι οποίοι χρησιμοποιούν όμως 2W , αντί για τα 20W.

Για τα μάτια του ρομπότ επιλέχθηκαν 2 WebCams της Microsoft (VX-6000) με μέγιστη ανάλυση σταθερής εικόνας 5.0 Megapixels και 1.3 για Video , ενσωματωμένο μικρόφωνο , και μέγιστο framerate 30 fps για εικόνες 640 x 480 pixels. Η δε εικόνα τους είναι πολυ καθαρή και διαθέτουν μεγαλύτερη οπτική γωνία (71 μοιρών) από τα αντίστοιχα μοντέλα που θα μπορούσαν να επιλεγούν διατηρώντας παράλληλα ένα λογικό κόστος.

Να σημειωθεί τέλος ότι όλα τα παραπάνω , εκτός από τις κάμερες , είναι ελεγμένα συμβατά τόσο με linux όσο και με windows καθότι η απόφαση για το λειτουργικό και τα προγραμματιστικά περιβάλλοντα δεν είχε παρθεί ακόμα.

Operating System :

Ο στόχος επιλογής λειτουργικού συστήματος ήταν ένα φθηνό , ασφαλές , αξιόπιστο και ελαφρύ υπολογιστικά λειτουργικό σύστημα αφού το project ουσιαστικά εμπίπτει κυρίως στην κατηγορία των embedded συστημάτων, και οι επιλογές εδώ θα μπορούσαν να είναι ένα Customized Linux , Windows Embedded , Windows Vista , Windows XP και Windows 98SE .

Ένας Linux Kernel μαζί με κάποια Customized διανομή (κατα πάσα πιθανότητα Gentoo , έτσι ώστε το λειτουργικό να είναι compiled για την CPU του Guard Dog) , βασικά services (apached , wifi , ssh terminal , ..) θα ήταν μια άψογη λύση με εγγυημένο uptime , βέλτιστο memory footprint και λειτουργία μόνο χρησιμοποιώντας την μνήμη RAM 512 MB , κάτι το οποίο είναι σημαντικό , καθότι η Compact Flash memory card έχει συγκεκριμένη διάρκεια ζωής (κύκλων ανάγνωσης/εγγραφής).

Τα Windows Embedded υποτίθεται ότι παρέχουν ένα cut down windows framework το οποίο όμως κατα την περίοδο επιλογής δεν χρησιμοποιούνταν πουθενά , ήταν πλήρως undocmented και θα ήταν μεγάλο ρίσκο για την ολοκλήρωση της εφαρμογής

Τα Windows Vista θα ήταν η απόλυτη υπερφόρτωση του συστήματος (1GB+ Memory Requirements) , το δε Mindstorm NXT δεν είναι καν συμβατό με Windows Vista , για αυτό ποτέ δεν τέθηκαν σοβαρά υπόψην.

Τα Windows XP θα παρείχαν την πιο mainstream δυνατή λύση με εγγυημένη συμβατότητα αλλά ένα αρκετά μεγάλο overhead για όλα τα περιττά πράγματα τα οποία κάνουν και δεν χρειάζονται σε ένα deticated σύστημα.

Τέλος τα Windows 98SE θα παρείχαν πλήρη συμβατότητα με το παραπάνω hardware , πολύ μεγάλη ταχύτητα και μικρό overhead λόγω του non-NT Kernel τους , αλλά θα είχαν μεγάλο πρόβλημα ασφάλειας καθότι είναι discontinued και ιδιαίτερα ευάλωττα σε επιθέσεις denial of service , ενώ τείνουν στο να δυσλειτουργούν κάποιες φορές (με πιο διάσημη την Blue screen of death κατα την παρουσίαση τους από τον Bill Gates) . Για λόγους λοιπόν μείωσης του ενδεχόμενου ρίσκου τέθηκαν γρήγορα εκτός σκέψης.

Τελικά επιλέχθηκαν τα Windows XP SP2 τα οποία είναι δωρεάν λόγο του MSDNAA , και η επιλογή αυτή έγινε κυρίως λόγο της κάπως μεγαλύτερης πείρας μου με το λειτουργικό αυτό και τις διάφορες παραμέτρους που θα απαιτούνταν για την ανάπτυξη των προγραμμάτων έτσι ;νστε να μειωθεί ο χρόνος ανάπτυξης του project. Επιπλέον δεν ήταν δυνατό να βρεθεί μια βιβλιοθήκη αντίστοιχη του SpeechAPI της Microsoft κάτι το οποίο ήταν πολύ σημαντικό για την διαδραστικότητα της όλης εφαρμογής.

Η επιλογή βέβαια των Windows XP είχε και τα αρνητικά της , γιατί παρότι ήταν ρυθμισμένα να μην χρησιμοποιούν paging file κατέστρεψαν ανεπανόρθωτα την Compact Flash memory “σκληρό δίσκο” μετά από μια μέρα χρήσης με αποτέλεσμα την αγορά ενός κανονικού σκληρού δίσκου ο οποίος θα μπορούσε να γίνεται thrash από το λειτουργικό αλλά ανέβασε το συνολικό ενεργειακό κόστος κατα 6Watt ή 30% .

Programming Frameworks / Main Program :

Για την ανάπτυξη του προγραμματιστικού κομματιού της εργασίας χρησιμοποιήθηκαν τα Win32API , Direct X 8.1 , Direct Draw , Microsoft Speech API , NetGear Drivers , LifeCam Drivers , Fantom NXT Drivers σαν βασικό υπόβαθρο.

Εξωτερικά προγράμματα :
Speed Fan για την μέτρηση θερμοκρασίας και αλλαγής τάσεων ,
VLC για το Web Streaming ήχου
Ammar Server σαν Web Server/Remote Terminal της εφαρμογής

Εσωτερικά προγράμματα μέρη του κεντρικού εκτελέσιμου :
FDLib – Face Detection Library -
http://www.kyb.mpg.de/bs/people/kienzle/facedemo/facedemo.htm
Βασισμένη σε ένα paper των Wolf Kienzle, G¨okhan Bakır, Matthias Franz and Bernhard Sch¨olkopf του ινστιτούτου Max Planck .[1]

VideoDLL – Μια βιβλιοθήκη δυναμικής σύνδεσης που αναπτύχθηκε για τους σκοπούς του project με κύριο μέλημα την ταχύτητα και την παροχή αρκετών διαθέσιμων φίλτρων που μπορούν να εφαρμοστούν στις λαμβανόμενες εικόνες πριν αυτές ακόμα μεταφερθούν από την μνήμη στην οποία αποθηκεύονται. Επίσης η χρήση του DLL επέτρεψε τον εύκολο διπλασιασμό της εισαγωγής εικόνων αφού το πρόγραμμα δέχεται είσοδο από 2 κάμερες
(περίπου 1683 γραμμές κώδικα γραμμένου σε C++ compiled με το Visual Studio 2003)

