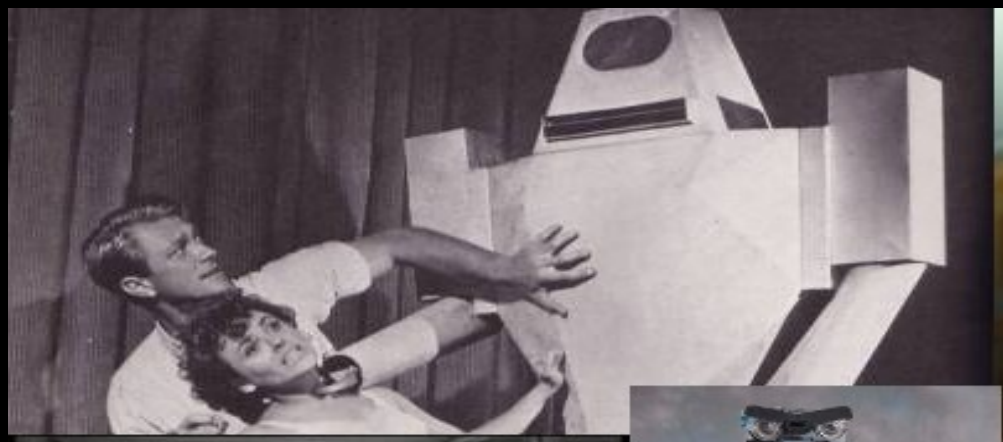
Guard Dog Project

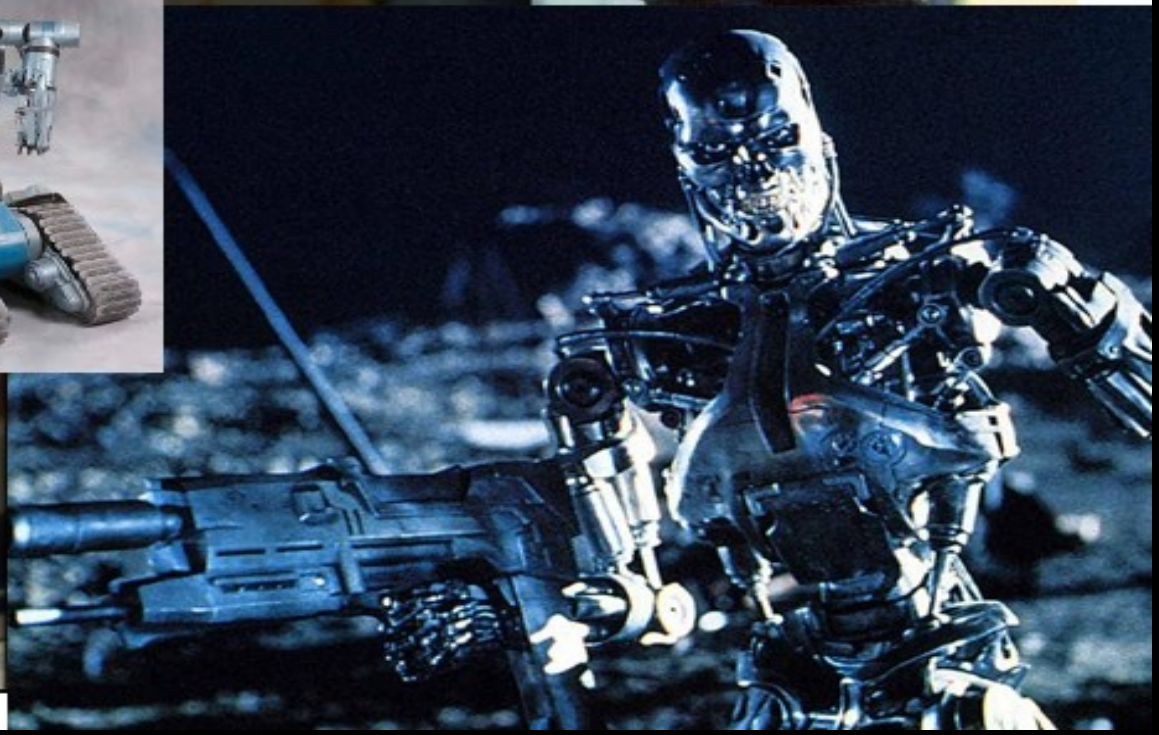
Πολύ σύντομη ιστορική αναδρομή

- 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

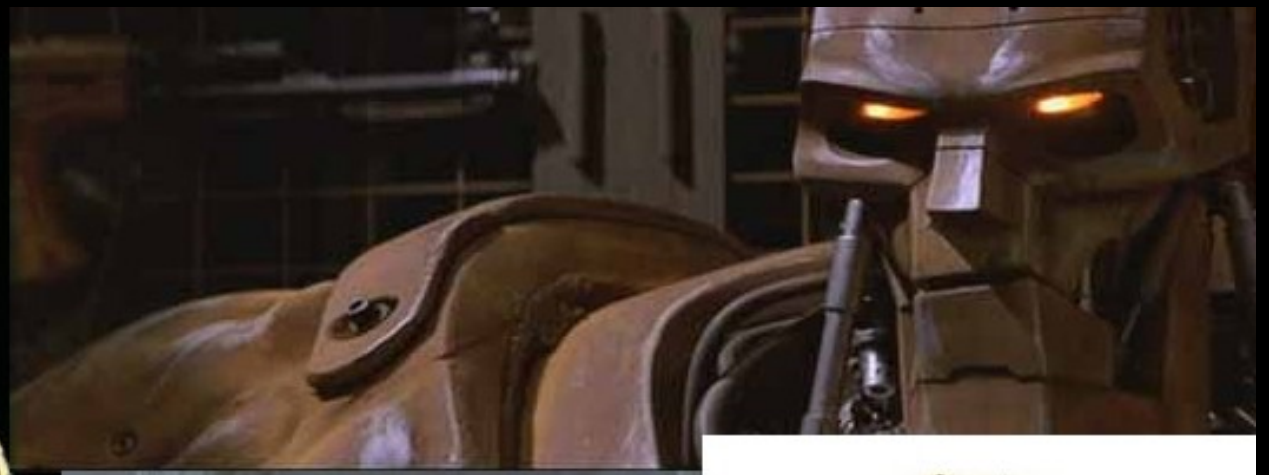
Τι είναι ένα ρομπότ ?

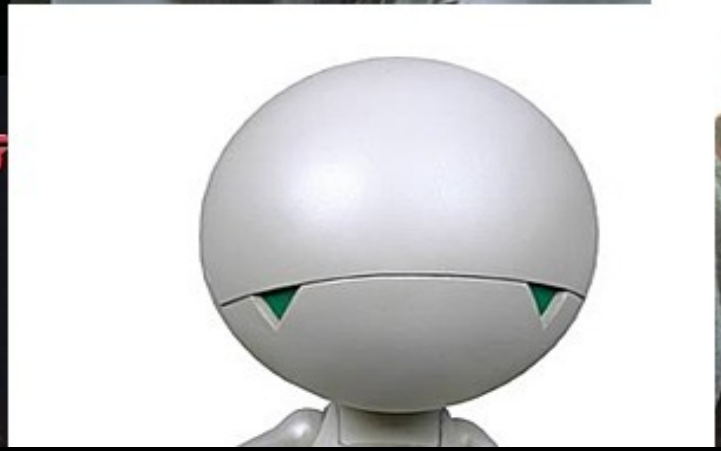
Hollywood says



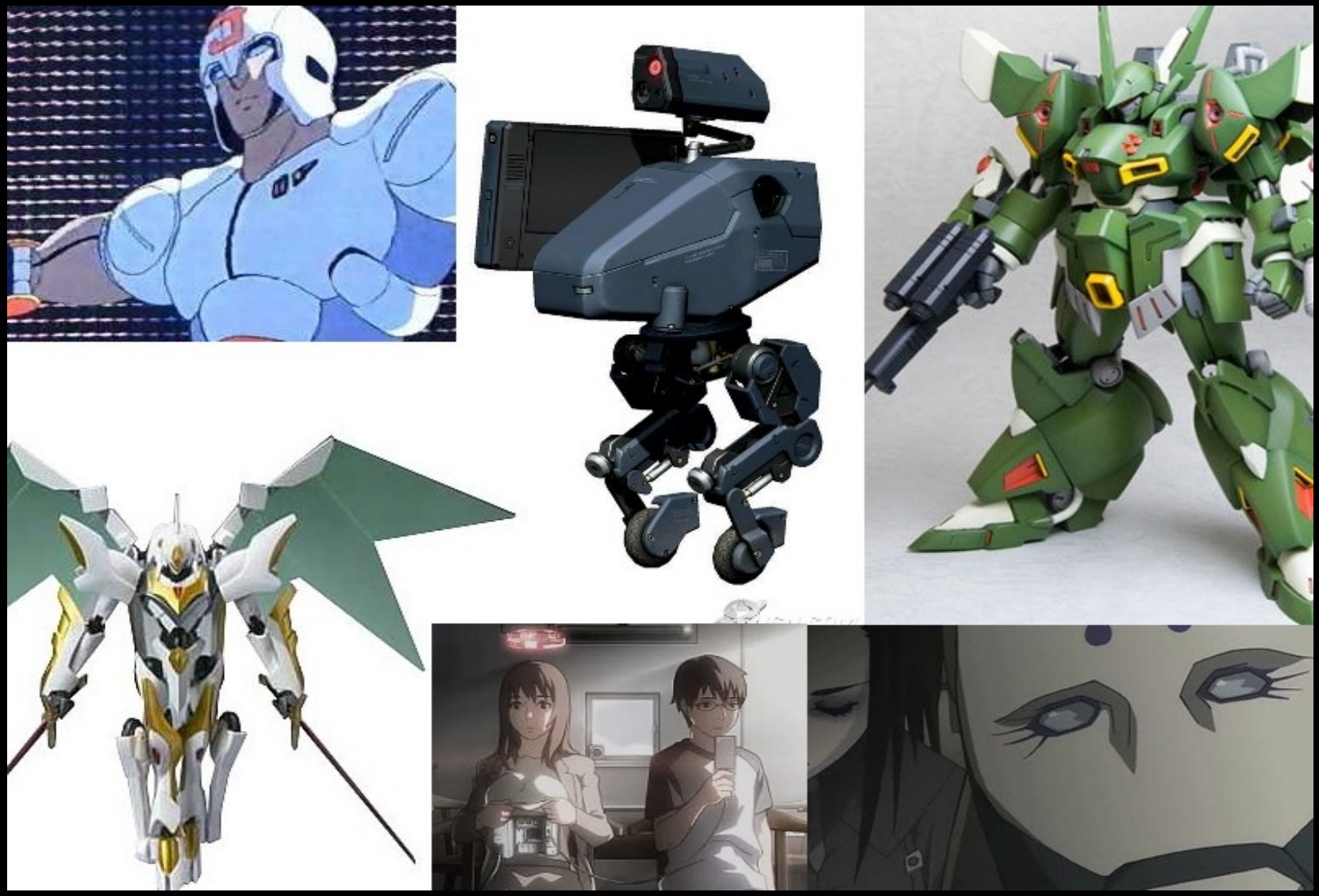


Hollywood says





Japan says

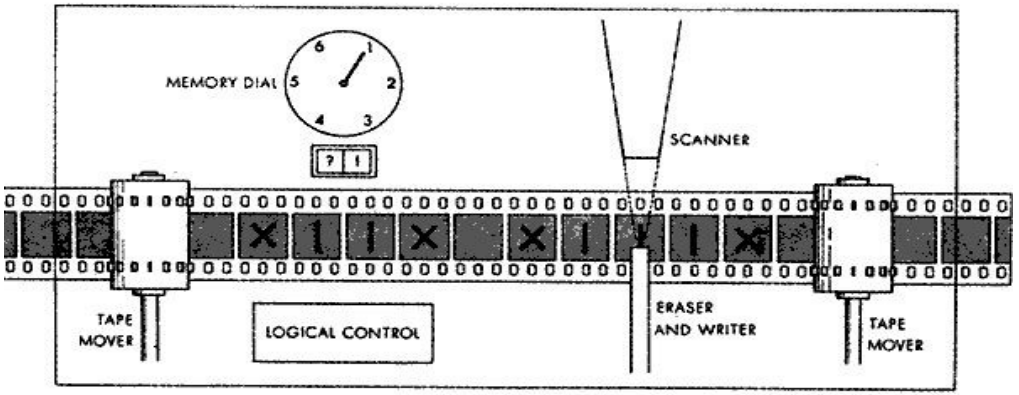


Ένα ρομπότ είναι..

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..

Η “ταινία γεμίζει” με χαρακτήρες από το περιβάλλον , μέσω των περιφερειακών και γράφοντας σε κάποιες θέσεις της ταινίας το μηχάνημα “αλληλεπιδρά”



Ένα ρομπότ δεν είναι..

- Κάτι εξωπραγματικό
- Πολύ δύσκολο στην κατασκευή

  (πριν 100 χρόνια ήταν)

  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..

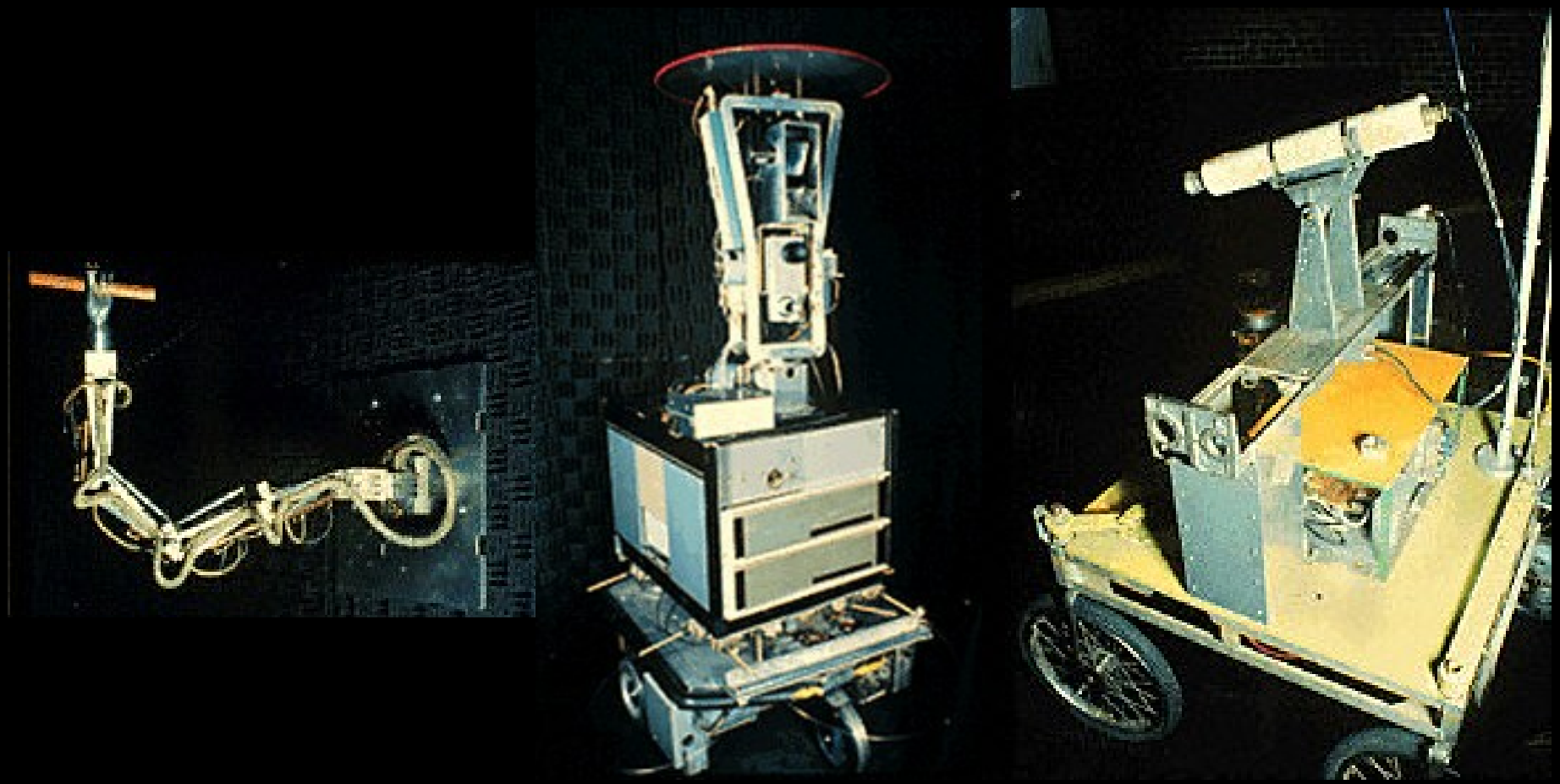
  Τηλεκατευθυνόμενο != Robot

Ένα ρομπότ δεν είναι κάτι καινούργιο

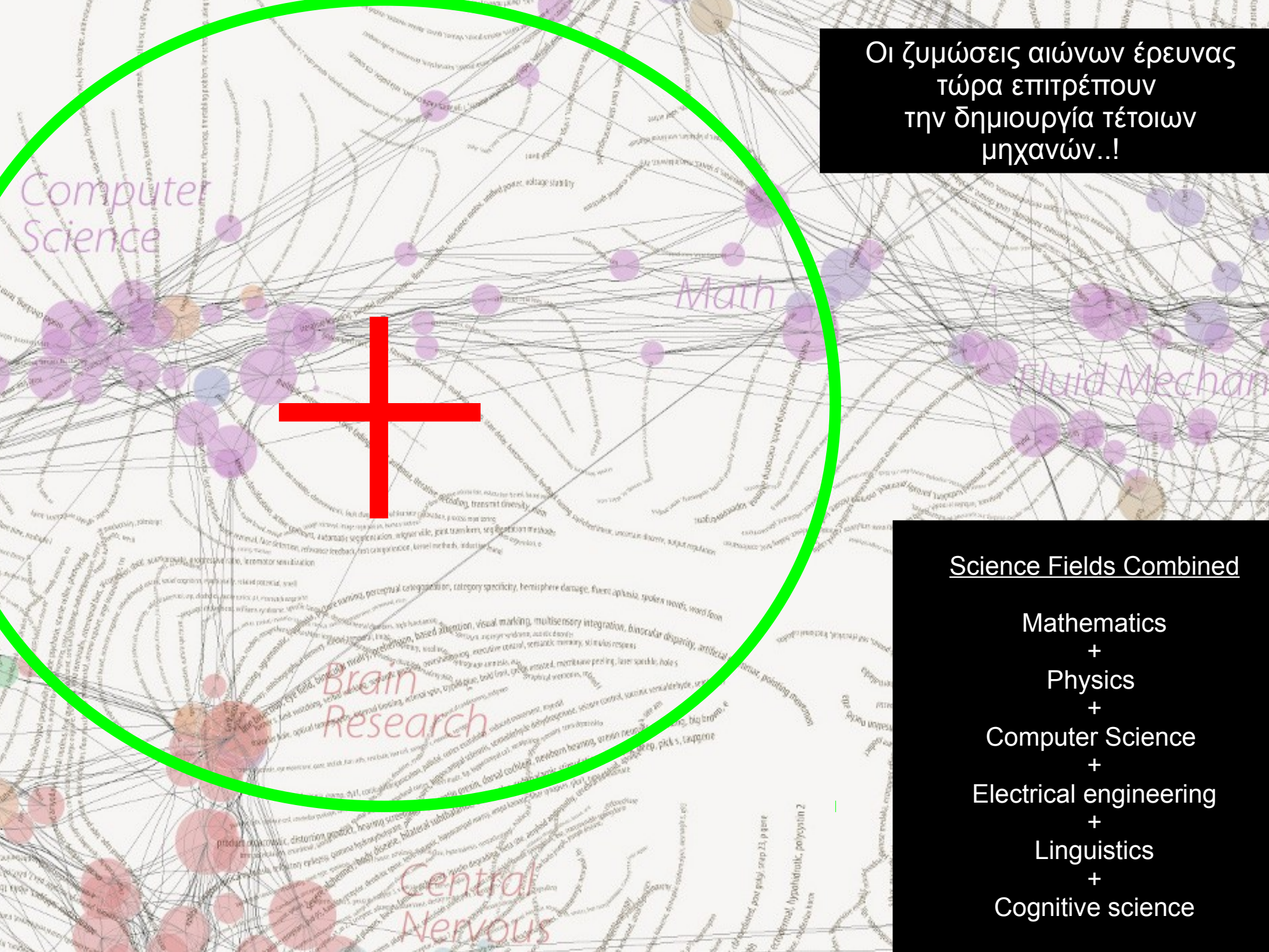
Karakuri ningyō (からくり人形?) are mechanized puppets or automata from Japan from the **17th century to 19th century**. The word karakuri means "mechanisms" or "trick". In Japanese ningyō is written as two separate characters, meaning person and shape. It may be translated as puppet, but also by doll or effigy.[1] The dolls' gestures provided a form of entertainment.

Three main types of karakuri exist. Butai karakuri (舞台からくり stage karakuri?) were used in theatre. Zashiki karakuri (座敷からくり tatami room karakuri?) were small and used in homes. Dashi karakuri (山車からくり festival car karakuri?) were used in religious festivals, where the puppets were used to perform reenactments of traditional myths and legends.

Ένα ρομπότ δεν είναι κάτι καινούργιο



Stanford A.I. Lab
1962 – 1970 – 1979
50 χρόνια πρίν



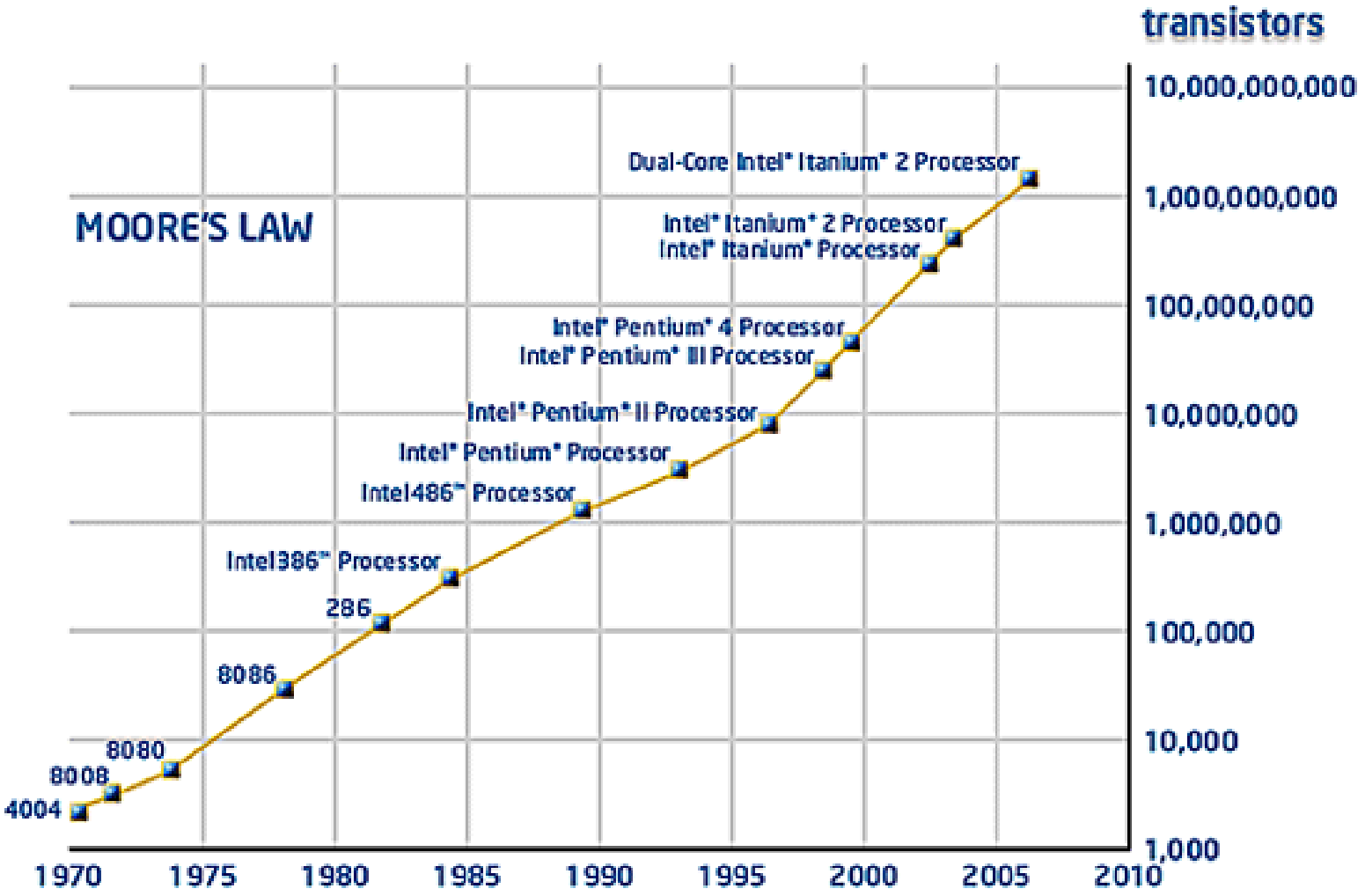
Οι ζυμώσεις αιώνων έρευνας
τώρα επιτρέπουν
την δημιουργία τέτοιων
μηχανών..!

Science Fields Combined

Mathematics
+
Physics
+
Computer Science
+
Electrical engineering
+
Linguistics
+
Cognitive science

Ένα ρομπότ είναι κάτι “σχεδόν” εφικτό

με μηδαμινό κόστος σε σχέση με το παρελθόν χάρη στην εκθετική βελτίωση της τεχνολογίας



Σκοπός της παρουσίασης

- Να δείτε πώς περίπου διαστρωματώνεται ένα ρομπότ
- Να μάθετε λίγα πράγματα για το GuarddoG
- Να διαπιστώσετε (καθότι computer scientists) οτι η πληροφορική είναι βασικό domain για ρομποτική και για embedded hardware
- Να κάνετε fork το guarddog!
- Να προλάβω να κάνω σε 30 λεπτά την παρουσίαση!

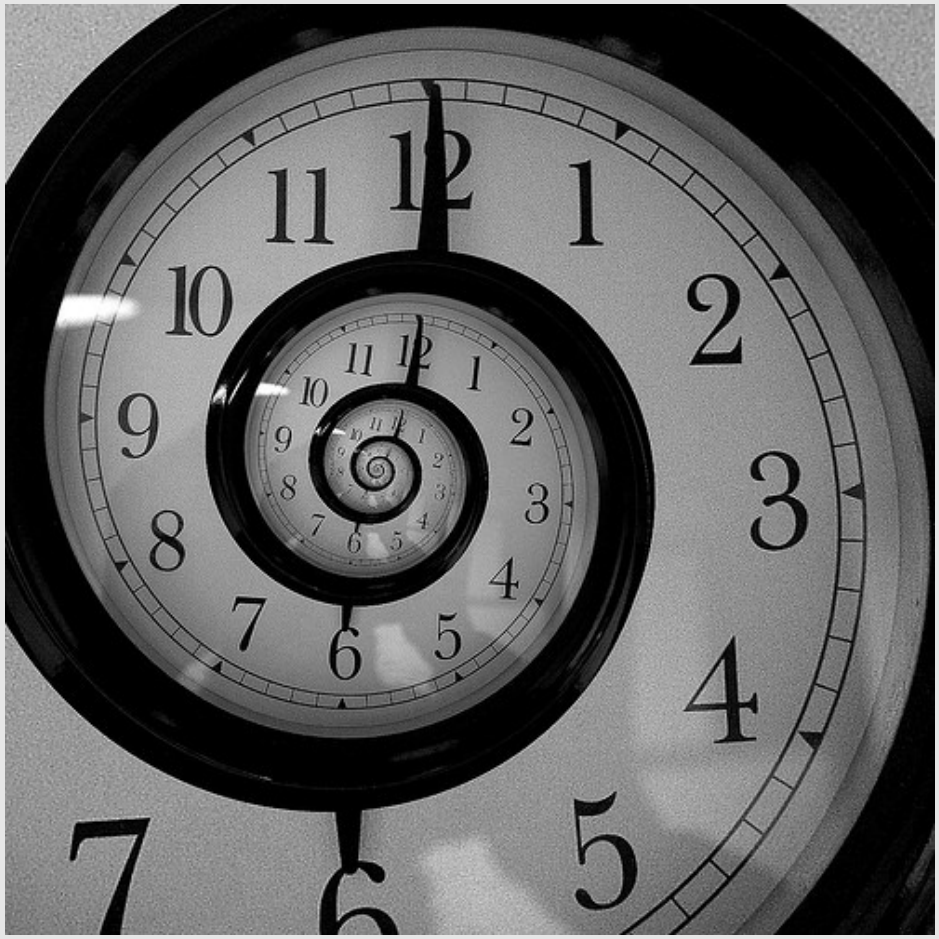
Το να μάθουμε είναι ο σκοπός του event!

- Υπάρχει πολύς χρόνος για ερωτήσεις
- Με τις ερωτήσεις μαθαίνουμε (και εσείς και εγώ) !
- Για όποια απορία μετά το event μπορείτε να με βρείτε online στα links που θα παραθέσω στο τέλος..

Μεγάλο Project
=
Μεγάλη παρουσίαση (ελπίζω ενδιαφέρουσα)

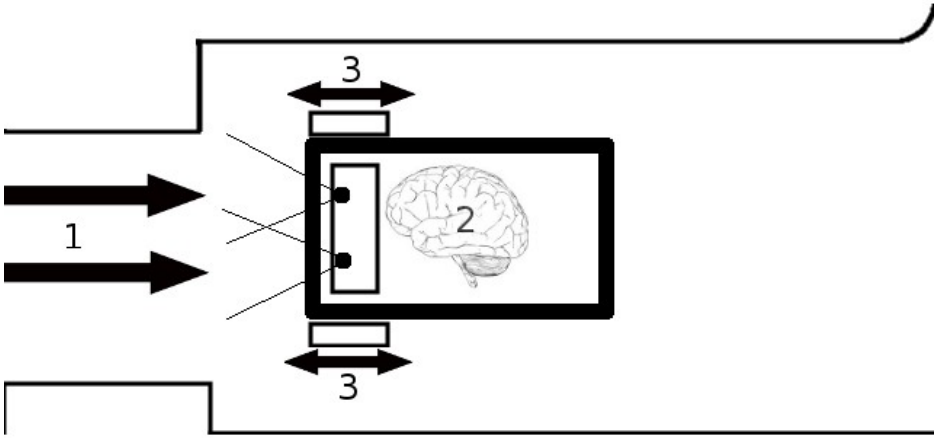
- 2 κομμάτια software / hardware ..
- Βασικοί αλγόριθμοι
- Διαστρωμάτωση
- Ερωτήσεις/συζήτηση, guarddog github repo

  φτιάξτε το δικό σας ! :)



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.



