ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ

Guard Dog Project

Ιστορική Αναδρομή

- Ο άνθρωπος πάντα λειτουργούσε με εργαλεία για να λύνει προβλήματα
- Βιομηχανική επανάσταση πολλαπλασιασμός μυικής δύναμης (αυτοκίνητο)
- Πληροφορική πολλαπλασιασμός πνευματικής δύναμης (google , wikipedia , etc :P)
- Ρομποτική = Merging των δύο παραπάνω

Τι είναι ένα ρομπότ ?

Ένα ρομπότ είναι..

Ένας ηλεκτρονικός υπολογιστής όπου αντί για Mouse / Πληκτρολόγιο / Οθόνες έχουμε Ρόδες , Αισθητήρες Υπερήχων , Ηχεία , Μικρόφωνα , Κάμερες κτλ.

Ένας υπολογιστής που να επεξεργάζεται τα παραπάνω και να “επικοινωνεί” με το περιβάλλον

Μια μηχανή turing με ρόδες..

Ένα ρομπότ δεν είναι..

- Κάτι εξωπραγματικό
- Πολύ δύσκολο στην κατασκευή

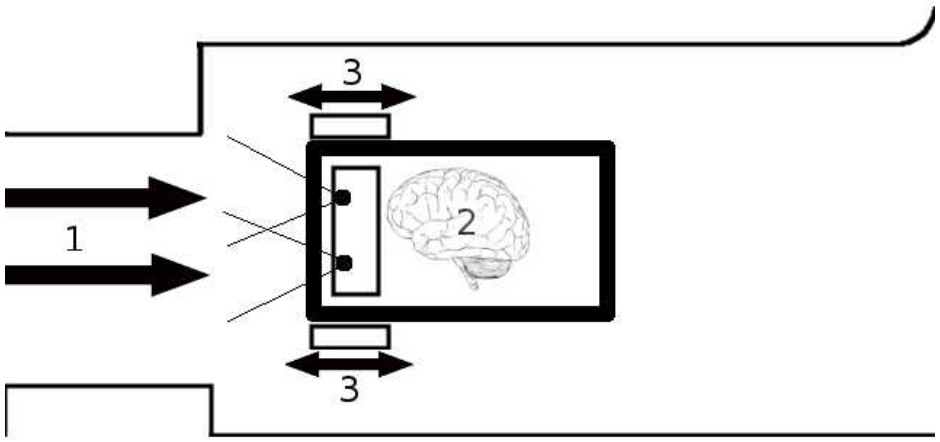
  Οι καφετιέρες , τα πλυντήρια , το αυτόματο πότισμα , όλα είναι ρομπότ υπό μία έννοια..

- Το δυσκολότερο προβλήμα είναι να φτιάξει κάποιος κάτι το οποίο να μην έχει απλά και μόνο αντανακλαστική συμπεριφορά..

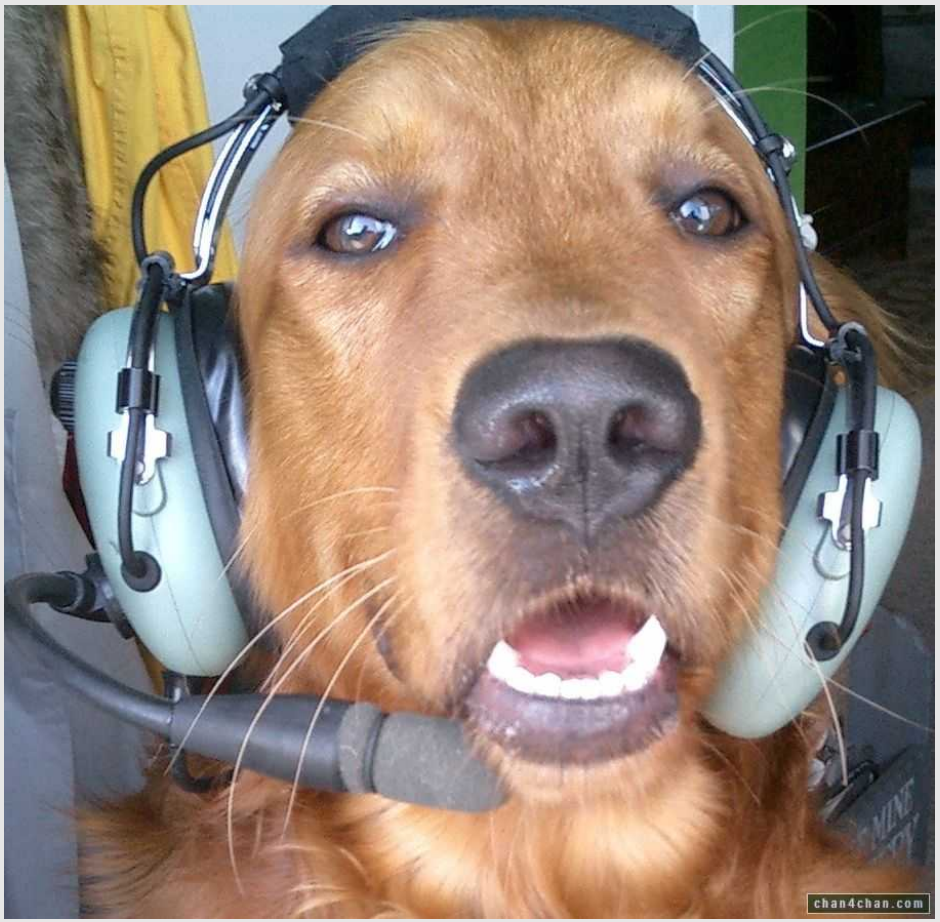
  Τηλεκατευθυνόμενο != Robot

GuarddoG Project
Ο Στόχος

- Δημιουργία ενός φύλακα χώρων ο οποίος χρησιμοποιώντας στεροσκοπική όραση να μπορεί να περιπολεί σε μια γνωστή διαδρομή , και σε περίπτωση που ανιχνεύσει εισβολή να καταδιώκει τον εισβολέα και να ειδοποιεί τον ιδιοκτήτη του.



Προφανώς όχι πραγματικό Guard “Dog”



- Στην φύση πήρε 5.000.000.000 χρόνια για να “φτιάξει” σκύλους

  Δυστυχώς δεν έχω τόσο χρόνο :P

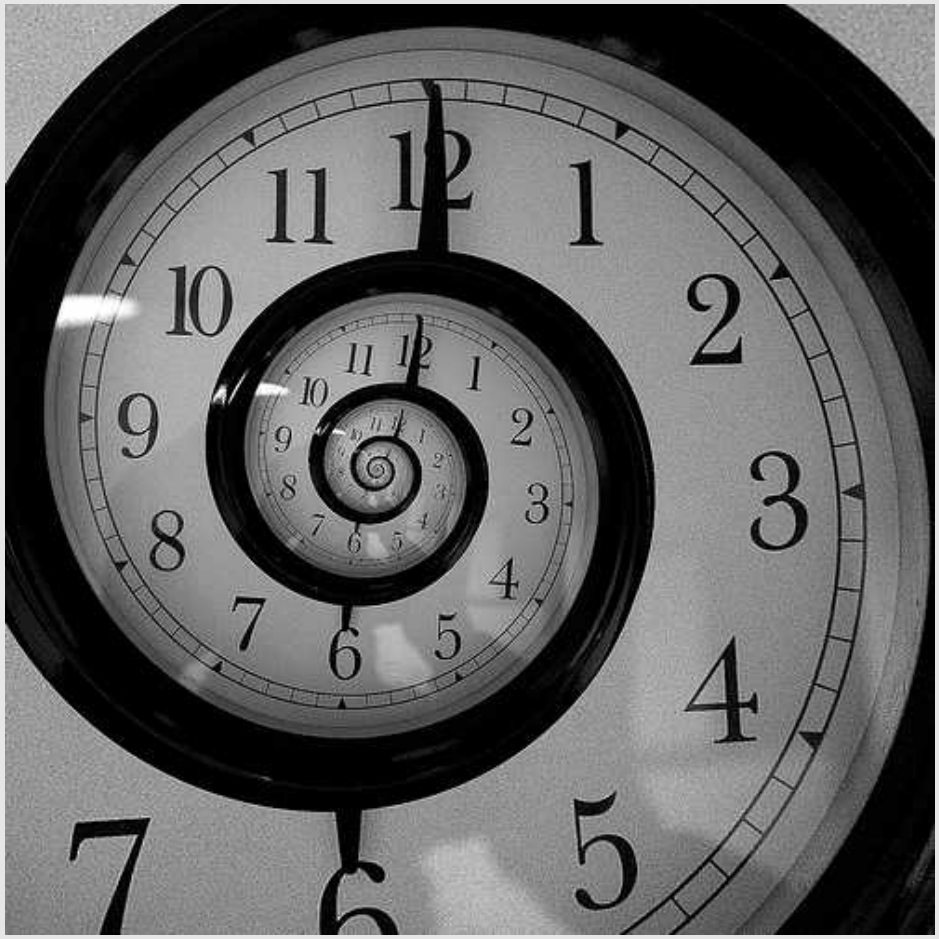
- Λειτουργία φύλακα και διαστάσεις σκύλου..

  Όχι πραγματικό σκυλί..

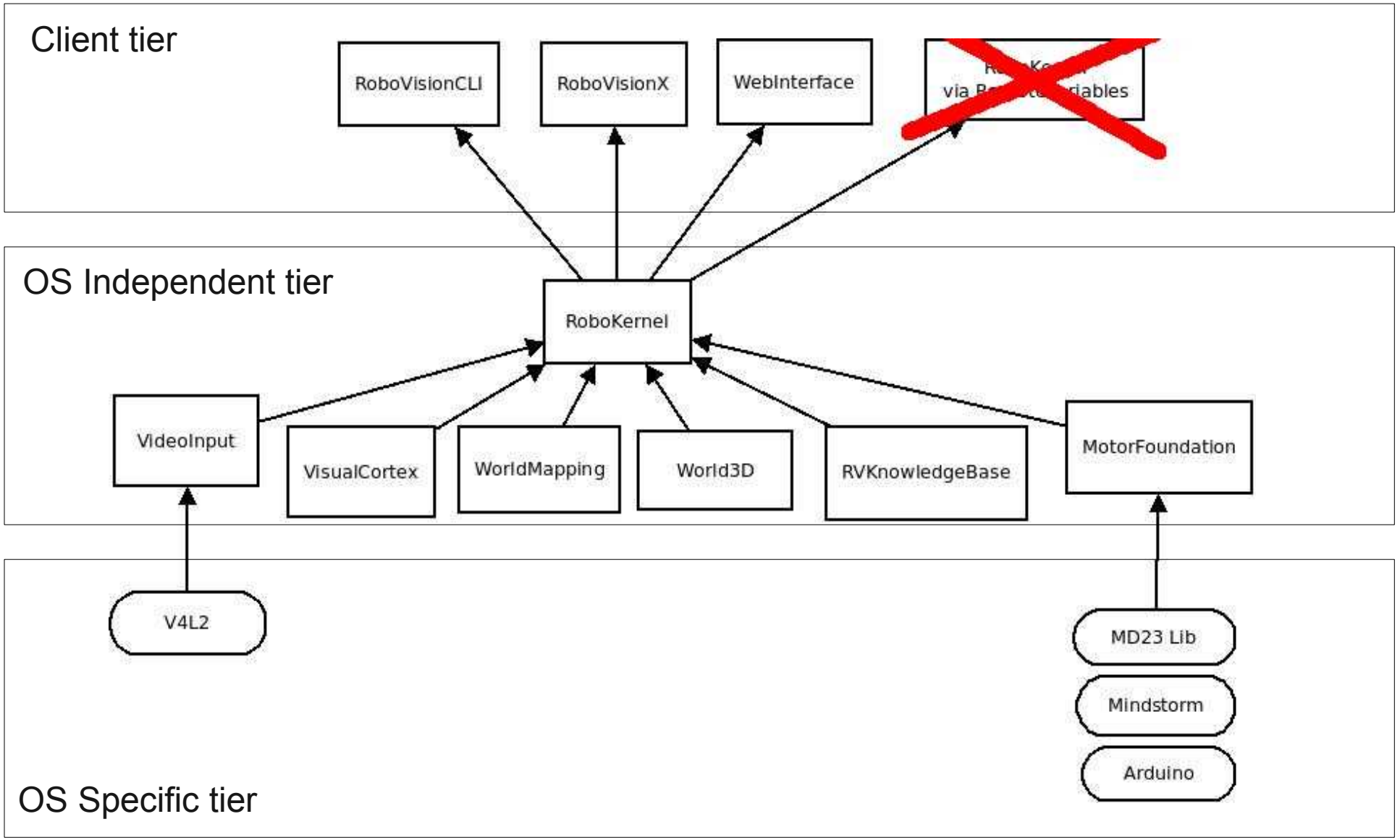
Μεγάλο Project
=
Μεγάλη παρουσίαση

- 2 κομμάτια software / hardware ..
- Βασικοί αλγόριθμοι
- Διαστρωμάτωση
- Ερωτήσεις/συζήτηση, guarddog github repo

  φτιάξτε το δικό σας ! :)



Αρχιτεκτονική του Software



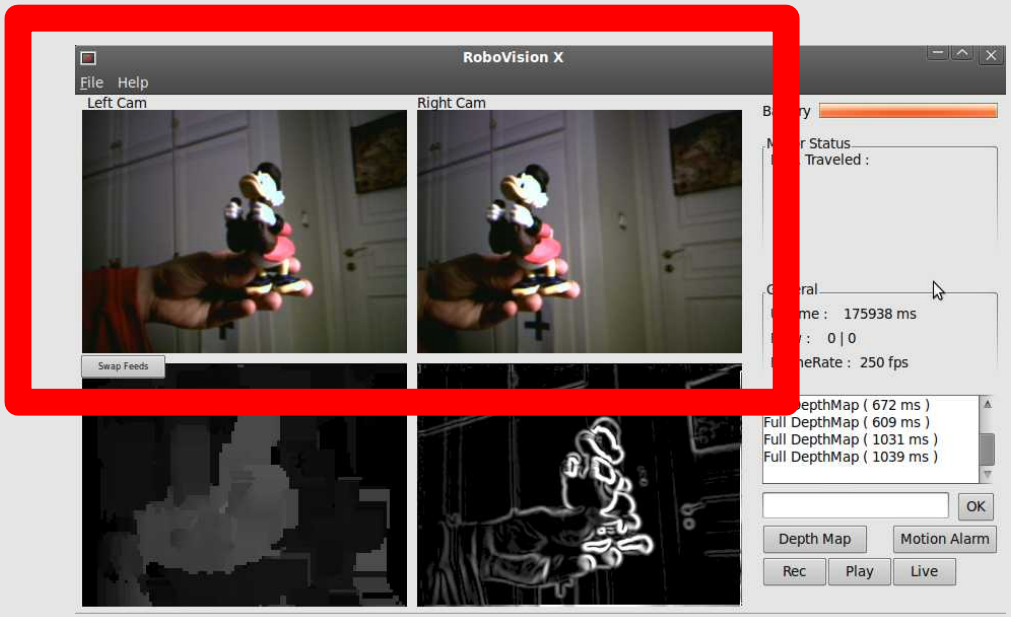
Το κάθε module έχει σαφώς καθορισμένη λειτουργία

Το κάθε ένα από τα modules εκτελεί μια σχετικά απλή λειτουργία , όλα μαζί κάνουν όμως κάτι πολύ πιο περίπλοκο

Video Input
Visual Cortex
World Mapping
RVKnowledgebase
Motor Foundation

Όλο το project είναι γραμμένο σε C (το GUI σε C++)και είναι statically linked για λόγους απόδοσης ..

Video Input

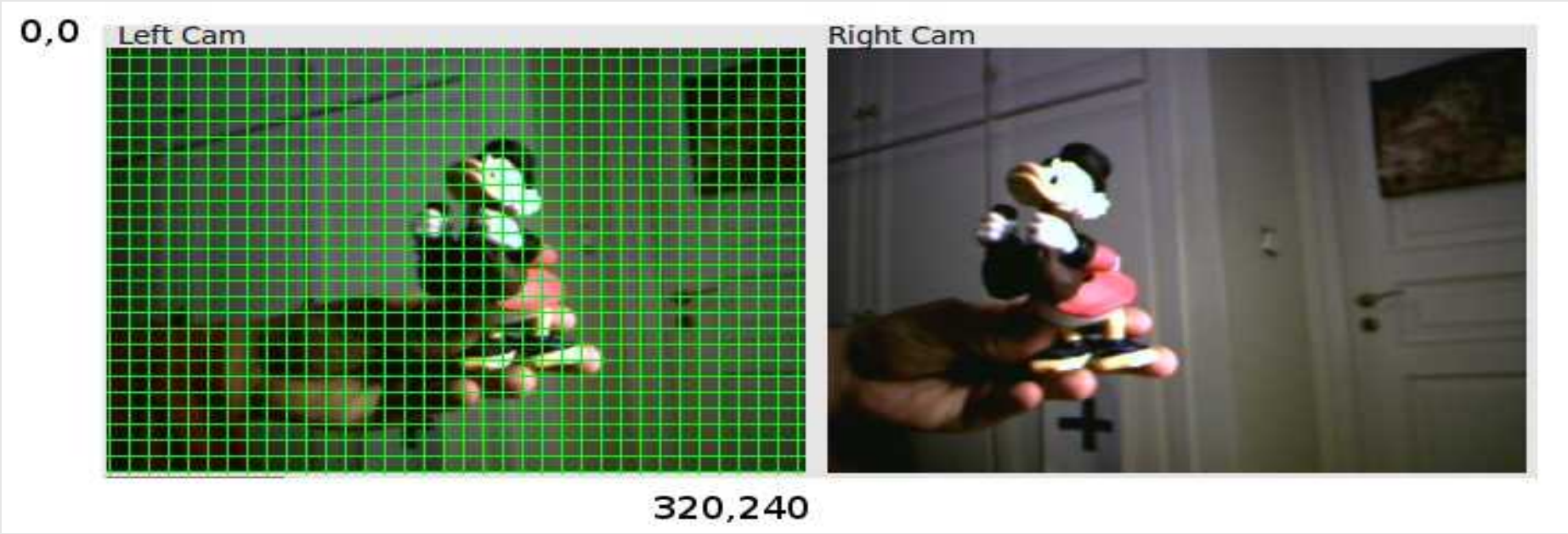


Αναλαμβάνει να μεταφέρει arrays με την εικόνα που βλέπουν οι 2 webcams

Σαν βιβλιοθήκη μπορεί κάποιος να το χρησιμοποιήσει για οποιοδήποτε project

Video Input

Πως αναπαριστάται μια εικόνα?