VideoSubsystem **–** Μια βιβλιοθήκη δυναμικής σύνδεσης , που αναπτύχθηκε για τους σκοπούς του project και αναλαμβάνει την διενέργεια των οπτικών συσχετήσεων , αποθήκευση δεδομένων και εξαγωγή συμπερασμάτων . Λειτουργεί δεχόμενη ένα stream από διπλές εικόνες και έχει εξόδους τα επιμέρους χαρακτηριστικά τους και στο μέλλον ιδανικά απλή γεωμετρία , textures και detected faces (χρησιμοποιώντας την FDLib)
(το κομμάτι αυτό του project είναι ακόμα υπο κατασκευή και συνεχή βελτίωση , βρίσκεται περίπου στις 1099 γραμμές κώδικα στις 11/07/08 3:53 , είναι γραμμένο σε C++ , compiled με το Visual Studio 2003 και είναι ταχύτατο καθώς χρησιμοποιεί παντού pointer arithmetic και μεγάλα optimizations. Ενδεικτικό είναι ότι τα πρώτα φίλτρα που είχα φτιάξει και δοκίμαζα με τον κύριο Παπαϊωάννου είχαν latency περίπου 800 ms και ήταν γραμμένα σε Freepascal , χρησιμοποιώντας δισδιάστατα arrays και καλώντας procedures των Windows για την μετατροπή RGB σε bytes , ενώ η νέα υλοποίηση βγάζει τα ίδια αποτελέσματα σε χρόνους λιγότερους των 60 ms)

SAPISpeech **–** Μια βιβλιοθήκη δυναμικής σύνδεσης , που αναπτύχθηκε για τους σκοπούς του project και αναλαμβάνει την είσοδο φωνητικών εντολών (Speech to text) και την έξοδο φωνητικού feedback (Text to speech) , με ένα πολύ εύκολο interface και χωρίς να απαιτεί σχεδόν καμμία προγραμματιστική γνώση.
(557 γραμμές κώδικα , γραμμένου σε C++ , compiled με το Visual Studio 2003)

CPPNXT **-** Μια βιβλιοθήκη δυναμικής σύνδεσης , που αναπτύχθηκε για τους σκοπούς του project και αναλαμβάνει τον ρόλο της παρεγγεφαλίδας του ρομπότ και της μεταβίβασης και εκτέλεσης με ακρίβεια των κινήσεων από το Mindstorm καθώς και το Polling των αισθητήρων του.
Να σημειωθεί εδώ ότι η συγκεκριμένη βιβλιοθήκη χρησιμοποιεί μια μεικτή τεχνική επικοινωνίας με το NXT Block αφού τρέχει ένα κομμάτι της στο ίδιο το Mindstorm επιτρέποντας την χρήση του μικροεπεξεργαστή του για τοπικό polling των μοτέρ και κινήσεις πολυ μεγάλύτερης ακριβείας από αυτές που προσφέρουν άλλες αντίστοιχες βιβλιοθήκες (που βασίζονται στον USB ή Bluetooth δίαυλο και πολλά επίπεδα ελέγχου , drivers , μετάδοσης κτλ)
(1223 γραμμές κώδικα γραμμένου σε C++ compiled με το Visual Studio 2003)

AmmarGUI **–** Η γραφική διεπαφή γραμμένη σε Freepascal

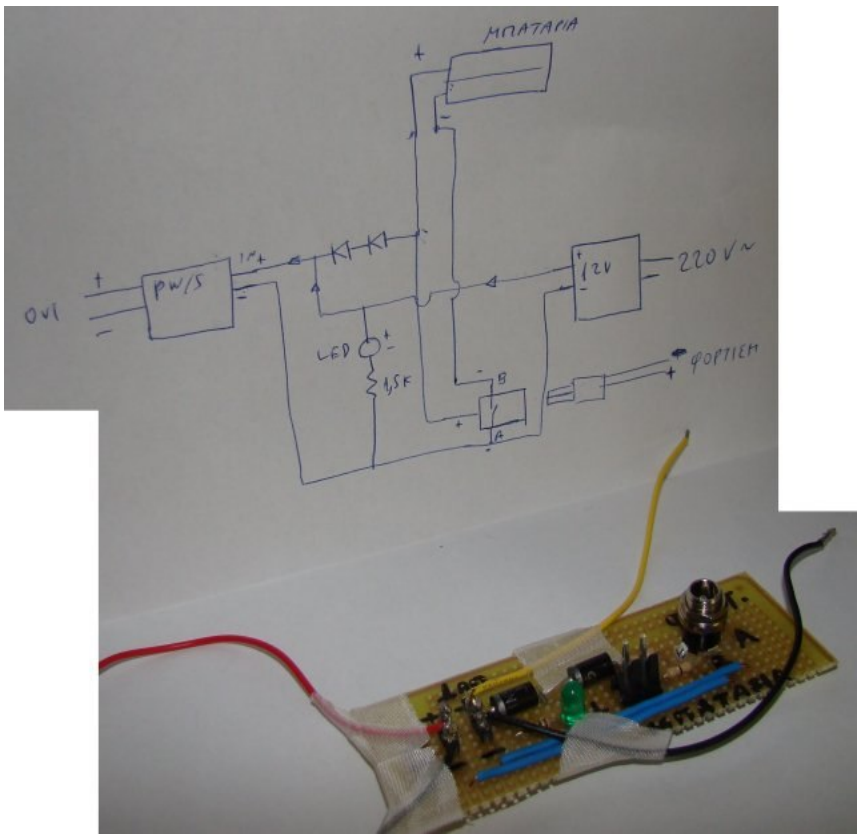
RoboVision Executable **–** Το κυρίως πρόγραμμα που αναλαμβάνει τον συνδυασμό των πάντων και δίνει ζωή στο robot
(3759 γραμμές κώδικα γραμμένου σε Pascal (Delphi) compiled με τον Freepascal Compiler σημείωση , σταδιακά αρκετά κομμάτια τα οποία , αρχικά για σκοπούς κυρίως δοκιμών βρίσκονταν στο executable , περνάνε στο VideoSubsystem όπου γίνονται optimized , οπότε σταδιακά το μέγεθος του main executable αναμένεται να αρχίσει να μικραίνει .
Η Freepascal επελέγη καθότι παρέχει ένα πολύ πιο συντηρίσιμο και ευανάγνωστο συντακτικό και που κατα την γνώμη μου επιβάλλεται στα Controlling processes. Τέλος ακόμα και η ανομοιομορφία της γλώσσας σε σχέση με τα DLLs σε καμμία περίπτωση δεν αυξάνει την πολυπλοκότητα , χάρη στην καλή κατανομή εργασιών σε αυτοτελή building blocks , ενώ επίσης δεν έχει αρνητικό αντίκτυπο στην ταχύτητα , αφού στα πιο κρίσιμα κομμάτια της εφαρμογής έχει χρησιμοποιηθεί η C++ με τον Visual Studio compiler.)