- Οι υπολογισμοί πηγαίνουν

1 → 2 → 3
←

Chicken and egg , χαοτικό φαινόμενο
Η θέση καθορίζει την κίνηση ή η κίνηση την θέση?

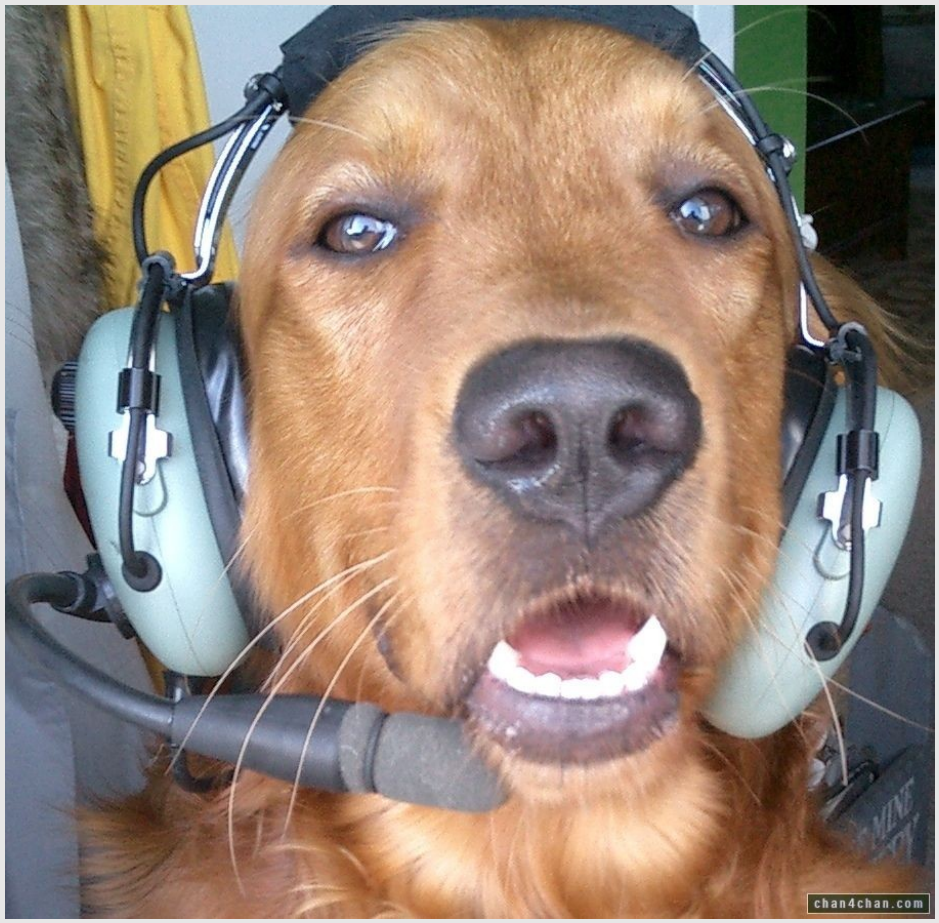
Προφανώς όχι πραγματικό Guard “Dog”

- 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..



Επίσης δεν είναι εντελώς έτοιμο..

- Θέλει αρκετή δουλειά ακόμα !
- Το παρόν που βλέπετε το έχω πάρει όπως είναι από το “εργαστήριο” μου ..
- Δεν είναι προς το παρόν (και μάλλον και το μέλλον) εμπορικώς implemented και plug and play σαν το Rovio πχ..
- Αλλά έχω κάνει και πάρα πολύ δουλειά ήδη..
- Τα standards είναι πολύ υψηλά για να θεωρηθεί τελειωμένο..
- Κυρίως δουλεύω τους αλγορίθμους με datasets οπότε το physical build δεν κάνει κάτι πολύ εντυπωσιακό .. Το “εντυπωσιακό” είναι το software και είναι αόρατο..

Παρότι έχω δαπανήσει πολύ χρόνο..

2008

2009

2010

2011

Πολύ μεγάλο Bus Factor

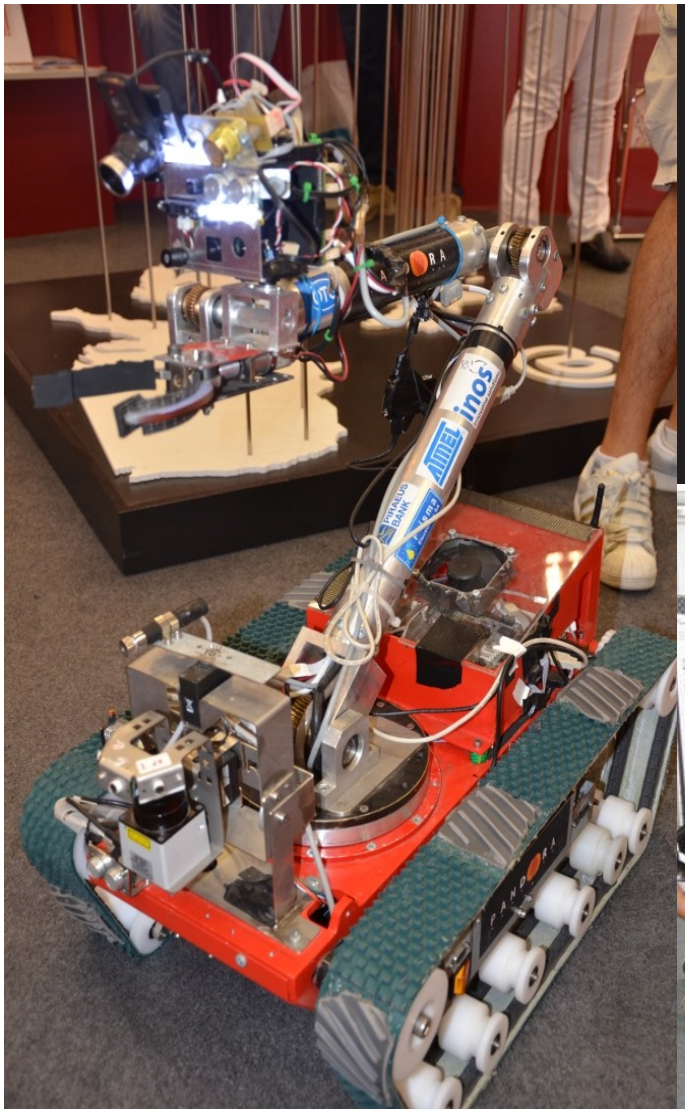
1 developer

In software development, a software project's bus factor is a measurement of the concentration of information in individual team members. The bus factor is the total number of key developers who would need to be incapacitated (as by getting hit by a bus) to send the project into such disarray that it would not be able to proceed; the project would retain information (such as source code) with which no remaining team member is familiar. A high bus factor means that many developers would need to be removed before the project would necessarily fail.

"Getting hit by a bus" could take many different forms. This could be a person taking a new job, having a baby, changing their lifestyle or life status, or literally getting hit by a bus: the effect would be the same. The term was commonplace in business management by 1998, was used in mental health in the same year,[1] was seen in software engineering papers in Association for Computing Machinery and Information Systems Frontiers by 1999, and the term "Bus Factor" was used in engineering by 2003.

Pandora Project

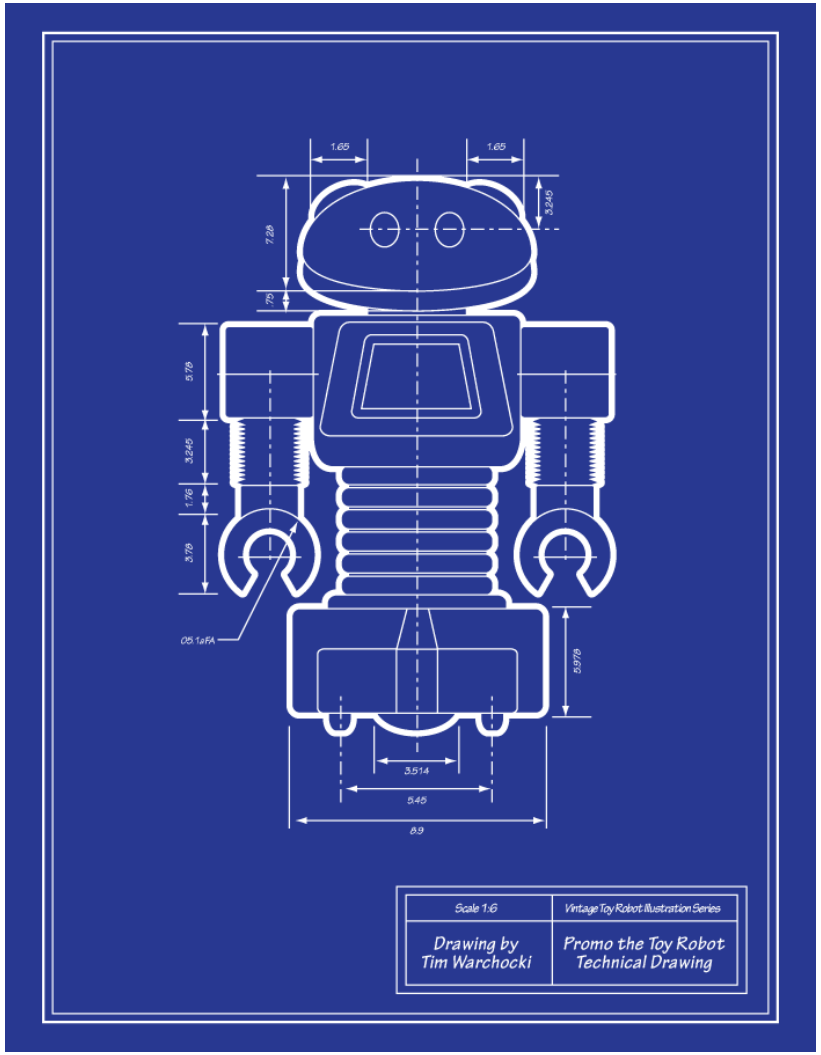
https://github.com/pandora-auth-ros-pkg/pandora-auth-ros-pkg



Αντίστοιχο Project του Α.Π.Θ. βασισμένο σε ROS

* 16 άτομα
* 25K– 30K € cost
* Το guarddog κοστίζει όλοκληρο περίπου
οσο το battery pack
του Pandora , το οποίο είχε πάρει και
φωτιά από βραχυκύκλωμα οπότε το άλλαξαν

Πως ξεκινάει κανείς να φτιάξει κάτι τέτοιο?



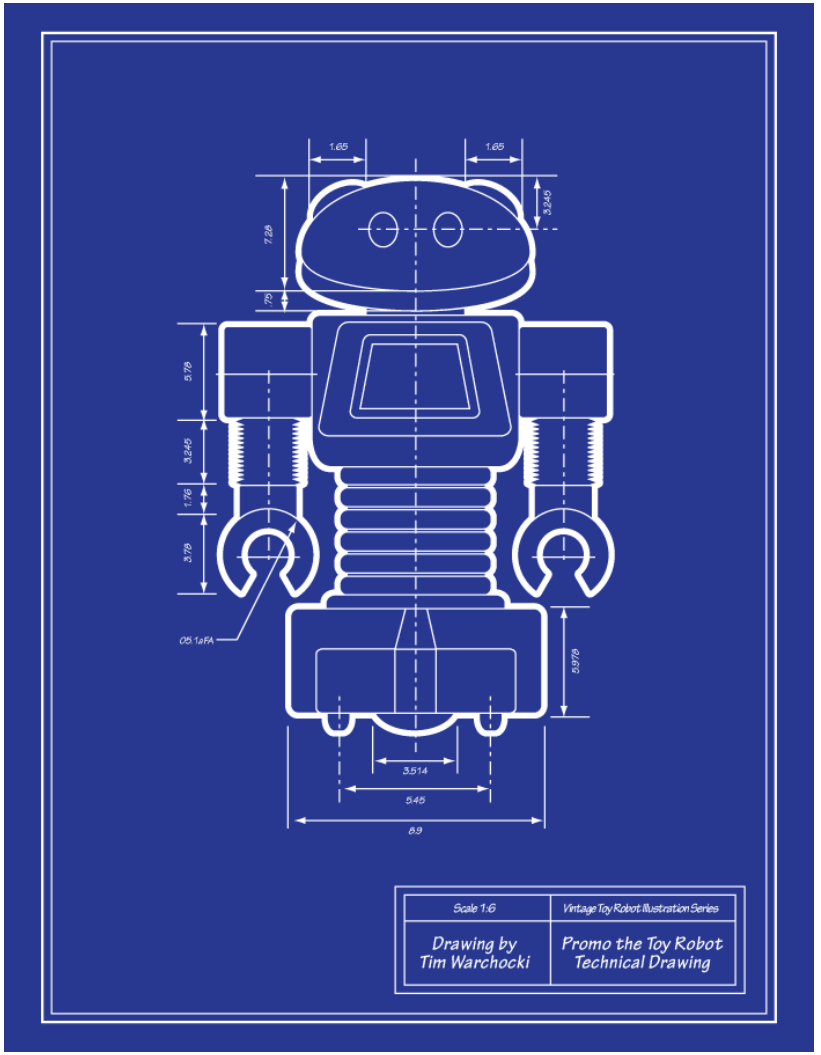
Στόχος : A robotics platform that can act as a guard , traverse a known path and fend off intruders. In case of a security breach it would signal the alarm and begin to follow the perpetrator and after a set distance would resume its previous path.

Ένας υπολογιστής που θα πρέπει να :

- Βλέπει , εμπόδια , πρόσωπα , χώρους
- Αντιλαμβάνεται πρόσωπα
- Κινείται ανεξάρτητα
- Χαρτογραφεί

Πως ξεκινάει κανείς να φτιάξει κάτι τέτοιο?

- Βλέπει -> είσοδος από κάμερες
- Αντίληψη βάθους -> χρήση στερεοσκοπίας αρα 2 πηγών εικόνας
- Αντίληψη προσώπων -> pattern recognition
- Κινείται ανεξάρτητα -> λειτουργία με μπαταρίες , low power consumption , custom body
- Χαμηλό κόστος , υλοποίηση από υλικά μαζικής παραγωγής



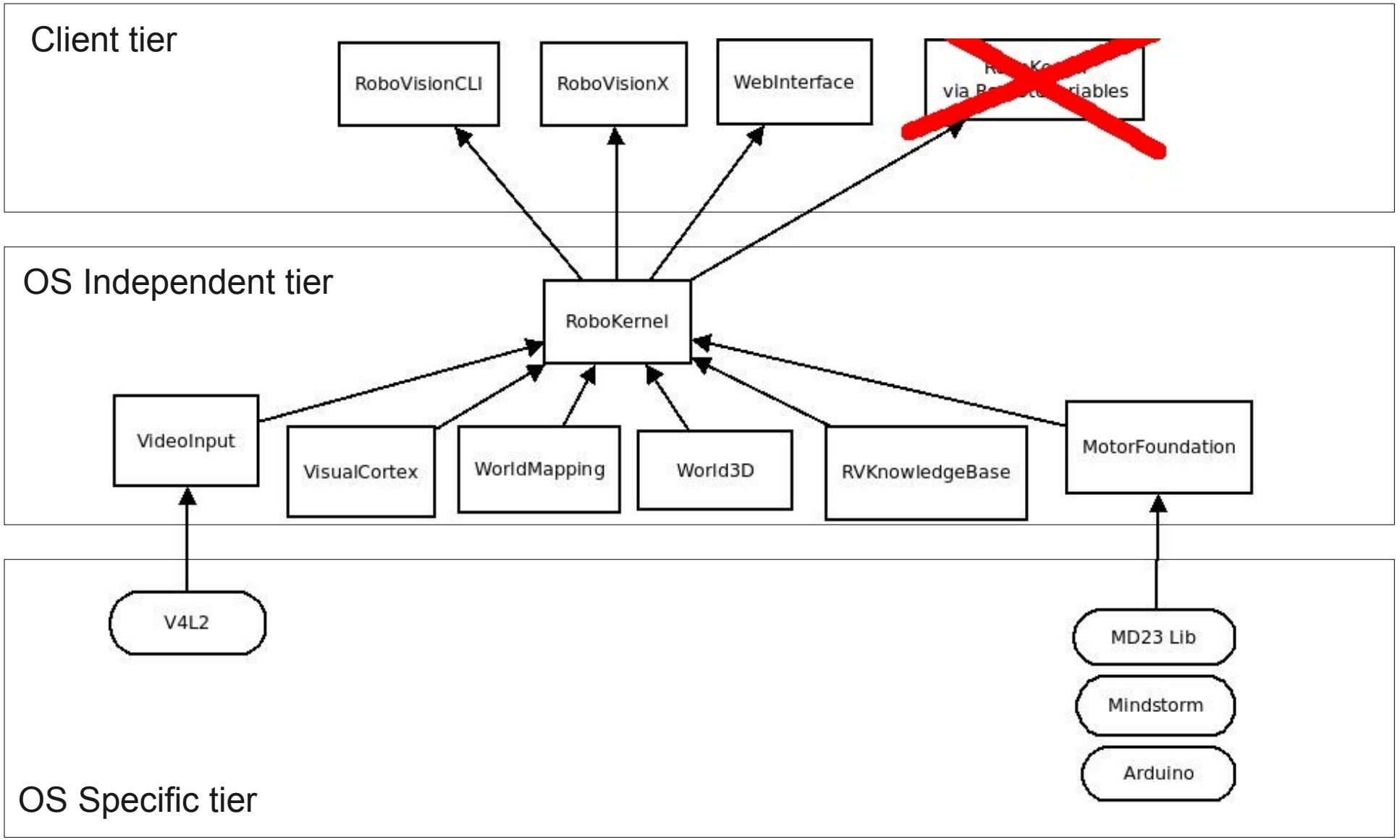
Μια αντίστροφη Κάρτα Γραφικών

Τα κομμάτια του Hardware

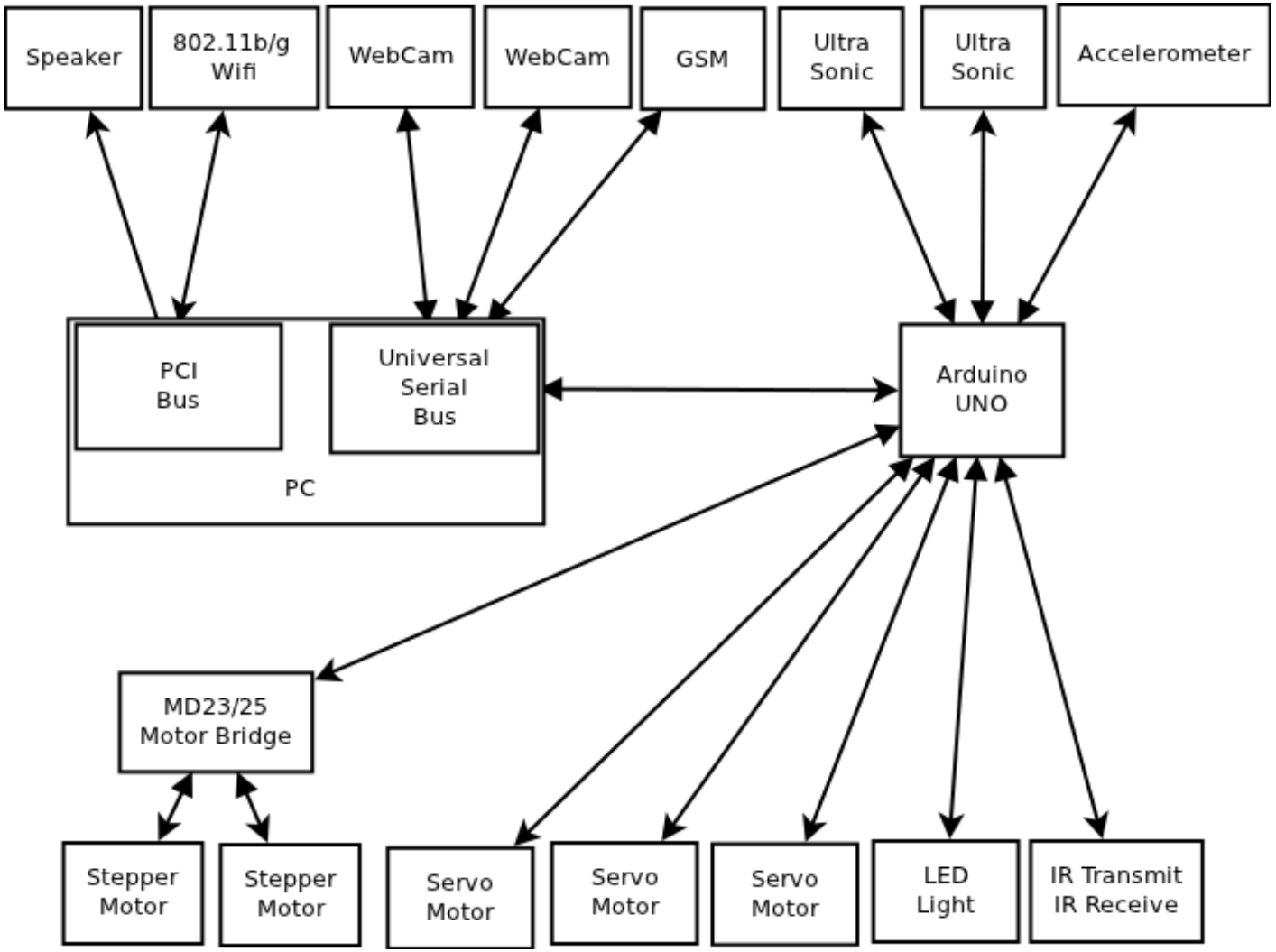
ένα PC με ρόδες

- 1.2Ghz Celeron
- Mini-ATX Motherboard
- 802.11 b/g WIFI
- 2 x Webcams
- 2 x Microphones
- 1 x Arduino
- 1 x RD01/02 kit , 2x Servo
- 2x Ultrasonics
- Dual Axis accelerometer

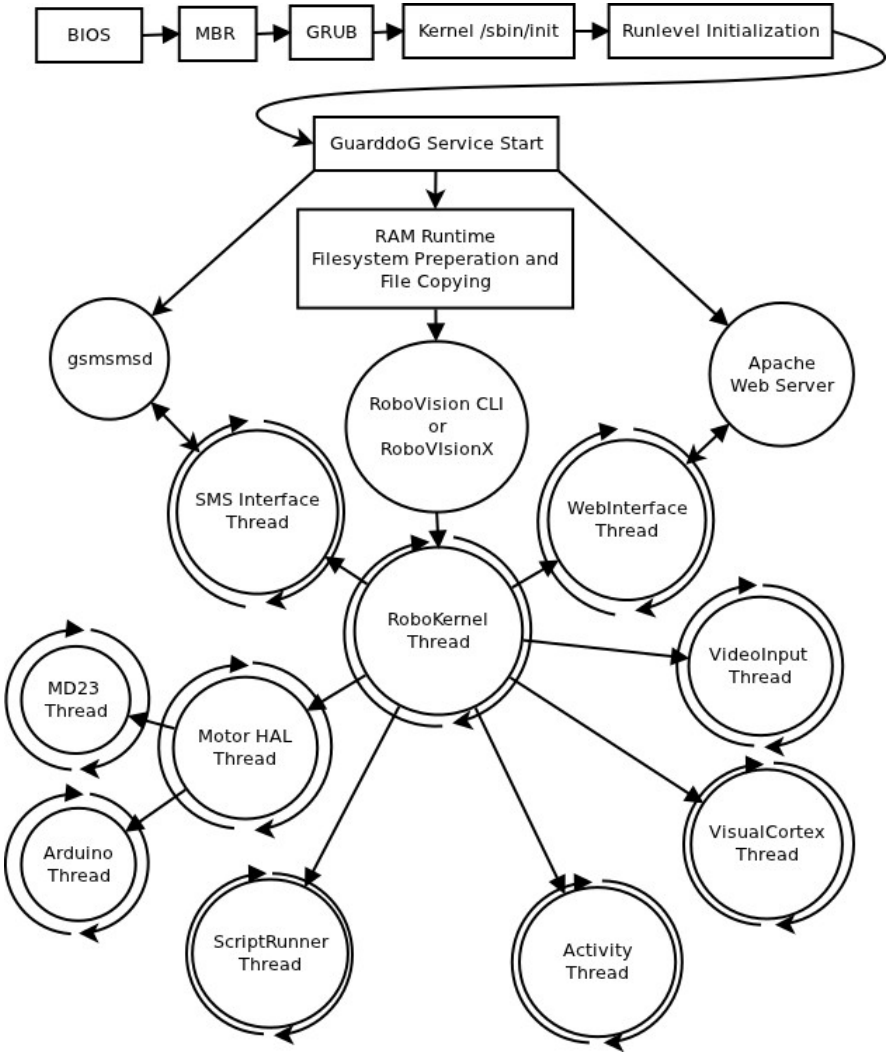
Αρχιτεκτονική του Software



Connection Diagram



GuarddoG Startup!



Το κάθε module έχει σαφώς καθορισμένη λειτουργία

Το κάθε ένα από τα modules εκτελεί μια σχετικά απλή ξεχωριστή λειτουργία , με το δικό του tester και lib , όλα μαζί κάνουν όμως κάτι πολύ πιο περίπλοκο

Video Input
Visual Cortex
World Mapping
RVKnowledge base
Motor HAL

RoboKernel

RoboVisionX
RoboVisionCLI

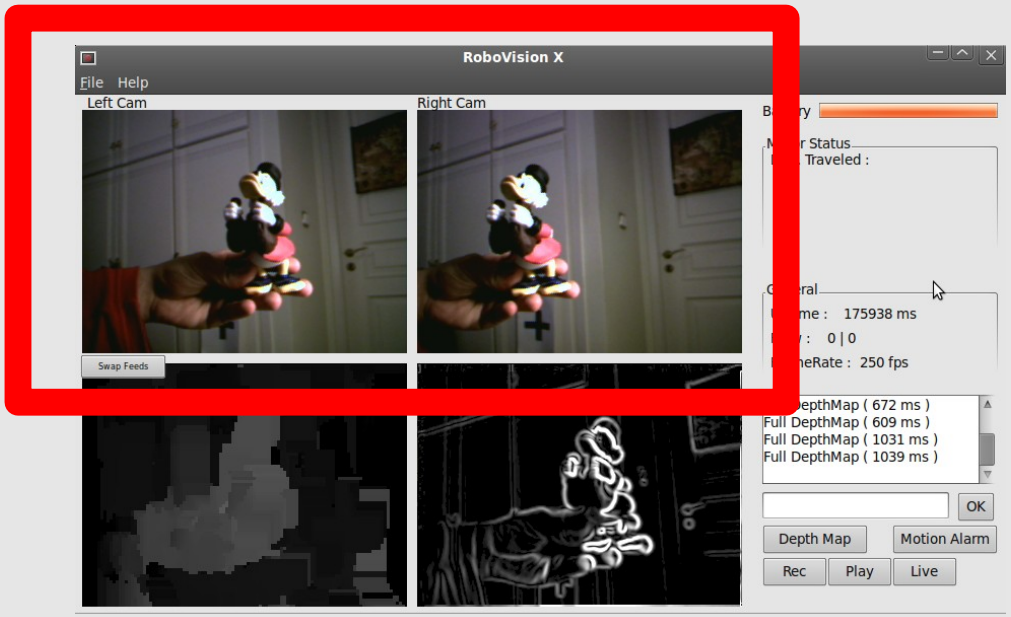
Όλο το project είναι γραμμένο σε C (το GUI σε C++) και είναι statically linked για λόγους απόδοσης ..

Πρόβλημα #0

Έχουμε 2 κάμερες (που βγάζουν δισδιάστατη εικόνα) και θέλουμε να σχηματίσουμε μια τρισδιάστατη αναπαράσταση του χώρου

Πρώτα απο όλα θα πρέπει να μπορούμε να λάβουμε input από τις κάμερες!

Video Input

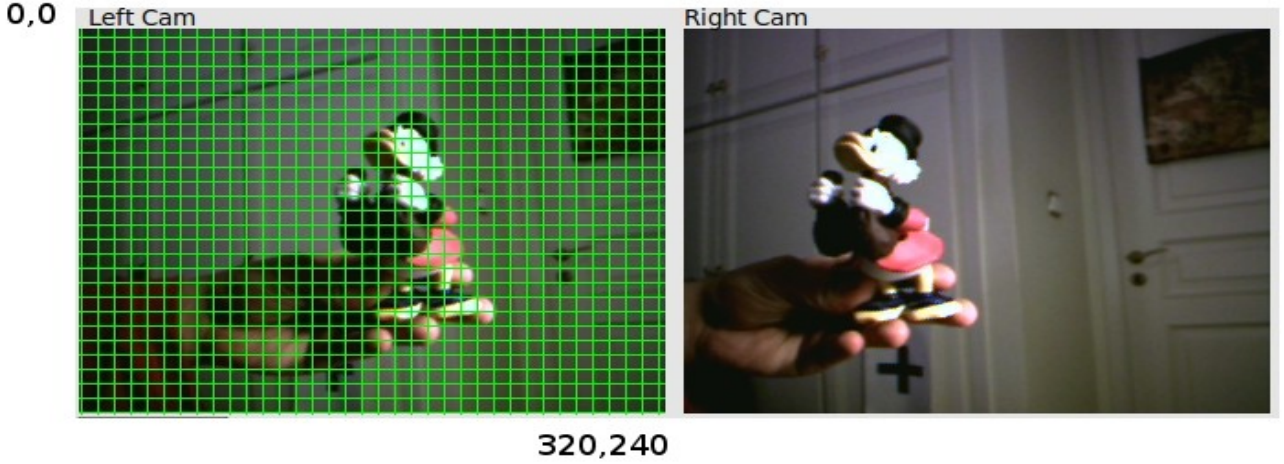


Αναλαμβάνει να μεταφέρει arrays με την εικόνα που βλέπουν οι 2 webcams

Σαν βιβλιοθήκη μπορεί κάποιος να το χρησιμοποιήσει για οποιοδήποτε project

Video Input

Πως αναπαριστάται μια εικόνα?