Χρήση ως εξής :

InitVideoInputs(2);
InitVideoFeed(“dev/video0”,320,240,3,1,0);
InitVideoFeed(“dev/video1”,320,240,3,1,0);

unsigned char * GetFrame(int webcam_id);

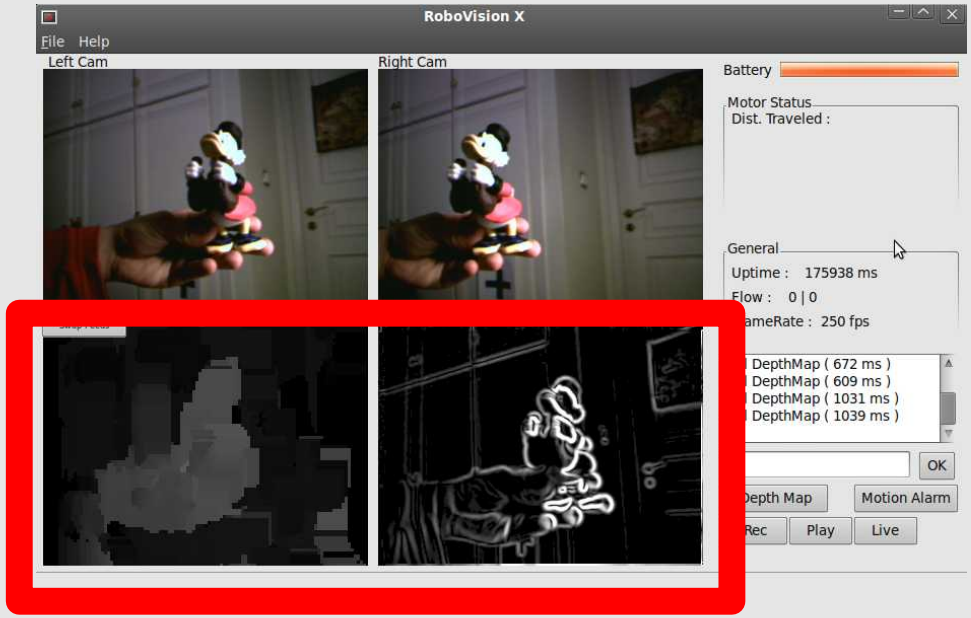
Video Input

Δουλειά του Video Input δεν είναι επεξεργασία εικόνας , απλά εξαγωγή και μεταφορά της..
Για να βρούμε το χρώμα του pixel X,Y κοιτάμε στο picture array ως εξής

picture = GetFrame(0);

picture[Y*3*320 + X*3] <- Red (0-255)
picture[Y*3*320 + X*3+1] <- Green (0-255)
picture[Y*3*320 + X*3+2] <- Blue (0-255)

Visual Cortex



- Ουσιαστικά “δέχεται” pointers από frames

  zero-copy (1 copy βασικά)
- Έχει ένα pipelining φίλτρων που τους εφαρμόζει για να μην υπάρχουν περιττές επαναλήψεις διαδικασιών
- Εξάγει frames τα οποία είναι μετασχηματισμός των frame εισόδου..

Visual Cortex

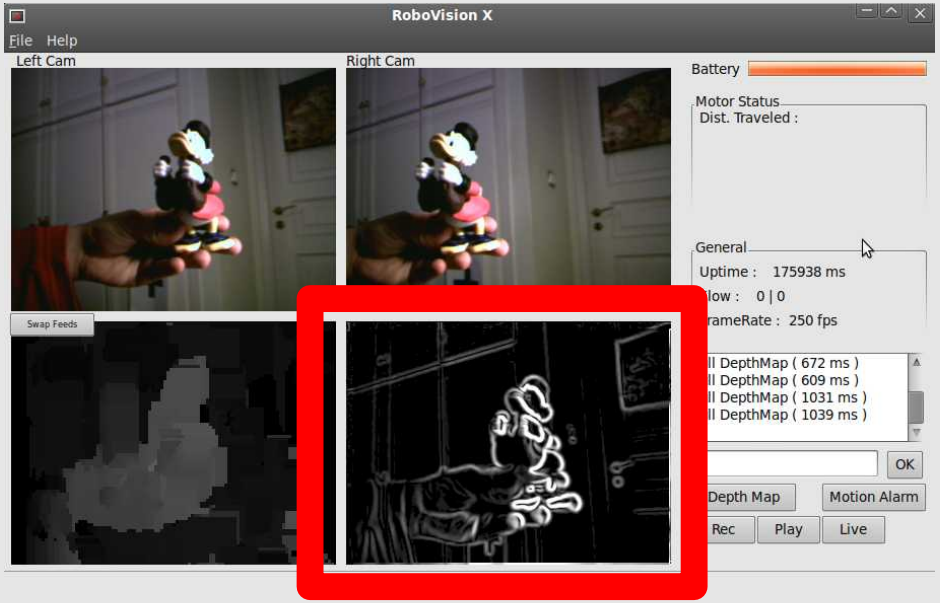
Ένα σύστημα με video registers και φίλτρα

Sobel Edge Detection
Gaussian Blurring Images
Image/Patch Histograms
Image/Patch Comparison
Tracking Changes between frames
Palette reductions
Disparity Mapping

SIFT / SURF
Face Detection
Object Detection (στο μέλλον)

Μπορείτε να ψάξετε το κάθε buzz-word στο internet τα τελευταία δύο είναι implemented μέσω άλλων βιβλιοθηκών

Visual Cortex




```
BOOLEAN  Sobel(unsigned char * image,int image_x,int image_y)
{

  unsigned int x=0,y=0;
  unsigned int x1=1,y1=1,x2=image_x,y2=image_y;


  if (image==0) { return(0); }

  unsigned char *proc_image;
  //proc_image = new unsigned char [ image_x * image_y * 3 ];
  proc_image = ( unsigned char * ) malloc ( sizeof(unsigned char) * image_x * image_y * 3 );

  BYTE *px;

  BYTE *r;
  BYTE *g;
  BYTE *b;

 BYTE p1=0,p2=0,p3=0,p4=0,p5=0,p6=0,p7=0,p8=0,p9=0;

 .....
```

Παράδειγμα implemented φίλτρου :

Sobel Edge Detection

Αναγνώριση ακμών , υπολογίζοντας την παράγωγο αλλαγής χρώματος..

Με απλά λόγια : εκεί που αλλάζει έντονα το χρώμα επιστροφή άσπρο , αν δεν αλλάζει καθόλου μαύρο ενδιάμεσες αλλαγές γκρί κτλ..

Αντίστοιχα φίλτρα blur κτλ είναι πολύ απλά..

Visual Cortex

Η κυρίως δουλειά του είναι να παίρνει μια ροή από 2 εικόνες
(αριστερή/δεξιά κάμερα) και να εξάγει depth maps

Έχουμε έτοιμους τους 2 pointers των εικόνων
video_register[LEFT_EYE] , video_register[RIGHT_EYE]
(που δείχνουν στην μνήμη του video input)

- Δημιουργία ιστογράμματος για κάθε τμήμα των 2 εικόνων και αποθήκευση του στους καταχωρητές l_video_register[HISTOGRAM_COMPRESSED_LEFT] ,l_video_register[HISTOGRAM_COMPRESSED_RIGHT]

- Gaussian blur για κάθε εικόνα , ώστε να εξομαλυνθεί ο θόρυβος και αποθήκευση του στον καταχωρητή video_register[EDGES_LEFT] , video_register[EDGES_RIGHT]

- Sobel Edge detection με είσοδο τους καταχωρητές video_register[EDGES_LEFT] video_register[EDGES_RIGHT] και αποθήκευση τους στον ίδιο χώρο

Visual Cortex

Disparity Mapping - VisualCortex/DisparityDepthMap.c
Βασική ιδέα , οι κάμερες κοιτάζουν παράλληλα αρα στον άξονα Y (ύψος) έχουμε ακριβώς ίδια σημεία , στον άξονα X όσο μεγαλύτερη η απόσταση , τόσο πιο κοντά

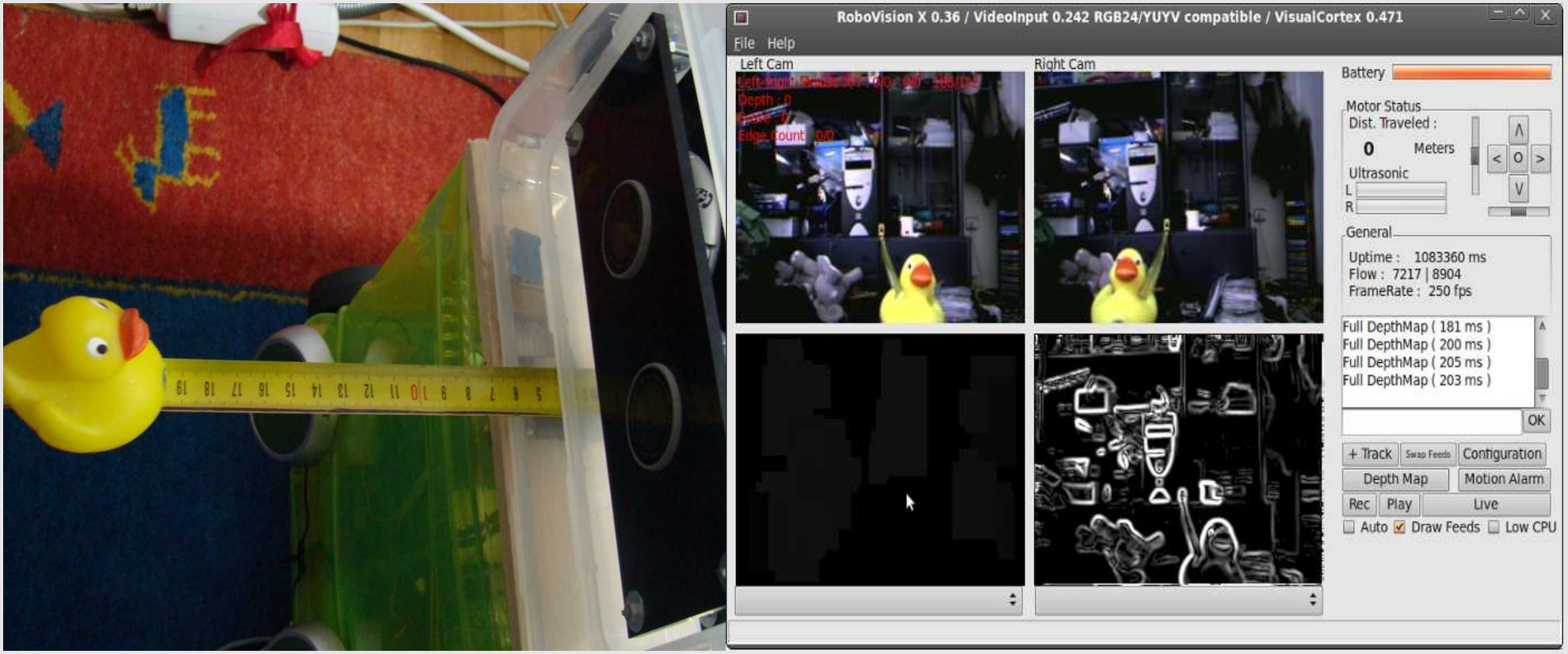
Συγκρίνουμε Patches , μετράμε τις αποστάσεις

Γενικά για μέγεθος Patch 30x50 πχ έχουμε
Για κάθε x από 1 έως 320 αριστερά , 320 συγκρίσεις στην χειρότερη με δεξιά

136 – 62 = distance “**74**”

Visual Cortex

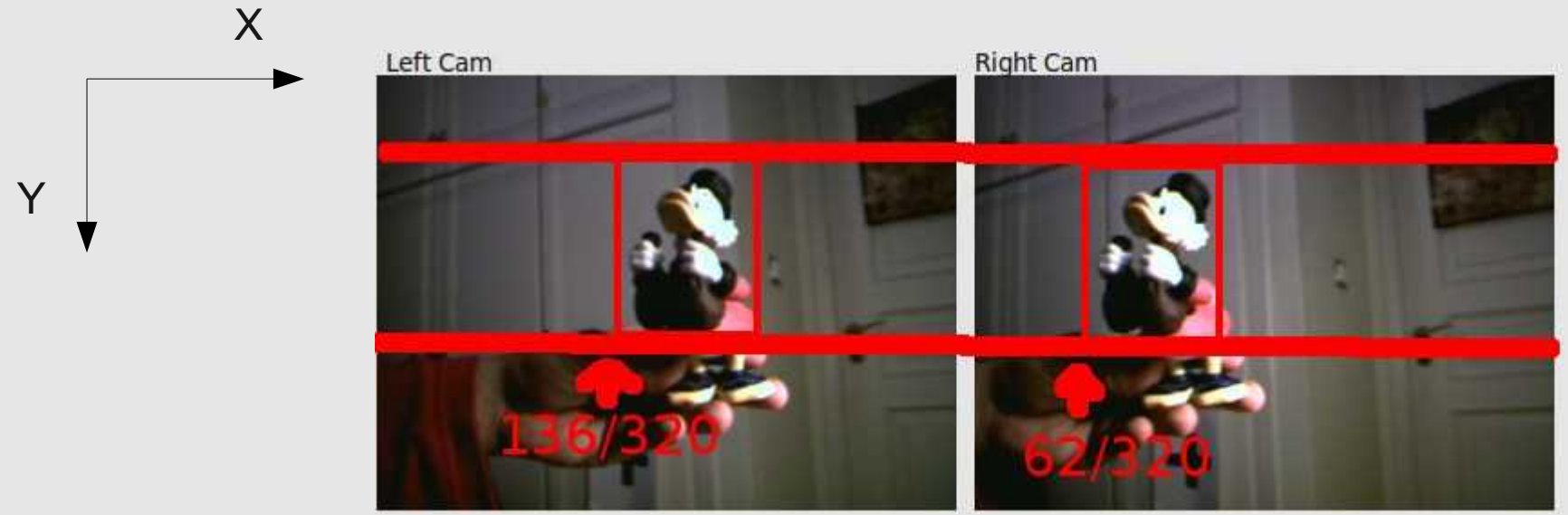
Εμπειρικές μετρήσεις



Με τις κάμερες μου (φακούς/παραμορφώσεις κτλ) σε απόσταση 6 cm
21cm = 92 , 22cm = 88 , 23cm = 87 , 24cm = 83 , 25cm = 82 , 26cm = 79
27cm = 77 , 28cm = 75 , 29cm = 71 , 30cm = 70 κτλ κτλ κτλ

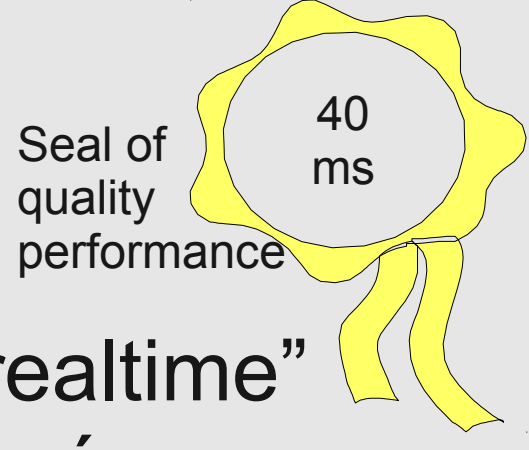
Visual Cortex

Η απόσταση της φιγούρας (74) σημαίνει ότι είναι γύρω στα 28-28.5 cm μακριά από το ρομπότ στο συγκεκριμένο screenshot !



136 – 62 = distance “**74**”

Visual Cortex



Για να γίνεται το disparity mapping “realtime” θέλουμε να παίρνει στην χειρότερη περίπτωση 40ms το κάθε scan(25 x 40 = 1000 ms , 25 fps)

Κάθε operation είναι π.χ. σύγκριση δύο 30x50 patches
Το GuarddoG πετυχαίνει περίπου 100-300 ms ανάλογα με τον υπολογιστή που τρέχει τον RoboKernel και τον φωτισμό του χώρου στον οποίο κινείται

320 x 320 x 240 / 40 = 24576000 /40 = 614400 operations /ms
640 x 640 x 480 / 40 = 196608000 /40 = 4915200 operations / ms
1024 x 1024 x 768 / 40 = 805306368 / 40 = 20132659 operations / ms
...
...
1920 x 1920 x 1024 / 40 = 3774873600 / 40 = 94371840 operations / ms

Visual Cortex

Βελτιώσεις :

Για κάθε αριστερό X , comparison δεξιά μέχρι το X αντί για το 320
Histogram Comparison Before Patch Comparison (faster candidate discarding)
Αντί για κάθε X,Y comparison για κάθε X/detail , Y/detail
Thresholding για γρήγορη απόρριψη
Multiple level comparison (διαφορετικά patch sizes)
Normalization
...
Και άλλα ..