Ενεργειακά θέματα :
Η ενέργεια είναι ένα από τα μεγαλύτερα θέματα της καθημερινότητάς μας και είναι φανερό πλέον οτι οι υπάρχουσες μορφές αποθήκευσης ηλεκτρισμού είναι ανεπαρκείς , ακριβές και συνήθως επιβλαβείς για το περιβάλλον (αφού χρησιμοποιούν τοξικά στοιχεία όπως ο Μόλυβδος , Κάδμιο και άλλα..) και αποτελούν ανοιχτό πρόβλημα στον τομέα της φυσικής / χημείας. Παρότι ξεφεύγει από τον ρόλο ενός προγραμματιστή , ωστόσο ασχολήθηκα εκτενώς με τις δυνατές εναλλακτικές , εξαντλώντας την μεγάλη πλειοψηφία πηγών ενέργειας , ενώ δοκιμαστικά τοποθετήθηκε και ένα μικρό φωτοβολταϊκό σύστημα σαν "καπέλο" του ρομπότ και το οποίο όμως σε καμμία περίπτωση δεν θα μπορούσε να τροφοδοτήσει το τρέχον σύστημα. Με την αλματώδη μείωση της κατανάλωσης από τους επεξεργαστές Atom της Intel που ανέφερα και πριν θα ήταν δυνατή η τροφοδοσία του υπολογιστή αποκλειστικά από ένα τέτοιο σύστημα.
Μετά από μετρήσεις κατανάλωσης της Motherboard αυτή βρέθηκε περίπου στα 4.5A @ 12V .
Αναφορικά έγιναν δοκιμές με συστοιχία σε σειρά μπαταριών AA σε αριθμούς 12 και 10 τεμαχίων , τόσο επαναφορτιζόμενων (1.2V όσο και κανονικών 1.5V με χωρητικότητα 2500mAh) , με μπαταρία αυτοκινήτου 12V 45 mAh , με 2x μπαταρίες Lead Acid 12V 12Ah ενώ τελικά χρησιμοποιήθηκε ένα πακέτο 10 μπαταριών μεγέθους C και χωριτικότητας 4500mAh μαζί με έναν φορτιστή.

Κατά την λειτουργία σε μπρίζα το ρομπότ παίρνει ρεύμα από ένα ελαφρύ τροφοδοτικό mounted πάνω του που δίνει έξοδο 12V 12.5 A . Το υπάρχον καλώδιο δίνει αυτονομία κίνησης περίπου 10 μέτρων αλλα κάνει δύσκολη την περιδιάβαση του χώρου αποτελώντας ένα μη προβλέψιμο εμπόδιο το οποίο βρίσκεται εκτός του οπτικού πεδίου του ρομπότ.

Για το δε switching μεταξύ Power Supply και μπαταρίας χρησιμοποιείται το παρακάτω κύκλωμα το οποίο σχεδίασαν και κατασκεύασαν στην εταιρεία Telcom Hellas (http://www.telcomel.gr).

Το mindstorm από την άλλη τροφοδοτείται με 6 επαναφορτιζόμενες μπαταρίες , οι οποίες αρχικά σχεδιαζόταν να ενοποιηθούν με της μπαταρίες της motherboard , αλλά λόγω του παλμικού ελέγχου των step motors και του ρεύματος που απαιτούν αυτό θα έκανε ασταθή την τάση των μπαταριών και η ιδέα εγκαταλήφθηκε.



Υλοποίηση – User Input & Feedback :
Η είσοδος εντολών από τον χρήστη μπορεί να γίνει με 5 τρόπους.

1) Ο πρώτος και απλούστερος είναι συνδέοντας ένα computer monitor , ένα πληκτρολόγιο και ένα δείκτη οπού το robot μετατρέπεται σε ένα κανονικό desktop σύστημα.

2) Δεύτερος κατα σειρά απλότητας τρόπος είναι κάνοντας remote desktop σύνδεση στο μηχάνημα οπού και πάλι μετατρέπεται σε ενα κανονικό desktop σύστημα , χωρίς όμως την ανάγκη για σύνδεση του οποιοδήποτε παραπάνω περιφερειακού.

3) Για τους σκοπούς του project δημιουργήθηκε ένα Web front-end του συστήματος , το οποίο υλοποιείται , χρησιμοποιώντας τον AmmarServer παρέχει ένα live feed από τις κάμερες του ρομπότ , καταγράφει την δραστηριότητά του καθώς και τα αποθέματα μπαταρίας , πιθανά προβλήματα που διαγνώσθηκαν , snapshots εισβολέων ή παράξενων γεγονότων και με το κατάλληλο routing του port στο οποίο ακούει ο AmmarServer , από τον κεντρικό υπολογιστή της εγκατάστασης την οποία θα φρουρεί το ρομπότ θα είναι δυνατή η παρακολούθηση του από οποιοδήποτε σημείο της γής.

4) Επίσης έχει πλήρως υλοποιηθεί η είσοδος Speech To Text και αυτό έχει σαν αποτέλεσμα το ρομπότ να μπορεί να λάβει φωνητικές εντολές στα αγγλικά και να εκτελέσει μια σειρά ενεργειών , όπως το να απενεργοποιηθεί , να τεθεί σε κατάσταση επιφυλακής , να σημάνει συναγερμό , να κάνει ησυχία , να μετρήσει πρόσωπα , να ακολουθήσει κάποιο σετ εντολών και άλλα.

5) Τέλος αρχικά σχεδιαζόταν να είναι δυνατόν να μπορούν να δοθούν εντολές με καρτέλες κάποιας χρωματικής κωδικοποίησης, οι οποίες θα περιείχαν κάποια βασικά σχήματα. Η ύπαρξη της ηχητικής αναγνώρισης βέβαια καθιστά την οπτική συνεννόηση δευτερεύουσας σημασίας , χωρίς όμως αυτό να σημαίνει πως η ιδέα αυτή έχει εγκαταληφθεί.