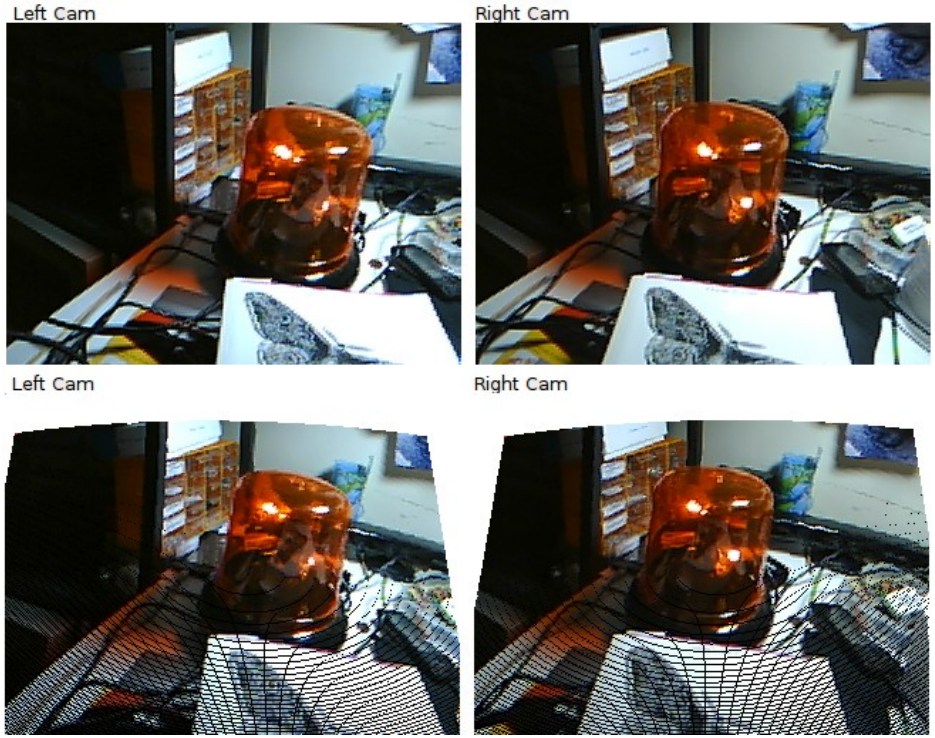


R	G	B	R	G	B
224	245	100	81	216	229
249	10	152	238	77	206
150	111	70	184	123	244
85	136	107	14	9	100
237	249	255	235	41	142
205	142	176	71	145	224
38	255	201	205	133	178
239	92	149	217	9	23
56	201	32	54	209	154
195	145	96	85	21	190
187	202	222	212	244	177
117	102	109	89	15	167
200	249	117	137	30	190
191	54	168	70	107	109
215	57	31	143	65	4
88	116	83	101	179	122

Ένα array από 320 * 240 * 3 bytes = 225KB

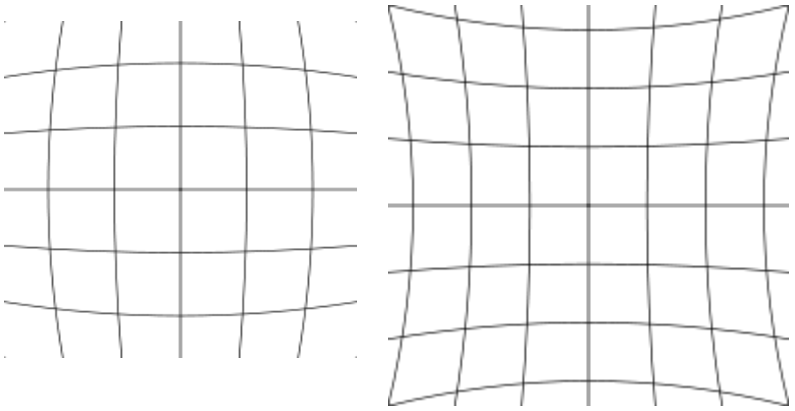
Video Input

- Η εικόνα που πήραμε αλλοιώνει την πραγματικότητα λόγω εργοστασιακών σφαλμάτων της κάθε κάμερας!



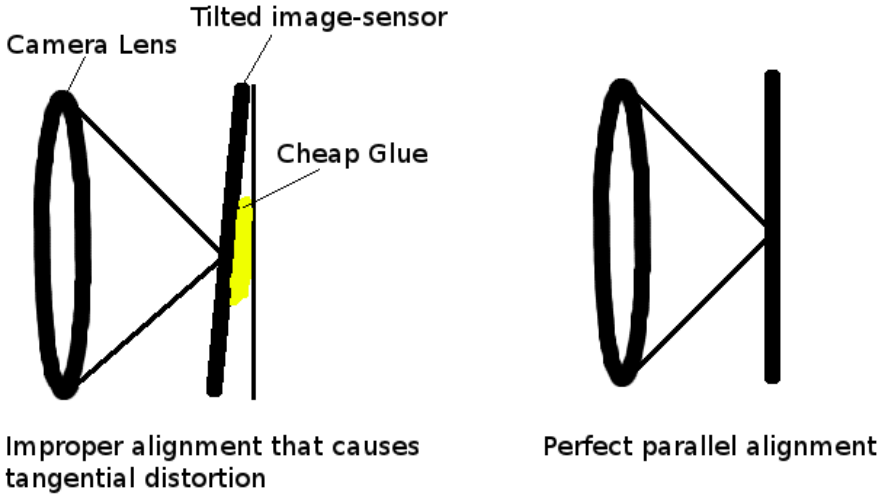
Camera distortions

Radial Distortions (Lens)



Barrel Distortion **Tangential Distortion**

Tangential Distortions (Assembly)

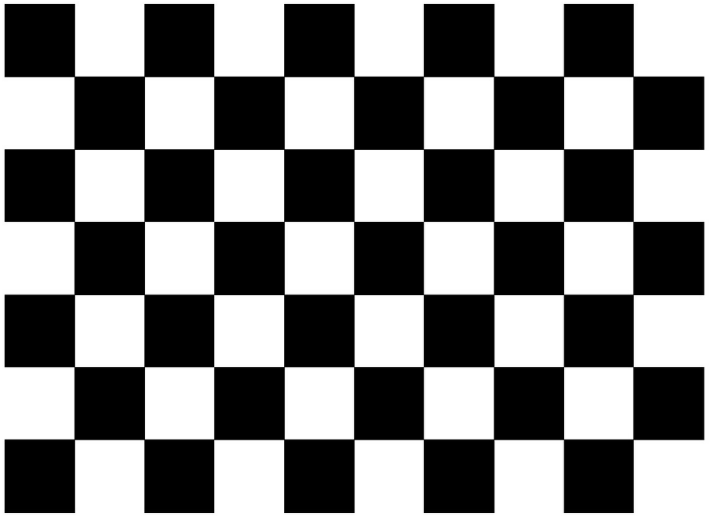


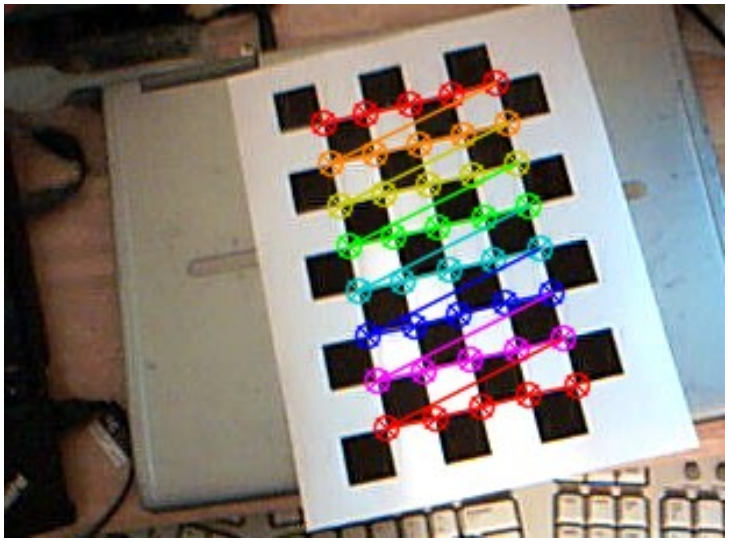
Improper alignment that causes tangential distortion

Perfect parallel alignment

- Δυο είδη παραμόρφωσης τα οποία συνδυάζονται..!
- Για να τα καταπολεμήσουμε χρησιμοποιούμε γνωστά σχήματα και παρατηρούμε πως παραμορφώνονται σαν είδωλα από τις κάμερες.

Image Rectification





Η μέθοδος που χρησιμοποιείται (Zhang / Sturm , συνήθης σε εφαρμογές με OpenCV) προϋποθέτει ένα εκτυπωμένο grid από τετράγωνα γνωστού μεγέθους και πλήθους. (στην εικόνα 10x7)

Οι ακμές εντοπίζονται και μετράται η απόκλιση ανάλογα με την απόσταση από το κέντρο της κάμερας (το οποίο δεν παραμορφώνεται)

Στην διαδικασία δίνουμε :
- skew coefficient (γ) usually zero
- principle point or image center (Cx , Cy)
- focal point (Fx , Fy) multiplied by a number that scales from pixels to distance (and is defined by the size of a pixel in the image sensor) .

Και μας δίνει :
coefficients for radial distortion (k1 , k2 , k3)
coefficients for tangential distoriton (p1 , p2)

Image Resectioning

Οι τιμές που λαμβάνουμε χρησιμοποιούνται στον παρακάτω μετασχηματισμό συντεταγμένων

$$s\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_1 \\ r_{31} & r_{32} & r_{33} & t_1 \end{pmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

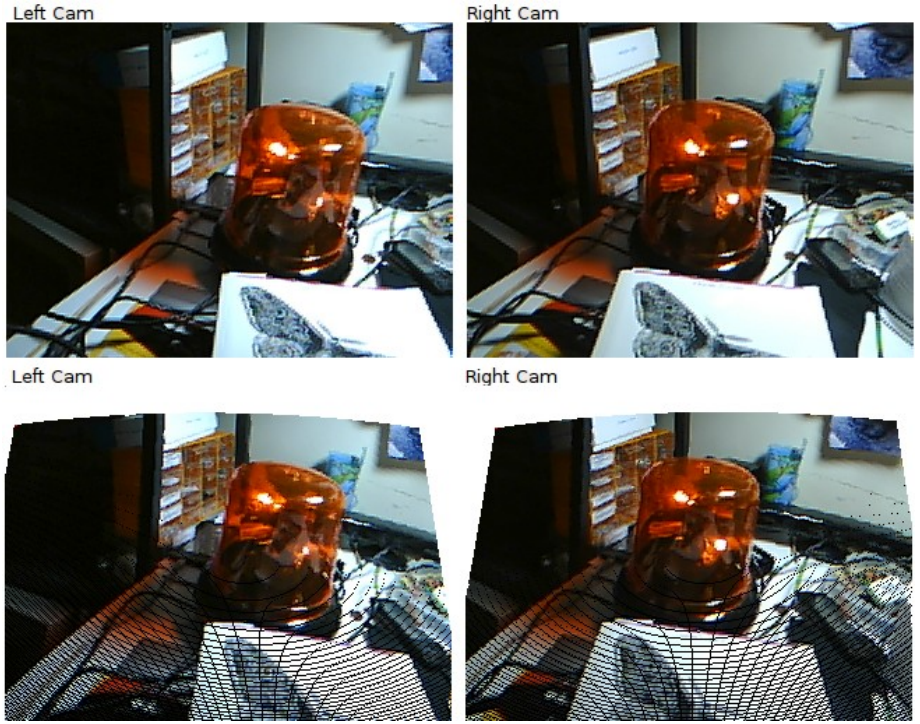
$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + t$$

$$x' = x/z$$
$$y' = y/z$$

$$r^2 = x'^2 + y'^2$$

$$x'' = x'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + 2\,p_1 x' y' + p_2\left(r^2 + 2x'^2\right)$$
$$y'' = y'\left(1 + k_1 r^2 + k_2 r^4 + k_3 r^6\right) + p_1\left(r^2 + 2y'^2\right) + 2p_2 x' y'$$

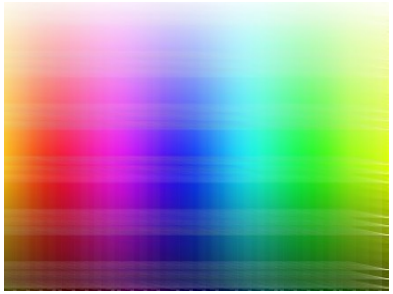
$$u = f_x x'' + c_x$$
$$v = f_y y'' + c_y$$



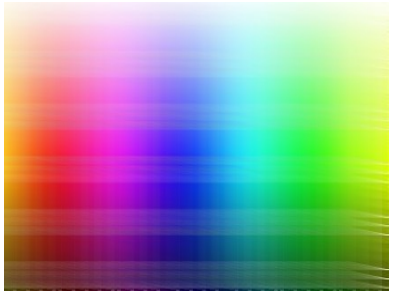
Mapping each of the x,y to u,v their real position

Video Input

Στόχος 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



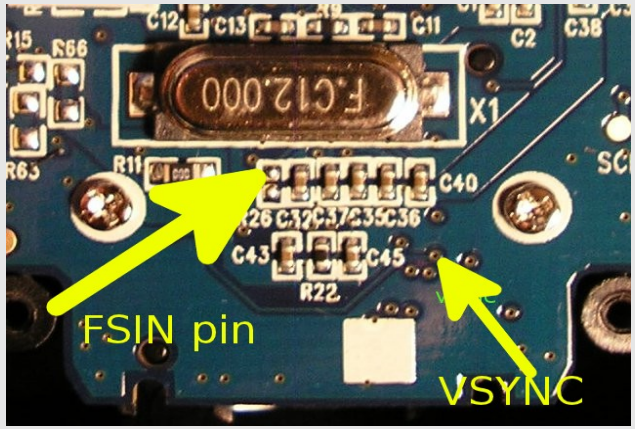
picture = GetFrame(0);



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)

Video Input

Small hardware based camera synchronization problems λόγω ανυπαρξίας κάποιου hardware clock can be improved using FSIN VSYNC pins



Κατα τα άλλα χαμηλό overhead , κοντά στο σύστημα , κυρίως hardware θέματα

(USB controller / Webcam Driver κτλ)

Πρόβλημα #0

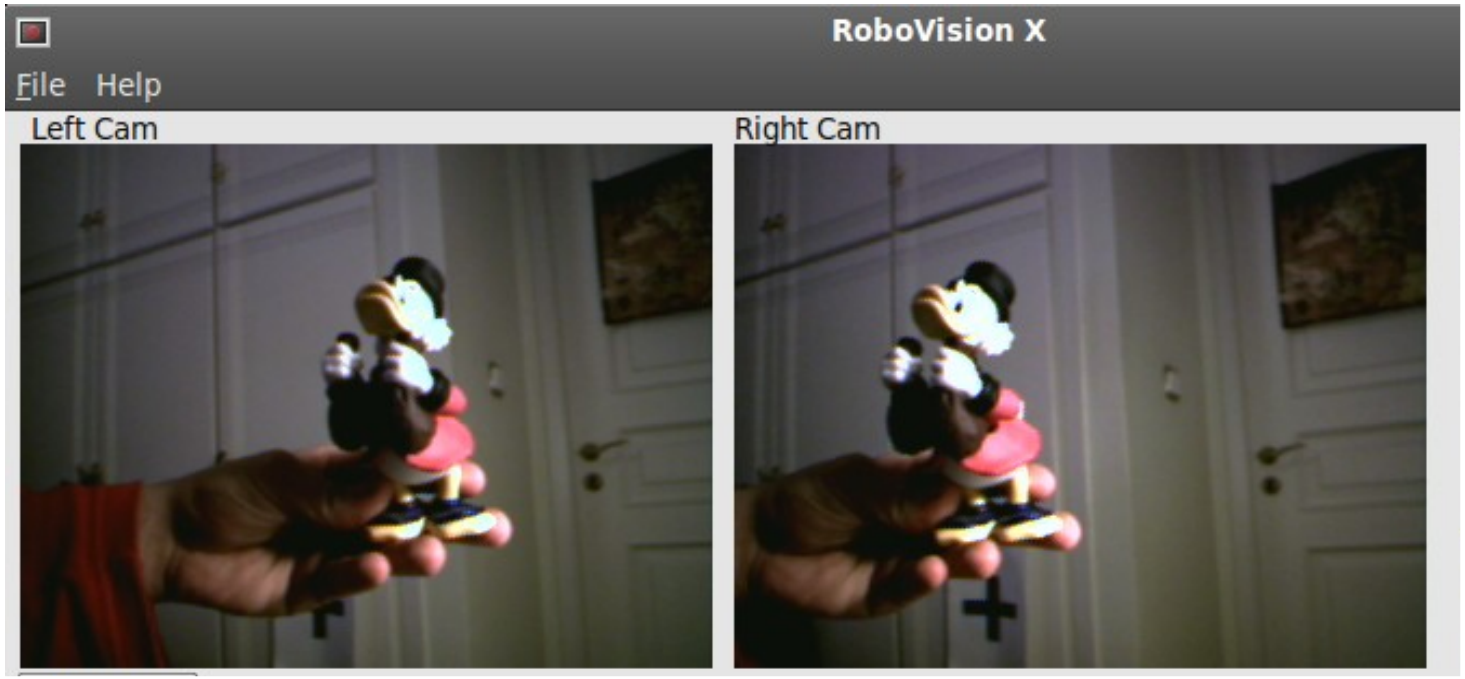
- Συνεχόμενο stream απο εικόνες
- Συγχρονισμένες
- Η εικόνα αναπαριστά την πραγματικότητα χωρίς αλλοιώσεις

Πρόβλημα #1

Έχουμε 2 κάμερες (που βγάζουν το δισδιάστατο είδωλο της σκηνής που βλέπει το ρομπότ) και θέλουμε να σχηματίσουμε μια τρισδιάστατη αναπαράσταση του χώρου

Θα πρέπει να μετασχηματίσουμε την σειρά 2 δισδιάστατων pixels εικόνων σε 3D points ή voxels ..!

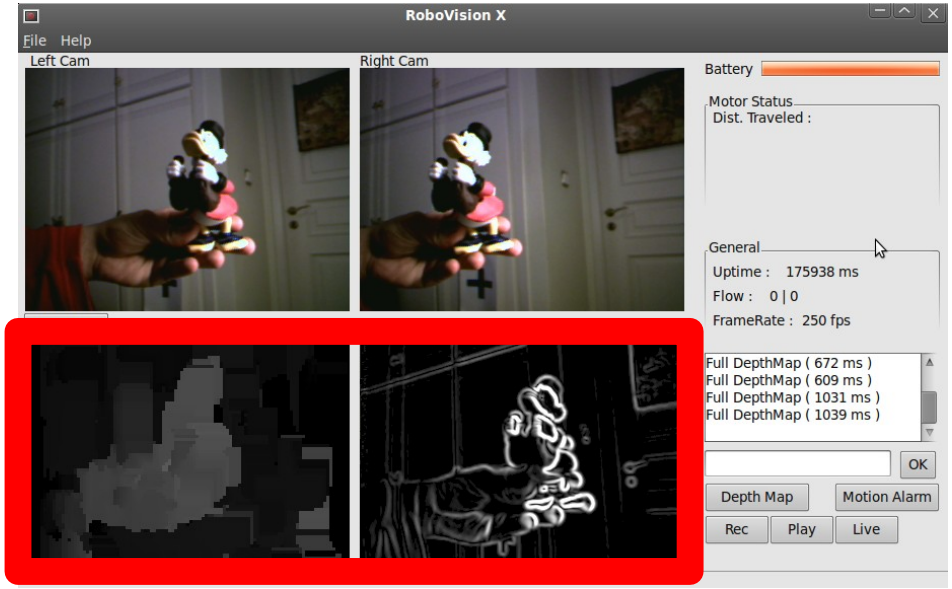
Μέχρι τώρα..



Έχουμε μια συνεχόμενη ροή εικόνας από 2 διαφορετικές οπτικές γωνίες και θέλουμε να μετασχηματιστεί σε πληροφορία τρισδιάστατου χώρου..

Ιδανικά θα θέλαμε να συμπεριφερθούμε στις 2 εικόνες σαν μεταβλητές που τις τοποθετούμε σε ένα μαύρο κουτί και μας εξάγει μια τρίτη εικόνα με πληροφορία βάθους..

Visual Cortex

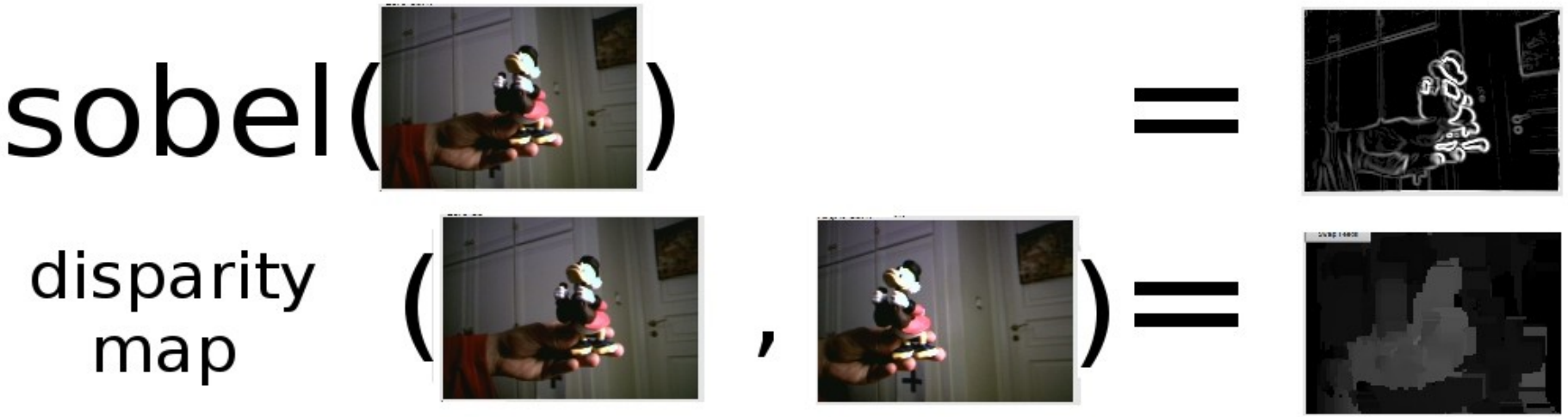


- Ουσιαστικά “δέχεται” pointers από frames

  zero-copy (1 copy βασικά)

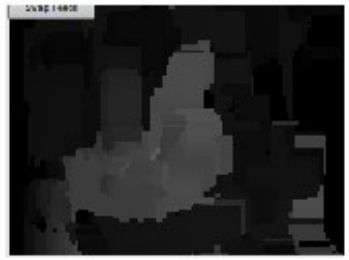
- Έχει ένα pipelining φίλτρων που τους εφαρμόζει για να μην υπάρχουν περιττές επαναλήψεις διαδικασιών

- **Εξάγει** frames τα οποία είναι **μετασχηματισμός** των frame **εισόδου..**



Visual Cortex

sobel() = 

disparity map ( , ) = 

Αντί για την “εικόνα” περνάμε έναν pointer !
Αλλά για να καταφέρουμε να βρούμε το βάθος πρέπει να μετασχηματίσουμε τις πληροφορίες σε πιο “βολική μορφή”

SobelFromSource(video_register[LEFT_EYE],video_register[LEFT_SOBEL],320,240);
Αυτό είναι το πρώτο ουσιαστικά..

Visual Cortex

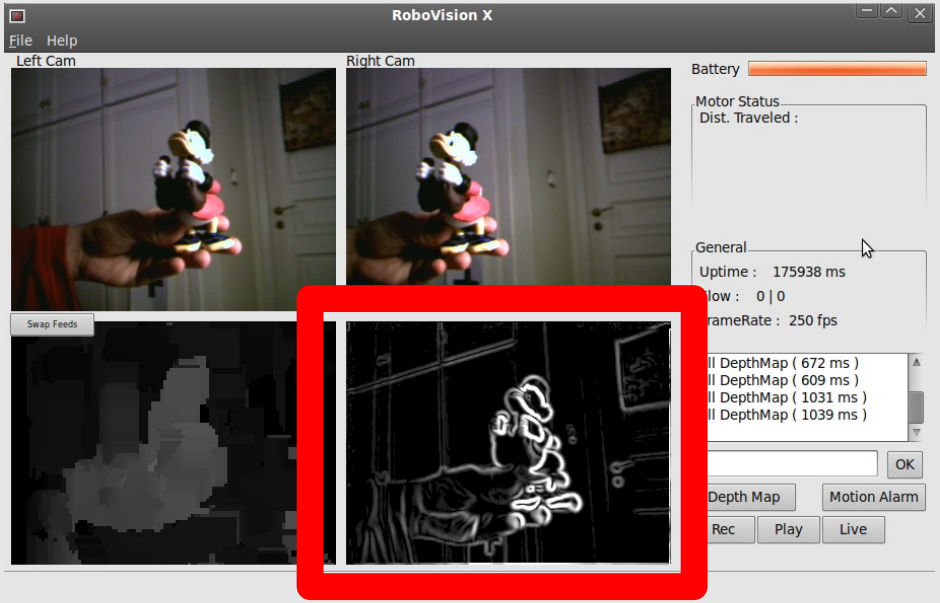
Ένα σύστημα με video registers και φίλτρα

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

SIFT / SURF (μέσω OpenSURF)
Face Detection (μέσω Fdlib, OpenCV)
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

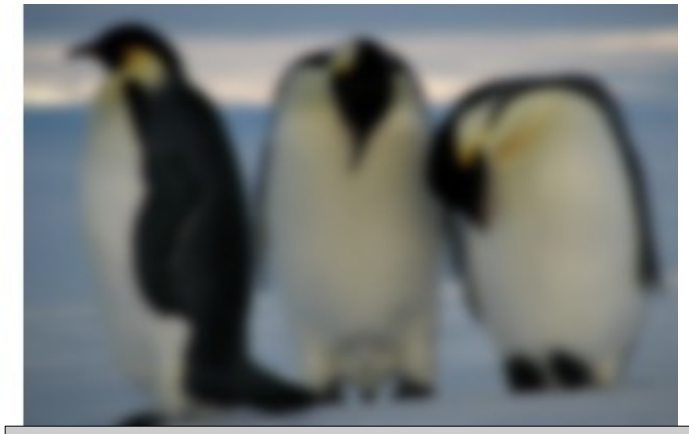
Αναγνώριση ακμών , υπολογίζοντας την παράγωγο αλλαγής χρώματος..

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 , αλλά θα σας τα αναφέρω περιληπτικά καθότι είναι τα βασικά building blocks για την διαδικασία..

Visual Cortex

Φίλτρα εξεργασίας εικόνας **πολύ συνοπτικά**



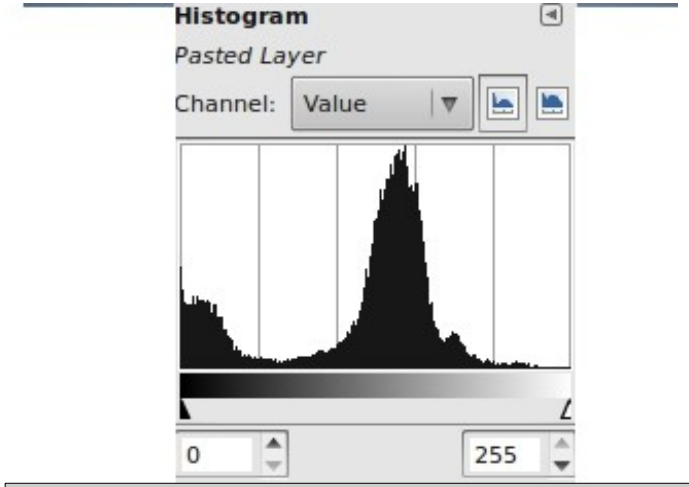
Gaussian Blur

Monochrome

Visual Cortex

Φίλτρα εξεργασίας εικόνας **πολύ συνοπτικά**

Sobel Edge Detection



Histograms

Visual Cortex

Φίλτρα εξεργασίας εικόνας **πολύ συνοπτικά**

Palette Reduction

Flood Fill

Visual Cortex

Filters as Convolution Matrices

1	1	1
1	**1**	1
1	1	1

3X3 Convolution Kernel
Divisor 9

As the anchor of the kernel passes from each element of the image array the value (marked blue) gets replaced by the addition of the neighboring elements multiplied with the according kernel element.

$$H(x,y)=\sum_{i=0}^{M_i-1}\sum_{j=0}^{M_j-1} I(x+i-a_i, y+j-a_j)G(i,j)$$

The anchor element on the light intensities array will become
(1x90+1x80 +1x70+1x90+1*80+1*70+ 1x90 + 1x80 + 1x70) / 9 which is 80

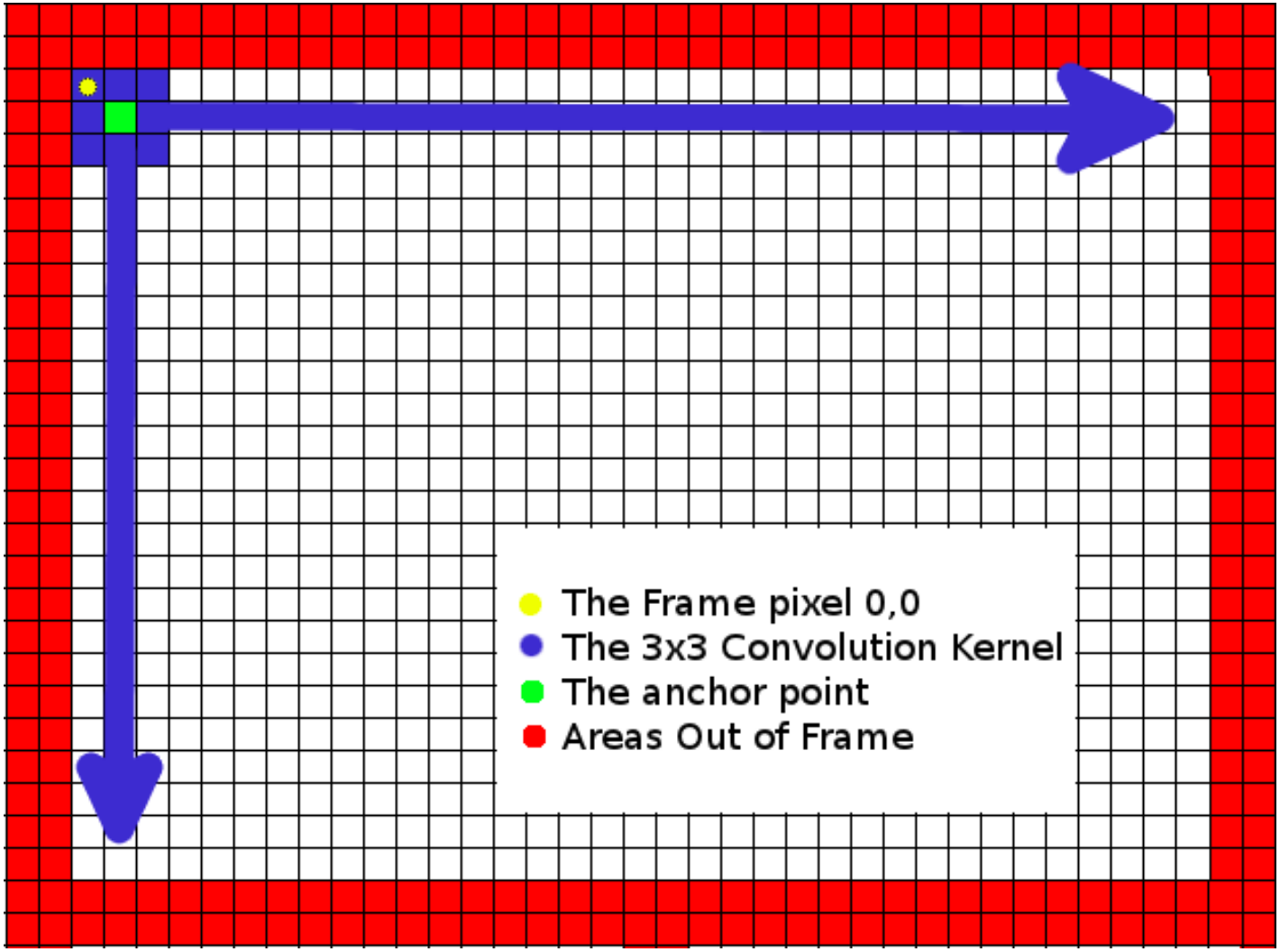
9 x 6 Original Light Intensities Captured

90	80	70	90	80	70	90	80	70
90	**80**	70	90	80	70	90	80	70
90	80	70	90	80	70	90	80	70
90	80	70	90	80	70	90	80	70
90	80	70	90	80	70	90	80	70
90	80	70	90	80	70	90	80	70

An important thing to be noted Is that values on the edges of the array (marked orange) can not be correctly calculated as not all neighboring elements exist , common solutions for this is “imagining” that there are zero elements when an element does not exist , using a different divisor to compensate for the missing elements or skipping the elements that can not be calculated correctlly .