Visual Cortex

My Patch Matching Function

```
unsigned int inline ComparePatches(
                    struct ImageRegion source_block,
                    struct ImageRegion target_block,
                        unsigned char *left_view,
                        unsigned char *right_view,
                    unsigned char *left_gauss_sobel,
                    unsigned char *right_gauss_sobel,
                    unsigned int best_score
                                                )
```

Visual Cortex

My Patch Matching Function

Είναι “καλός” αλγόριθμος καθότι συγκρίνει κυρίως τις ακμές μεταξύ τους αλλά παίρνει υπ`όψην και το χρώμα του patch οπότε βελτιώνει το αποτέλεσμα και πρακτικά δουλεύει..

```
ScoreThreshold = 17000 // για παράδειγμα
If ( ScoreThreshold > best_score )  { ScoreThreshold = best_score;  }

matching_score=0; // ( lower is better )
For each pixel of patch1,patch2
 {
   If difference of patch1.sobel[pixel] and patch2.sobel[pixel] > 15 then sobel_mismatch=1;
   If difference of patches < 40 and both patches > 30 then sobeled = 1;
   matching_score += color_difference_of(patch1.sobel[pixel],patch2.sobel[pixel])
   matching_score += sobel_difference_of(patch1.sobel[pixel],patch2.sobel[pixel])
   If ( sobel_mismatch ) matching_score +=
                    sobel_difference_of(patch1.sobel[pixel],patch2.sobel[pixel]) * 8

   If ( matching_score > ScoreThreshold ) break;
}
return matching_score;
```

Visual Cortex

Using Histograms to Speed up Patch Matching

Λόγω της επαναληπτικής φύσης της διαδικασίας κυρίως στην δεξιά εικόνα τα ίδια blocks περνιούνται ξανά και ξανά και ξανά για αυτό τον λόγο μια καλή (και γρήγορη όταν υλοποιήθει) ιδέα για ένα φίλτρο που να γλυτώνει περιττές συγκρίσεις είναι να συγκρίνουμε τον μέσο όρο των καναλιών R G B..!
Έτσι έχοντας για παράδειγμα 2 blocks εικόνων

10 123 165 200 165 123 10	20 140 180 220 180 140 20
10 123 165 200 165 123 10	20 140 180 220 180 140 20
10 123 165 200 165 123 10	20 140 180 220 180 140 20
10 123 165 200 165 123 10	20 140 180 220 180 140 20
10 123 165 200 165 123 10	20 140 180 220 180 140 20
10 123 165 200 165 123 10	20 140 180 220 180 140 20
Median : 113.7	Median : 128.5

Με κατάλληλη υλοποίηση επιταχύνει 10-20% βελτίωση ταχύτητας στο Patch Comparison

Visual Cortex

Using Histograms to Speed up Patch Matching

Η οικονομία γίνεται ως εξής

X X X X X X X X X	***X X X X X X X*** X X
X 10 123 165 200 165 123 10	***X 20 140 180 220 180*** 140 20
X 10 123 165 200 165 123 10	***X 20 140 180 220 180*** 140 20
X 10 123 165 200 165 123 10	***X 20 140 180 220 180*** 140 20
X 10 123 165 200 165 123 10	***X 20 140 180 220 180*** 140 20
X 10 123 165 200 165 123 10	***X 20 140 180 220 180*** 140 20
X 10 123 165 200 165 123 10	X 20 140 180 220 180 140 20

Σε κάθε μετακίνηση του παραθύρου (patch) απλά προσθέτουμε τους όρους στην άκρη δεξιά πχ και αφαιρουμε τους όρους στην άκρη αριστερά ! Έτσι γλιτώνουμε παρα πολλές πράξεις

Visual Cortex

Using Histograms to Speed up Patch Matching

Η οικονομία γίνεται ως εξής

X	***X***	***X***	***X***	***X***	***X***	***X***	X	X
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	10	123	165	200	165	123	10	

X	***X***	***X***	***X***	***X***	***X***	***X***	X	X
X	***20***	***140***	***180***	***220***	***180***	140	20	
X	***20***	***140***	***180***	***220***	***180***	140	20	
X	***20***	***140***	***180***	***220***	***180***	140	20	
X	***20***	***140***	***180***	***220***	***180***	140	20	
X	***20***	***140***	***180***	***220***	***180***	140	20	
X	20	140	180	220	180	140	20	

Σε κάθε μετακίνηση του παραθύρου (patch) απλά προσθέτουμε τους όρους στην άκρη δεξιά πχ και αφαιρουμε τους όρους στην άκρη αριστερά ! Έτσι γλιτώνουμε παρα πολλούς υπολογισμούς

Visual Cortex

Ευριστικές βελτίωσης

Το αποτέλεσμα από όλες τις παραπάνω διαδικασίες πολύ συχνά έχει κενά σημεία (σημεία μακρυά από ακμές), σημεία στα οποία τοπικά λόγω θορύβου μπορεί να υπάρχει κάποια έντονη αιχμή και άλλες ατέλειες.

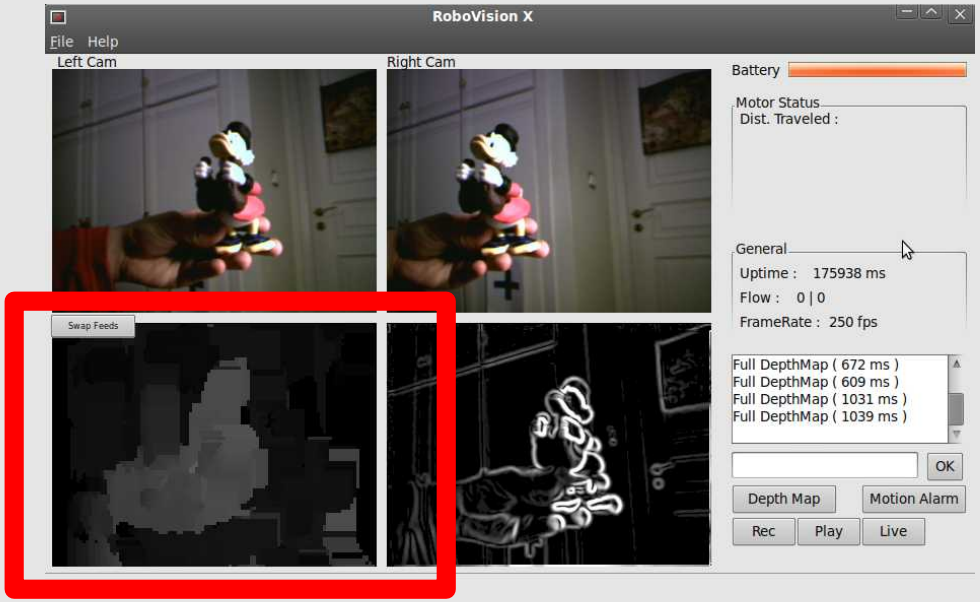
Για να βελτιωθεί το αποτέλεσμα κάποιες άλλες τεχνικές που εφαρμόζονται είναι :

Μεταβλητό μέγεθος patch
Γέμισμα κενών κάθετα , για όσο δεν υπάρχουν ακμές
“Μαντεψιά” της επόμενης αντιστοίχησης (θεωρόντας ότι συνεχίζει την προηγούμενη)
Αντιστοίχηση των σημείων που κινούνται μεταξύ τους
Και άλλα..

Visual Cortex

- Το αποτέλεσμα των αλγορίθμων του Visual Cortex είναι μια τρισδιάστατη φέτα του κόσμου , με τιμές από 0 (μακρυά) έως 320 (θεωρητικό κοντά όριο)

- Συνδυάζοντας την με τις πληροφορίες χρώματος έχουμε ένα 3D ανάγλυφο



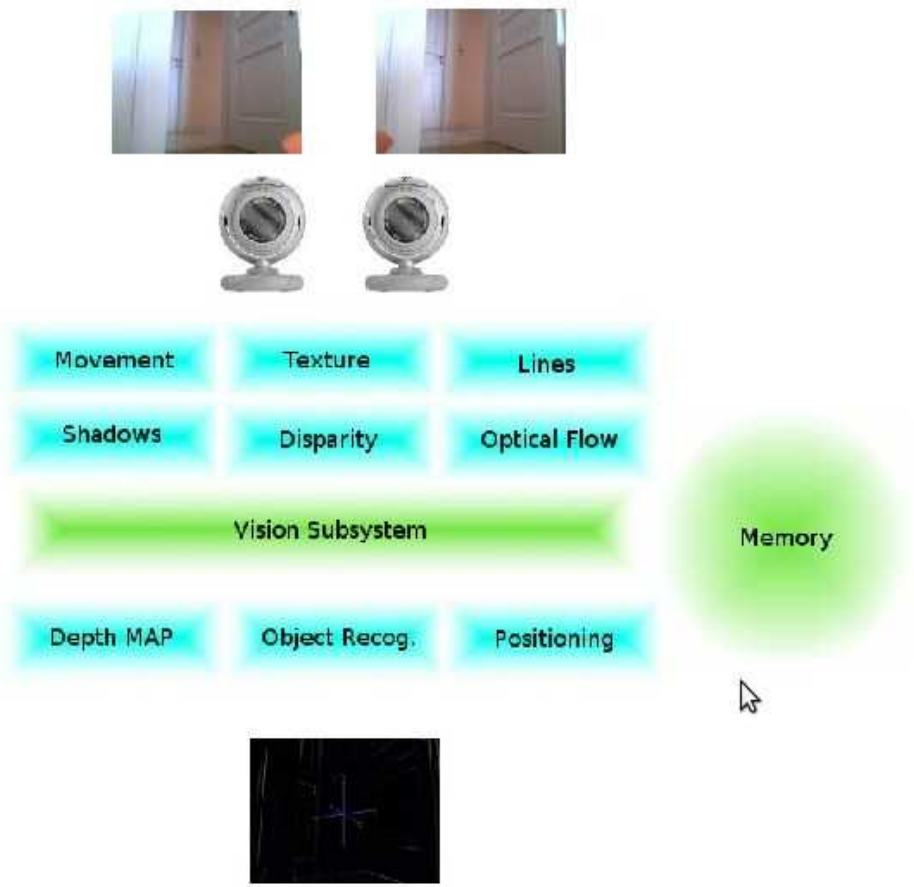
Visual Cortex

- Χρησιμοποιώντας τους ίδιους αλγόριθμους για patch comparison είναι δυνατό το tracking σε features , συγκρίνοντας τα με την “γειτονιά” τους.
- Στο Visual Cortex υπάρχει ένα δικό μου πρόχειρο implementation που δεν αποδίδει πολύ καλά
- Έτοιμες συμβατές βιβλιοθήκες/λύσεις είναι η OpenCV ή το OpenSURF

Visual Cortex

Ιδανικά :

- Optical Flow tracking σε κάθε κάμερα
- Disparity Mapping
- Light/Shadow Depth estimation
- Line Comparison



World3D

(actual guarddog voxel rendering)

World3D

Προς το παρόν λόγω μη λειτουργικότητας της κεφαλής 2 αξόνων ελευθερίας κάθε εικόνα που εξάγει το GuarddoG είναι κάθετη στο επίπεδο το οποίο κινείται , προφανώς λοιπόν μετρώντας το state των encoders των μοτέρ (πόσες μοίρες έχουν γυρίσει) , το input από το accelerometer 2 αξόνων , την είσοδο των ultrasonic sensors και το κάθετο depth map , ενώ θα μπορούσε να προβάλλεται η κάθε τρισδιάστατη τομή σε ένα συνολικό 3D mesh προς το παρόν σχηματίζεται , μόνο μια δισδιάστατη αναπαράσταση του χώρου (κενό / όχι κενό) στον οποίο μπορεί ανάλογα να προχωρήσει ή όχι το guarddog..

Το World3D είναι ένα stub κομμάτι του project το οποίο στο μέλλον συγκρίνοντας γνωστά features του χώρου (και χρησιμοποιώντας το Visual Cortex) θα πρέπει να προσφέρει πλήρη 3D αναπαράσταση του χώρου και όχι το απλό δισδιάστατο μοντέλο του στο οποίο βρίσκει μονοπάτια το guarddog..

World3D

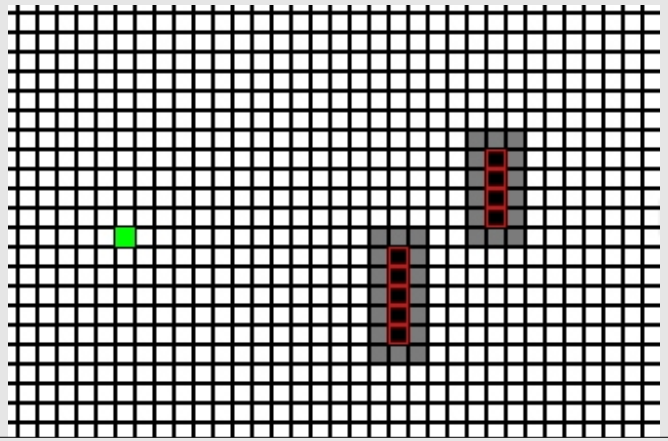
Ο σωστός τρόπος για να γίνει ονομάζεται SLAM
Έχουμε ένα σύννεφο από σημεία στην δεξιά κάμερα , έχουμε ένα αντίστοιχο σύννεφο στην αριστερή κάμερα και την αντιστοίχηση μεταξύ τους

Με το που αλλάξει το viewpoint της σκηνής , έχουμε την μετακίνηση του σύννεφου σημείων οπότε ιδανικά θα μπορούσαμε να υπολογίσουμε τον πίνακα με τον οποίο “πολλαπλασιάστικαν” τα σημεία έτσι ώστε να περιστραφούν

World3D

Προς το παρόν η υλοποίηση λαμβάνει απλά την θέση των encoders των μοτέρ και υποθέτει έναν δισδιάστατο χώρο , όπου τα ορατά εμπόδια μπλοκάρουν όλο το επίπεδο μπροστά

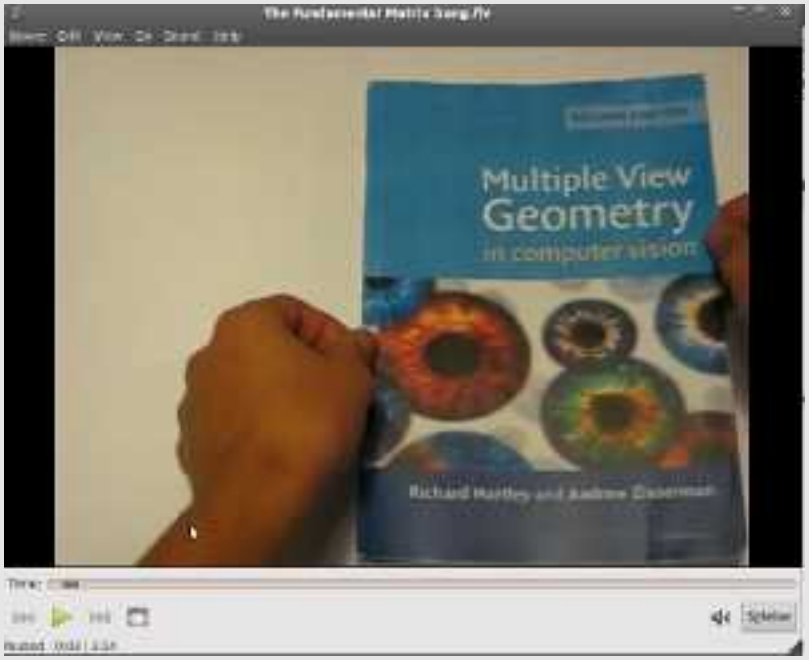
Το setting/unsetting των εμποδίων στην μνήμη του GuarddoG γίνεται εξίσου απλά τρέχοντας τον αλγόριθμο Bresenham (2D Line casting) , αυτό είναι σίγουρα πολύ πιο γρήγορο από μια πλήρη 3D υλοποίηση , επίσης είναι σίγουρα παρα πολύ πιο ανακριβές , μια πιο καλή υλοποίηση είναι το επόμενο πράγμα το οποίο θα πρέπει να γίνει implement!