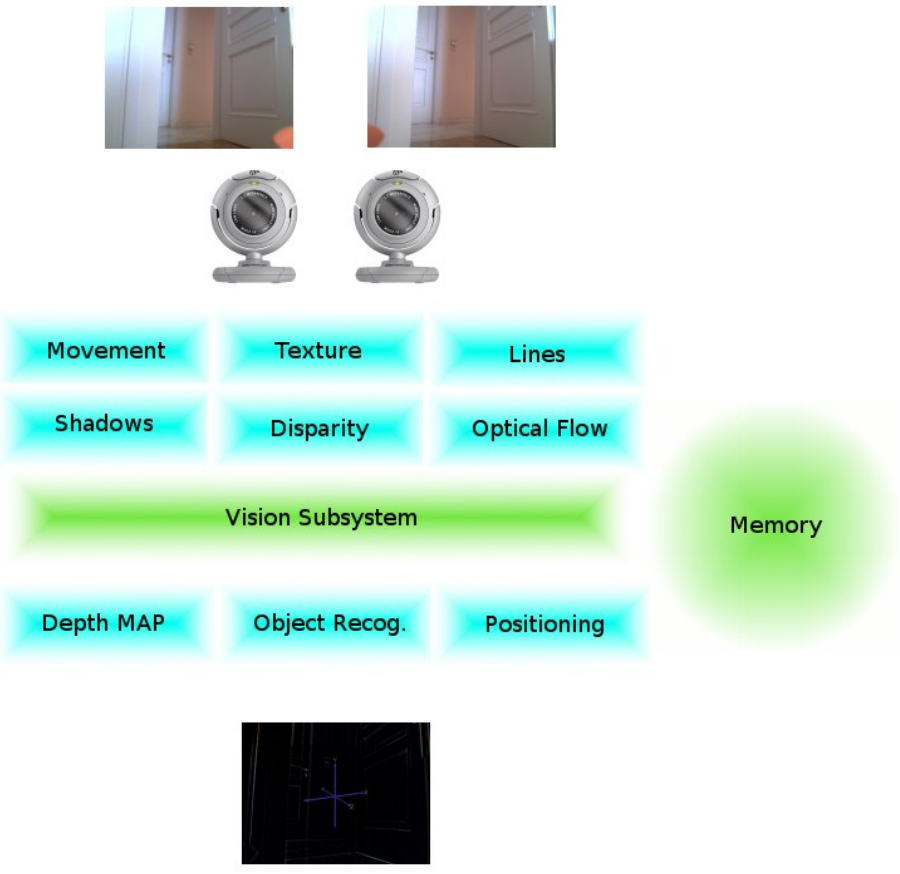
Υλοποίηση – Job Management :
Για την υλοποίηση των διαφόρων tasks δημιουργήθηκε μια scripting γλώσσα που γίνεται interpreted κατευθείαν από το RoboVision executable και περιλαμβάνει high level περιγραφές για εντολές μετακίνησης εκτέλεσης ελέγχου εικόνας , ηχητικής σήμανσης ακόμα και ξυπνητήρια που μπορεί να κάνουν το ρομπότ να ξεκινάει μια περιπολία μεταξύ 5:00 και 10:00 κάθε βράδυ και στην συνέχεια να επιστρέφει σε idle state.
Η γλώσσα αυτή προς το παρόν είναι αρκετά απλή ενώ στο μέλλον θα πρέπει να προστεθούν και aliases για της διάφορες κινήσεις κάτι το οποίο θα επιτρέπει ευκολότερο προγραμματισμό των δρομολογίων , ενώ ιδανικά το ίδιο το Guard Dog θα χαρτογραφούσε τα δωμάτια σε έναν χάρτη δίνοντας ονόματα σε κάθε πολυγονική περιοχή και επιτρέποντας ανθρωποειδή προγραμματισμό του τύπου :

default.trip .....
Ξύπνα στις 5:00
Σε περίπτωση παραβίασης του χώρου πλήρης συναγερμός
Κατά την λειτουργία σου να είσαι αθόρυβο
Επανέλαβε
Έρευνα στα δωμάτια 3,4,5
Μέχρι η ώρα να είναι 10:00
Επέστρεψε στην αρχική θέση
Επέστρεψε σε κατάσταση αναμονής
.....

Προς το παρόν το κομμάτι έρευνα στα δωμάτια είναι του τύπου
Στρίψε δεξιά 25 μοίρες
Προχώρα μπροστά 10 εκατοστά κτλ..

Υλοποίηση – Σύστημα συναγερμού
Το σύστημα συναγερμού αποτελείται από 2 κομμάτια , το πρώτο είναι μια δικλείδα ασφαλείας που επιτρέπει την διαπίστωση κάποιου καταστροφικού συμβάντος (πχ το ρομπότ σταμάτησε να εκπέμπει πριν 5 δευτερόλεπτα άρα πιθανόν έχει βγεί εκτός εμβέλειας , να λαμβάνει παράσιτα στο κανάλι 802.11 οπότε κάποιος να επιχειρεί εισβολή , μπορεί να έμεινε χωρίς μπαταρία ή ακόμα και να καταστράφηκε από κάποιον εισβολέα.) και σε κάθε περίπτωση σημαίνει γενικό συναγερμό προωθώντας το μήνυμα με e-mail , sms , ακόμα και ενεργοποιώντας μια πιθανή σειρήνα που μπορεί να υπάρχει στην εγκατάσταση! Το δεύτερο τμήμα του μηχανισμού ασφαλείας τρέχει μέσα στο Guard Dog και τροφοδοτείται από τις αισθήσεις του robot. Οταν τεθεί σε κατάσταση συναγερμού σημαίνει έναν πολύ δυνατό ήχο από τα μεγάφωνα του ρομπότ και εκπέμπει σήμα κινδύνου στον κεντρικό υπολογιστή , ενώ παράλληλα κάνει log τοπικά το event που προκάλεσε τον συναγερμό.

Υλοποίηση – Οπτικές πληροφορίες :
Αυτό είναι το σημαντικότερο και πιο ενδιαφέρον κομμάτι του Project. Η διαστρωμάτωση του μηχανισμού εξαγωγής οπτικής πληροφορίας είναι η παρακάτω.



Για την επίτευξη του Computer Vision κομματιού του Project χρησιμοποιούνται οι αλγόριθμοι Sobel , Gaussian Blur ενώ σε ορισμένες περιπτώσεις Monochrome conversions και Palette Reducing , ενώ έχει κατασκευαστεί ένας πρωτότυπος αλγόριθμος για Twin View depth map ο οποίος εξηγείται παρακάτω. Η ανίχνευση της κίνησης κατά την κίνηση του ιδίου του ρομπότ θα γίνεται με την χρήση 2 μεθόδων. Αφενός χρησιμοποιώντας Optical Flow detection και αφετέρου με ένα υποσύστημα το οποίο αναπτύσσω τώρα και αντιστοιχίζει κατευθείαν πληροφορίες οι οποίες αλλάζουν στο ένα μάτι με αντίστοιχες πληροφορίες που αλλάζουν στο δεύτερο προσφέροντας ταχύτατη ανανέωση του depth map χωρίς την σάρωση ολόκληρης της εικόνας.