Visual Cortex

Filters as Convolution Matrices



Visual Cortex

Filters as Convolution Matrices

GAUSSIAN BLUR

1	1	1
1	**1**	1
1	1	1

Divisor 9

SOBEL DERIVATIVE

1	-2	1
2	**-4**	2
1	-2	1

Divisor 1

SECOND-ORDER DERIVATIVE

-1	0	1
0	**0**	0
1	0	-1

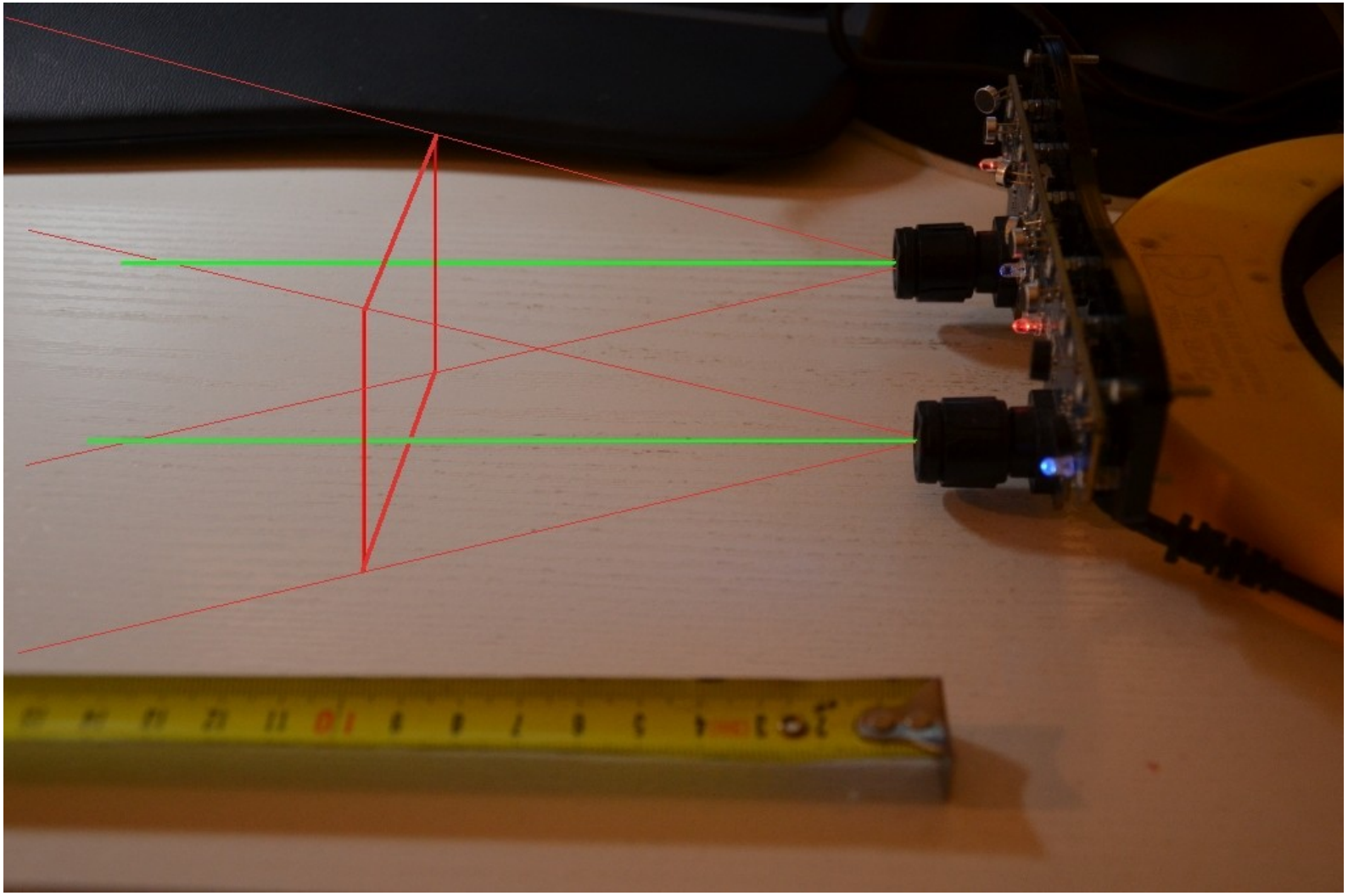
Divisor 3

Visual Cortex

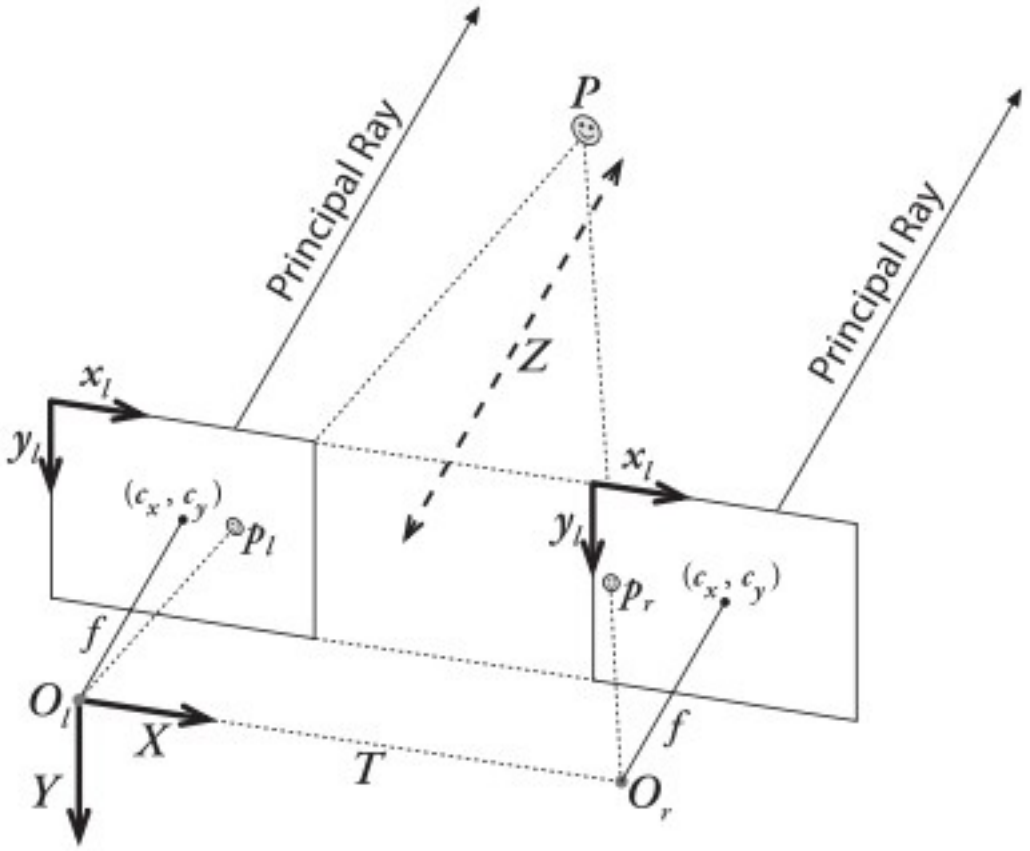
Κυρίως πρόβλημα Patch Matching!

Τι ταιριάζει πού ?

Visual Cortex



Visual Cortex



Προσπαθούμε να κάνουμε match το P από την αριστερή στην δεξιά κάμερα Pl με Pr

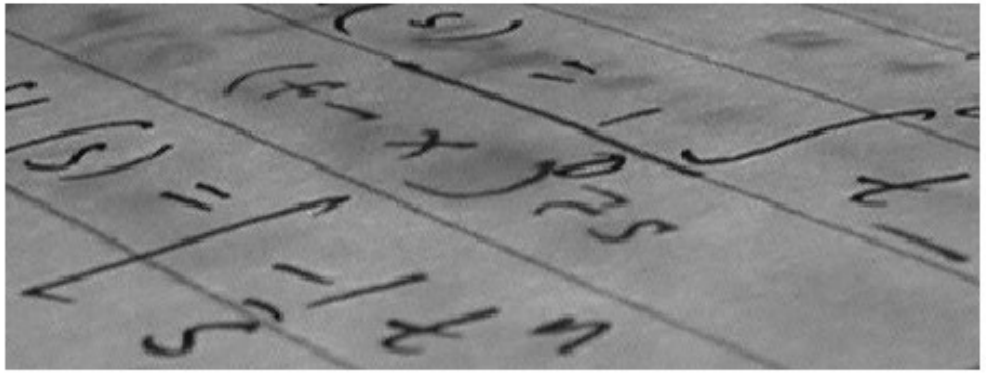
Visual Cortex

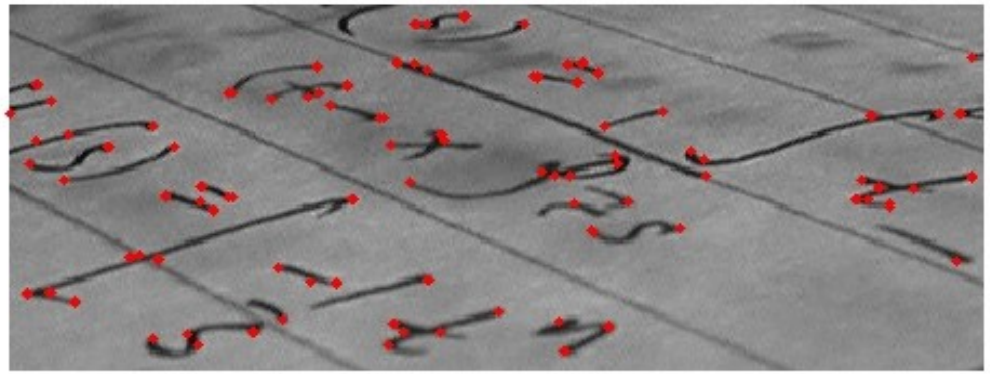
Τα ίδια αντικείμενα με :

* Ελαφρώς διαφορετική γωνία στον χώρο
* Στον άξονα του χρόνου
* Ανάλογα με τον φωτισμό
* Ηλεκτρομαγνητικό Θόρυβο στο CMOS
* Απόσταση από το focal point
* Lens Imperfections

Έχουν πολύ διαφορετική απεικόνιση !

Feature Detection



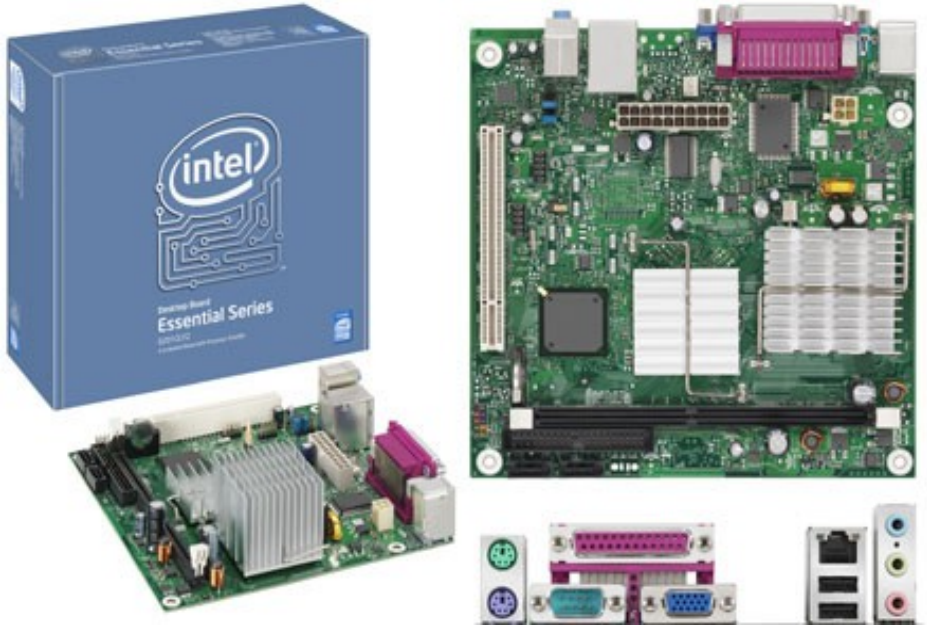


Common feature detectors and their classification:

Feature detector	Edge	Corner	Blob
Canny	X		
Sobel	X		
Harris & Stephens / Plessey	X	X	
SUSAN	X	X	
Shi & Tomasi		X	
Level curve curvature		X	
FAST		X	
Laplacian of Gaussian		X	X
Difference of Gaussians		X	X
Determinant of Hessian		X	X
MSER			X
PCBR			X
Grey-level blobs			X

Περιορισμένοι Πόροι

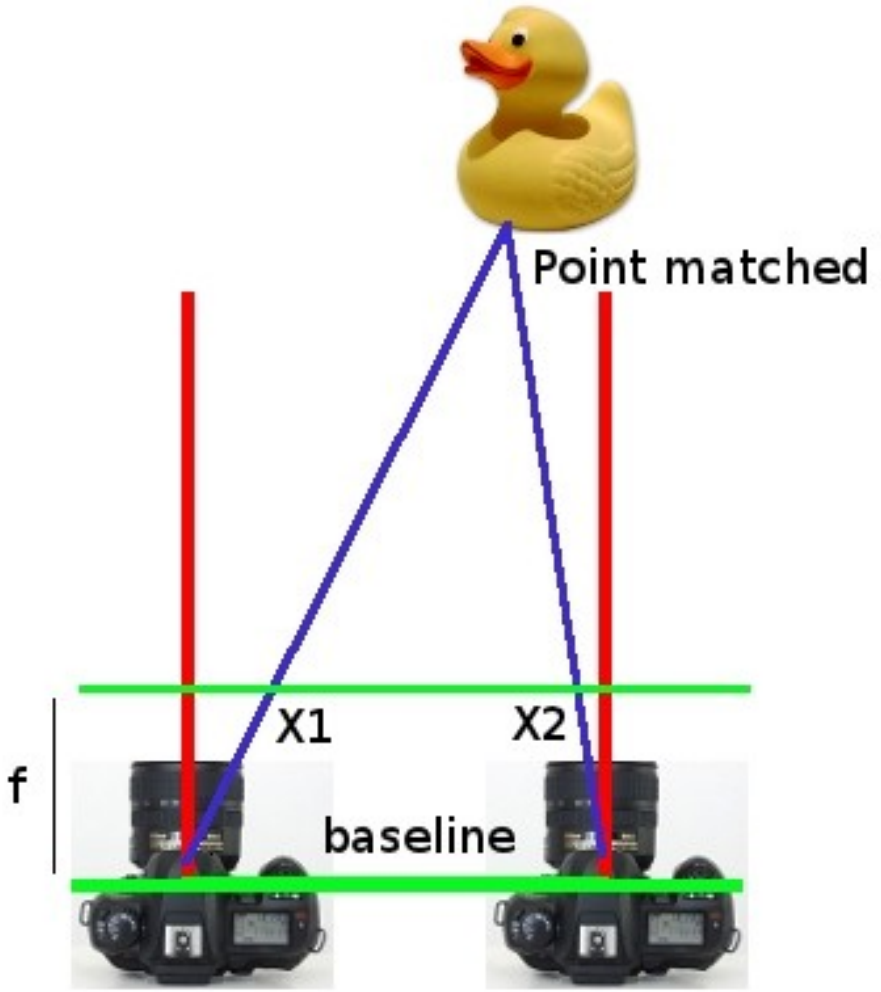
Intel Celeron : **1.2Ghz , 512 MB Ram , 800FSB**



- Ανάγκη για μια οσο το δυνατόν απλούστερη υπολογιστικά διαδικασία

- Οι 2 κάμερες είναι τοποθετημένες παράλληλα και μπορούμε να “εκμεταλλευτούμε” το context αυτό , ώστε να έχουμε κάποιο feasable αποτέλεσμα!

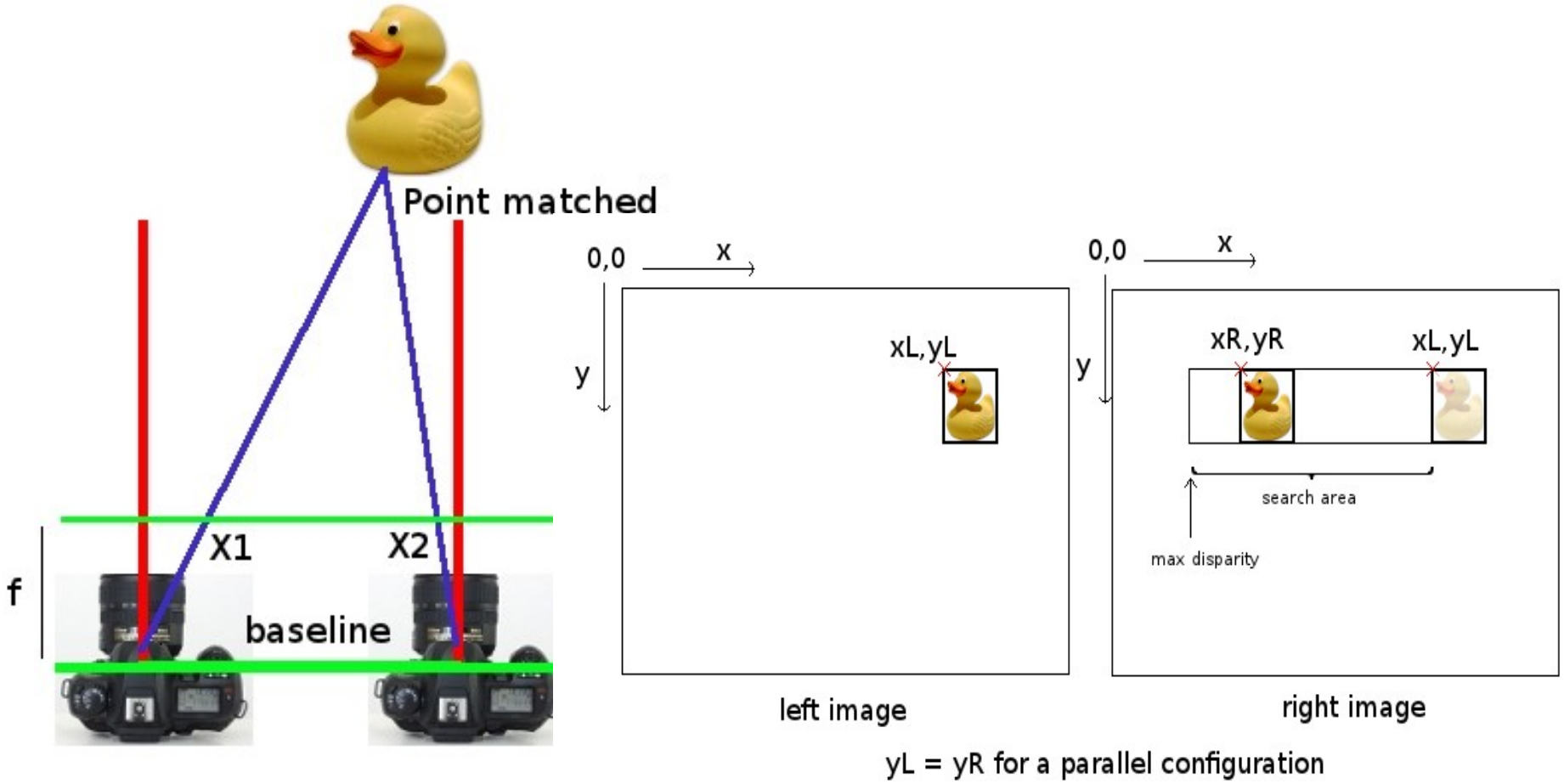
Στερεοσκοπία



$Z = (\text{baseline} * f) / (X1 - X2)$

$X = X1 * Z / f$
$Y = Y1 * Z / f$

Στερεοσκοπία



Visual Cortex

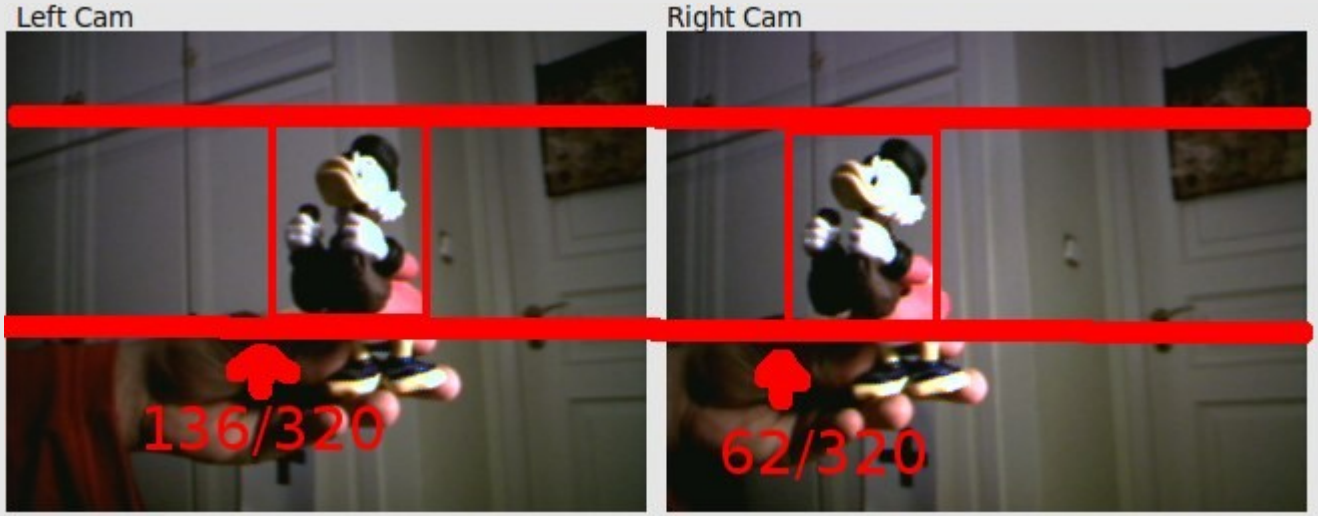
Disparity Mapping - VisualCortex/DisparityDepthMap.c
Βασική ιδέα , οι κάμερες κοιτάζουν παράλληλα αρα στον άξονα Y (ύψος) έχουμε ακριβώς ίδια σημεία , στον άξονα X όσο μεγαλύτερη η απόσταση , τόσο πιο κοντά

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

X

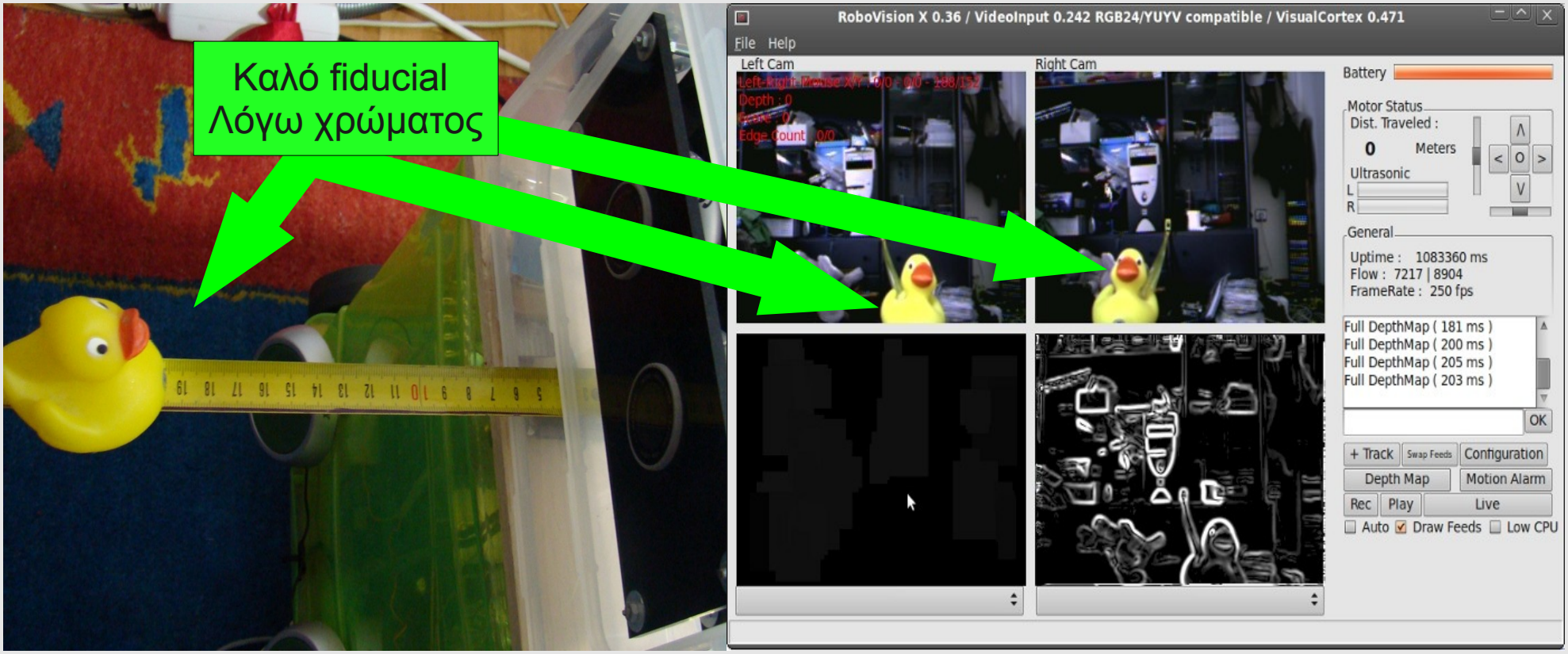
Y



136 – 62 = distance “**74**”
σημαίνει ότι είναι γύρω στα 28-28.5 cm μακριά από το ρομπότ στο συγκεκριμένο screenshot !