2D GuarddoG represenatation

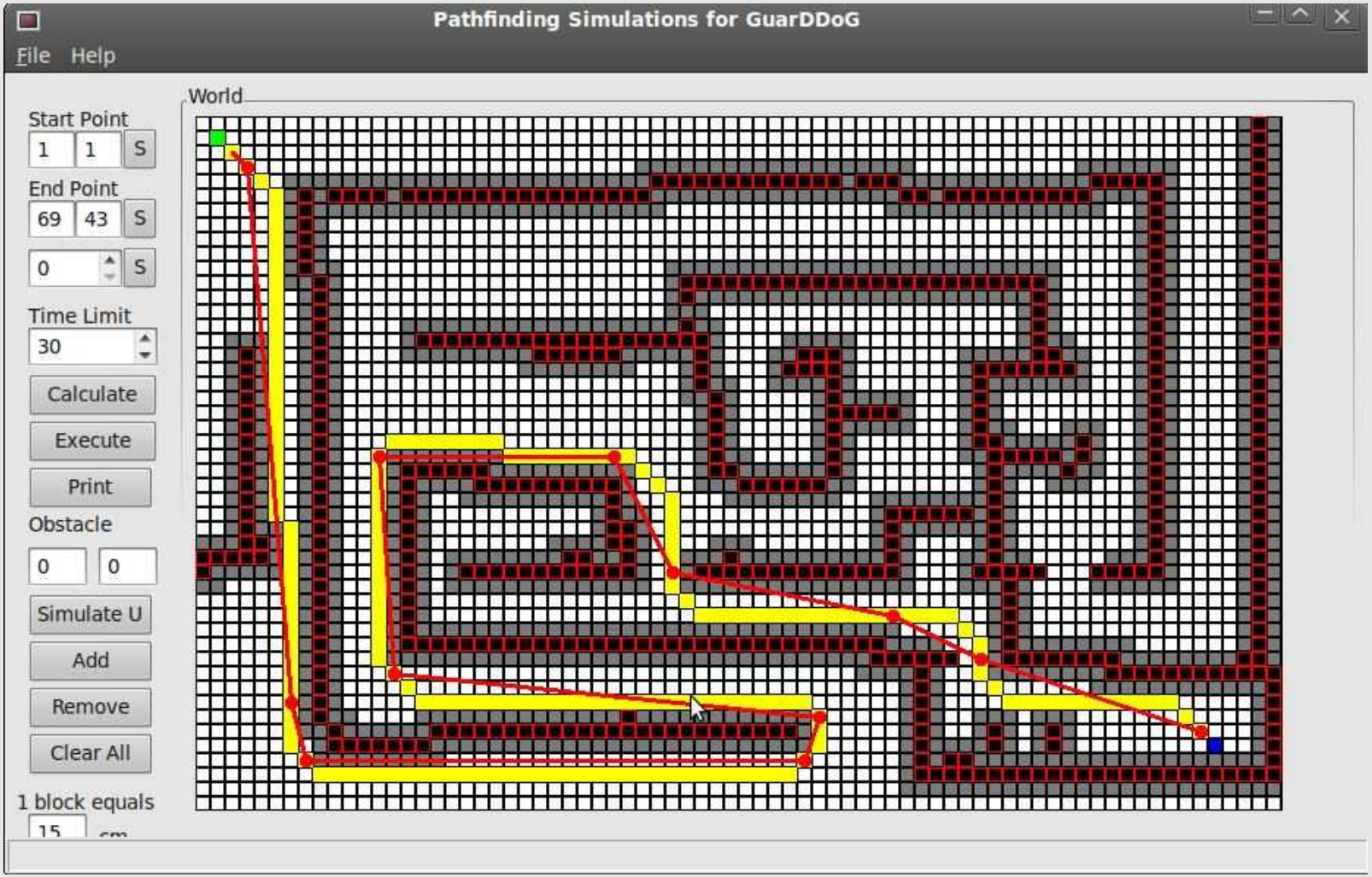
Much better 3d representation

Word3D



http://tinyurl.com/fundamentalmatrix

World Mapping



World Mapping

A * algorithm , with a twist !
LOGO output

struct Map * CreateMap(unsigned int world_size_x,unsigned int world_size_y,unsigned int actors_number);

int SetObstacle(struct Map * themap,unsigned int x,unsigned int y,unsigned int safety_radious);

int FindSponteneousPath(struct Map * themap,unsigned int agentnum,unsigned int x1,unsigned int y1,unsigned int x2,unsigned int y2,unsigned int timeout_ms) ;

int GetRoutePoints(struct Map * themap,struct Path * thepath) ;

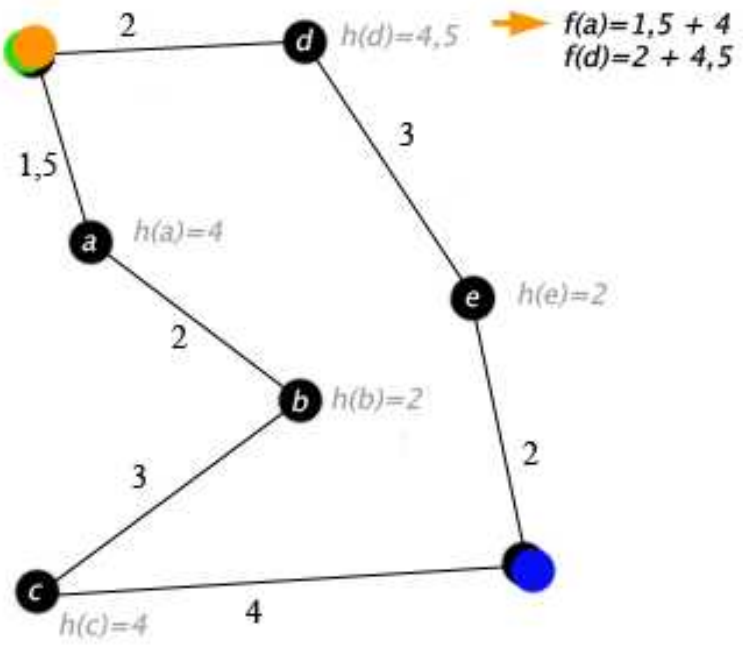
int GetRouteWaypoint(struct Map * themap,unsigned int agentnum,unsigned int count,unsigned int *x,unsigned int *y) ;

int GetStraightRouteWaypoint(struct Map * themap,unsigned int agentnum,unsigned int count,unsigned int *x,unsigned int *y);

int DeleteMap(struct Map * themap) ;

World Mapping

Λίγα λόγια για τον A*



A * algorithm

* Optimal

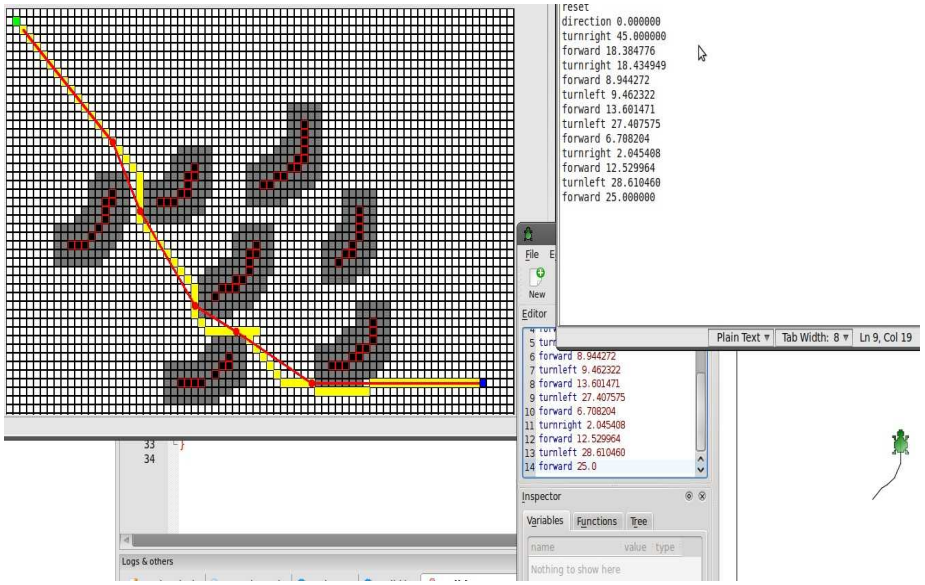
* Manhattan distance heuristic + στροφές!

* it is polynomial when the search space is a tree, there is a single goal state, and the heuristic function h meets the following condition:

| h(x) − h * (x) | = O(logh * (x))

where h * is the optimal heuristic, the exact cost to get from x to the goal. In other words, the error of h will not grow faster than the logarithm of the “perfect heuristic” h * that returns the true distance from x to the goal

World Mapping
Logo Output



Output σε γλώσσα “υψηλού επιπέδου”

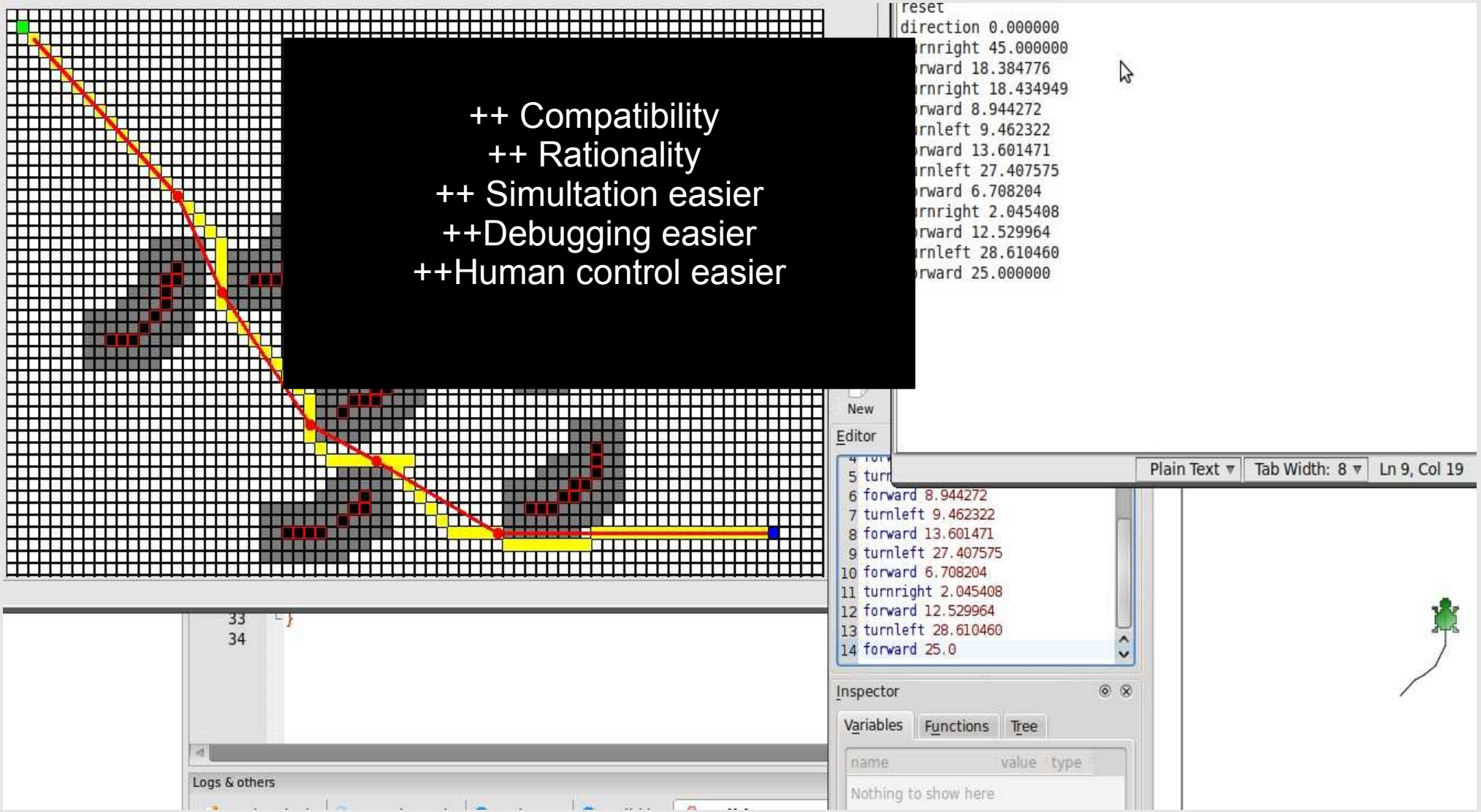
Για παράδειγμα :

Turnright 45 /* μοίρες */

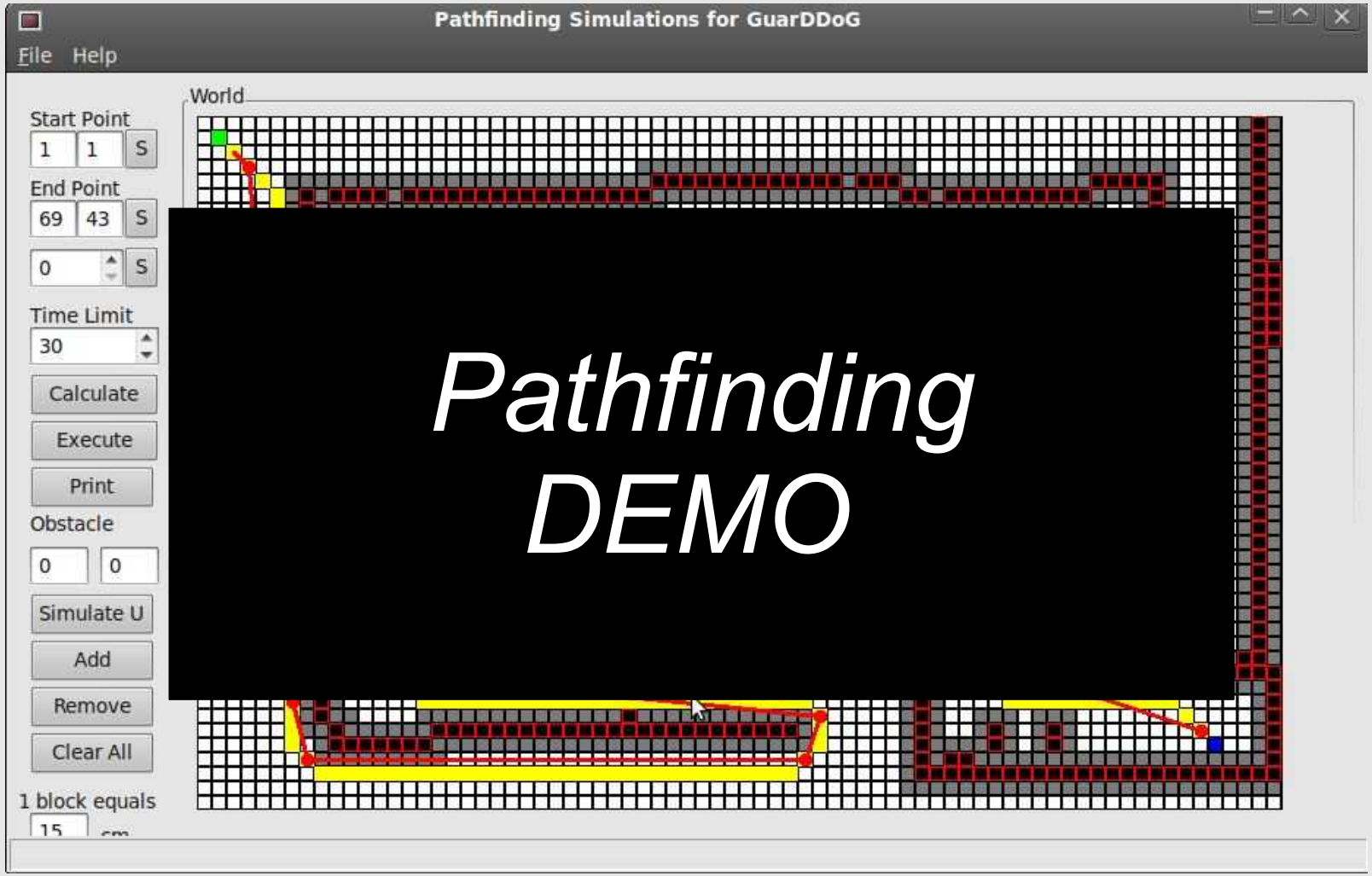
Forward 15 /* cm */

Κτλ κτλ..

World Mapping
LOGO Output



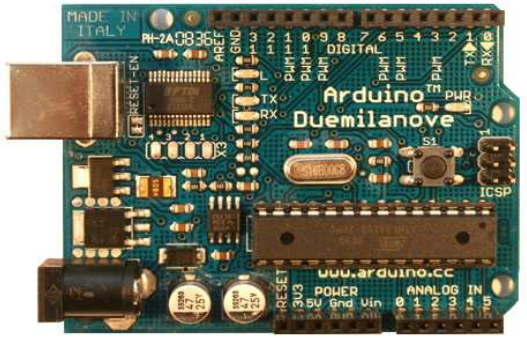
World Mapping



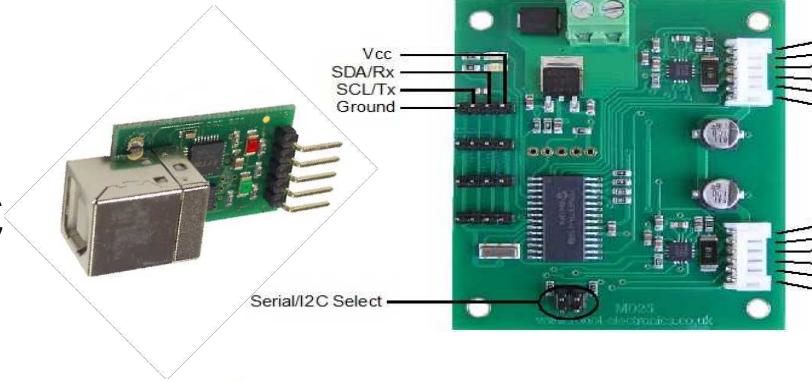
Motor Foundation

MotorHAL

Arduino



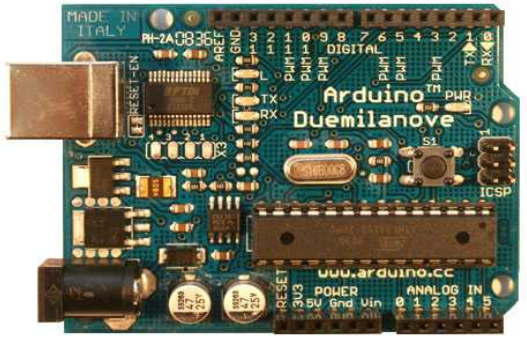
MD23
Via USB 2 I2C



Mindstorm

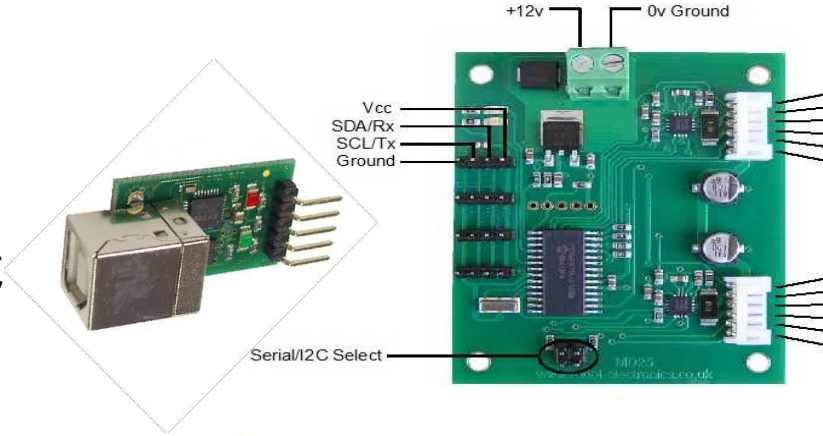
Motor Foundation

Arduino



MotorHAL

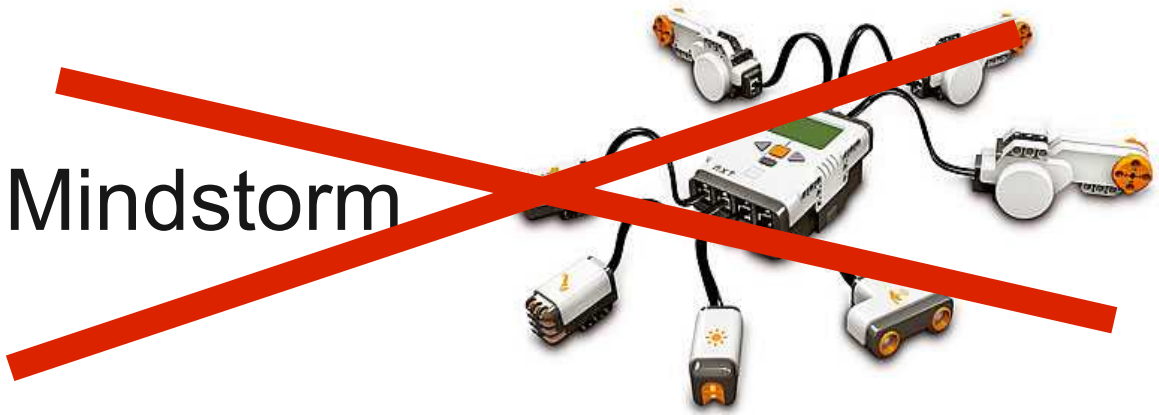
MD23
Via USB 2 I2C



Παρ`όλα αυτά πολύ καλή ιδέα για μια αρχή στην επικοινωνία με microcontrollers κτλ

Προσωπικά πιστεύω είναι το πιο εύκολο πράμα που μπορεί να χρησιμοποιήσει κάποιος για ξεκίνημα αλλά **πολύ** ακριβό!!