Επίσης με μια απλή αφαίρεση της μιας εικόνας από την άλλη είναι δυνατή η εξαγωγή συμπερασμάτων για αντικείμενα , τα οποία αποτελούν backdrop και βρίσκονται πολύ μακρυά , ενώ κατά πάσα πιθανότητα η αφαίρεση αυτή θα είναι και ένας πολυ καλός δείκτης για την εντόπιση occlusion μεταξύ των δυο εικόνων από ένα αντικείμενο πάρα πολυ κοντά σε μια από τις οπτικές πηγές.

Για την διαδικασία του depth mapping απαιτούνται 2 ικανότητες.. Η πρώτη είναι η δυνατότητα σύγκρισης ενος κομματιού εικόνας με κάποιο άλλο και δεύτερον η δυνατότητα αποθήκευσης ολων των αποτελεσμάτων έτσι ώστε να επιλεγεί το καλύτερο δυνατό.
Η υλοποίηση μου χρησιμοποιεί έναν Linear Classifier οπού απλώς όποια κομμάτια μιας scanline είναι πιο όμοια επιλέγονται πάνω από τα ήδη υπάρχοντα και για ευριστική την συνάρτηση που περιγράφεται παρακάτω.

Πρώτα από όλα στην μνήμη του υπολογιστή διατηρείται μια εικόνα του τρέχοντος βάθους του κάθε pixel καθώς και με ποιο αντιστοιχεί από την μια κάμερα στην άλλη , το score του και το region στο οποίο κατατάχθηκε.

Για να βαθμολογηθούν τα διάφορα image patches γίνεται μια απευθείας αφαίρεση των RGB intesities των δυο pixel και ο αριθμός που βγαίνει πολλαπλασιάζεται επι 4 έτσι ώστε μεγάλες διαφορές να αυξάνουν γεωμετρικά το score. Σε περίπτωση που και τα 2 pixels αντιστοιχούν σε sobel edges το score υποδιπλασιάζεται ενώ εάν μόνο το ένα αντιστοιχιστεί (άρα παμε να αντιστοιχίσουμε sobel edge σε κάποιο flat σημείο εικόνας το score πολλαπλασιάζεται επι 2). Με την παραπάνω τακτική έχουμε πολυ καλά δοκιμαστικά αποτελέσματα με σχετικά μικρό υπολογιστικό κόστος , ενώ η σύγκριση περιλαμβάνει όχι μόνο παραγώγους (όπως οι κλασσικοί αλγόριθμοι του είδους) , αλλά και σύγκριση με βάση πληροφορίες χρώματος που είναι επίσης πολυ σημαντικές , όπως και στο ανθρώπινο μάτι!

Επιπλέον ο παραπάνω αλγόριθμος ακόμα και αν οι 2 κάμερες παράγουν διαφορετική εικόνα (δλδ ακόμα και αν η μια κάμερα έχει RGB range 1 – 170 ενώ ή άλλη 100 – 255) λόγω της γενικότερης διαφοράς αυτής είναι εξίσου αποδοτικός.

Να σημειωθεί τέλος ότι όλα τα παραπάνω έχουν υλοποιηθεί με όσο το δυνατόν καλύτερο τρόπο , precalculation arrays , optimizations για μεγαλύτερη ταχύτητα (τα οποία βέβαια κάνουν τον κώδικα λίγο πιο δυσανάγνωστο) , pointer arithmetic και CPU specific compilations για ένα οσο το δνατόν ταχύτερο τελικό εκτελέσιμο.

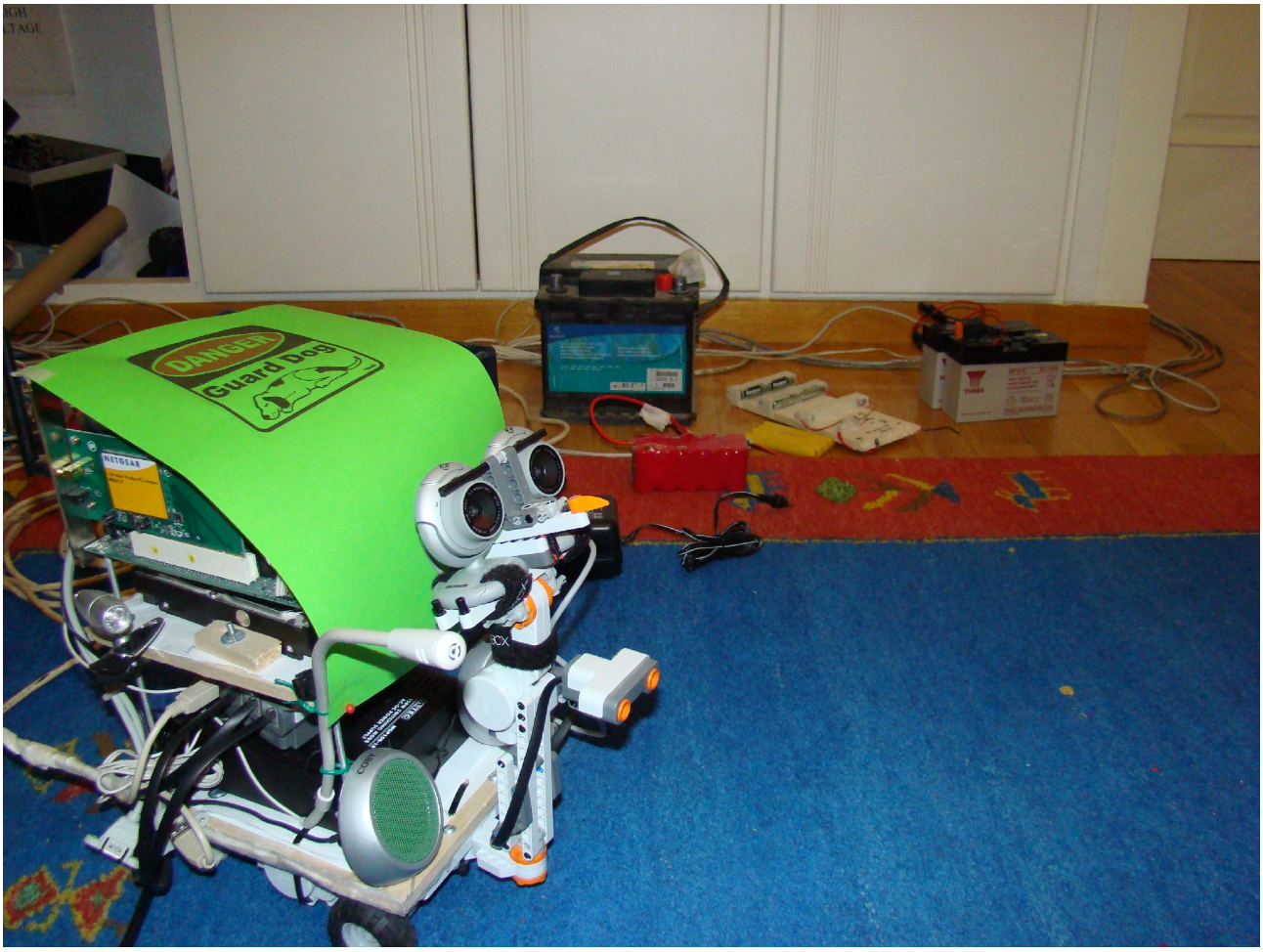
Τα υπόλοιπα κομμάτια του οπτικού συστήματος είναι πλήρως ρευστά οπότε θα περιγραφούν οταν ολοκληρωθούν.