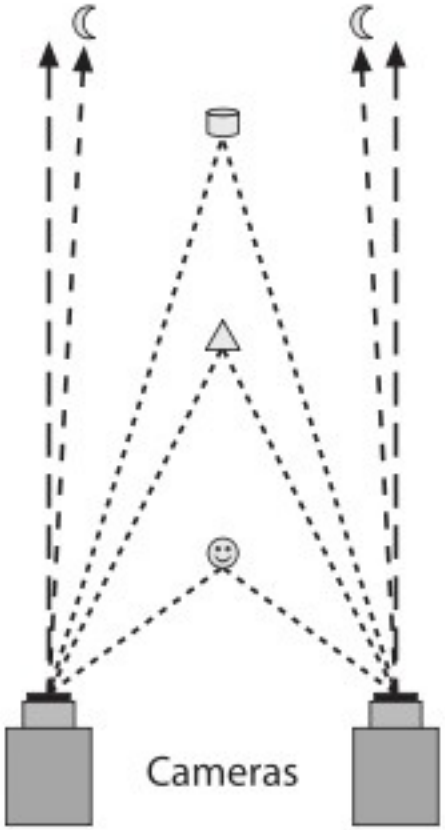
Visual Cortex

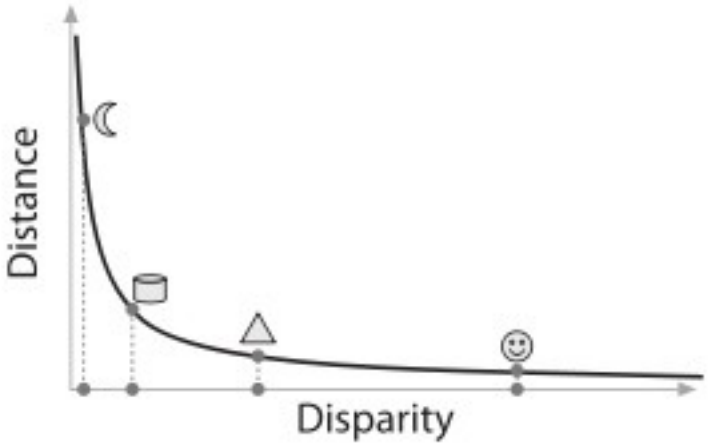
Εμπειρικές μετρήσεις



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

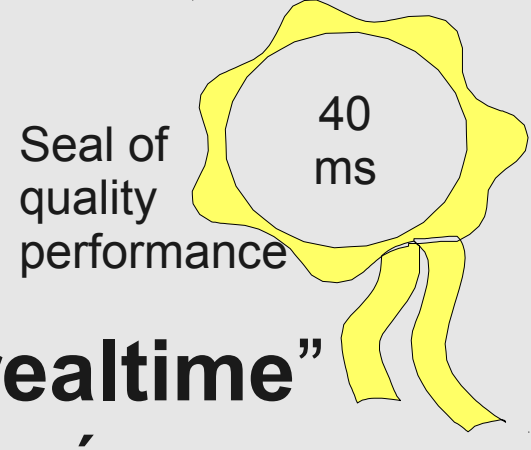




GuarddoG cameras 6.5cm απόσταση..
Works good for distances 20cm to 3m

(Εσωτερικοί χώροι)

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 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

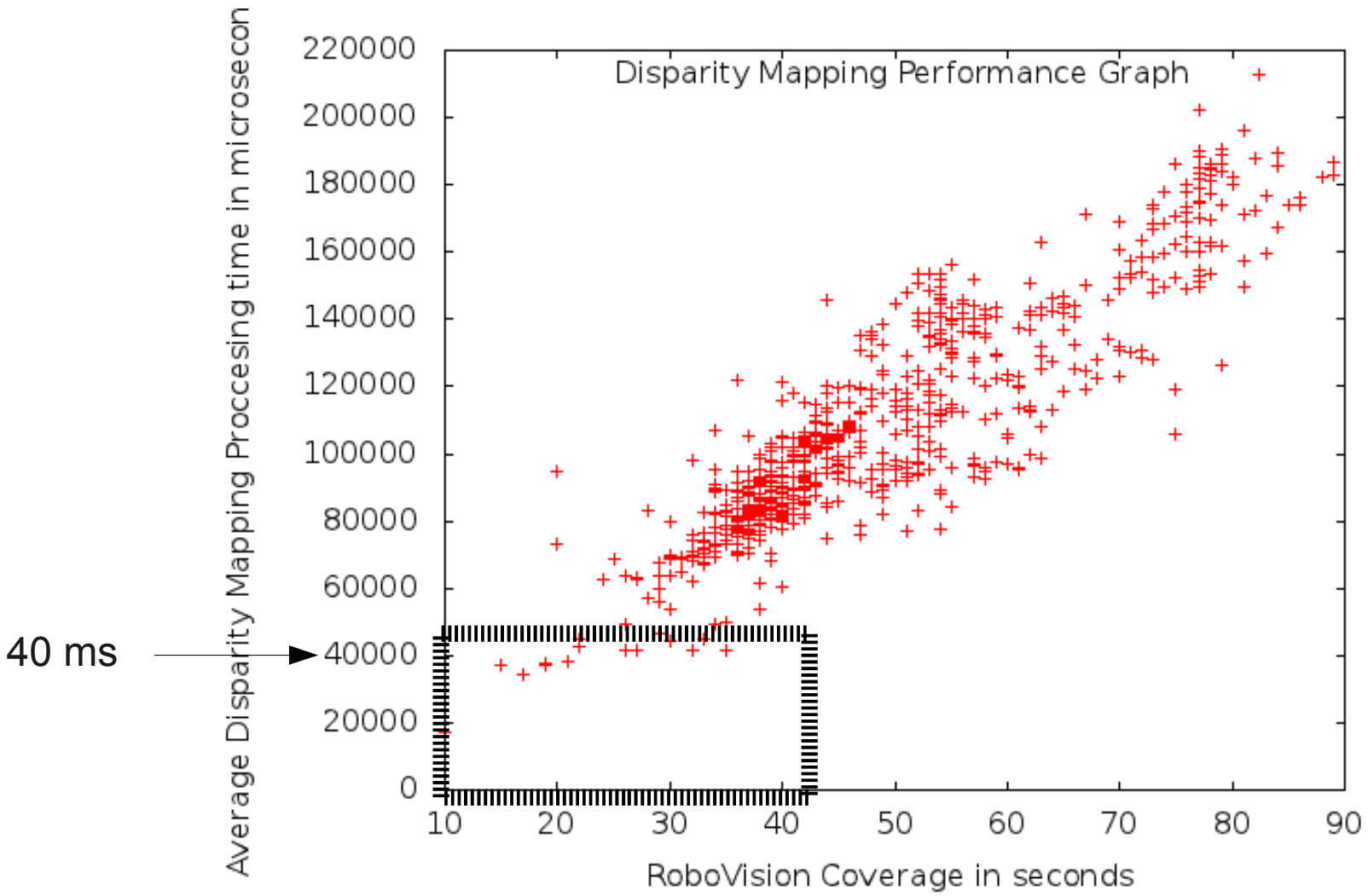
1 * 320 x 320 x 240 / 40 = 24576000 / 40 = **614400 operations / ms**
8 * 640 x 640 x 480 / 40 = 196608000 / 40 = **4915200 operations / ms**
24 * 1024 x 1024 x 768 / 40 = 805306368 / 40 = **20132659 operations / ms**

...

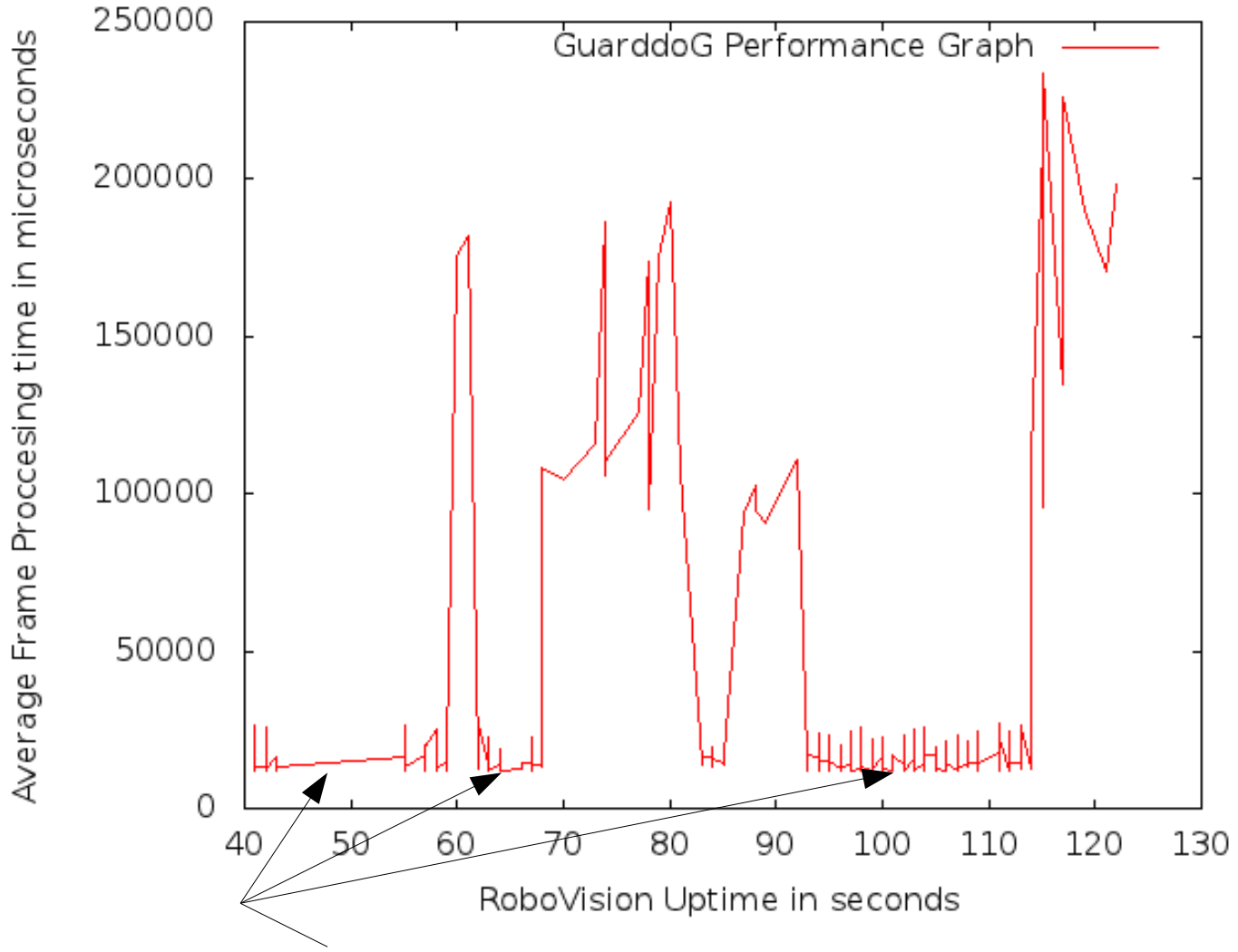
...

110 * 1920 x 1920 x 1024 / 40 = 3774873600 / 40 = **94371840 operations / ms**

Visual Cortex



Visual Cortex



No scene changes , saving CPU time and power

Visual Cortex

Σε περίπτωση που είχα ειδικό εξοπλισμό και όχι off-the shelf cameras
(Ακραίο παράδειγμα Large Hadron Collider shoots 60 megapixel photos at 40 million frames per second.)

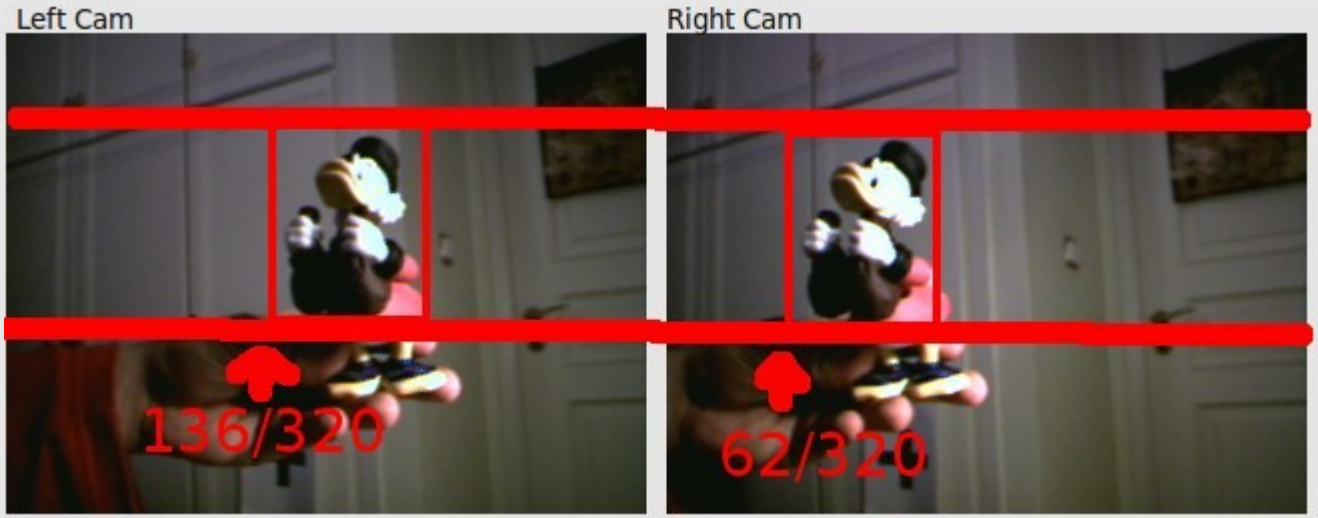
60 MP @ 40.000.000 fps >> 0.1 MP @ 25 fps

Ένα ταχύτερο framerate και η μεγαλύτερη εικόνα θα βοηθούσε πολύ περισσότερο για ένα ακριβές αποτέλεσμα

Οι νέες κάμερες έχουν 320x240 @ 120 fps

Επίσης θα έπρεπε να γίνεται ακόμα πιο γρήγορα η όλη διαδικασία!!

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 , πυραμίδες)
Normalization

... Και άλλα ...

Visual Cortex

My Patch Matching Function
Input & Calling it

```
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 OLD :P 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 = 30000 // 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..!

Μέθοδος summed area table

(την ξανα"εφηύρα" 30 χρόνια μετά από την orignal εφεύρεση της.. :P)

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
Όπως είπα και πιο πριν έχουμε RGB bytes

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 αφαιρούμε τους όρους στην άκρη αριστερά ! Έτσι γλιτώνουμε παρα πολλούς υπολογισμούς

Visual Cortex

Chi-square (method = CV_COMP_CHISQR)

$$d_{\text{chi-square}}(H_1, H_2) = \sum_i \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)}$$

For *chi-square*,* a low score represents a better match than a high score. A perfect match is 0 and a total mismatch is unbounded (depending on the size of the histogram).

Intersection (method = CV_COMP_INTERSECT)

$$d_{\text{intersection}}(H_1, H_2) = \sum_i \min(H_1(i), H_2(i))$$

For *histogram intersection*, high scores indicate good matches and low scores indicate bad matches. If both histograms are normalized to 1, then a perfect match is 1 and a total mismatch is 0.

Bhattacharyya distance (method = CV_COMP_BHATTACHARYYA)

$$d_{\text{Bhattacharyya}}(H_1, H_2) = \sqrt{1 - \sum_i \frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}}}$$

- Το Guarddog κάνει απλώς μια αφαίρεση των 2 histogram αθροισμάτων (δεξιά/αριστερή κάμερα) και αν το αποτέλεσμα είναι μεγαλύτερο από ένα threshold τα απορρίπτει , **δεν τα χρησιμοποιεί ώστε να κάνει matching**, τα χρησιμοποιεί ώστε να αποφύγει περιττά matches , σαν speed boost

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.

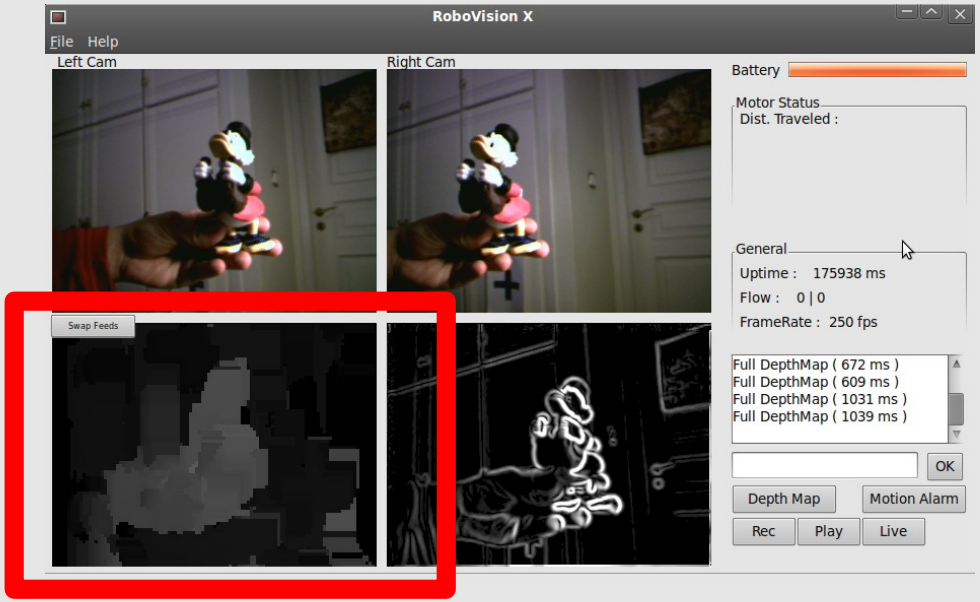
Για να βελτιωθεί το αποτέλεσμα κάποιες άλλες τεχνικές που εφαρμόζονται είναι :

-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

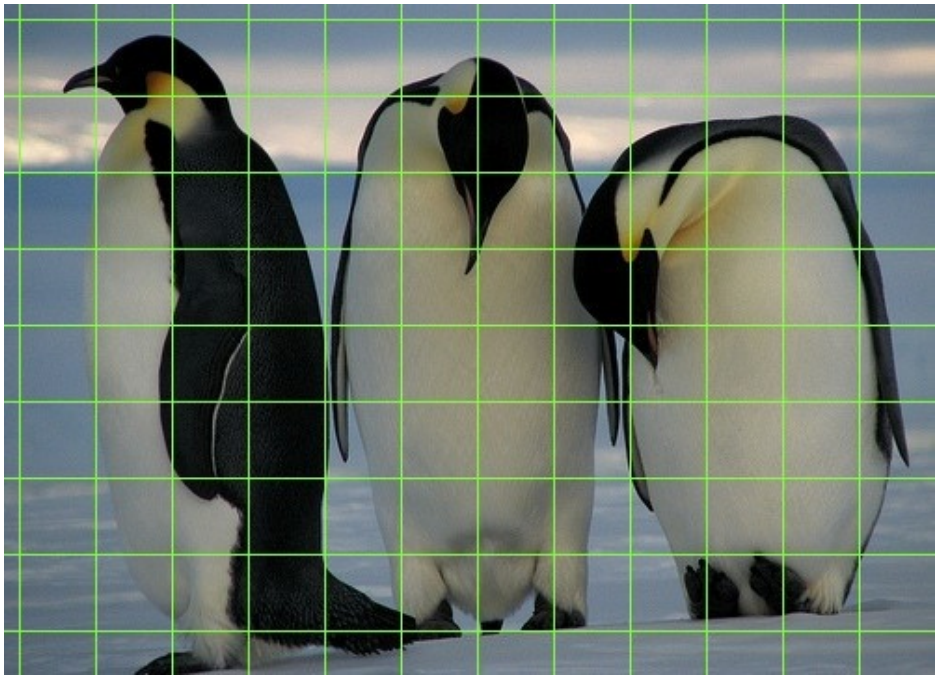
- Το αποτέλεσμα των αλγορίθμων του Visual Cortex είναι μια τρισδιάστατη φέτα του κόσμου , με τιμές από 0 (μακρυά) έως 320 (θεωρητικό κοντά όριο)

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



Visual Cortex

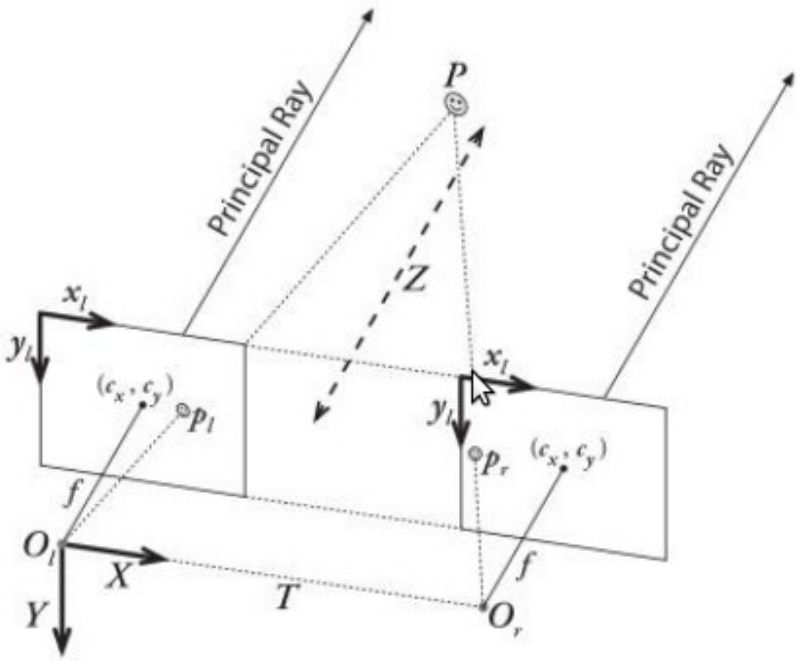
Disparity Mapping is by its nature a parallel task



- Μπορούμε να “τεμαχίσουμε” την εικόνα και να δώσουμε τα κομμάτια σε διαφορετικούς επεξεργαστές..
- Το πρόβλημα είναι εκ φύσεως παράλληλης επεξεργασίας
- CPU/(GPU?) task

Disparity Mapping

- Άλλα αξιοσημείωτα implementations για disparity mapping
- Σύγκριση με βάση υπάρχοντα test sets , Tsukuba , Middlebury stereo datasets..



-StereoSGBM ,Hirschmuller

-LibELAS ,Geiger

Disparity Mapping

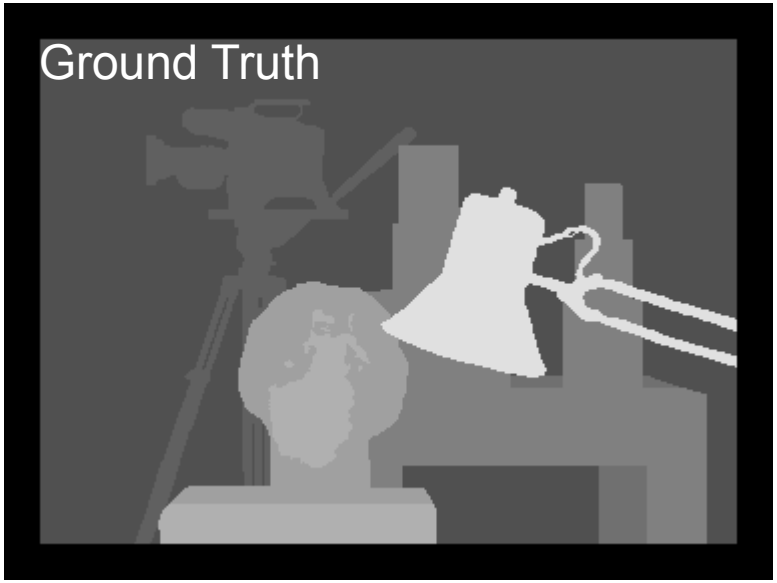
GuarddoG (traditional) disparity mapping algorithm pseudocode

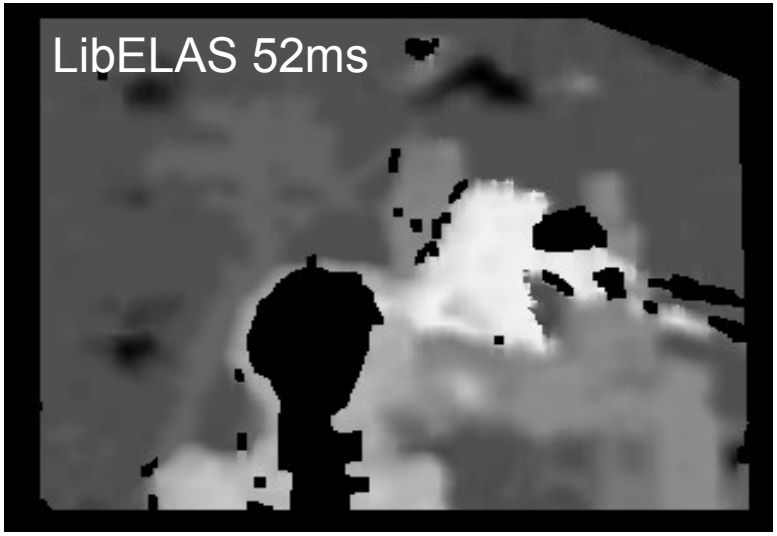
```
xL_Limit = height
yL_Limit = width
x_step = matching_window_width / detail
y_step = matching_window_height / detail

while ( yL < yL_Limit )
      {
        xL = 48; // Starting point , typically 15% of the image size therefore 48 for a 320x240 image
        while ( xL < xL_Limit )
         {
           best_match = Infinity;
           if ( //Filtering low texture areas to reduce errors
                EdgesOnInputWindow(
                                       xL,yL,
                                       matching_window_width,
                                       matching_window_height
                                     ) > edges_required_to_process_threshold)
           {
             MatchWithHorizontalScanline (
                       xL,yL
                       matching_window_size_x,matching_window_size_y
                       &best_match,
                       &xR,&yR
                                           )

             if ( best_match != Infinity )
               {
                 /* WE FOUND A MATCH */
                 RegisterDisparity(xL,yL,xR,yR,window_width,window_height)
               } else
               {  /* AREA IS EMPTY :P */ }
           }
            xL+=x_step
         }
         yL+=y_step
       }
```

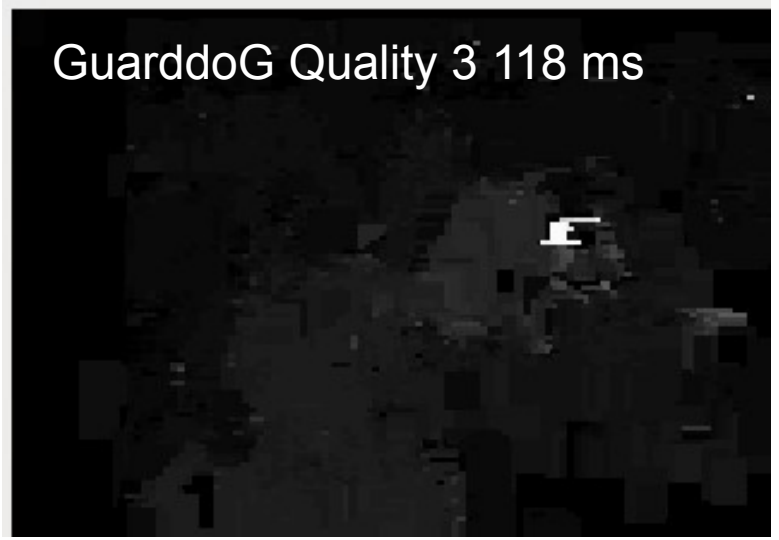
Disparity Mapping