Mindstorm



Motor Foundation
Abstraction Layer

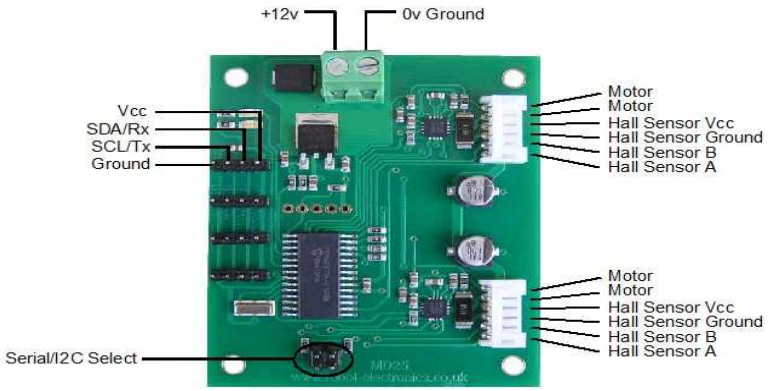
unsigned int RobotInit(char * md23_device_id,char * arduino_device_id);
unsigned int RobotClose();
void RobotWait(unsigned int msecs);

unsigned int RobotRotate(unsigned char power,signed int degrees);
unsigned int RobotStartRotating(unsigned char power,signed int direction);
unsigned int RobotMove(unsigned char power,signed int distance);
unsigned int RobotStartMoving(unsigned char power,signed int direction);
unsigned int RobotManoeuvresPending();
void RobotStopMovement();

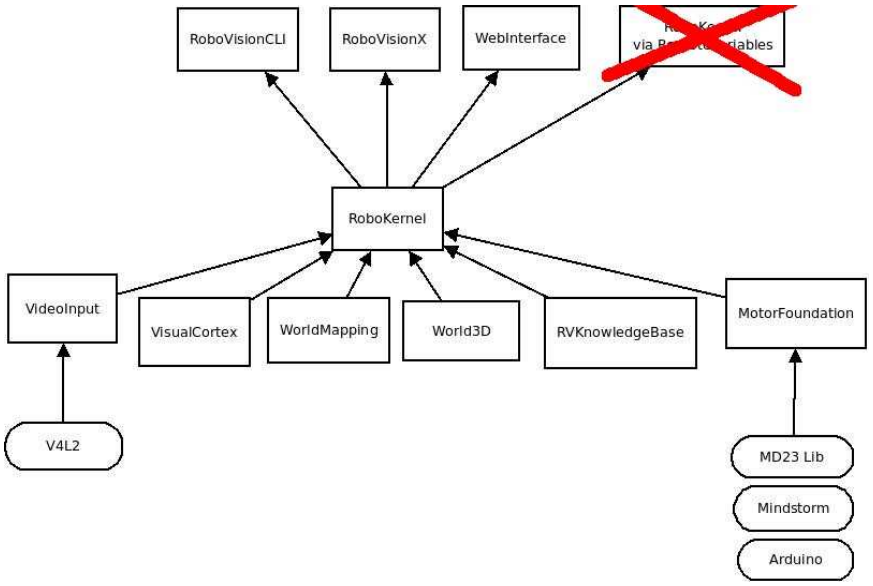
int RobotGetUltrasonic(unsigned int dev);
int RobotGetAccelerometerX(unsigned int dev);
int RobotGetAccelerometerY(unsigned int dev);
int RobotSetHeadlightsState(unsigned int scale_1_on,unsigned int scale_2_on,unsigned int scale_3_on);
int RobotIRTransmit(char * code,unsigned int code_size);

Motor Foundation

Καλή ιδέα το abstraction layer



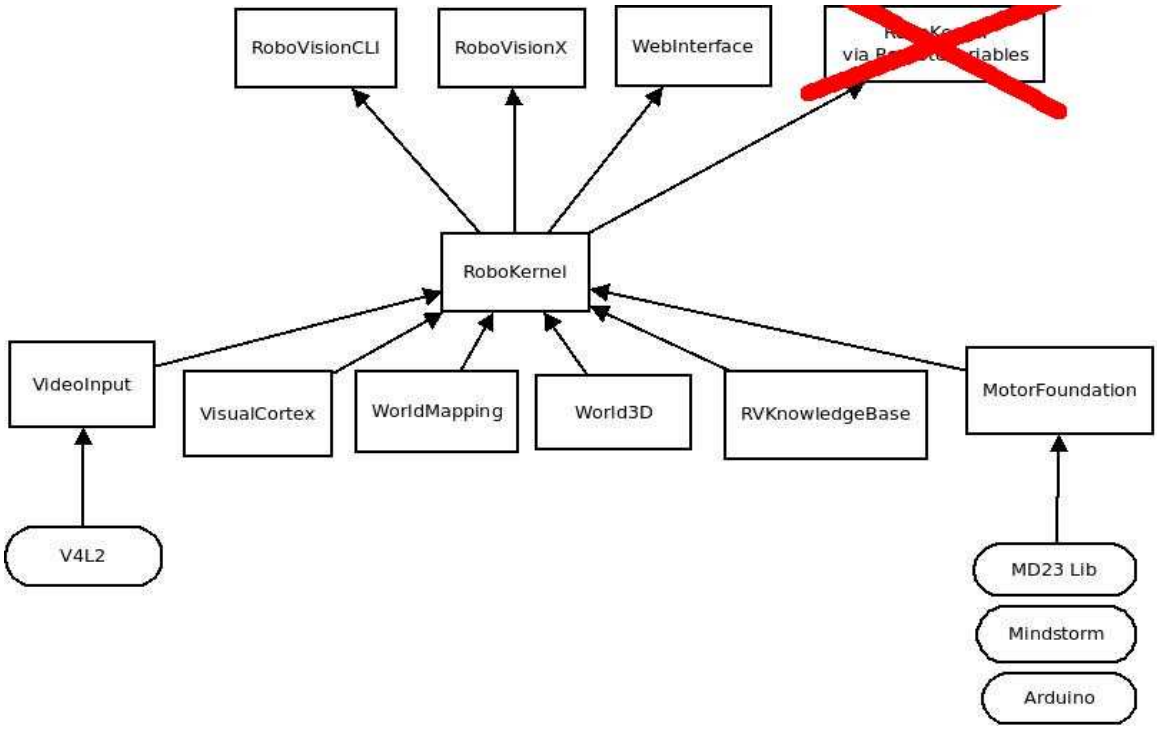
- Ουσιαστικά ο,τι μηχανική αλλαγή και να κάνω (αλλαγή controller , αλλαγή chasis , κτλ κτλ) , το μόνο που χρειάζεται είναι να την υλοποιήσω από κάτω και να κάνω redirect το RobotMove function
- Η αλλάγή από mindstorms σε MD25 παρότι το ένα kit για παιδιά και το άλλο “επαγγελματικό” έγινε παρα πολύ ανώδυνα χάρη σε αυτό τον σχεδιασμό..



RV knowledge base

Κάτι σαν το http://openmind.media.mit.edu/

High level εντολές και ενοποιημένο pipelining για την επεξεργασία τους



GuarddoG in numbers

- 4+ χρόνια
- 3 complete rewrites
- Περίπου 639 euro construction cost το συγκεκριμένο prototype
- C 58% , C++ 39% BAShell 1% , Arduino C 1% , PHP 1%

GuarddoG in numbers
via Code::Blocks Code Statistics

Libs
Visual Cortex - 3550 loc (64% code , 13% comments , 21% empty)
Video Input - 2560 loc (56% code , 30% comments , 15% empty)
Path Planning - 1850 loc (67% code , 9% comments , 20% empty)
RoboKernel - 1130 loc (73% code , 6% comments , 19% empty)
MD23/25 Lib – 911 loc (70% code , 4% comments , 19% empty)
InputParser_C – 603 loc (54% code , 23% comments , 21% empty)
Arduino Com lib – 316 loc (70% code , 4% comments , 20% empty)
MotorHAL – 295 loc (71% code , 4% comments , 21% empty)

GUIs
RoboVisionX – 1722 loc (76% code , 7% comments , 17% empty)
WorldMapping – 846 loc (73% code , 12% comments , 15 % empty)
RoboVisionCLI – 34 loc :P (68% code , 32% empty)

Προς το παρόν περίπου 13817 loc written by me..

To add :
RVKnowledgebase , World3D , etc

Master Foo and the ten thousand lines

Master Foo once said to a visiting programmer: “There is more Unix-nature in one line of shell script than there is in ten thousand lines of C.”
The programmer, who was very proud of his mastery of C, said: “How can this be? C is the language in which the very kernel of Unix is implemented!”
Master Foo replied: “That is so. Nevertheless, there is more Unix-nature in one line of shell script than there is in ten thousand lines of C.”
The programmer grew distressed. “But through the C language we experience the enlightenment of the Patriarch Ritchie! We become as one with the operating system and the machine, reaping matchless performance!”
Master Foo replied: “All that you say is true. But there is still more Unix-nature in one line of shell script than there is in ten thousand lines of C.”
The programmer scoffed at Master Foo and rose to depart. But Master Foo nodded to his student Nubi, who wrote a line of shell script on a nearby whiteboard, and said: “Master programmer, consider this pipeline. Implemented in pure C, would it not span ten thousand lines?”
The programmer muttered through his beard, contemplating what Nubi had written. Finally he agreed that it was so.
“And how many hours would you require to implement and debug that C program?” asked Nubi.
“Many,” admitted the visiting programmer. “But only a fool would spend the time to do that when so many more worthy tasks await him.”
“And who better understands the Unix-nature?” Master Foo asked. “Is it he who writes the ten thousand lines, or he who, perceiving the emptiness of the task, gains merit by not coding?”

Upon hearing this, the programmer was enlightened.

GuarddoG in numbers

GuarddoG Construction Cost
Until 12 / 10 /2010

Chassis
2 x Tupper = 5 euro
1 x IKEA Bucket = 15 euro
1 x Wooden Board = 10 euro
1 x Balsa board = 5 euro
2x Supermarket Wheels :P = 5 euro
Nuts , bolts , rails , cables , etc = 20 euro
Total : 60 euro

Embedded Electronics
1x Arduino = 30 euro (Duemillenove)
3x Infrared Led = 3 euro
1x RD-01 (or RD-02 Devantech motors) = 130 euro
2x Desktop Microphones (GENIUS MIC-01A) = 5 euro
2x Buttons (power -on) = 2 euro
2x Switches (power supply) = 2 euro
2x LED HeadLights = 10 euro
2x Ultrasonic Devantech SRF-05 with mounting = 40 euro
1x Dual Axis Accelerometer (memsic 2125) = 30 euro
Total : 252 euro

Computer Hardware
1x Fan = 5 euro
1x Mini-Itx Motherboard = 65-75 euro (Currently on guarddog Intel D201GLY2)
1x PicoPSU 90W = 45 euro
1x AC-DC 12 V Converter = 30 euro
2x Webcams (On guarddog MS VX-6000) = 92 euro , LOGITECH C510 HD
1x WIFI PCI card (WG311T) = 30 euro
1x USB Flash Drive 8GB + = 20 euro
1x 512-2048MB RAM DIMM (on guarddog 512MB DDR2) = 30 euro
Total : 327 euro

Total : 639 euro

Πρώτα βήματα GuarddoG mk1

- Όλο το κατασκευαστικό κομμάτι με Lego Mindstorms
- 2 x Webcams

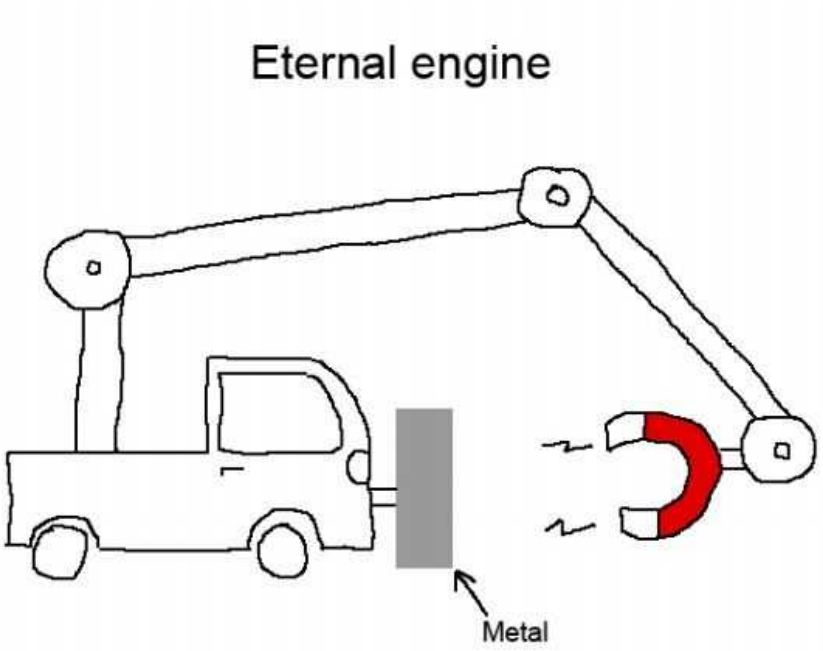
Κακό Calibration , ακτίνα όσο το καλώδιο USB (+ το USB hub)

Trials & Errors

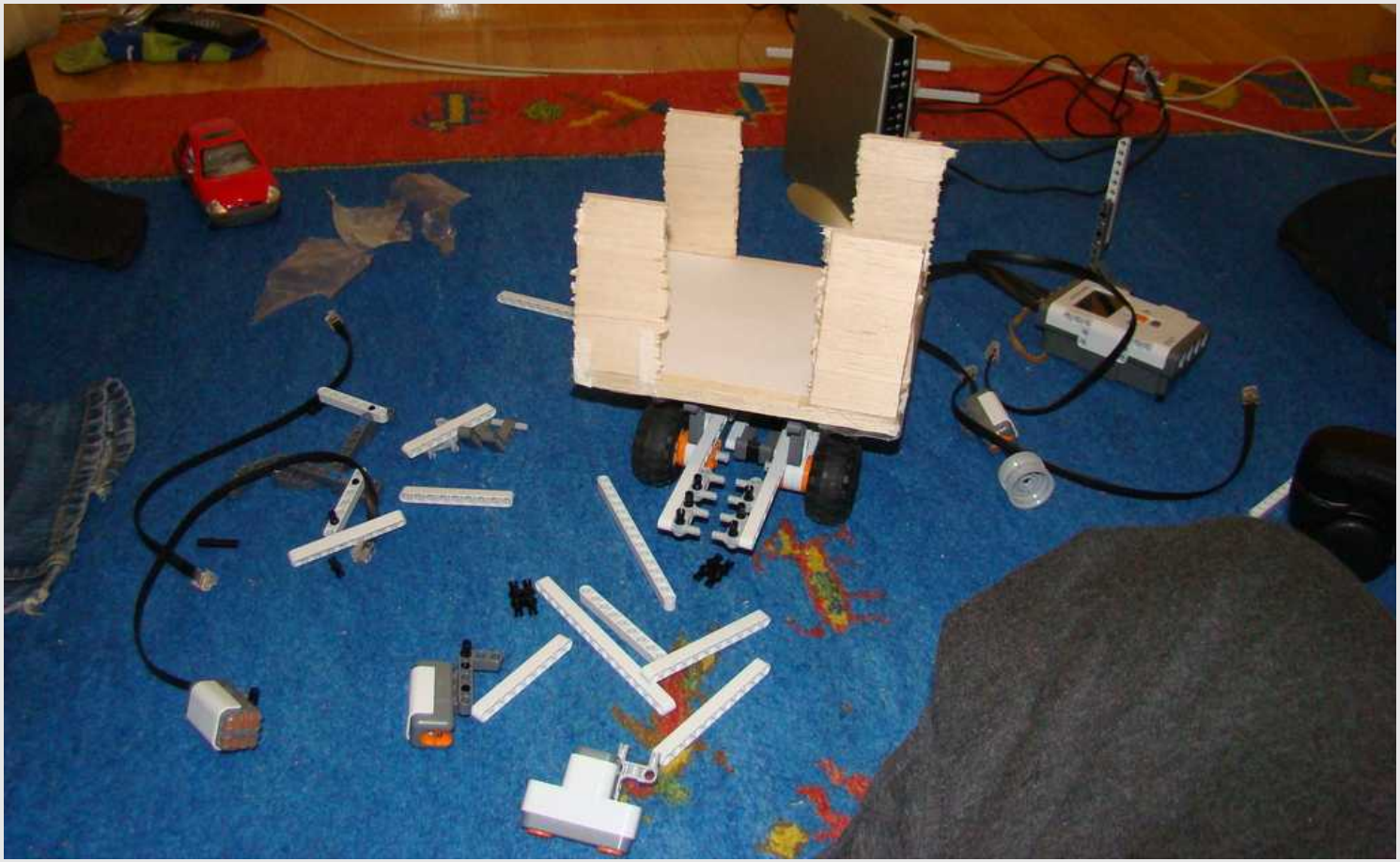
Πολλές ιδέες πολλές αποτυχίες..