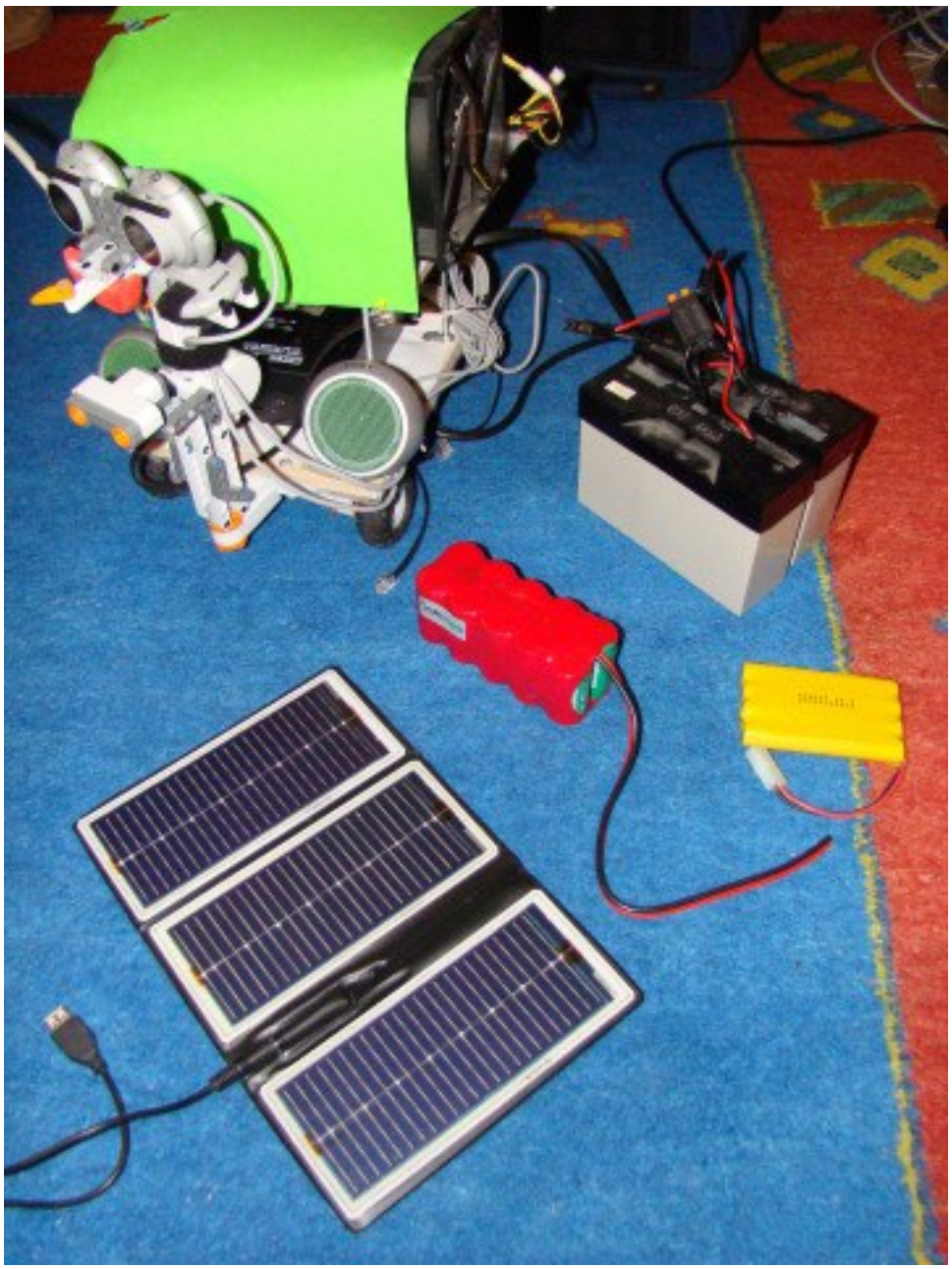
Φωτογραφικό Υλικό

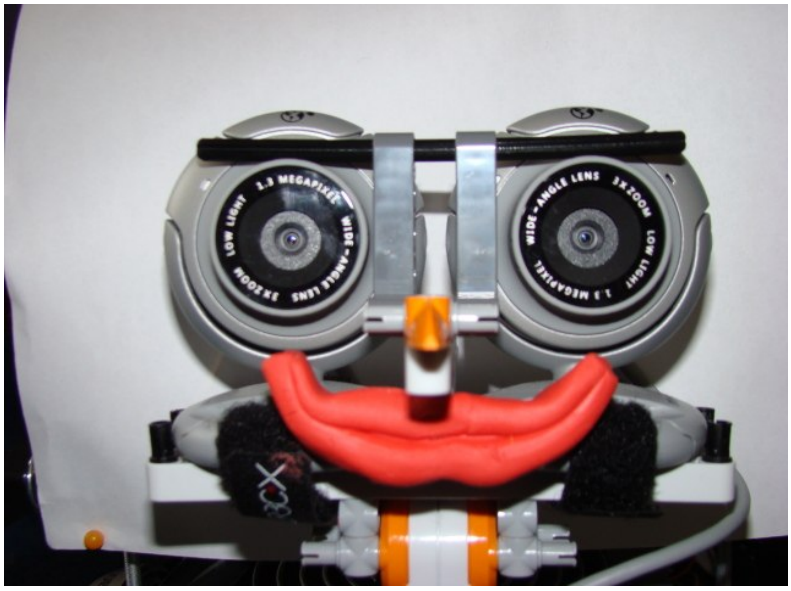
Στο youtube έχω προσθέσει και κάποια video από τις αρχικές φάσεις συναρμολόγησης του robot , είναι (με σειρά προσθήκης)