```
MatchWithHorizontalScanline
(
  xL,yL
  matching_window_size_x,matching_window_size_y
  &best_match,
  &xR,&yR
)
{
  xR_Limit=xL
  yR_Limit=yL // this can be an offset used for bad calibration situations
  best_score=Infinity
  while ( yR <= yR_Limit )
      {
        if (xR_Limit>MaxDisparity )  { xR = xR_Limit-MaxDisparity; } else
                                     { xR = 0; }
        while ( xR < xR_Limit )
         {
           score = ComparePatches
                    (xL,yL
                     xR,yR
                     window_width,
                     window_height
                    ); // This function uses integral images to extract a score
                       // and this is the speed up of the guarddog algorithm
           if ( best_score < score )
             { //New best result
               best_match=abs(xL-xR)
             }
          ++xR
         }
        ++yR
      }
}
```

Disparity Mapping
Comparison

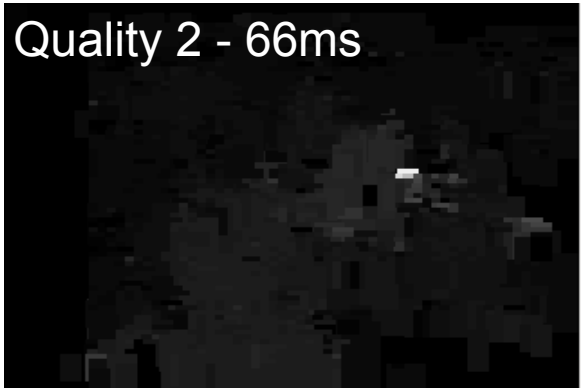


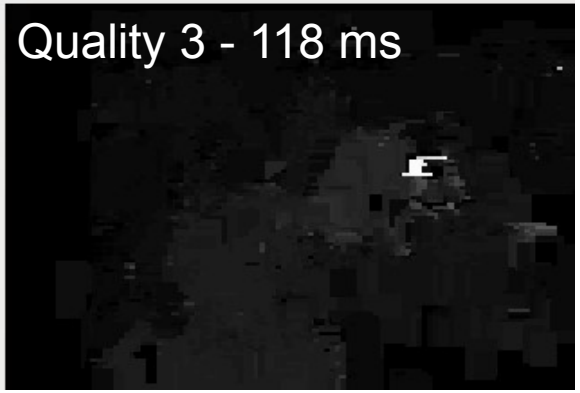


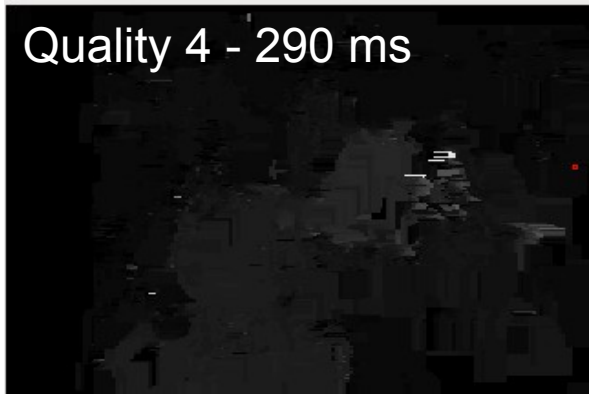


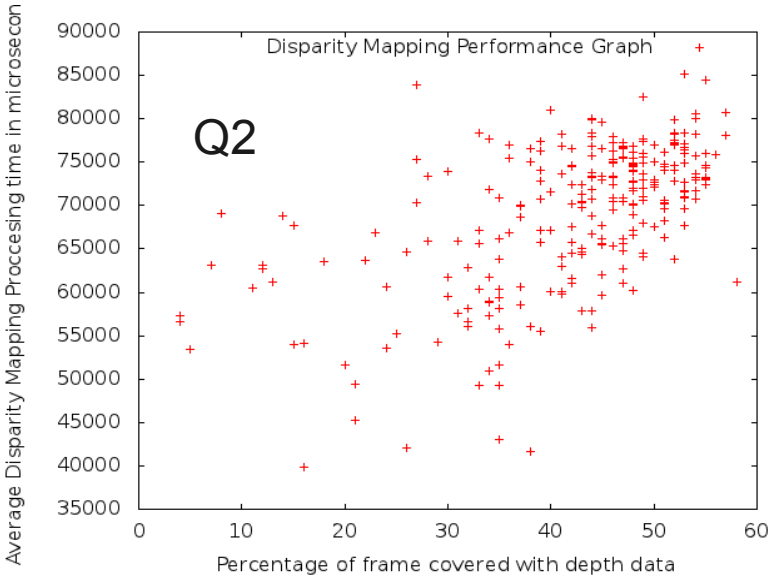
Disparity Mapping

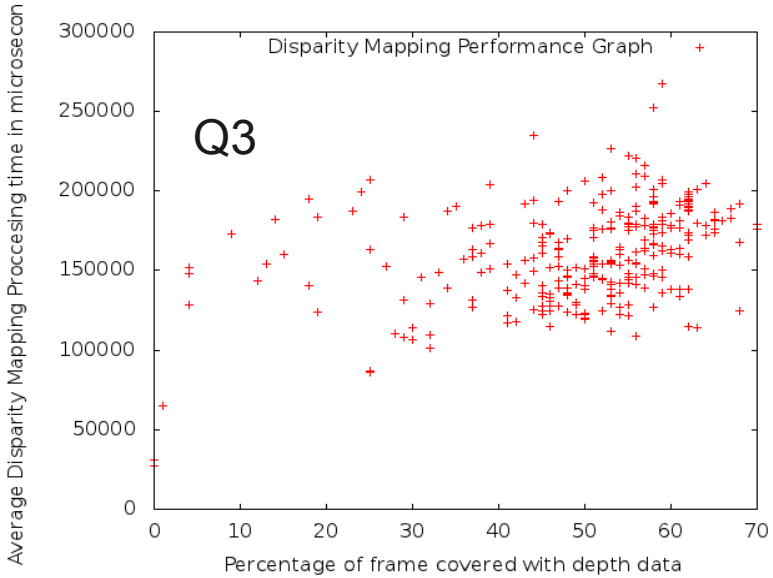
GuarddoG algorithm , time vs quality

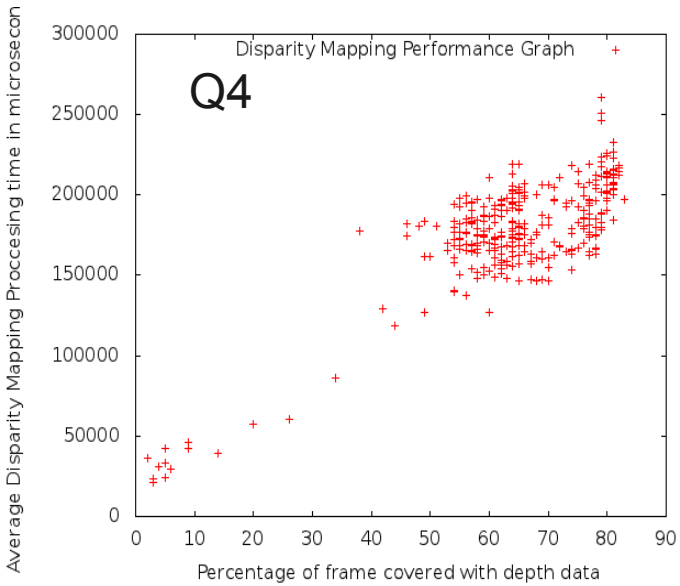






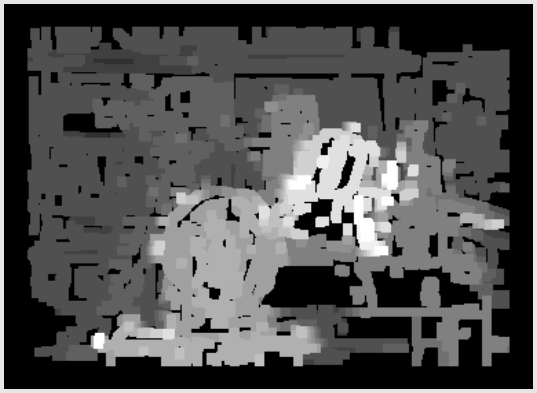




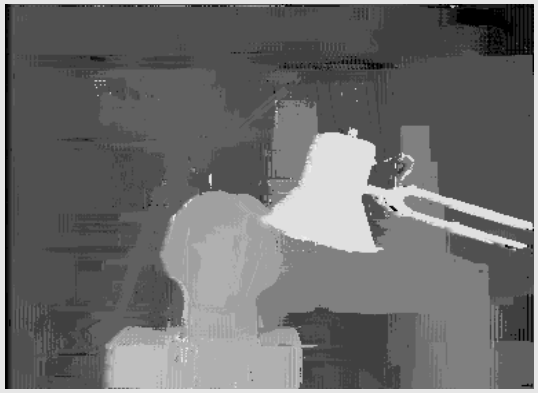


Disparity Mapping

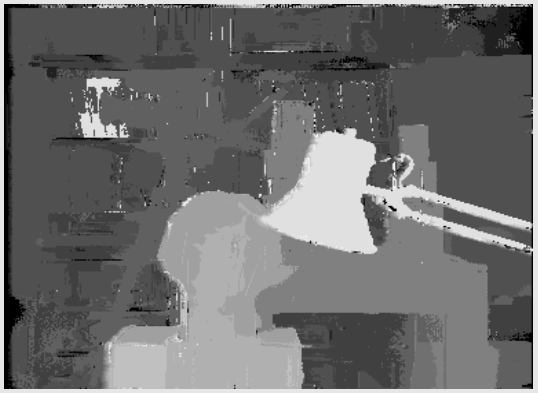
Extensive Comparison



Line Seg 1300+ ms



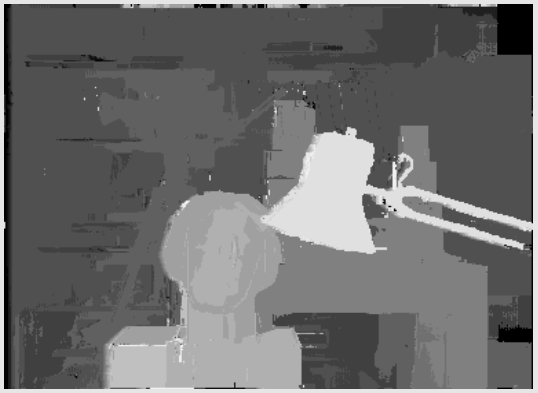
Fast Bilateral 32000ms



Adaptive Weights 1221000 ms

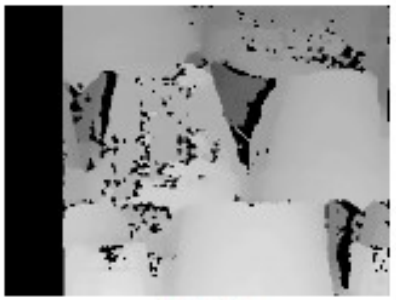
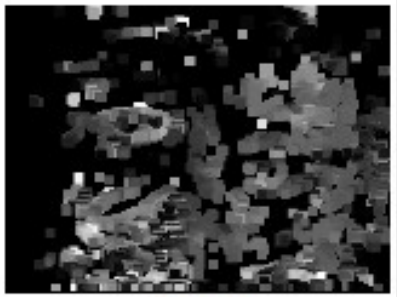
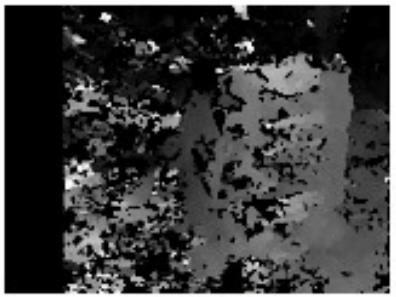
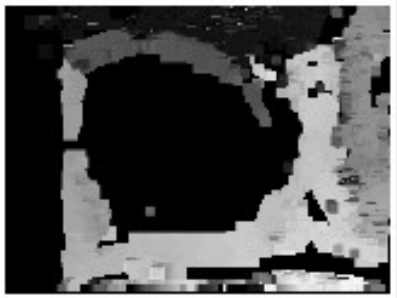
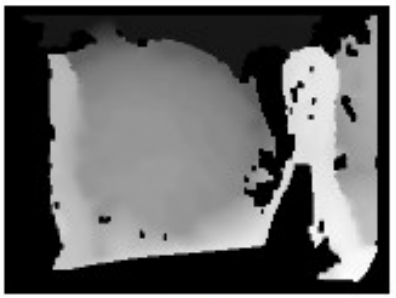
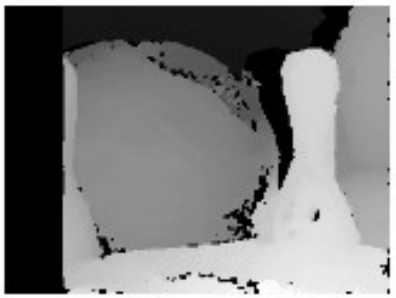
Segment Based 2000ms

Variable Window 26000ms

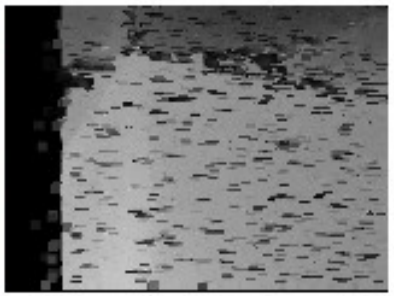
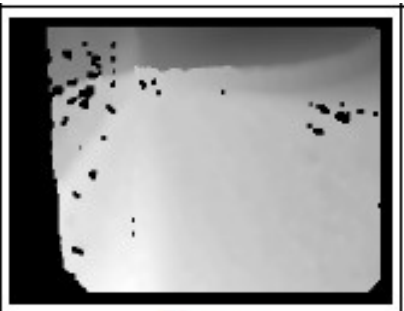
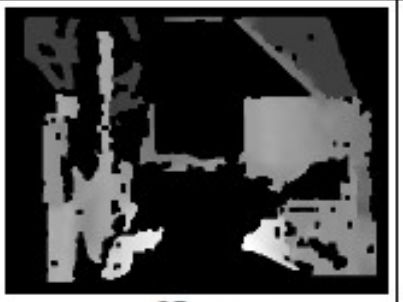
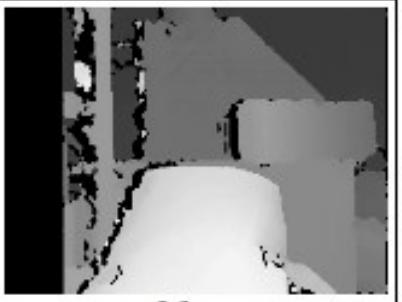
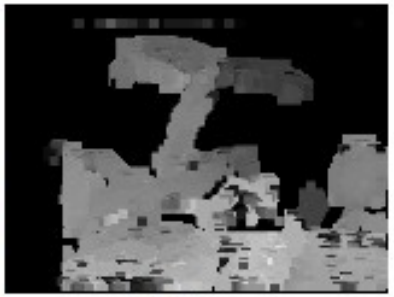
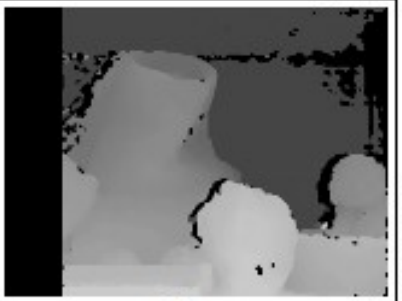


Segment Support 2358000 ms

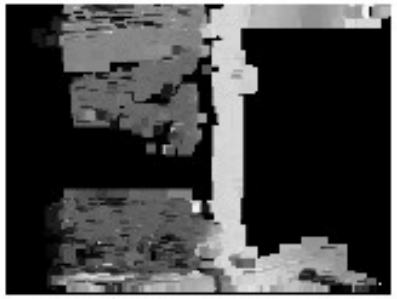
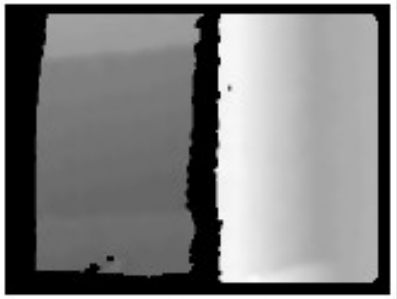
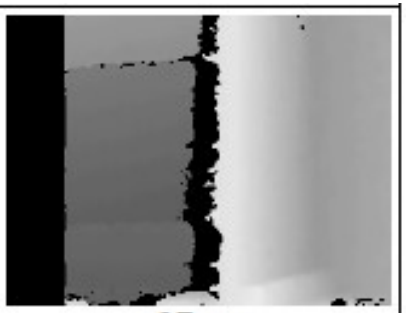
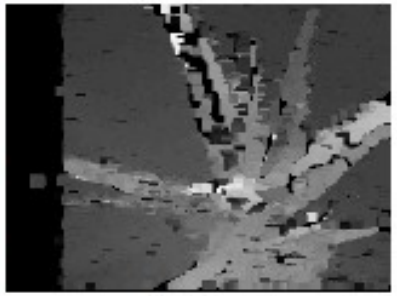
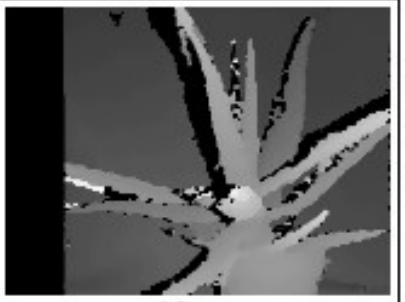
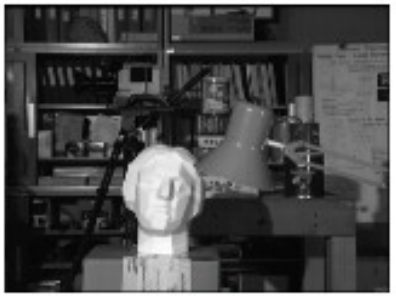
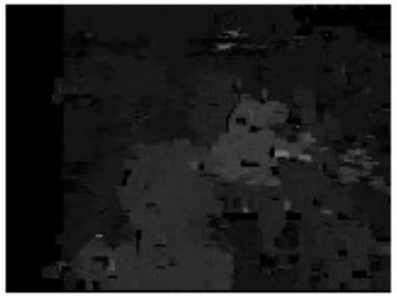
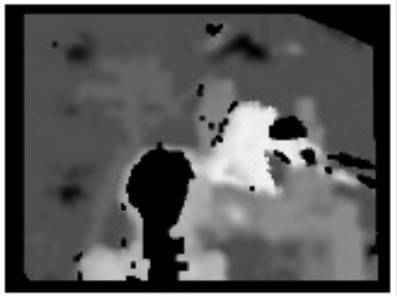
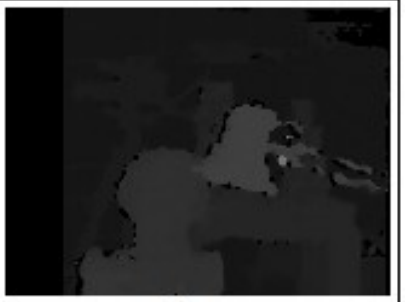
Disparity Mapping shootout

Test Images	GuarddoG	libELAS	StereoSGBM
 flowerpots	 142 ms	 51 ms	 38 ms
 gddg (custom)	 249 ms	 28 ms	 36 ms
 bowling	 173 ms	 48 ms	 40 ms

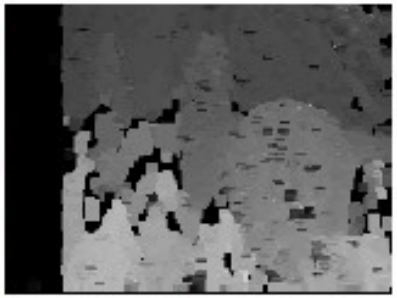
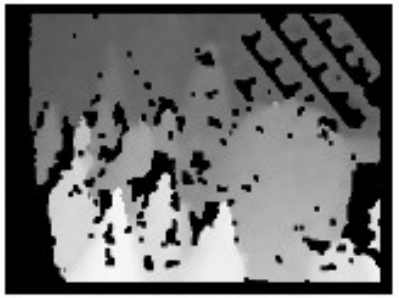
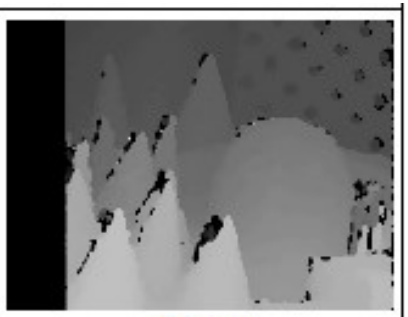
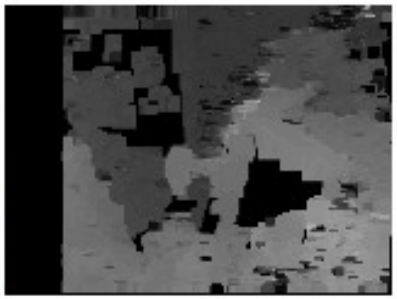
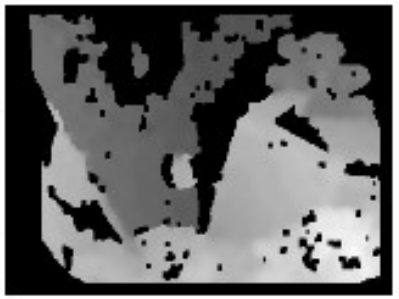
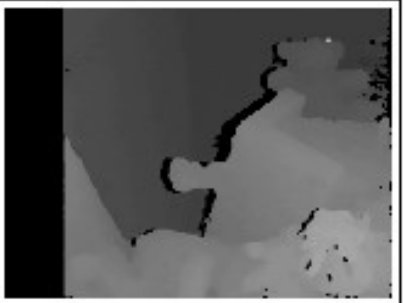
Disparity Mapping shootout

Test Images	GuarddoG	libELAS	StereoSGBM
 cloth	 433 ms	 51 ms	 31 ms
 lampshade	 147 ms	 39 ms	 36 ms
 middleburry	 171 ms	 27 ms	 37 ms

Disparity Mapping shootout

Test Images	GuarddoG	libELAS	StereoSGBM
 wood	 181 ms	 40 ms	 37 ms
 aloe	 346 ms	 52 ms	 38 ms
 tsukuba	 205 ms	 41 ms	 35 ms

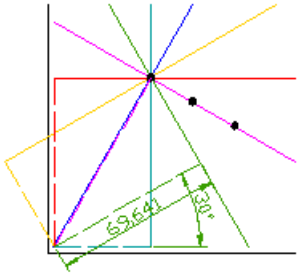
Disparity Mapping shootout

Test Images	GuarddoG	libELAS	StereoSGBM
 cones	 251 ms	 52 ms	 32 ms
 teddy	 200 ms	 40 ms	 33 ms

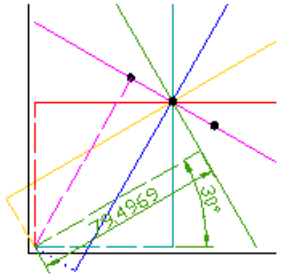
And the winner is StereoSGBM!

Άλλες τεχνικές
Hough transformation , lines

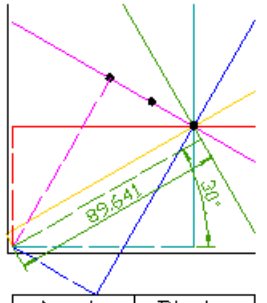
- Line Detection for matching lines , and improving things
- Ακόμα περισσότερη υπολογιστική δουλειά για το GuarddoG
- Ακόμα καλύτερα Contours αντί για lines κ.ο.κ
- TODO list!



Angle	Dist.
0	40
30	69.6
60	81.2
90	70
120	40.6
150	0.4



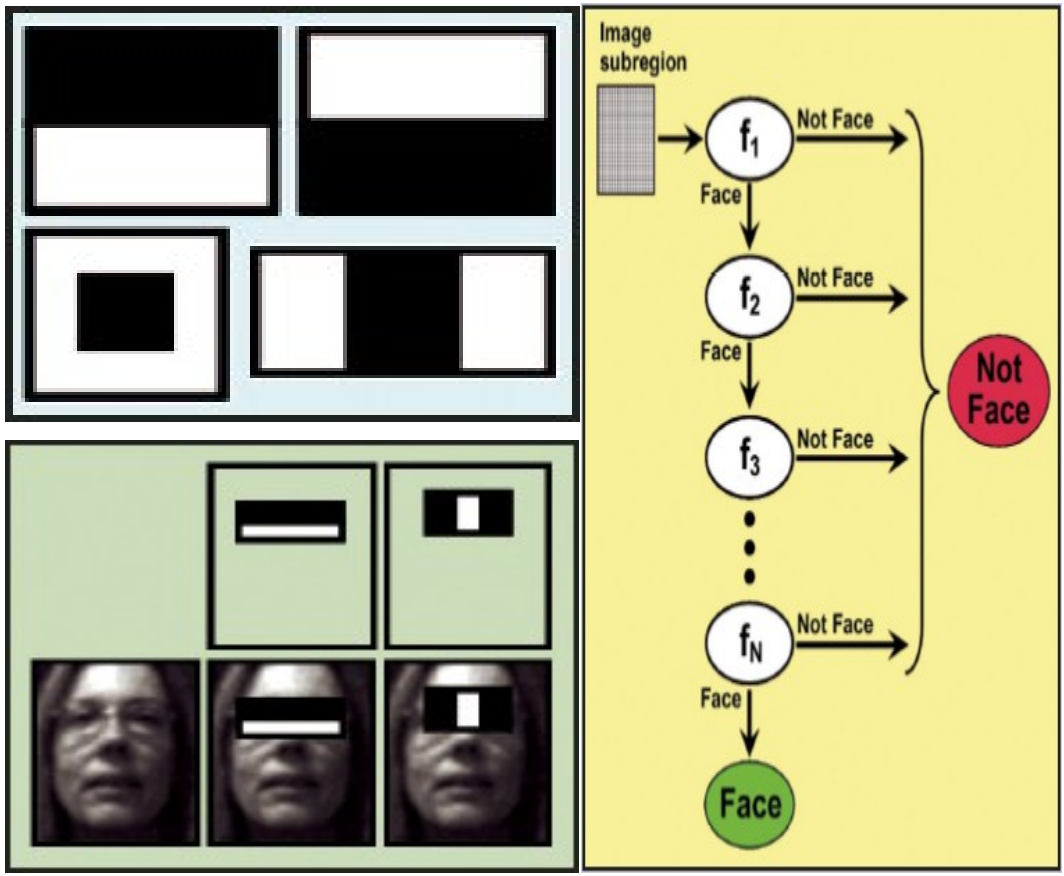
Angle	Dist.
0	57.1
30	79.5
60	80.5
90	60
120	23.4
150	-19.5



Angle	Dist.
0	74.6
30	89.6
60	80.6
90	50
120	6.0
150	-39.6

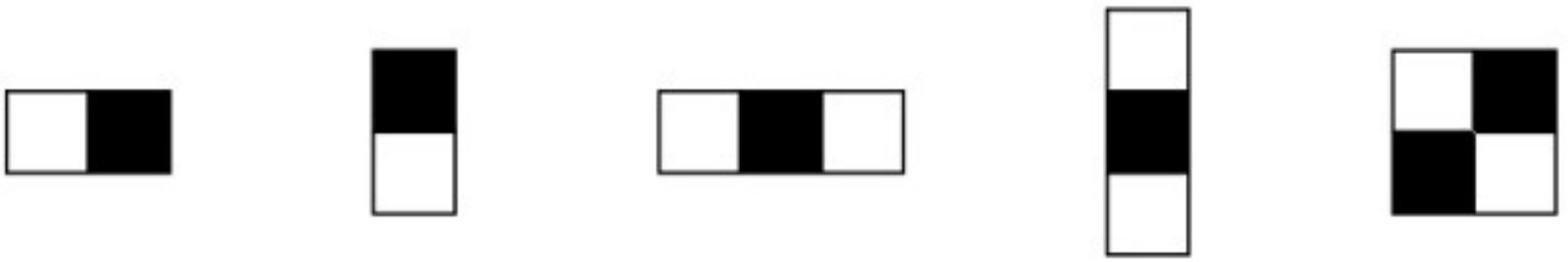
Άλλες τεχνικές
Face Detection (Haar features)

- The presence of a Haar feature is determined by subtracting the average dark-region pixel value from the average light-region pixel value. If the difference is above a threshold (set during learning), that feature is said to be present.



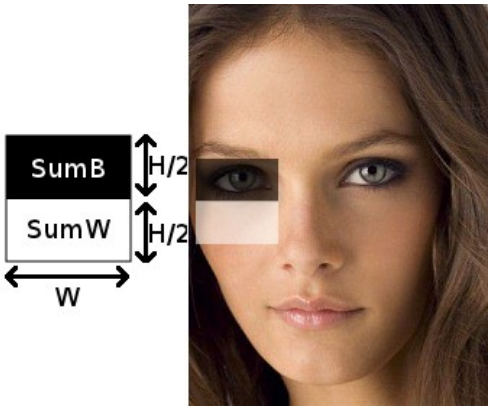
OpenCV uses this (Viola-Jones detector)

HAAR Wavelet Face Detection



- Black (Low) and

  White (High) areas