- Υλικά του ρομπότ
- Στήριξη με Balsa
- Relay της εικόνας ασύρματα και επεξεργασία κάπου αλλού
- Επεξεργασία της εικόνας “onboard”
- Τεχνικά θέματα

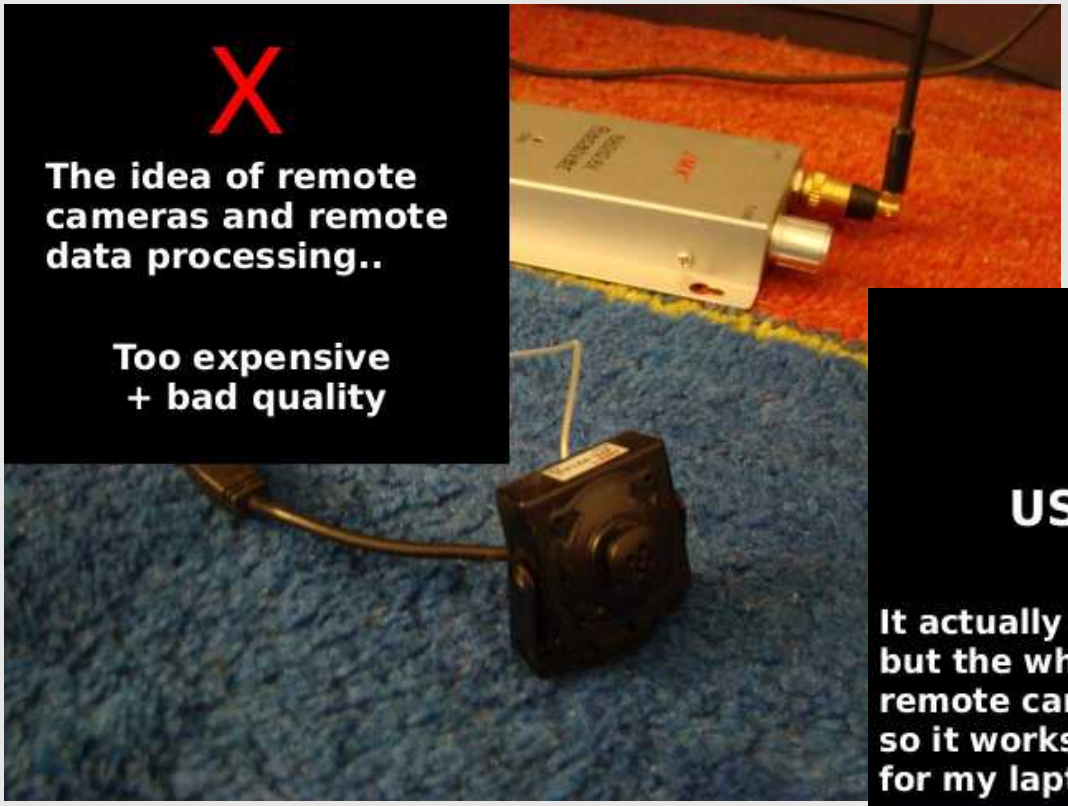
Σχεδόν τίποτα δεν δούλεψε όπως το σχεδίαζα στην αρχή ..



Balsa , όχι καλές στατικές ιδιότητες



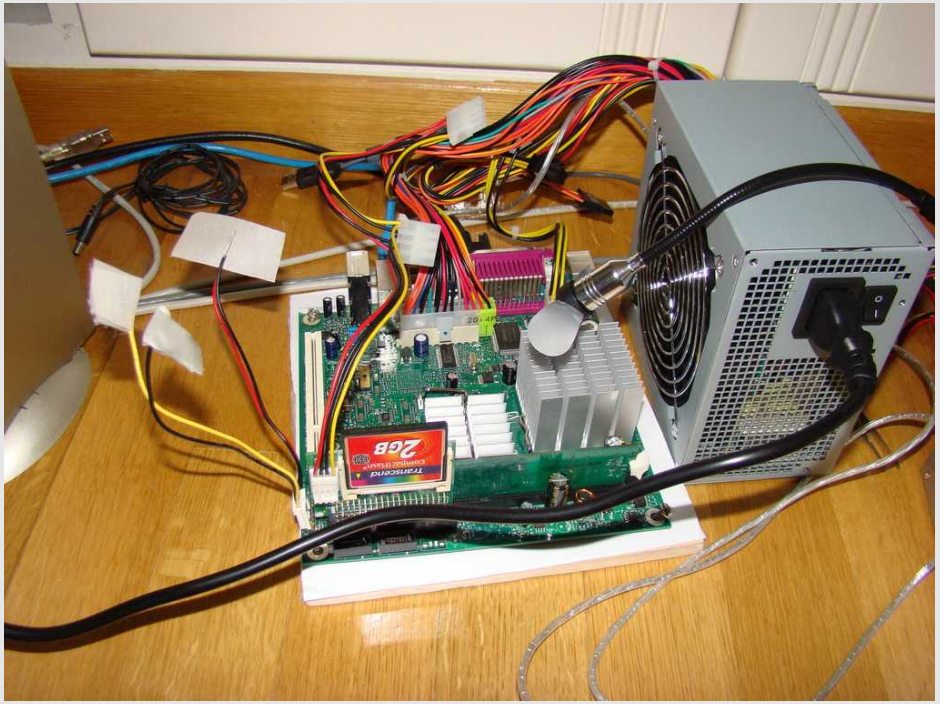
Wireless Cameras



Bank Balance = -150 euro

+ Θέματα ασφαλείας
(Unencrypted Video transmission)
κτλ

Επεξεργασία Onboard



- Άλλο ένα PC στο design..
- Extra Βάρος
- Τι λειτουργικό θα τρέχει
- Πόσο ρεύμα καταναλώνει κτλ
- Πόση επεξεργαστική ισχύς
- Κόστος

  Νέα Προβλήματα..

WinXP Epic Fail

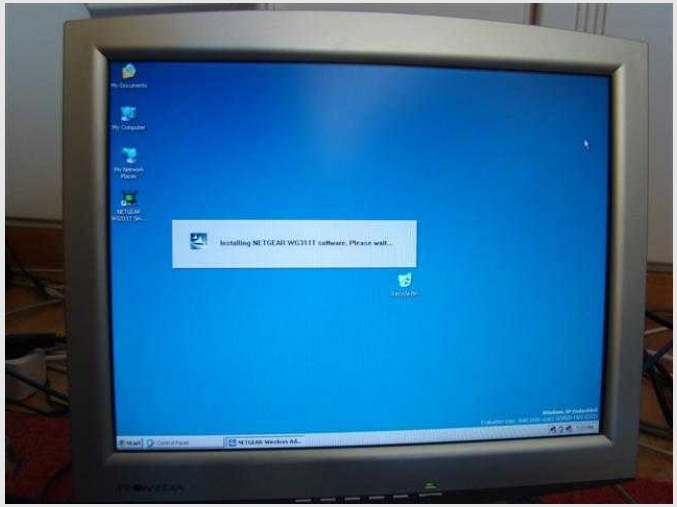
- Lock in , πολλά πράματα που είχα ήδη φτιάξει με DirectX
- Για χαμηλή κατανάλωση ρεύματος και αντοχή στην κίνηση θα πρέπει το PC να λειτουργεί χωρίς σκληρό δίσκο
- WinXP thrashed to death my CF card in 4 hours

WinXP Embedded

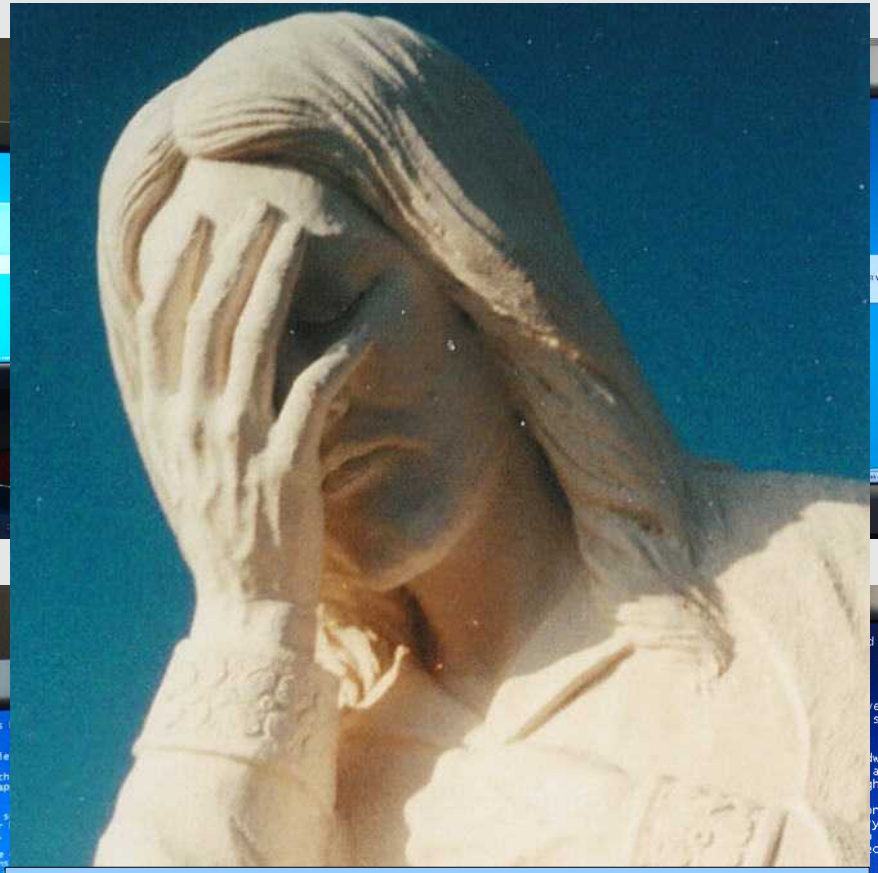
- OK με την (καινούργια) CF
- Binaries “Συμβατά” από WinXP
- Οι drivers για wifi κτλ μετά απο πολλά updates δούλεψαν

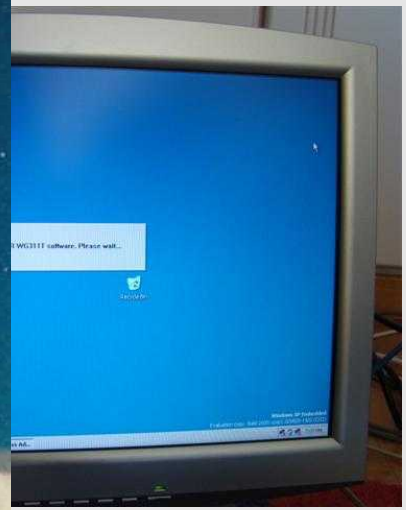
αλλά..

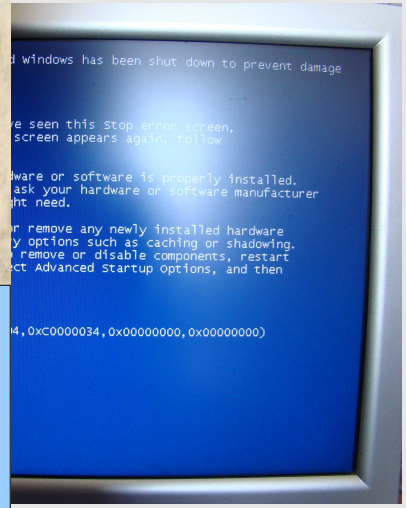
WinXP Embedded



WinXP Embedded







No Offence.. :P

Windows Γενικά

- Για να βάλω text to speech

  WinXP/Embedded έως SAPI 4.1

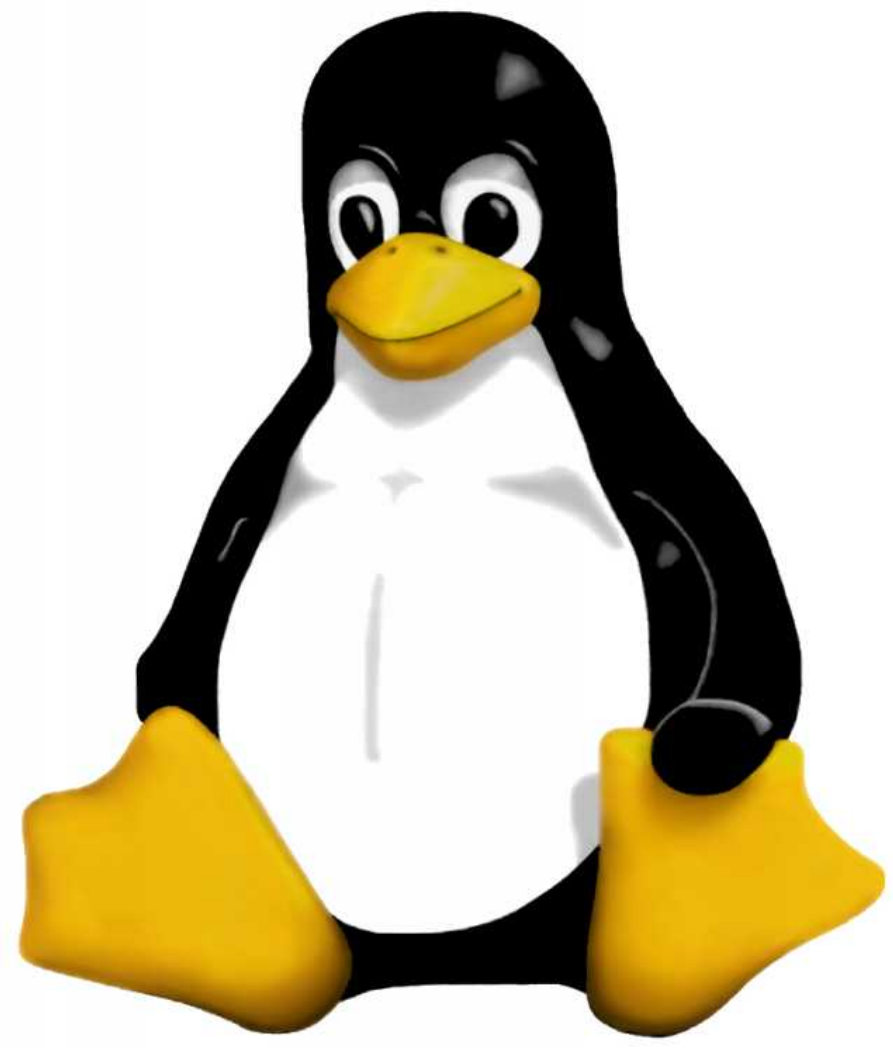
  Vista έως SAPI 5.0

  Win 7 έως SAPI 5.3
- 50+ euro per guarddog license
- Visual Studio X και πάνω μόνο
- Windows SDK 4GB , DirectX Sdk και άλλα τόσα για να κάνω κάτι απλό ..!
- Documentation μόνο για “binaries”

Περιορισμοί , περιορισμοί περιορισμοί ...

Αντίστοιχα σε Linux (Ubuntu/Debian-πχ) όπως δυστυχώς ανακάλυψα αργότερα..