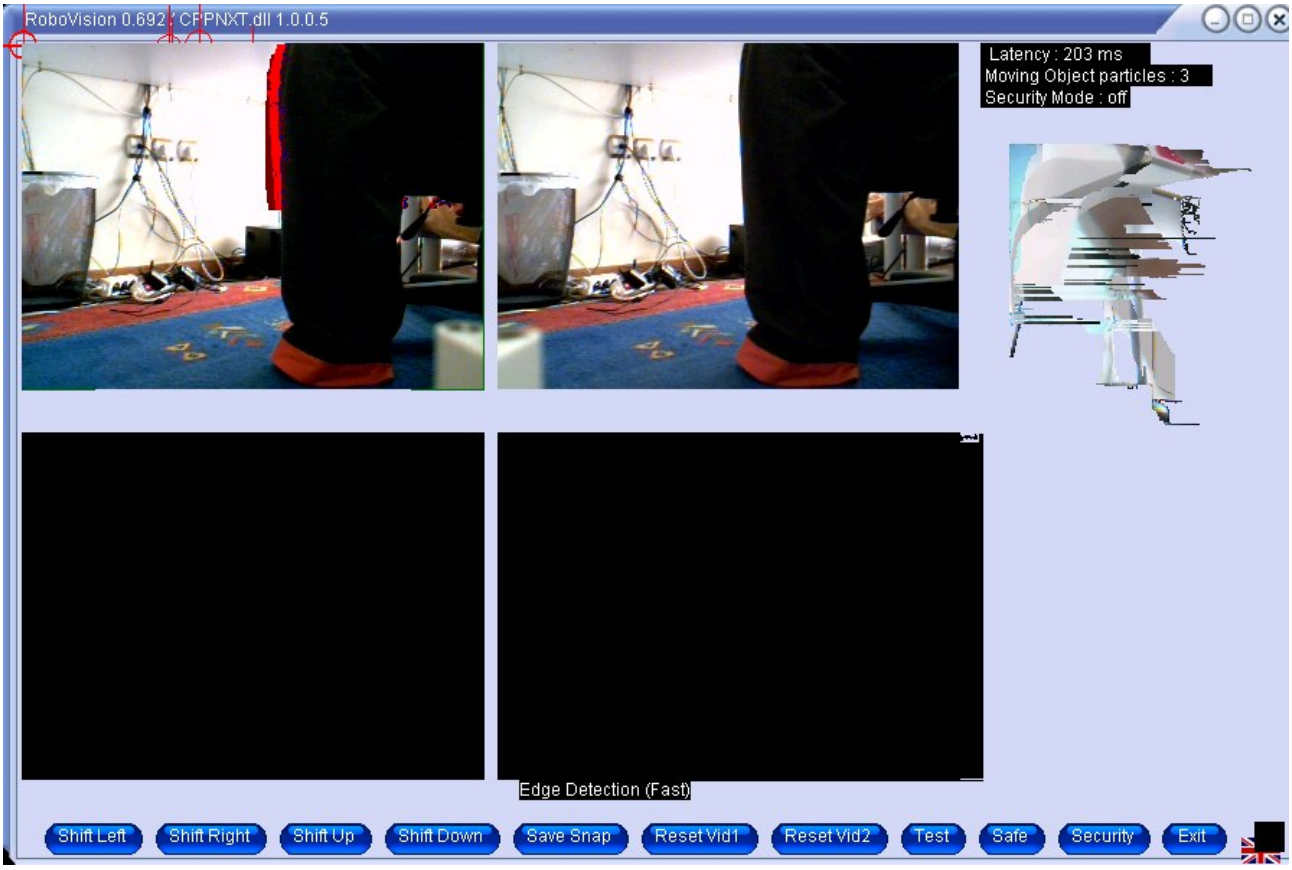
http://uk.youtube.com/watch?v=htE8DwA6NqU
http://uk.youtube.com/watch?v=SP2uFDsQfVY
http://uk.youtube.com/watch?v=t1-_TRhLFYw
http://uk.youtube.com/watch?v=Ja3Yieg-IS0
http://uk.youtube.com/watch?v=NS3-0cBFBCE

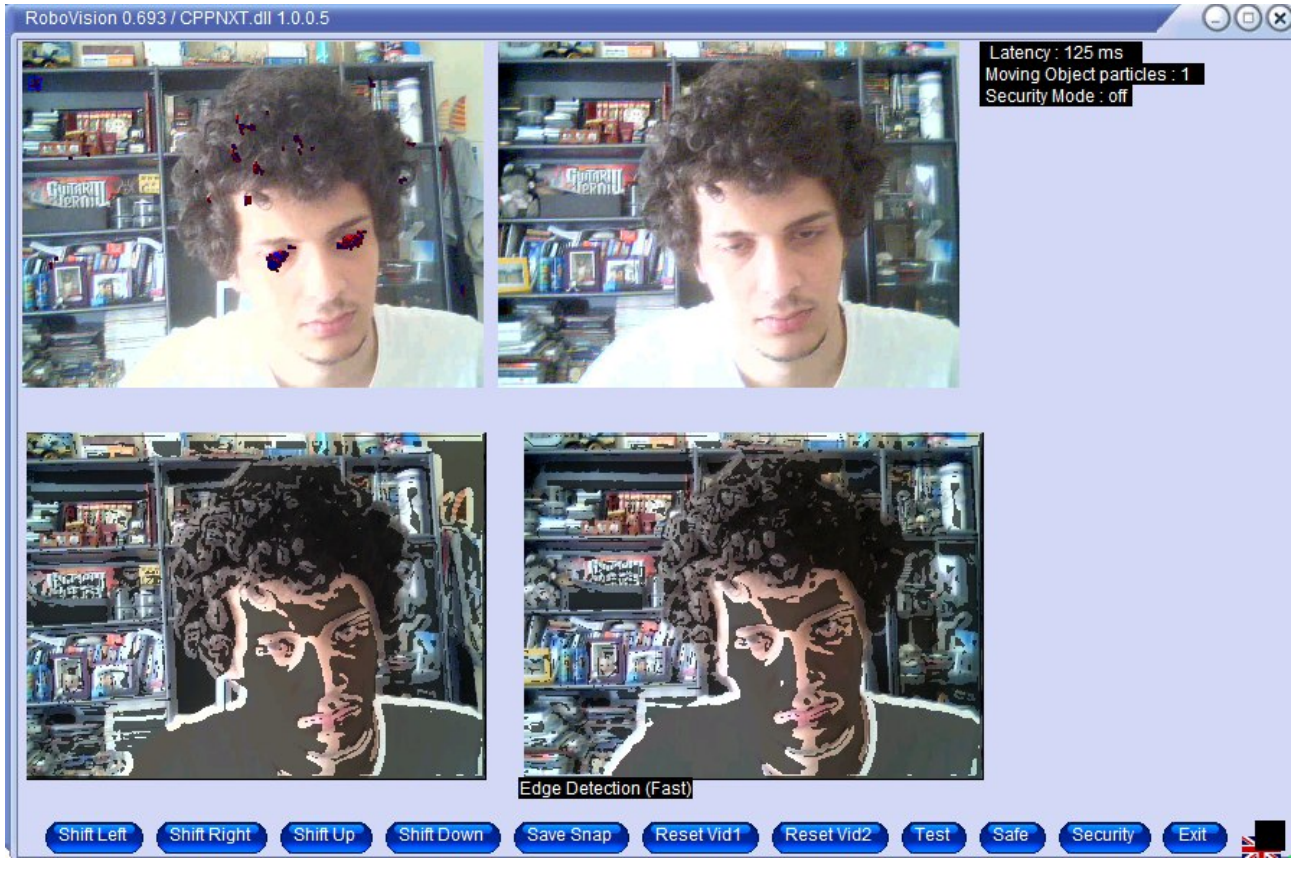
Ακολουθούν φωτογραφίες!