- Training using false and true images , OpenCV training file has over 5000 samples for upright faces..

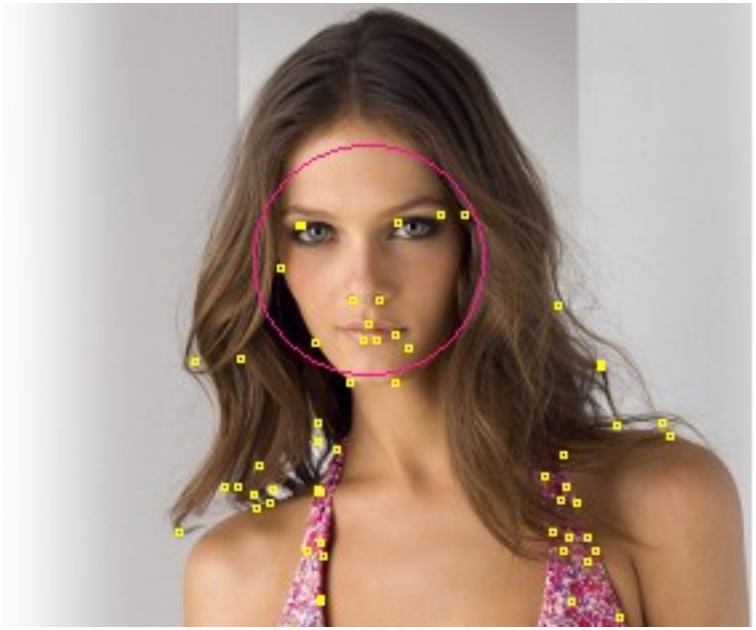


SumB = The Sum of color intensities in black area
SumW = The Sum of color intensities in white area

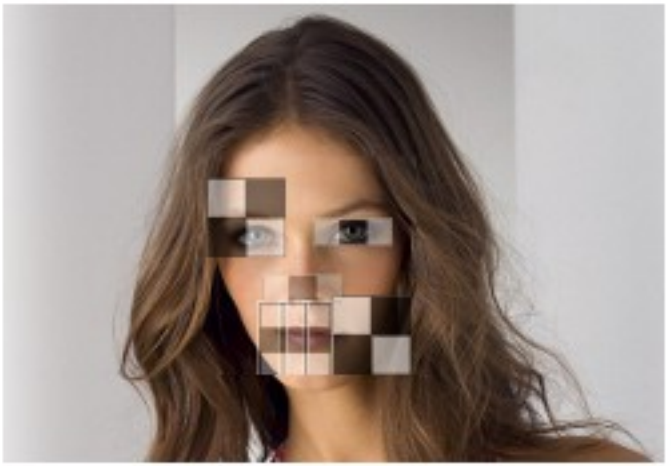
FeatureValue = SumW – SumB

If (FeatureValue > Threshold) { FeatureValue=1 }
else { FeatureValue=-1 }

HAAR Wavelet Face Detection



Features and the face detected

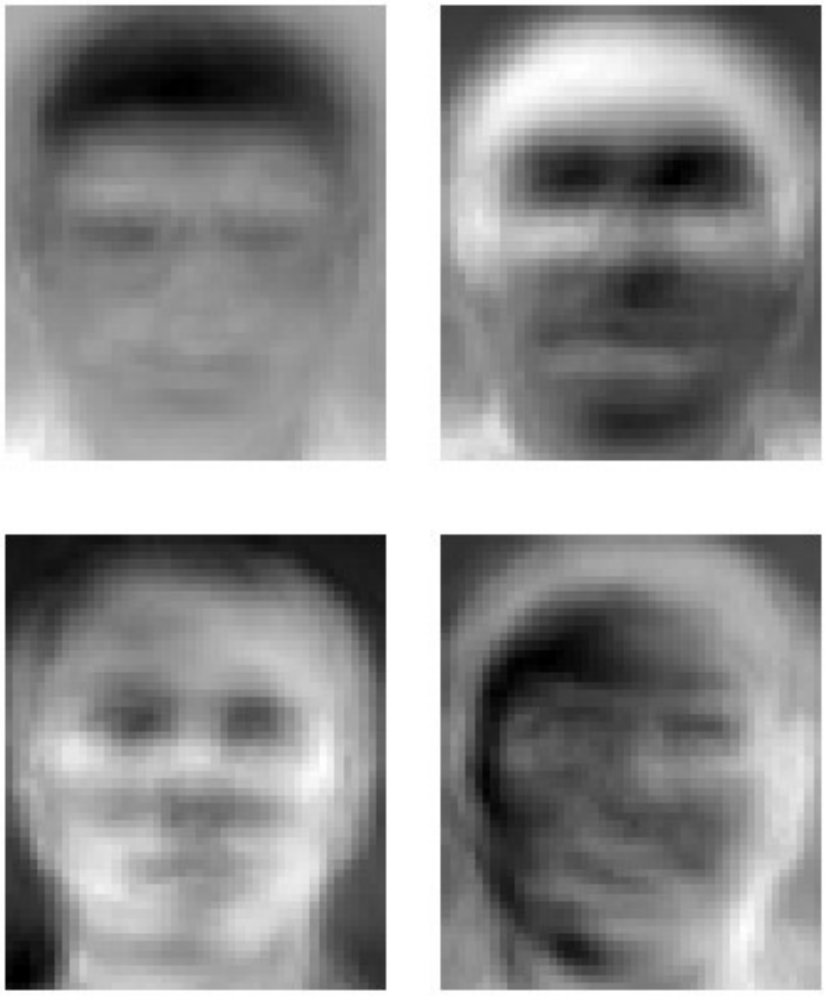


A Manual HAAR Cascade for dramatization

Testing Face Detection

Random faces out of a 4500+ faces collection gathered during IFT 2011

Άλλες τεχνικές
EigenFaces

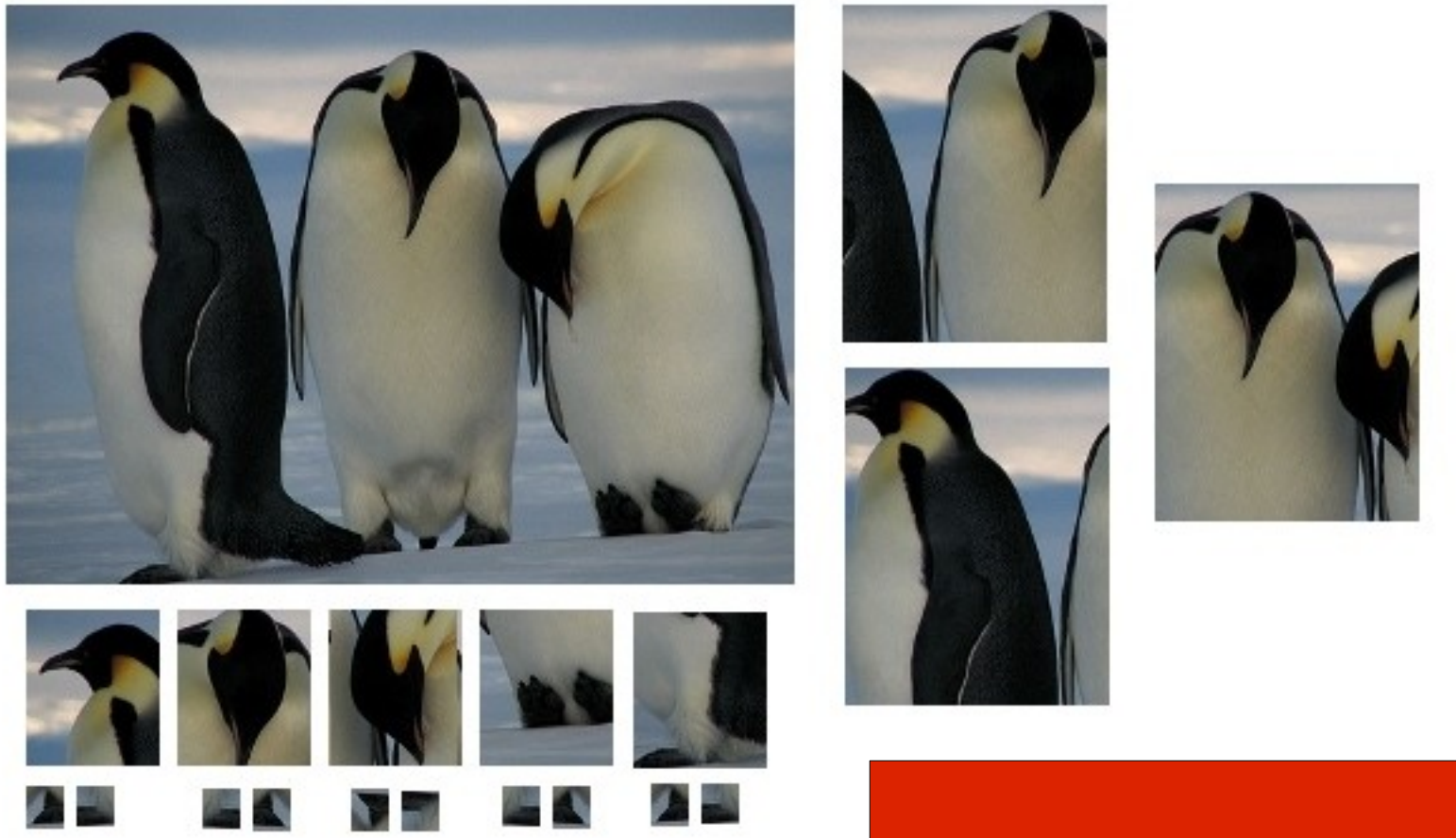


- Ουσιαστικά μέσοι όροι πολλών προσώπων
- Το κάθε πρόσωπο αναπαριστάται σαν

  15% A , 25%B , 0%C
  20% C

Visual Cortex

Όπως είπαμε και πριν κυρίως πρόβλημα Patch Matching!



Τι ταιριάζει πού ?

Visual Cortex

(actual guarddog voxel rendering)

Πρόβλημα #2

Έχουμε ένα 3D ανάγλυφο από σημεία του τι βλέπουμε μια στιγμή..

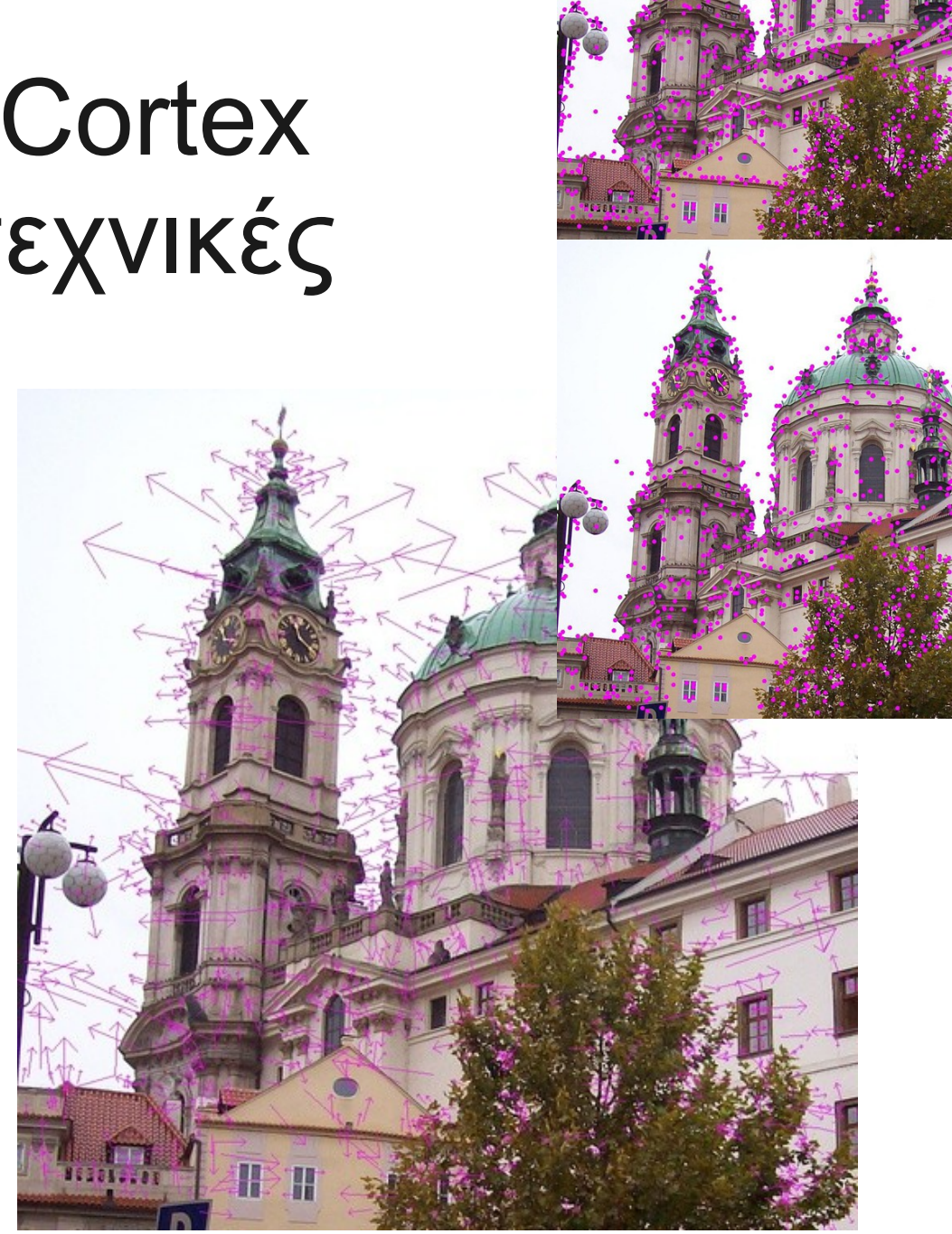
Πώς μπορούμε να φτιάξουμε έναν χάρτη από αυτό ?

Πώς μπορούμε να περιηγηθούμε στον χάρτη ?

Πώς μπορούμε να φτιάχνουμε/ανανεώνουμε αυτόματα τον χάρτη του τι βλέπουμε καθώς κινούμαστε ?

Visual Cortex
Άλλες τεχνικές

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

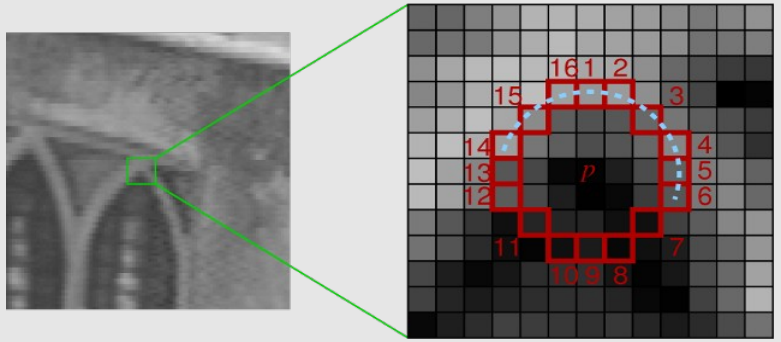


Μοναδικά Σημεία

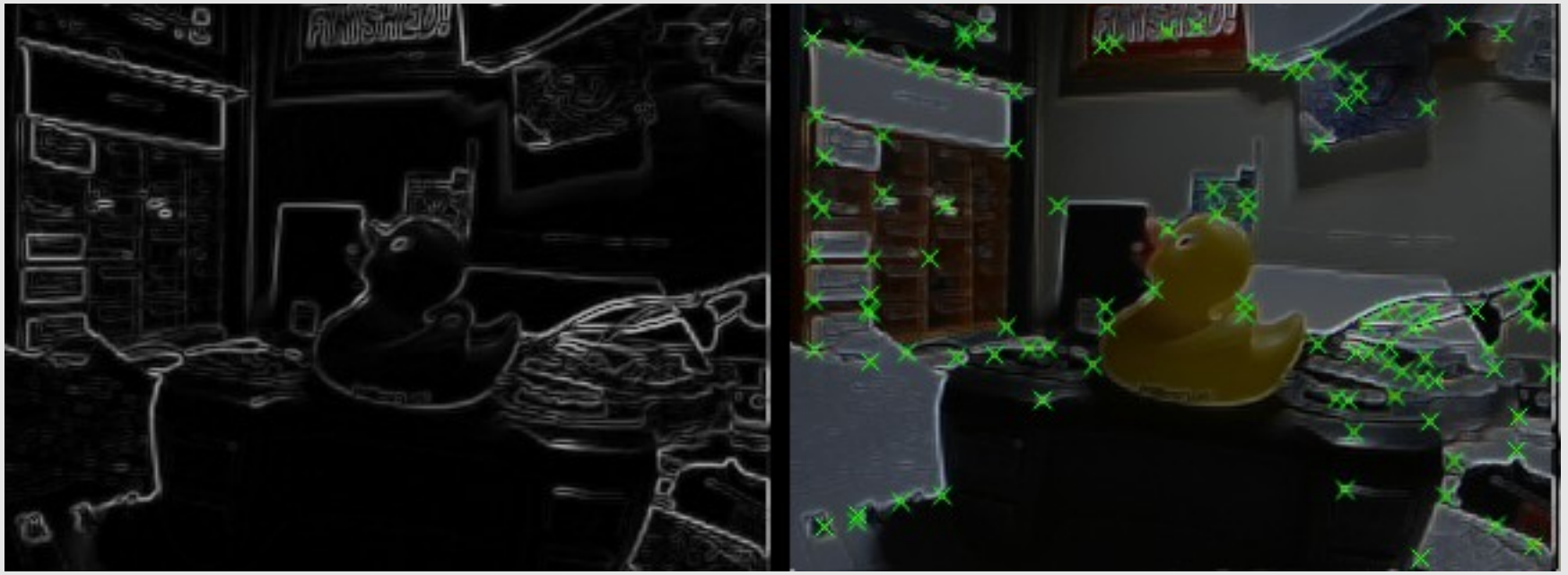
FAST Corner Detection

Edges αλλάζουν γρήγορα

Corners , αλλάζουν με πολυ χαμηλότερο ρυθμό , είναι “μοναδικές”

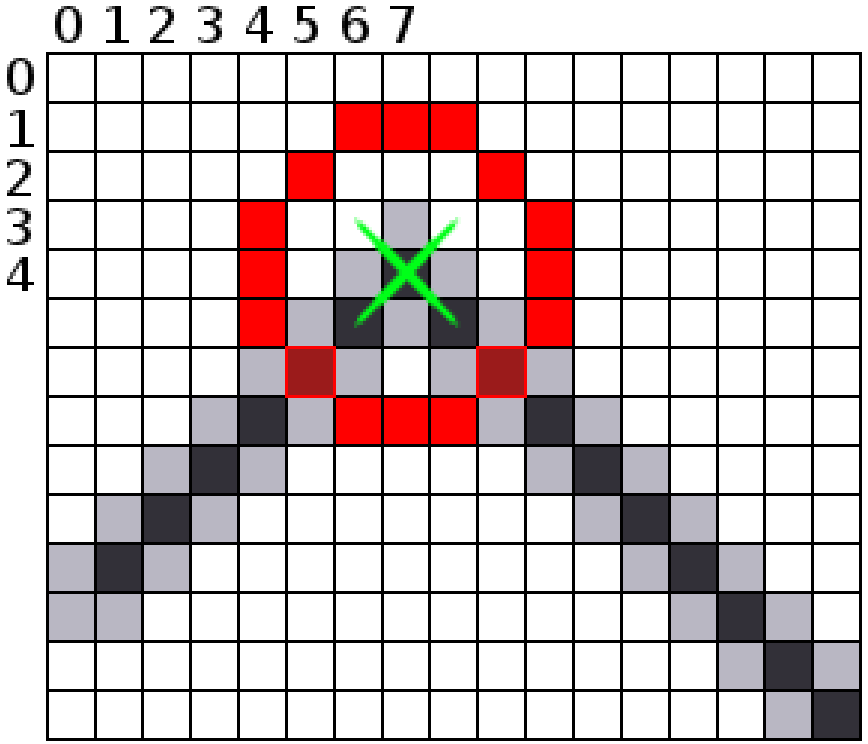


Sparse Feature Tracking



- Features are “unique” points and they are detected on a frame that has highlighted the edges (sobel , second deriv. etc)

Feature / Corner Detection



FAST , Edward Rosten : Κάνουμε Cast έναν κύκλο γύρω από κάθε τονισμένο σημείο σαν ακμή και σε περίπτωση που βρούμε παραπάνω από 3 θετικά pixel (μαζί με το κεντρικό) μαρκάρουμε το σημείο σαν “γωνία”

OpenCV Shi&Tomasi : cvGoodFeaturestoTrack
Υπολογίζουμε τις ελάχιστες ιδιοτιμές που μαρτυρούν μεγάλη αλλαγή έντασης ακμής

$$M = \begin{pmatrix} \sum (dI/dx)^2 & \sum (dI/dx * dI/dy) \\ \sum (dI/dx * dI/dy) & \sum (dI/dy)^2 \end{pmatrix}$$

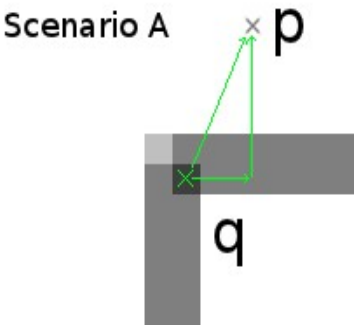
The minimal eigenvalue is then picked since :

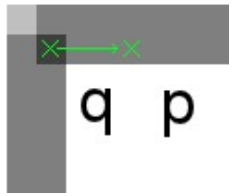
x_1, y_1 *corresponds with* λ_1

x_2, y_2 *corresponds with* λ_2

and compared to a threshold

Το αποτέλεσμα μπορεί να βελτιωθεί ακόμα Περισσότερο με subpixel ακρίβεια..!

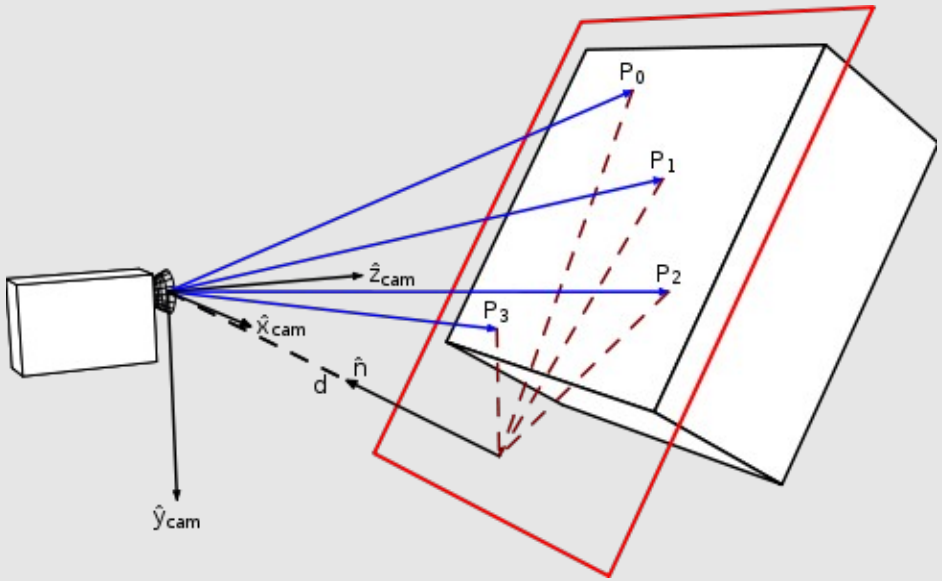




$$\langle \nabla I(p), q - p \rangle = 0$$

The dot product of the Gradient of pixel p with q – p is in both cases zero

Tracking Camera Pose



- Homography estimation for each of the cameras
- Camera Pose Tracking through 2x Homography estimations

Homography estimation

Έχουμε τα σημεία :

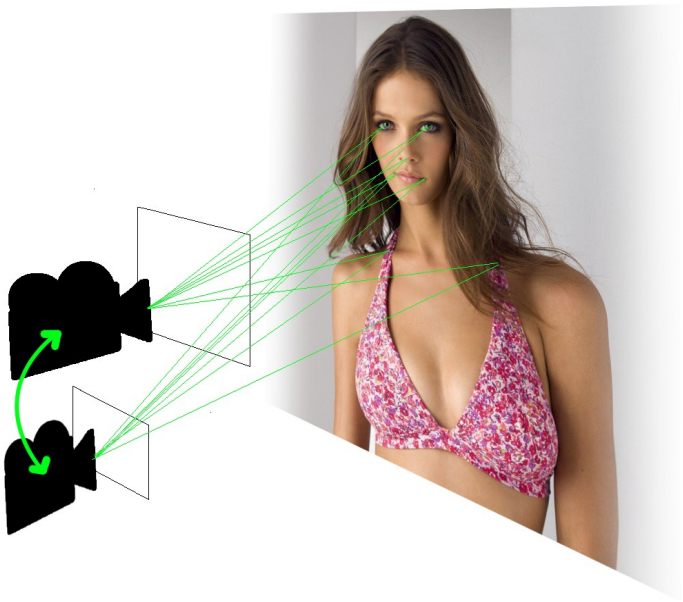
$p_1 (x_1 , y_1 , 1) , p_2 (x_2 , y_2 , 1) \ \dots p_n (x_n , y_n , 1)$

που αντιστοιχούν με τα :

$p'_1 (x'_1 , y'_1 , 1) , p'_2 (x'_2 , y'_2 , 1) \ \dots p'_n (x'_n , y'_n , 1)$

Θέλουμε να βρούμε έναν 3x3 πίνακα H έτσι ώστε

$p'_i = H \ p_i$ για κάθε i από 1 εως n



$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = H \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ z_i \end{bmatrix}$$

performing the multiplication

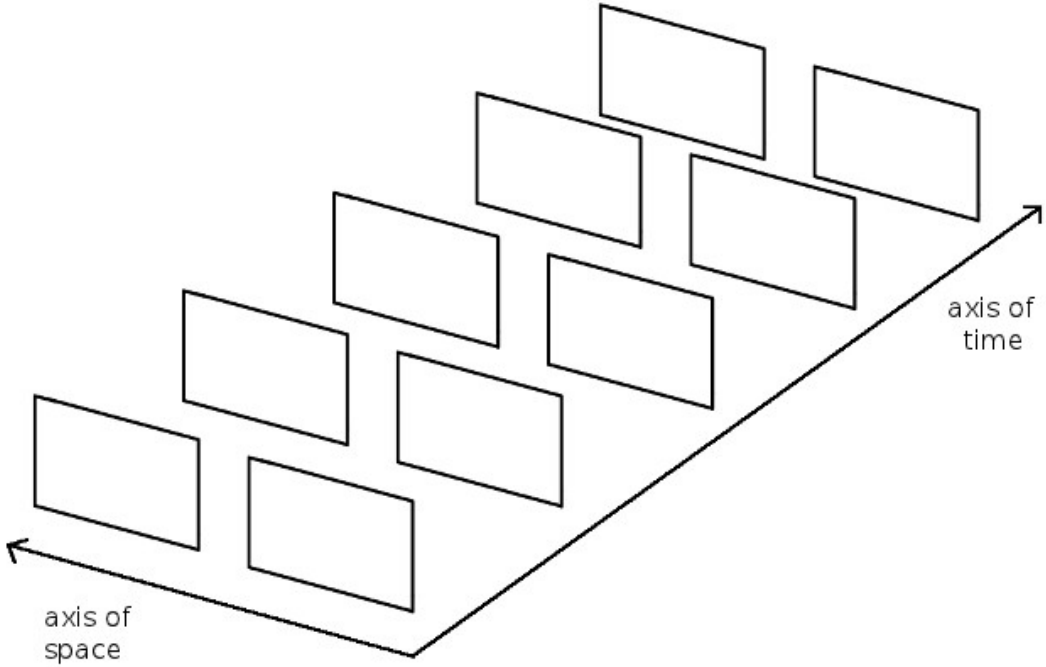
$$\begin{bmatrix} x'_i \\ y'_i \\ z'_i \end{bmatrix} = \begin{bmatrix} h_{11}x_i + h_{12}y_i + h_{13}z_i \\ h_{21}x_i + h_{22}y_i + h_{23}z_i \\ h_{31}x_i + h_{32}y_i + h_{33}z_i \end{bmatrix}$$

for inhomogenous coordinates

$$\begin{bmatrix} x'_i / z'_i \\ y'_i / z'_i \\ 1 \end{bmatrix} = \begin{bmatrix} \dfrac{(h_{11}x_i + h_{12}y_i + h_{13}z_i)}{(h_{31}x_i + h_{32}y_i + h_{33}z_i)} \\ \dfrac{(h_{21}x_i + h_{22}y_i + h_{23}z_i)}{(h_{31}x_i + h_{32}y_i + h_{33}z_i)} \\ 1 \end{bmatrix}$$

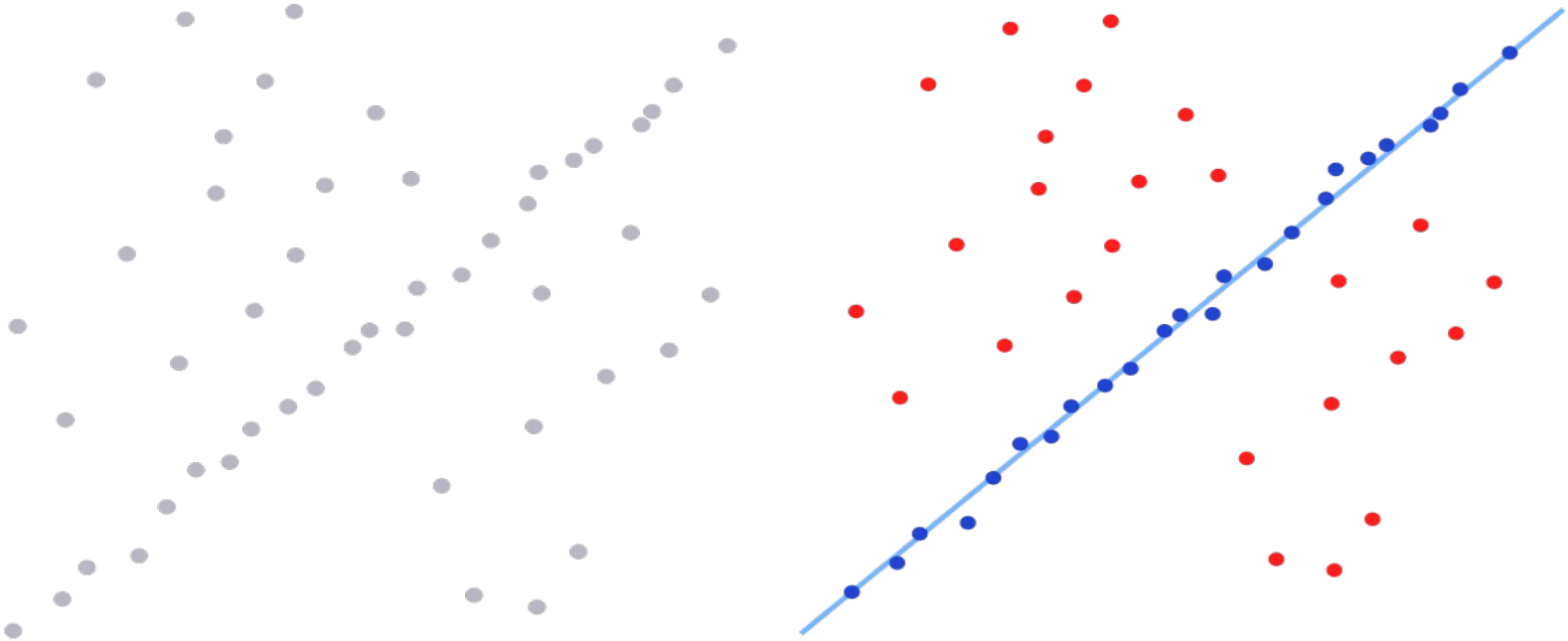
Homography Estimation

- Στον άξονα του χώρου και του χρόνου μπορούμε να συγκρίνουμε με εικόνες μπροστά και πίσω για να μας βοηθήσουν να προσδιορίσουμε την θέση μας



RANSAC

To find the best matching pair we use RANSAC



Left : A collection of points that form a line with a high number of incorrect measurments , Right : RANSAC given criteria to match points along a line can successfully reject outliers and recover the line , Images from Wikipedia , public domain

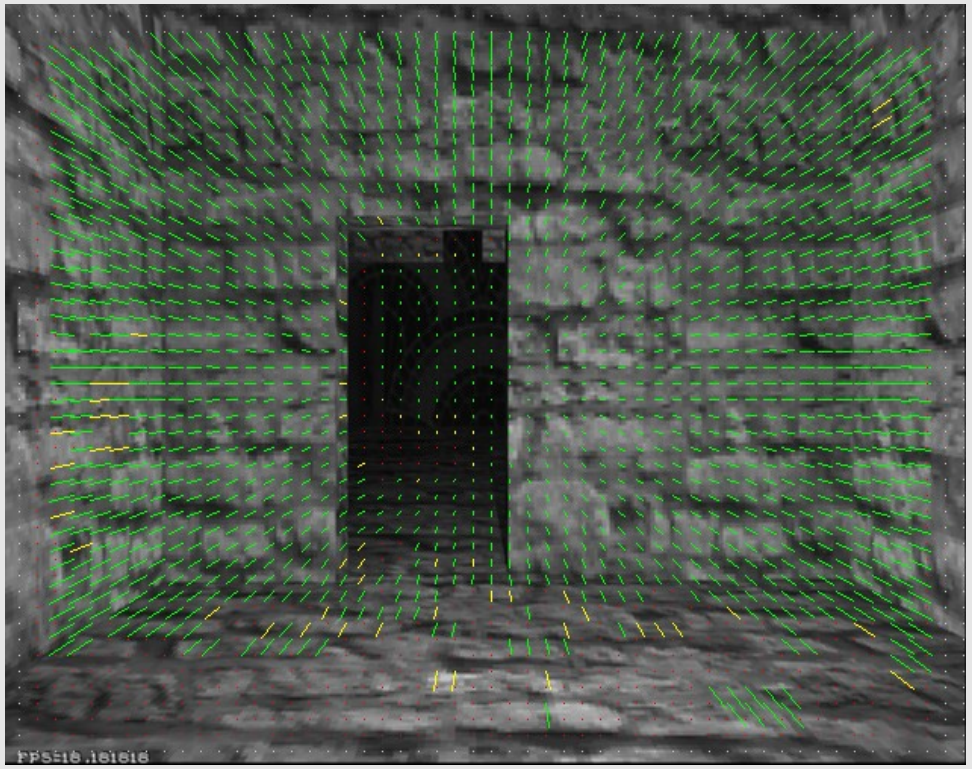
RANSAC Algorithm pseudocode