- Για να κάνεις text to speech απαιτούνται οι εξής δύσκολες διαδικασίες



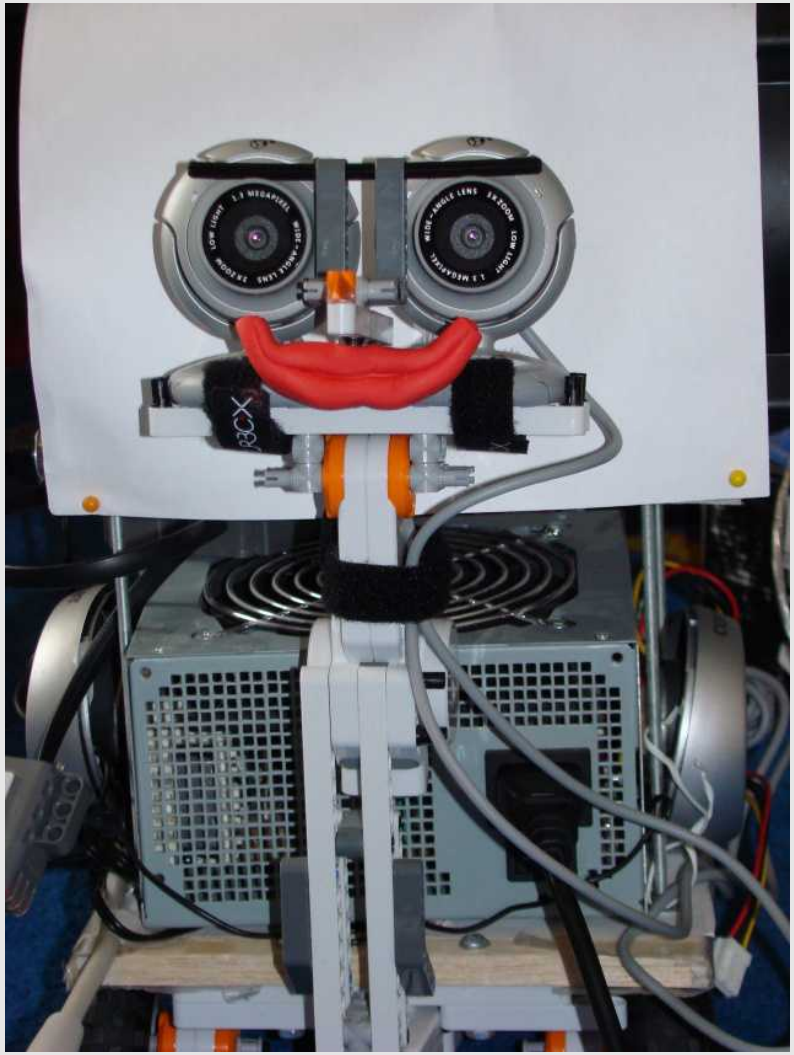
- sudo apt-get install festival
- echo “Text string” | festival –tts

  ή από C πχ
- system(“echo \“Text string\” | festival –tts”);

Εν το μεταξύ , για να επανέλθω ..

Παράλληλα με όλα τα κατασκευαστικά θέματα οι βασικοί αλγόριθμοι vision , αρχίζουν να υλοποιούνται..

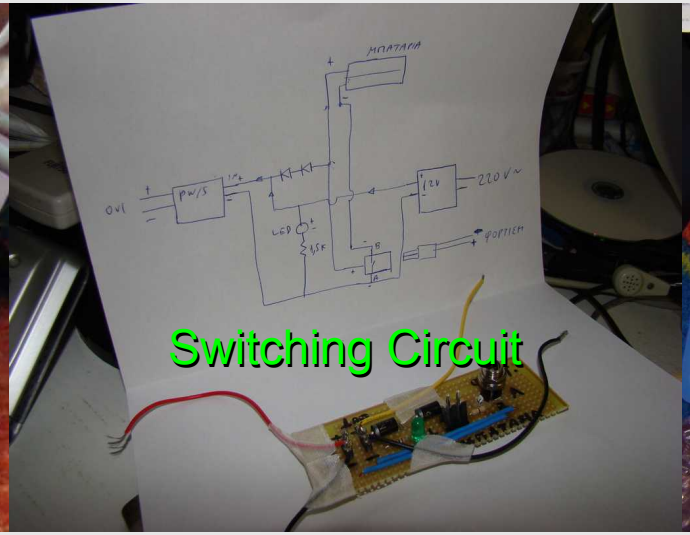
GuarddoG mk2

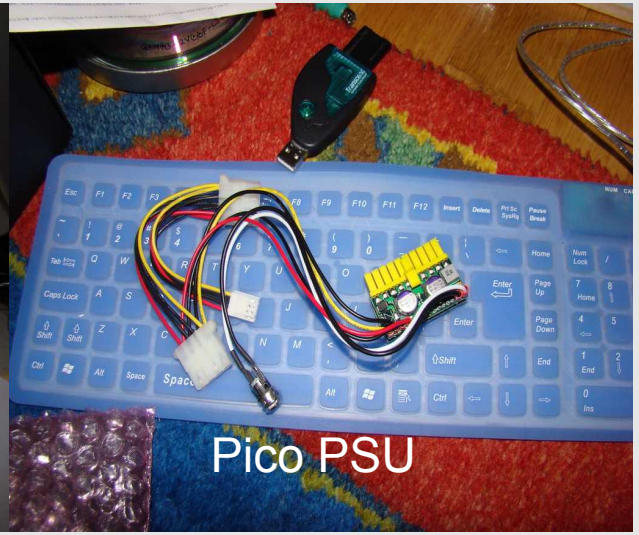


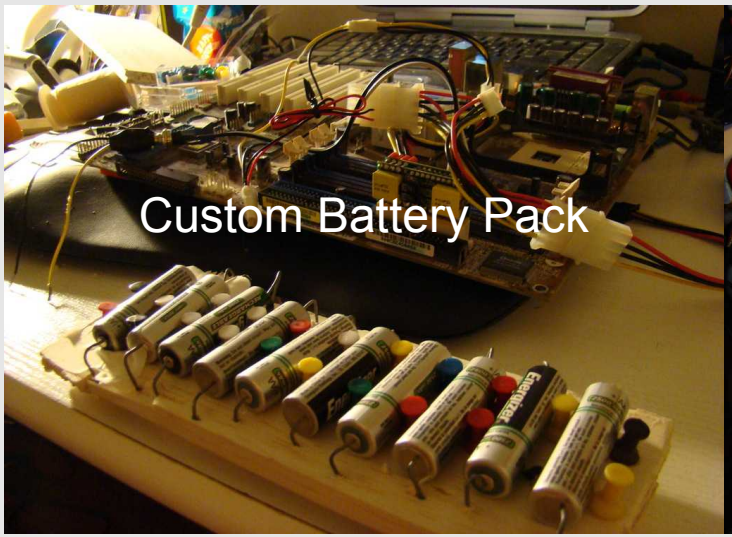
- Βαρύ τροφοδοτικό
- Γενικά μεγάλο βάρος για τα mindstorm motors
- Κακό alignment καμερών
- Software σε πρώιμο “μονοκόμματο” στάδιο

GuarddoG mk2 -> mk3

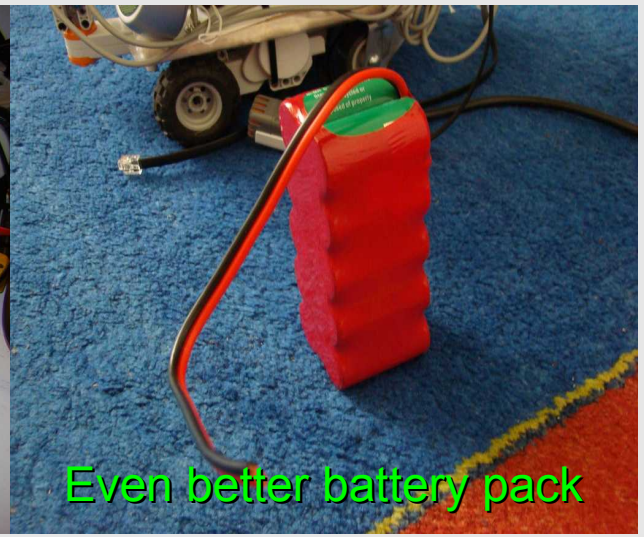

Γέμισμα τροχών με βαμβάκι


Switching Circuit

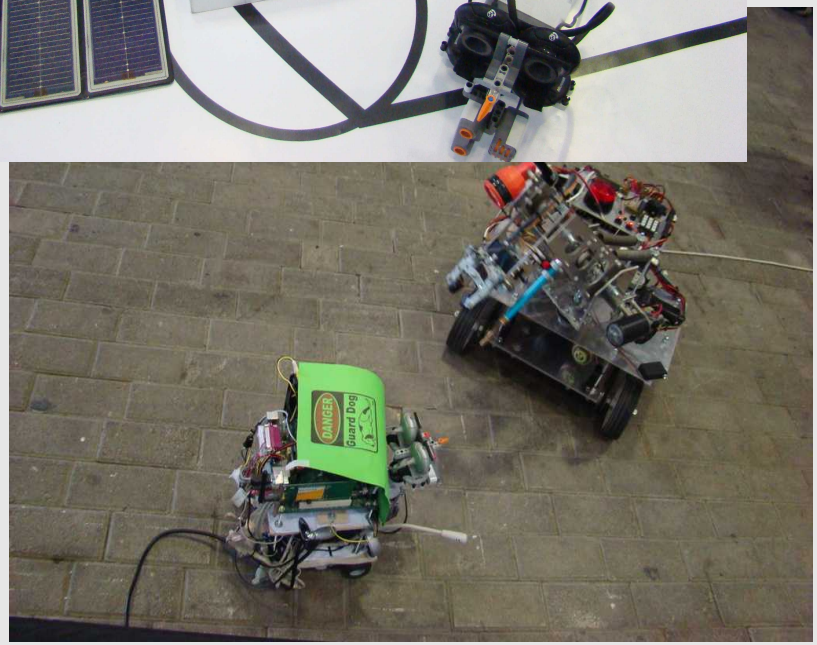

Pico PSU


Custom Battery Pack


Better Battery Pack


Even better battery pack

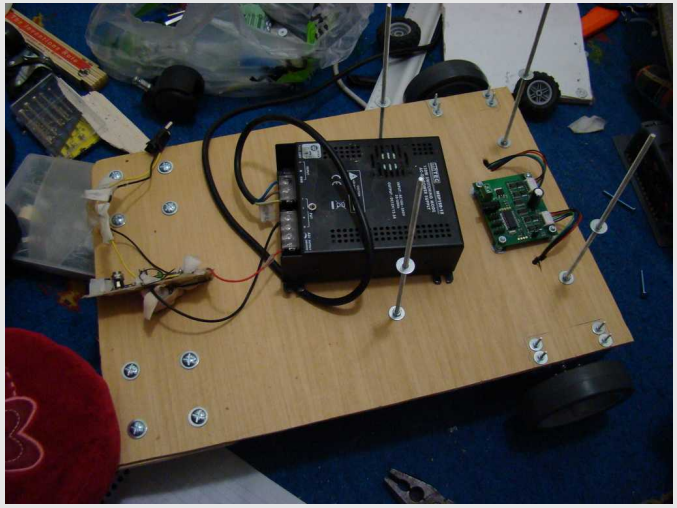
GuarddoG mk3

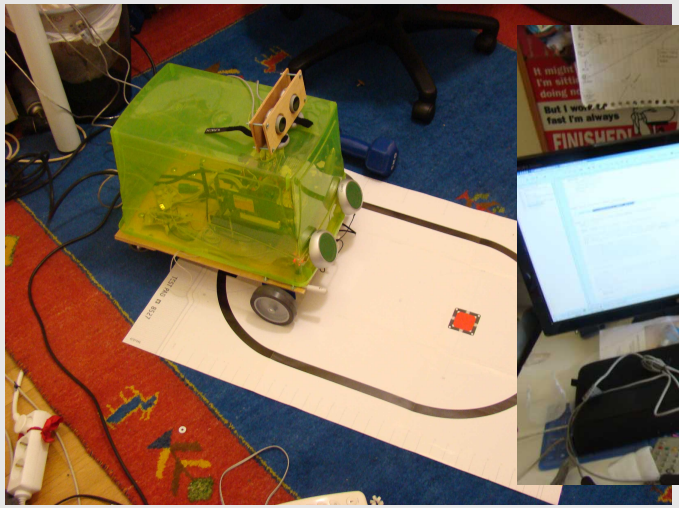


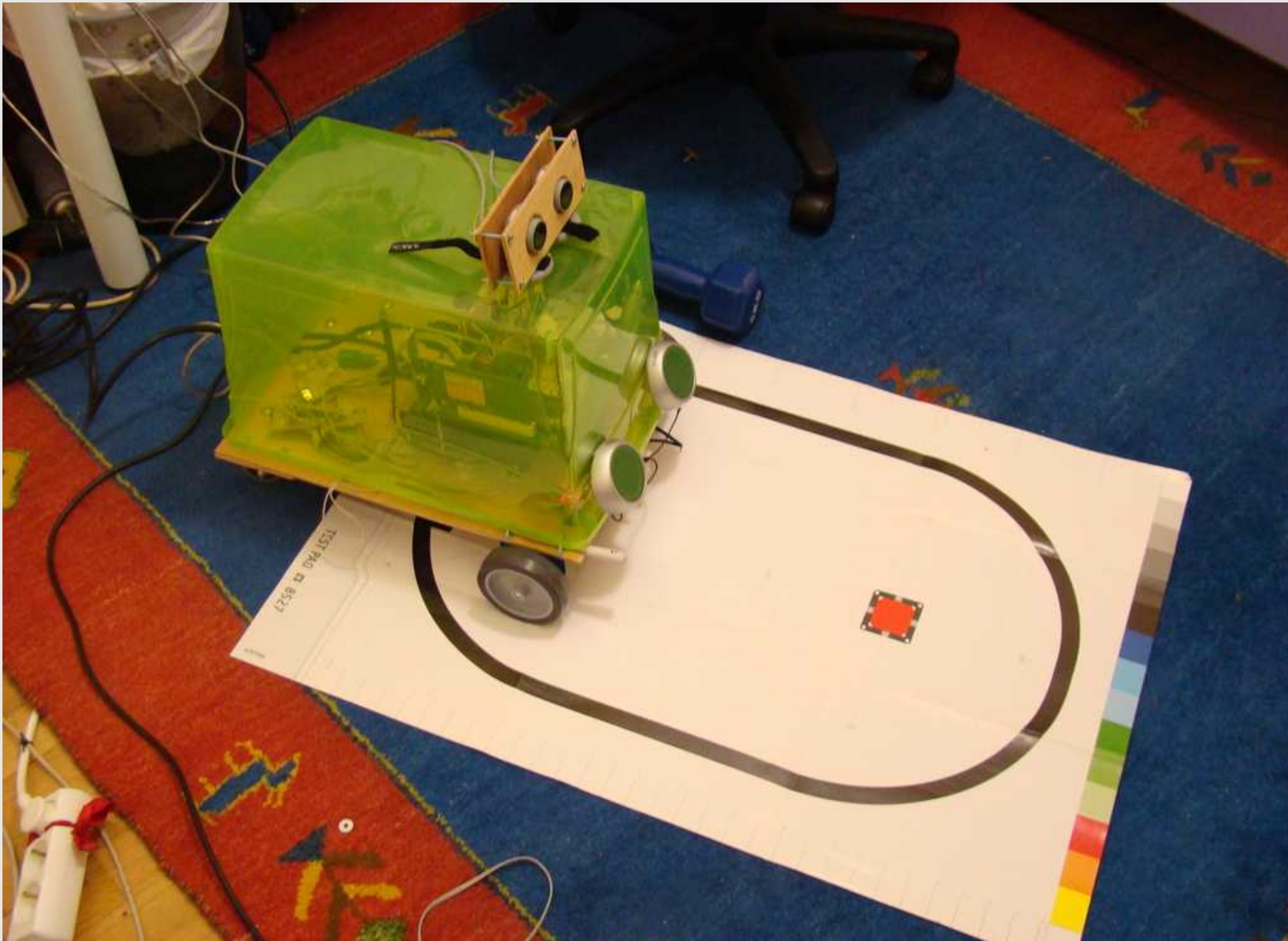
- Βραβεύση στην Athens Digital Week 2008

  με το extra budget απόφαση για remake from scratch όλου του project με πολύ υψηλότερα standards :)

GuarddoG mk4 (building)







Με λίγη φαντασία..

Αλλαγές Hardware -> Αλλαγές Software

Αρχικά η όλη διαστρωμάτωση του project ήταν ένα GUI στο οποίο έτρεχαν τα διάφορα φίλτρα..