DANGER
Guard Dog
NETGEAR

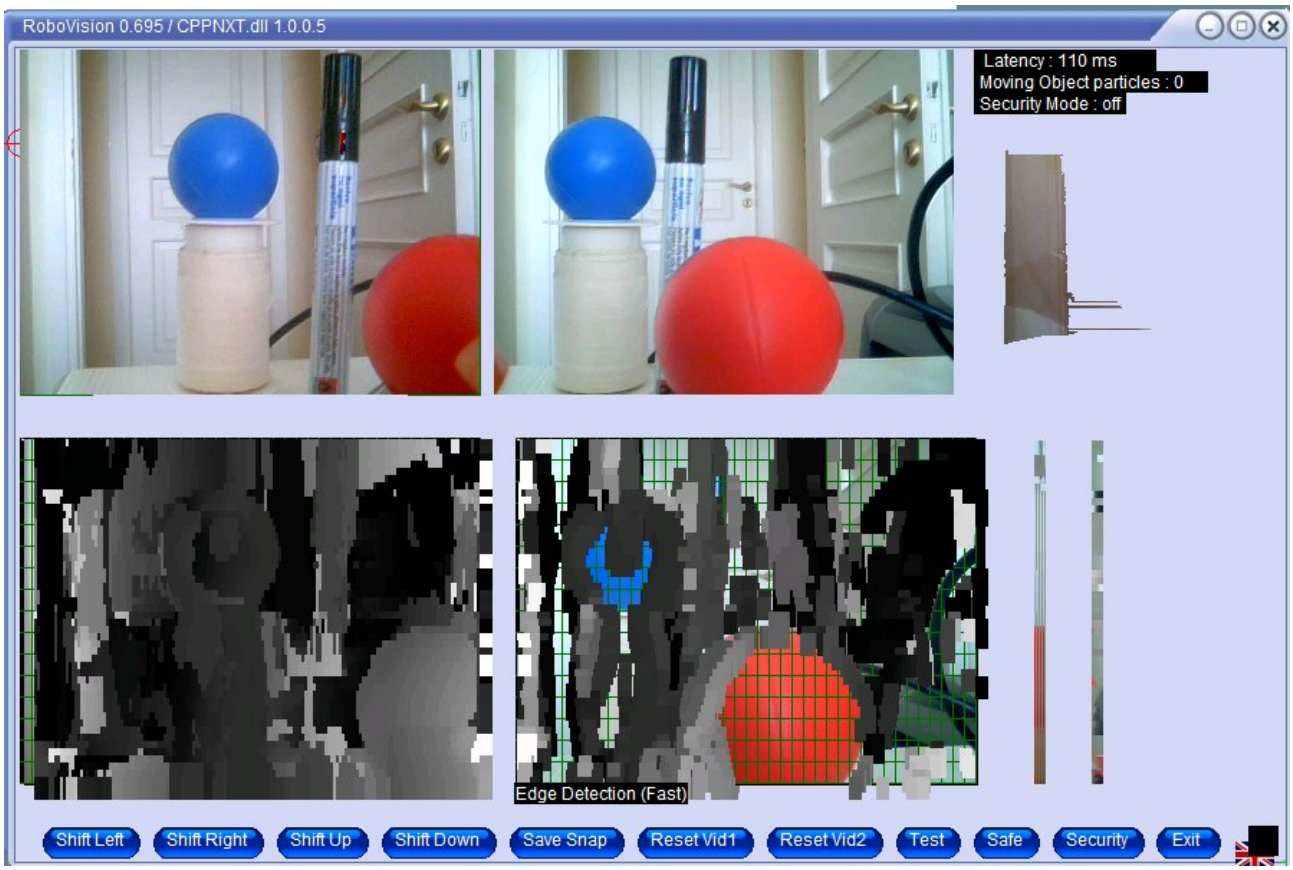



3.3 MEGAPIXEL
WIDE-ANGLE LENS
3X ZOOM
LOW LIGHT
WIDE-ANGLE LENS
3X ZOOM
LOW LIGHT
1.3 MEGAPIXEL


RoboVision 0.692 / CPPNXT.dll 1.0.0.5
Latency : 203 ms
Moving Object particles : 3
Security Mode : off
Edge Detection (Fast)
Shift Left
Shift Right
Shift Up
Shift Down
Save Snap
Reset Vid1
Reset Vid2
Test
Safe
Security
Exit


RoboVision 0.693 / CPPNXT.dll 1.0.0.5
Latency : 125 ms
Moving Object particles : 1
Security Mode : off
Edge Detection (Fast)
Shift Left
Shift Right
Shift Up
Shift Down
Save Snap
Reset Vid1
Reset Vid2
Test
Safe
Security
Exit


RoboVision 0.694 / CPPNXT.dll 1.0.0.5
Latency : 203 ms
Moving Object particles : 2
Security Mode : off
Edge Detection (Fast)
Shift Left
Shift Right
Shift Up
Shift Down
Save Snap
Reset Vid1
Reset Vid2
Test
Safe
Security
Exit


RoboVision 0.695 / CPPNXT.dll 1.0.0.5
Latency : 110 ms
Moving Object particles : 0
Security Mode : off
Edge Detection (Fast)
Shift Left
Shift Right
Shift Up
Shift Down
Save Snap
Reset Vid1
Reset Vid2
Test
Safe
Security
Exit