Προφανείς αλλαγές αλλάζοντας τους εξωτερικούς μικρο ελεγκτές και για μια portable αρχιτεκτονική :

GUI -> Background Service
Windows -> Linux
DirectX -> V4L2
Mindstorm -> Arduino , MD23

FOSS and Contributions

Μπορείτε να :

- Κατεβάσετε
- Χρησιμοποιήσετε
- Μελετήσετε
- Βελτιώσετε

http://www.github.com/AmmarkoV/RoboVision

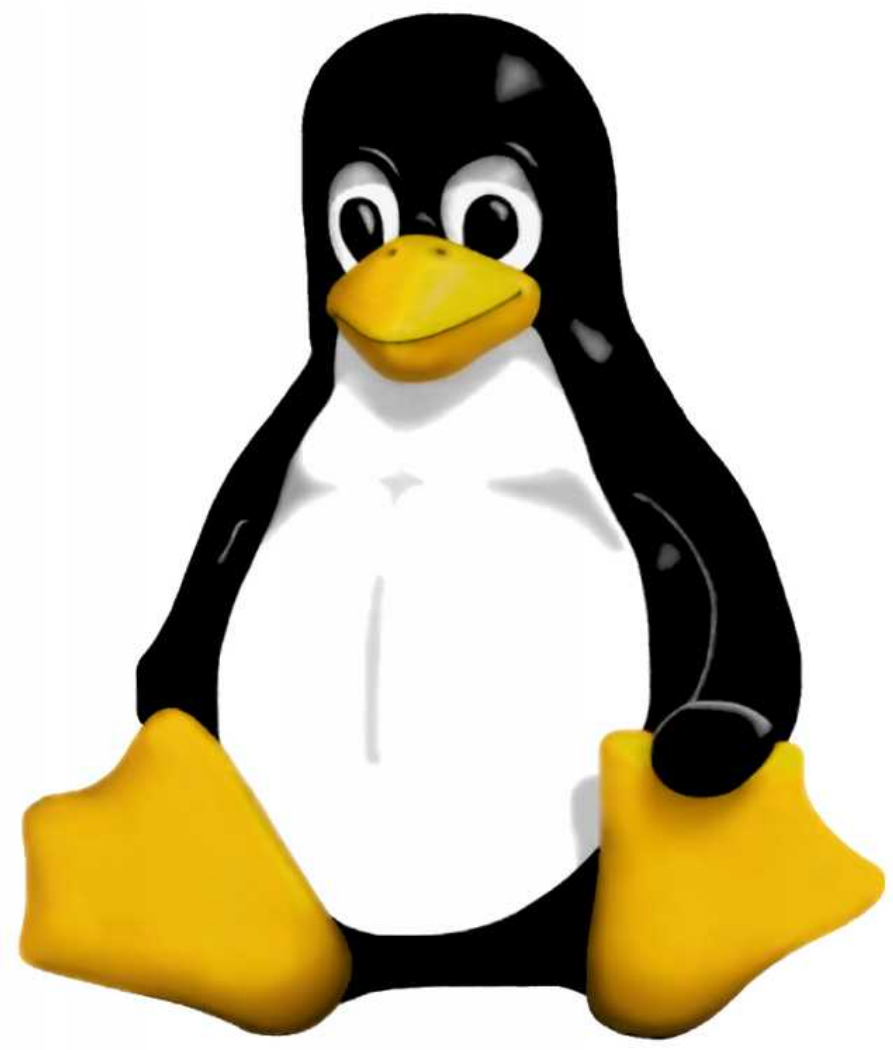
τον κώδικα !

Getting started , Checklist

Για το Vision κομμάτι , χρειάζεται:

- GNU/Linux OS (Debian/Ubuntu apt-get dependency scripts)
- Code::Blocks IDE (για να ανοίγει τα workspaces κτλ)
- 2x V4L2 Compatible Webcams

  (Logitech UVC driver ++)

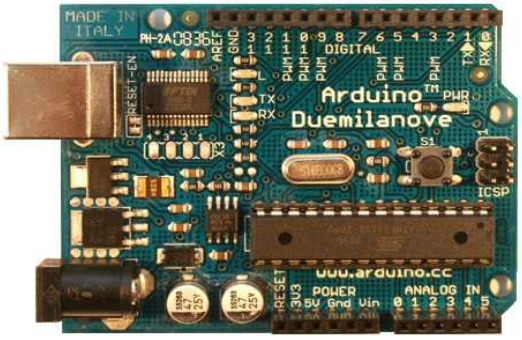


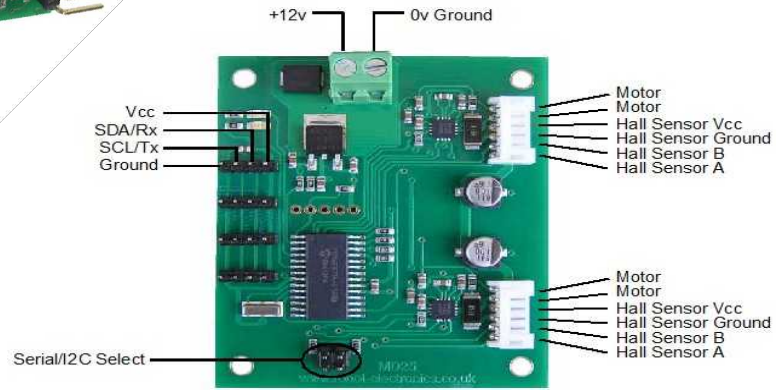
Getting Started , Testing Movement

Για “κίνηση” :

- Arduino Duemillenove
- MD25 Motor Kit
- USB 2 I2C

και άλλες μικρές αγορές ..





Connections , πλήρης κατάλογος κτλ στο documentation του repository (σύντομα..)

Replicating GuarddoG

Ουσιαστικα φτιάχνοντας μια πιθανόν διαφορετική βάση και συνδυάζοντας τα επι μέρους software/hardware κομμάτια (με οποιαδήποτε modifications , την οποία επίσης στο μέλλον ελπίζω να μπορεί να την διανείμω σαν source code ώστε να την παραγγείλει κάποιος με τα CAD σχέδια)

Κάποιος μπορεί να έχει το δικό του GuarddoG!

TODO – Contributions Wishlist

Computer Vision / Linear Algebra

Depth from light (Σκιές / Φώς)
Voxel Matching / Recognition
Object Recognition (από 3d+color data)
3D path planning (physics engine ?)

SLAM efficient implementation

TODO - Contributions Wishlist

etc..

English/Greek STT(Speech to text , πχ Sphinx)
Greek TTS (Text to speech , πχ Festival)
Stereo sound recognition (πχ moo..)

CAD / Plexiglass frame
RVKnowledgebase + NLP - (πχ MIT Openmind)

TODO – Physical things to do

New Plexiglass lasercut , chassis !!!!
Better cameras
etc..

GuarddoG Repository!

http://www.github.com/AmmarkoV/RoboVision

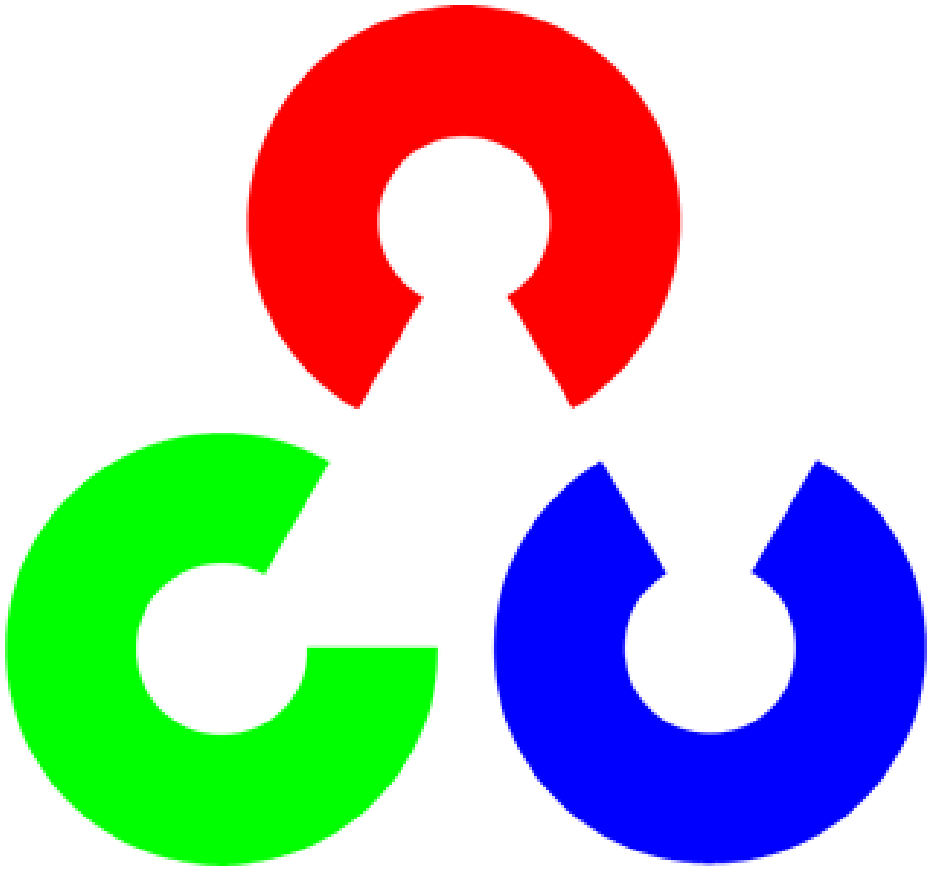
FOSS Aueb !

http://foss.aueb.gr/
http://foss.aueb.gr/irc
Mumble Server : foss.aueb.gr

IRC : irc.freenode.net --> chan #foss-aueb

I am **AmmarkoV**

OpenCV



Πάρα πολλά έτοιμα πράγματα , optimized από την Intel , BSD License , χρησιμοποιείται ανάμεσα σε άλλα για :

* 2D and 3D feature toolkits
* Egomotion estimation
* Facial recognition system
* Gesture recognition
* Human-Computer Interface (HCI)
* Mobile robotics
* Motion understanding
* Object Identification
* Segmentation and Recognition
* Stereopsis Stereo vision: depth perception from 2 cameras
* Structure from motion (SFM)
* Motion tracking

AR Toolkit

* Single camera position/orientation tracking.

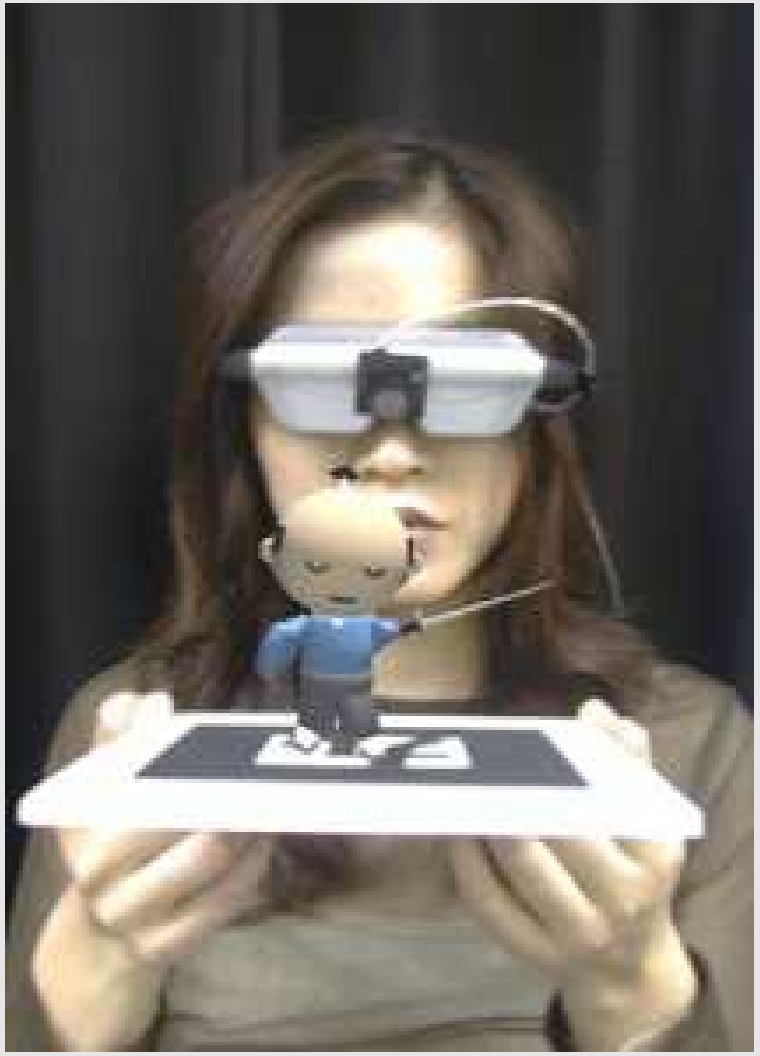
* Tracking code that uses simple black squares.

* The ability to use any square marker patterns.

* Easy camera calibration code.

* Fast enough for real time AR applications.

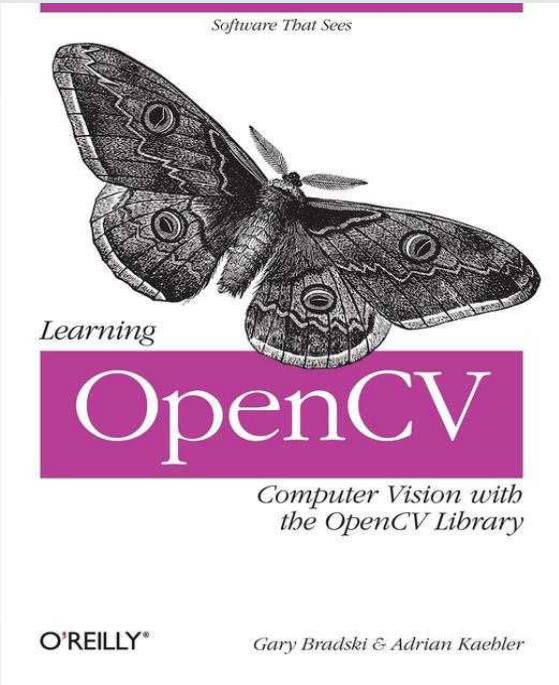
* Free and open source.

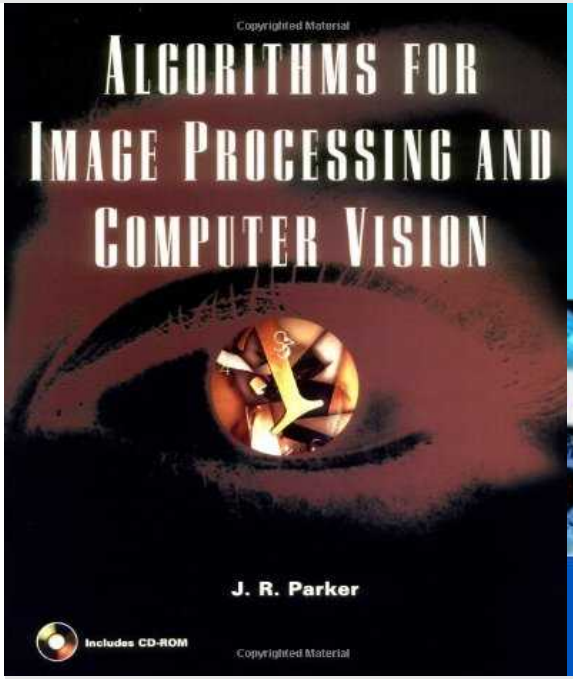


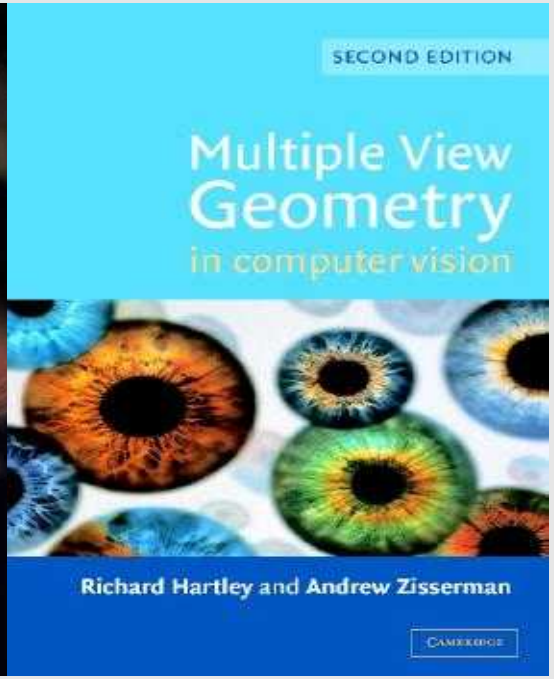
OpenSURF

- GPL v3
- Several times faster than SIFT
- Easy to use
- Robust
- It Works!

Suggested reading
for computer vision

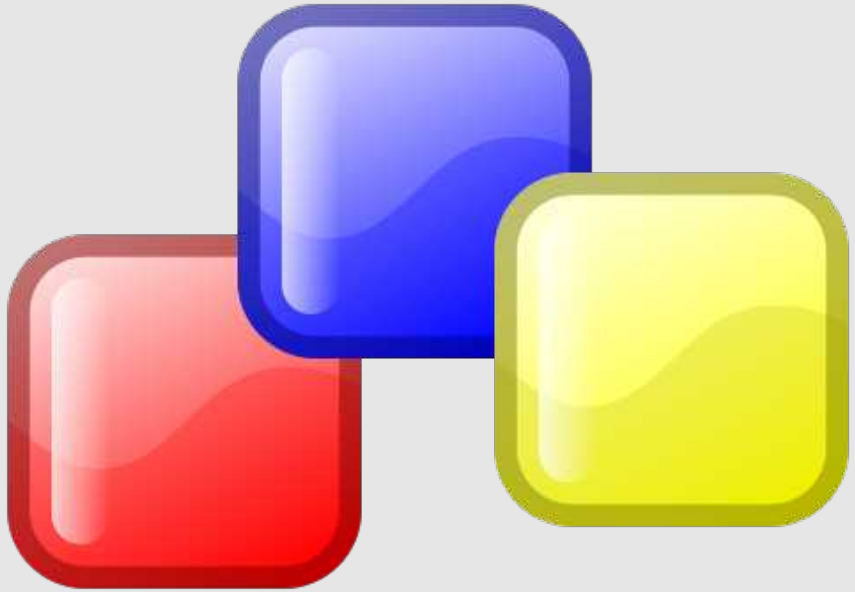




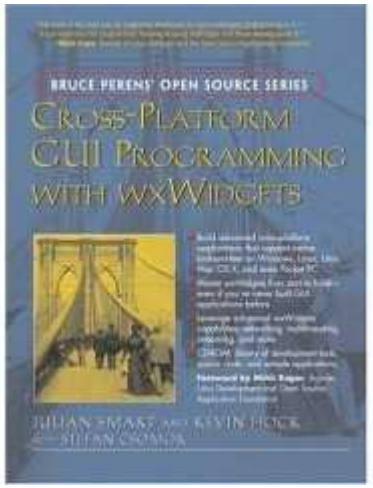


WxWidgets

- Crossplatform
- Native Controls
- Easy
- Object Oriented in a good way :)
- Πολλές παρατρεχάμενες libs



Suggested Reading για GUIs




That's all Folks!

but..
I ' LL BE BACK!