```
input:
    data - a set of observations
    model - a model that can be fitted to data
    n - the minimum number of data required to fit the model
    k - the number of iterations performed by the algorithm
    t - a threshold value for determining when a datum fits a model
    d - the number of close data values required to assert that a model fits well to data
output:
    best_model - model parameters which best fit the data (or nil if no good model is found)
    best_consensus_set - data points from which this model has been estimated
    best_error - the error of this model relative to the data

iterations = 0
best_model = 0
best_consensus_set = 0
best_error = Infinity
while ( iterations < k )
  {
    maybe_inliers = n randomly selected values from data
    maybe_model = model parameters fitted to maybe_inliers
    consensus_set = maybe_inliers

    for every point in data not in maybe_inliers
        if ( point fits maybe_model with an error smaller than t )
            { add point to consensus_set }

    if ( the number of elements in consensus_set is > d )
        /*this implies that we may have found a good model,
        now test how good it is*/
        this_model = model parameters fitted to all points in consensus_set
        this_error = a measure of how well this_model fits these points
        if (this_error < best_error )
          {
            /*we have found a model which is better than any of the previous ones,
            keep it until a better one is found*/
            best_model = this_model
            best_consensus_set = consensus_set
            best_error = this_error
          }
     ++iterations;
    }
return best_model, best_consensus_set, best_error
```

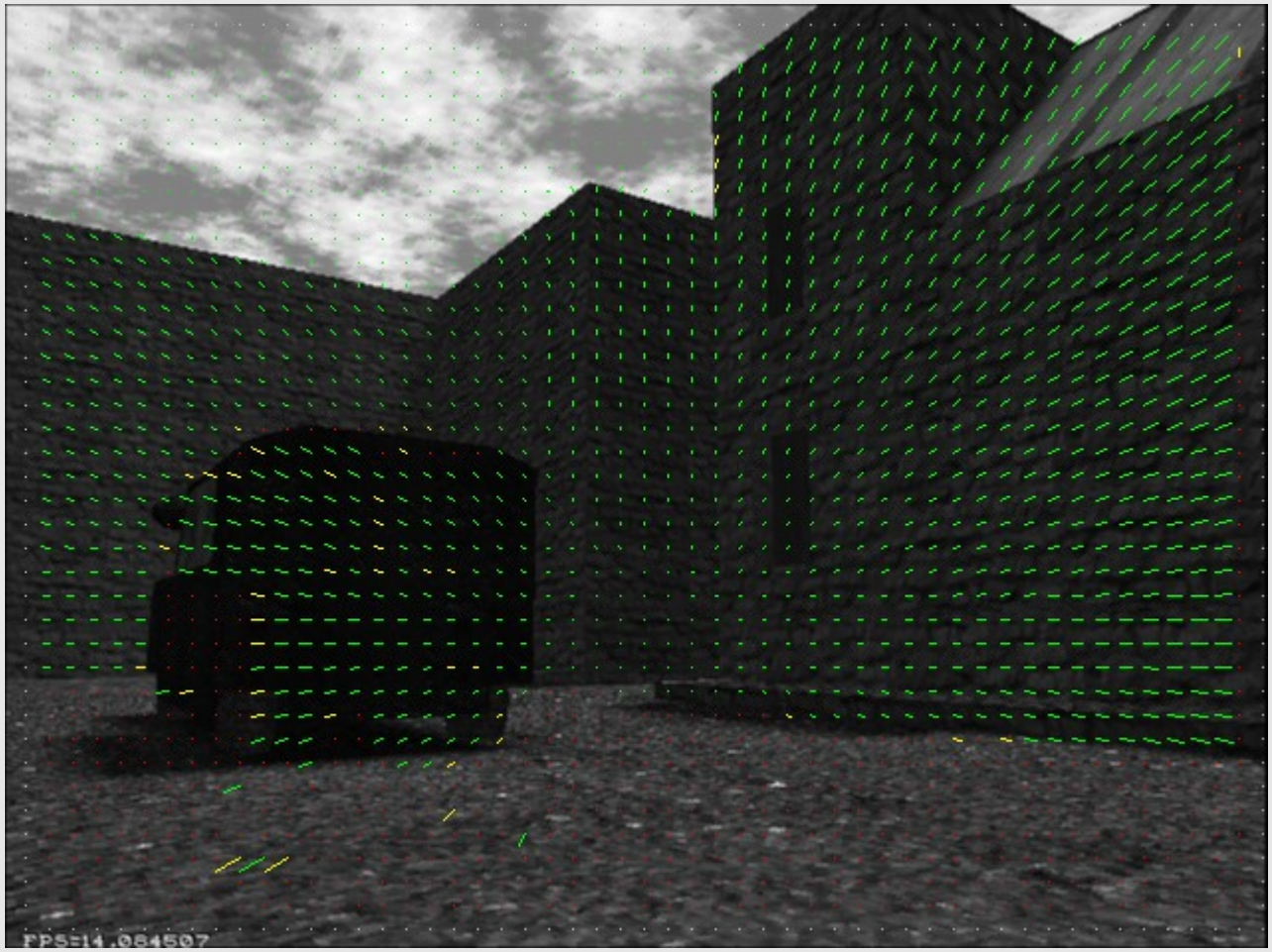
Dense Feature Tracking = Optical Flow

- Optical flow μας δίνει επίσης αποτελέσματα για depth συγκρίνοντας frames μεταξύ τους στον **χρόνο! ,** όχι στον χώρο (Αριστερή/Δεξιά) κάμερα

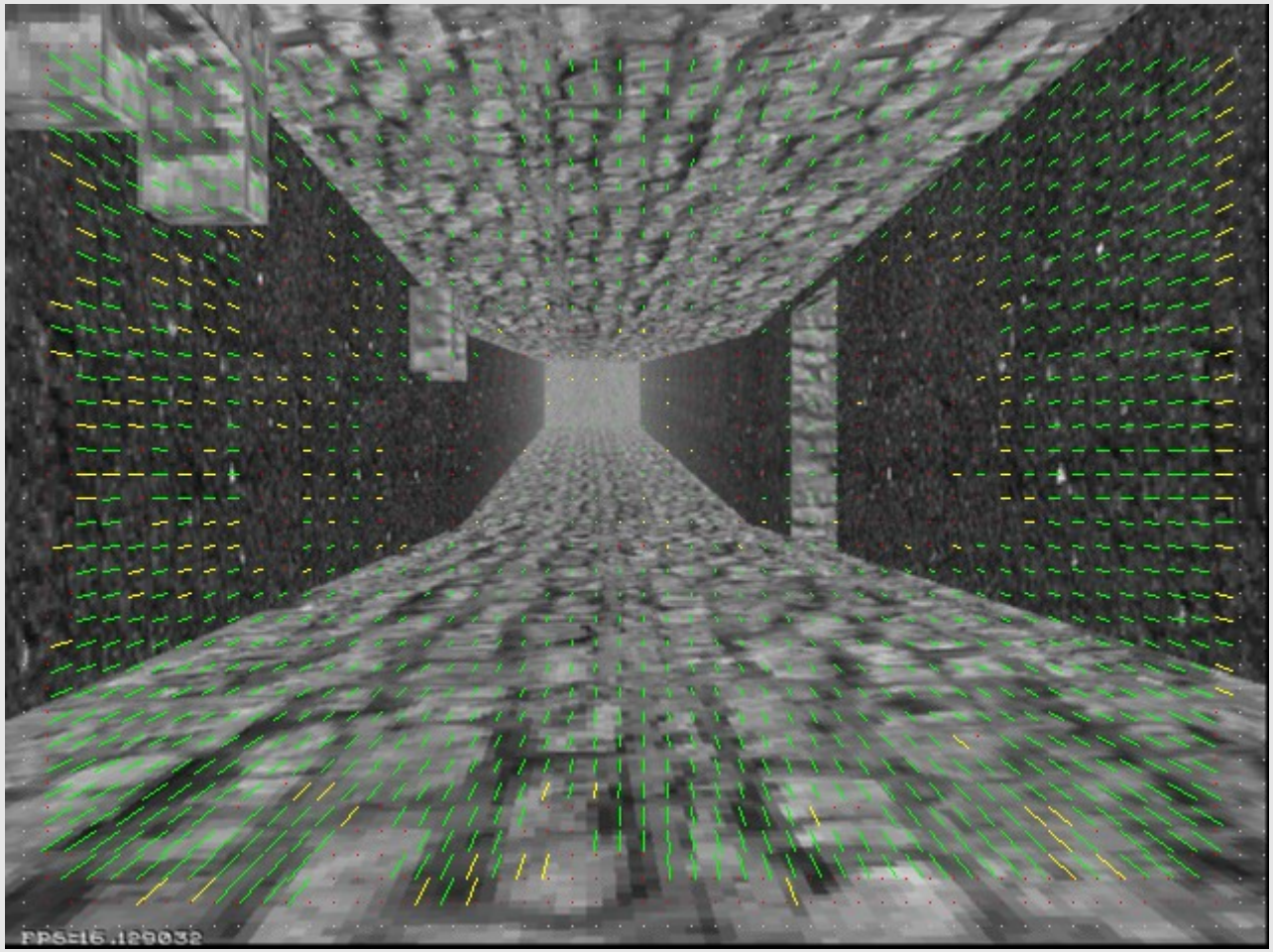
- Η βασική “πράξη” είναι η ίδια , Compare Patches!



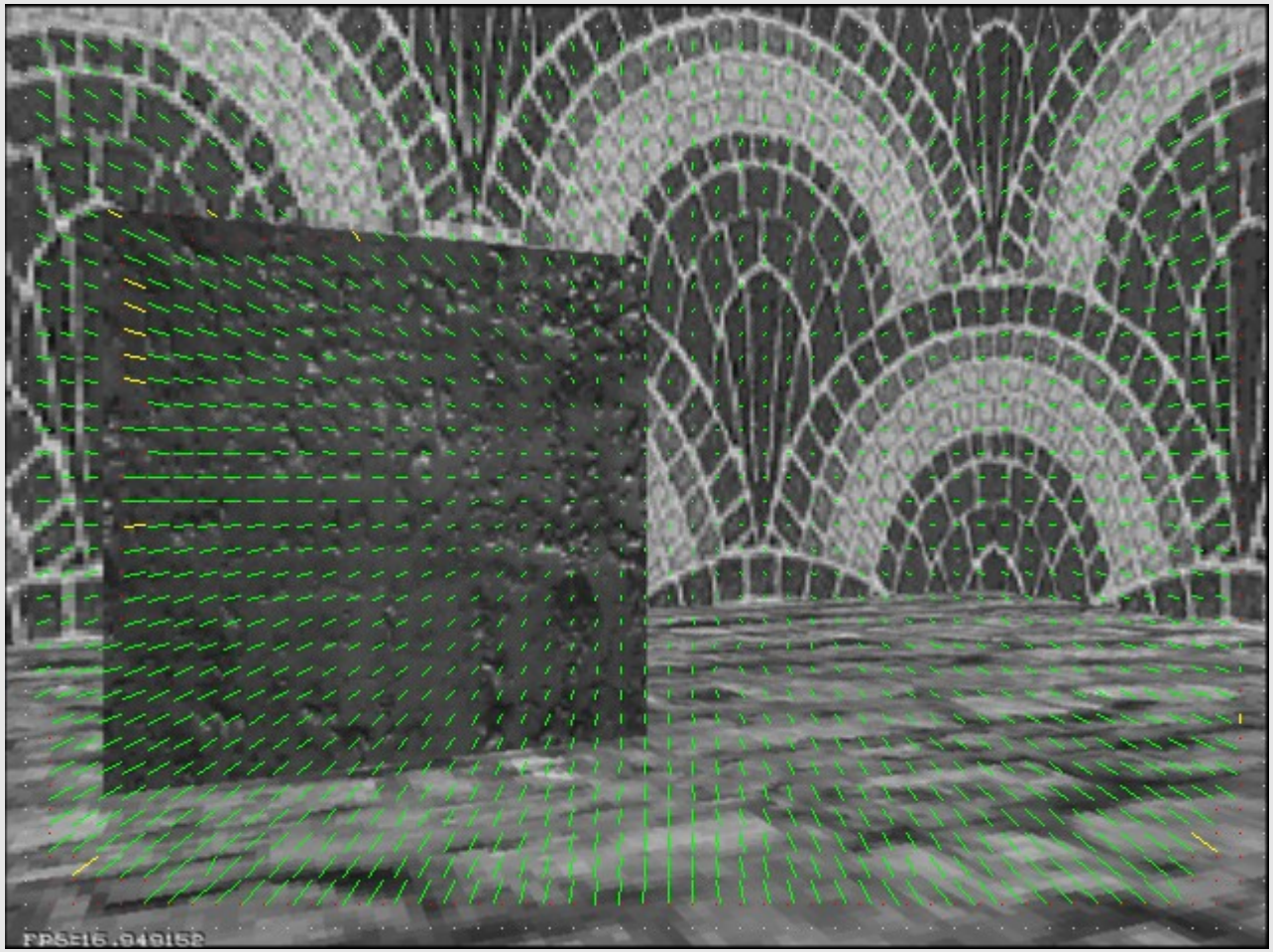
Dense Feature Tracking



Dense Feature Tracking



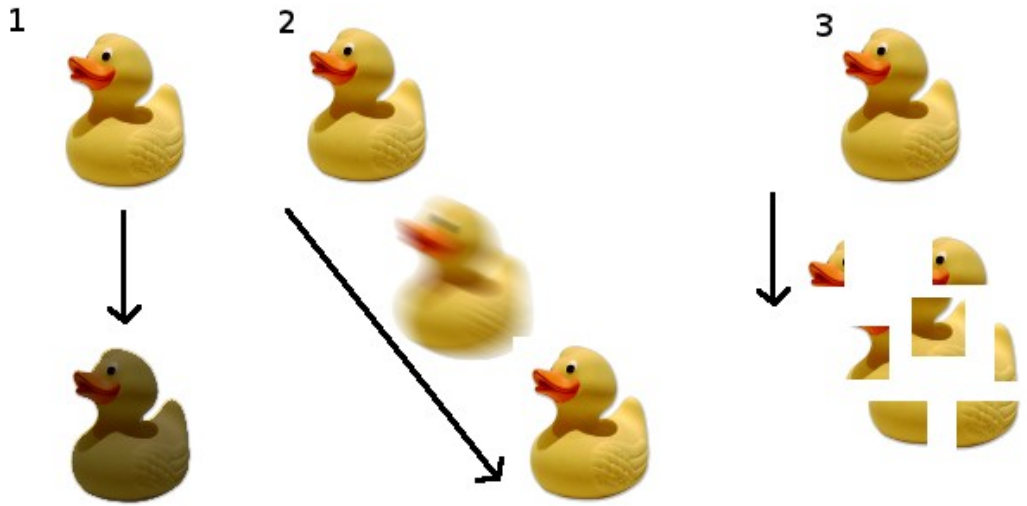
Dense Feature Tracking



Optical Flow

Assumptions (Lukas Kanade pyramid) :

- Brightness Constancy
- Temporal Persistence
- Spatial Coherence



bad instances on the optical flow problem

Optical Flow

Assumptions	Details	Weaknesses
Brightness Constancy	Tracked surfaces retain the same color between frames	shadow changes , illumination changes , blinking lights , camera exposure changes , image noise
Temporal Persistence	The rate of movement is sufficiently small between frames.	fast motion , rapid movement , large computation times between frames lead to slower frame rate and thus larger movement between frames
Spacial Coherence	Large" enough surfaces move in groups	small particles moving in different directions

Brightness constancy $\frac{\partial f(x)}{\partial t}=0$

Temporal persistence $I(x,y,t)=I(x+\Delta x,y+\Delta y,t+\Delta t)$

$$I(x+\Delta x,y+\Delta y,t+\Delta t)=I(x,y,t)+\frac{\partial I}{\partial x}\Delta x+\frac{\partial I}{\partial t}\Delta y+\frac{\partial I}{\partial t}\Delta t=0$$

dividing with Δt *gives us*

$$(I(x+\Delta x,y+\Delta y,t+\Delta t)-I(x,y,t))\frac{1}{\Delta t}=\frac{\partial I}{\partial x}\frac{\Delta x}{\Delta t}+\frac{\partial I}{\partial t}\frac{\Delta y}{\Delta t}+\frac{\partial I}{\partial t}=0$$

$$\ldots \qquad =\frac{\partial I}{\partial x}V_x+\frac{\partial I}{\partial t}V_y+\frac{\partial I}{\partial t}=0$$

$$\frac{\partial I}{\partial x}V_x+\frac{\partial I}{\partial t}V_y=-\frac{\partial I}{\partial t}$$

Optical Flow

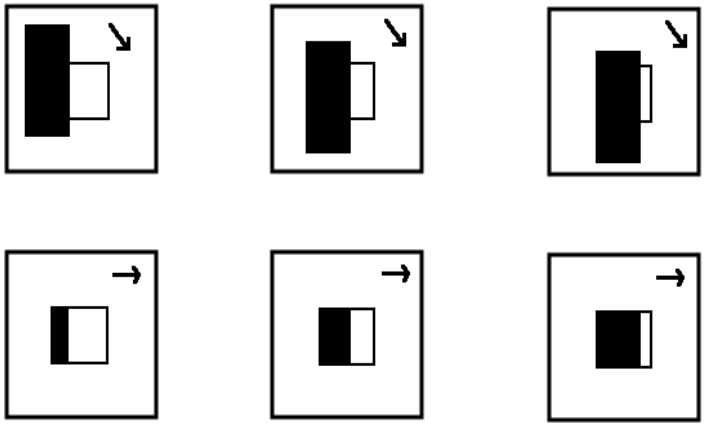
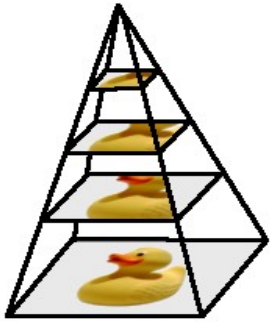


Illustration : The aperture problem.
First row : We have a black rectangle moving diagonally over a small detection window
Second row : Inside the detection window movement appears to be horizontal



A gaussian window pyramid

For a 5x5 window we have the following over constrained system of equations

$$\begin{bmatrix} I_x(P_1) & I_y(P_1) \\ I_x(P_2) & I_y(P_2) \\ \dots & \\ I_x(P_{24}) & I_y(P_{24}) \\ I_x(P_{25}) & I_y(P_{25}) \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} I_t(P_1) \\ I_t(P_2) \\ \dots \\ I_t(P_{24}) \\ I_t(P_{25}) \end{bmatrix}$$

$$A \; v = b$$

This is then solved using a least squares minimization

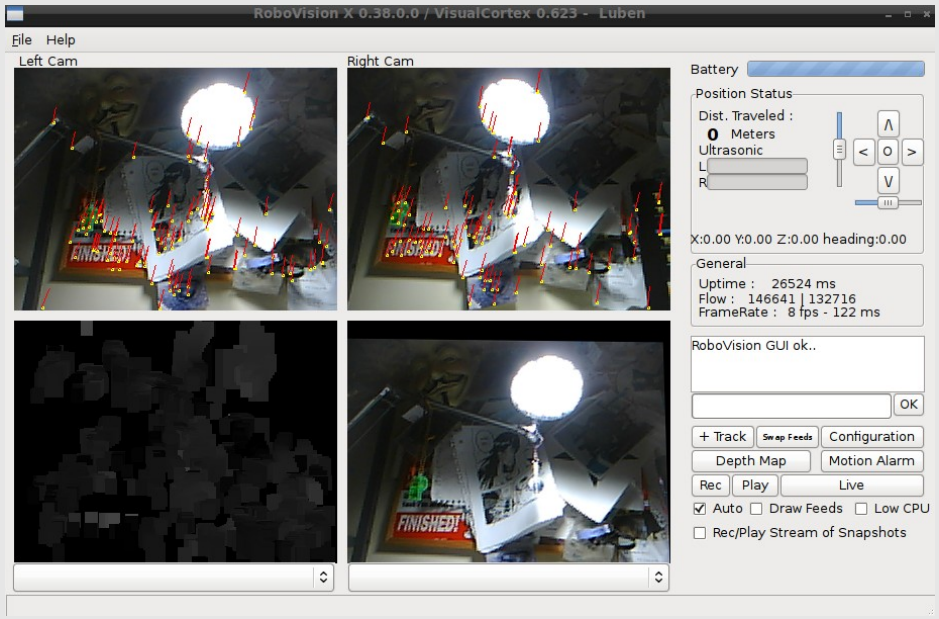
$$A \; v = b$$
$$A^T A \; v = A^T b$$
$$v = \frac{A^T b}{(A^T A)}$$

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_i)^2 & \sum_i I_x(p_i) I_y(p_i) \\ \sum_i I_x(p_i) I_y(p_i) & \sum_i I_y(p_i)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_i) I_t(p_i) \\ -\sum_i I_y(p_i) I_t(p_i) \end{bmatrix}$$

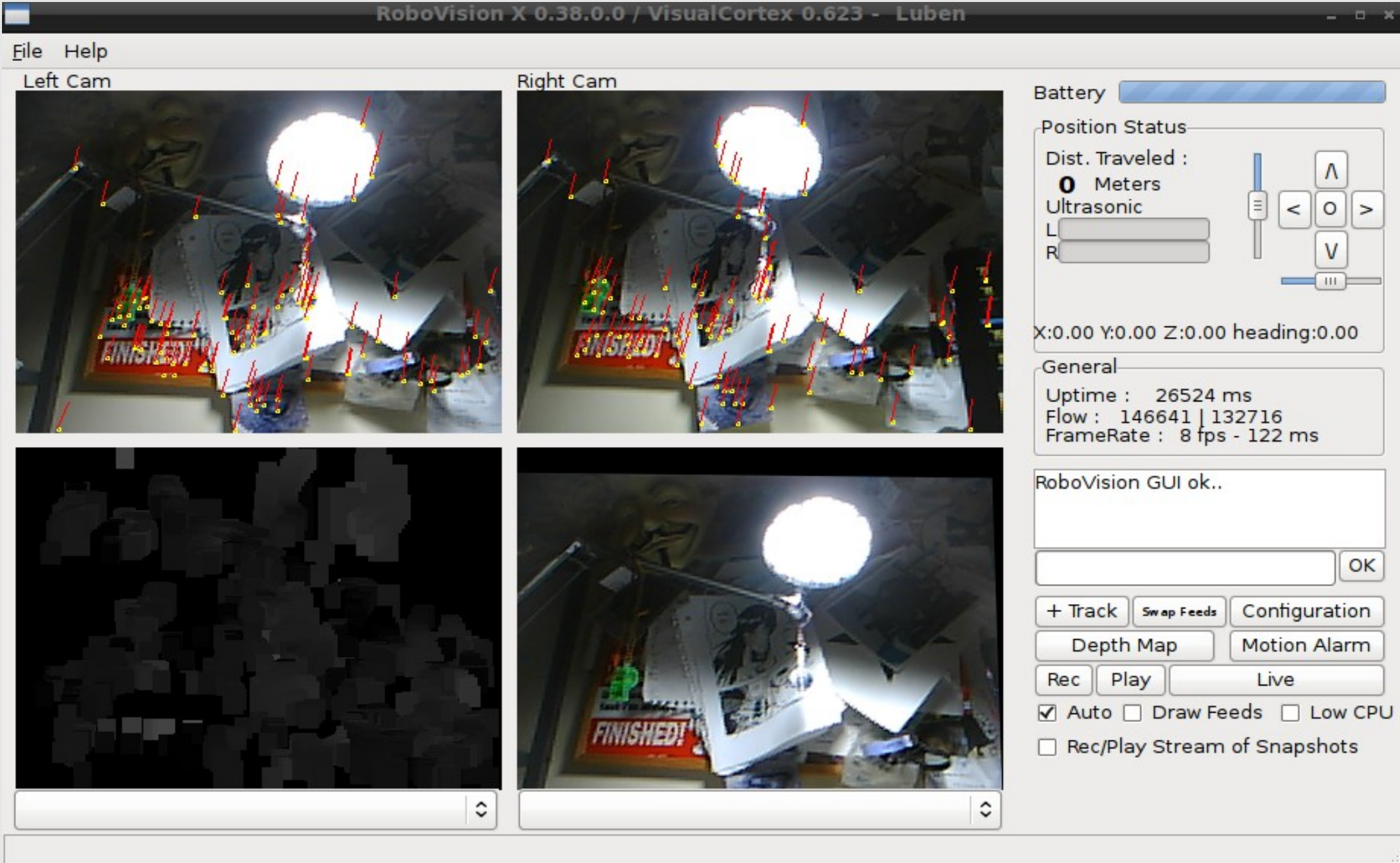
World 3D SLAM

- Ξέρουμε με ποιο πίνακα πολλαπλασιάστικαν τα σημεία έτσι ώστε να κατάληξαν σε αυτή τι θέση

  Αρα τι κίνηση έκανε η κάμερα



World 3D


RoboVision X 0.38.0.0 / VisualCortex 0.623 - Luben
File Help
Left Cam
Right Cam
Battery
Position Status
Dist. Traveled :
0 Meters
Ultrasonic
L
R
X:0.00 Y:0.00 Z:0.00 heading:0.00
General
Uptime : 26524 ms
Flow : 146641 | 132716
FrameRate : 8 fps - 122 ms
RoboVision GUI ok..
OK
+ Track
Swap Feeds
Configuration
Depth Map
Motion Alarm
Rec
Play
Live
Auto
Draw Feeds
Low CPU
Rec/Play Stream of Snapshots

Visual Cortex

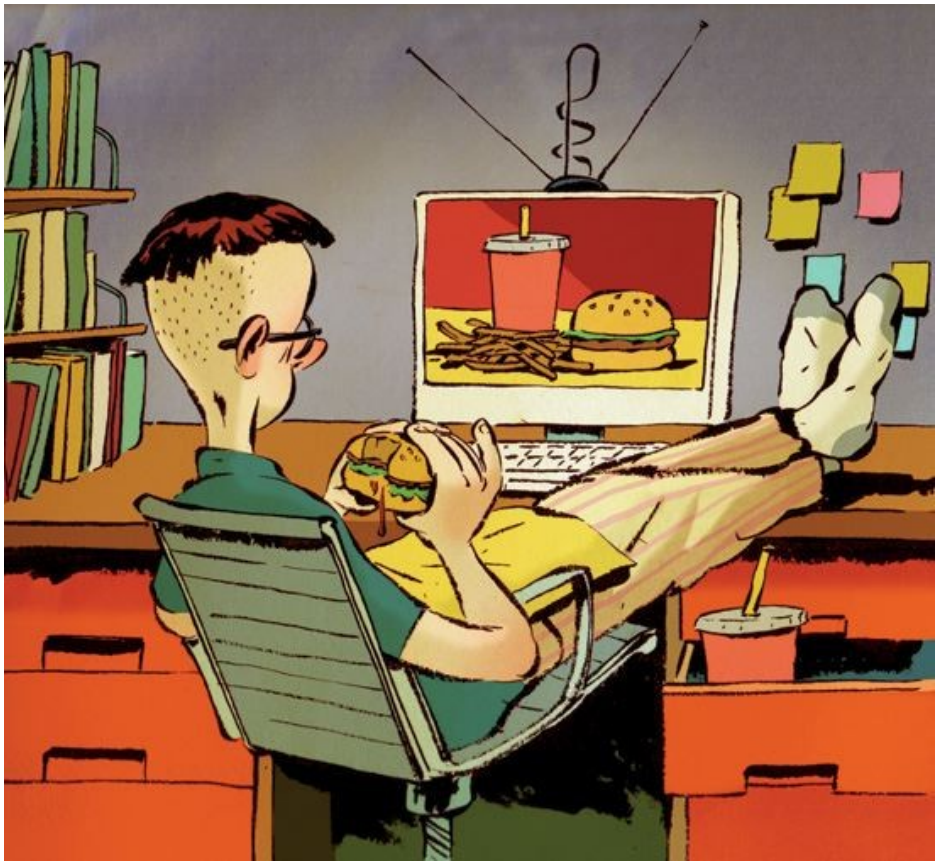
Άλλη πιθανή χρήση ..

Helping the blind in the city

Αντί για το μπαστούνι , ένα σύστημα
το οποίο να αντιλαμβάνεται εμπόδια
κινδύνους (αυτοκίνητα) και να βοηθά στην
Στην ασφαλή περιήγηση στην πόλη

Σαν GPS πεζών

Visual Cortex
Άλλη πιθανή χρήση ..



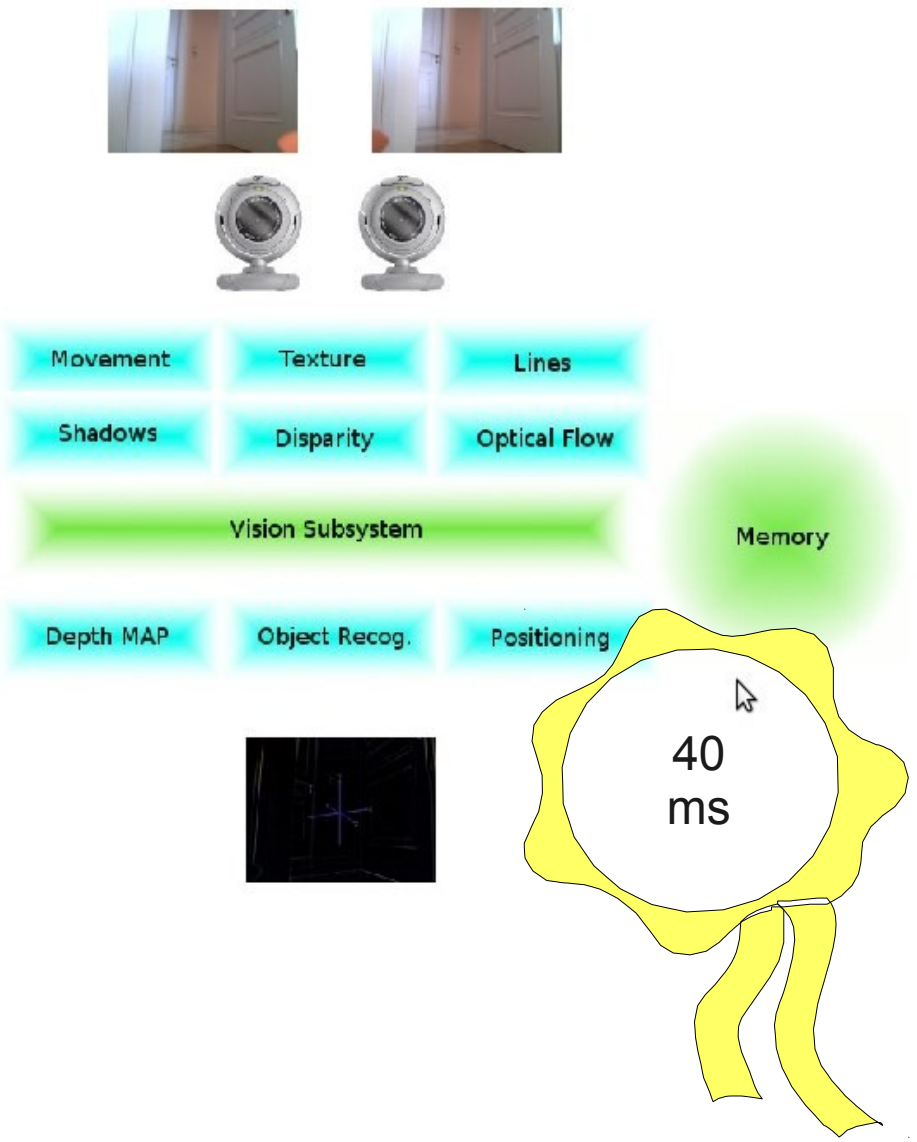
- TV tuners use the same interface (V4L2)
- TV Spam filter , updated over internet
- TV ads do not dynamically change
- HD-TV Spam-Filter box :D
- Patch matching ..
- Υποθέτοντας ότι έχει νόημα κάποιος να βλέπει τηλεόραση..

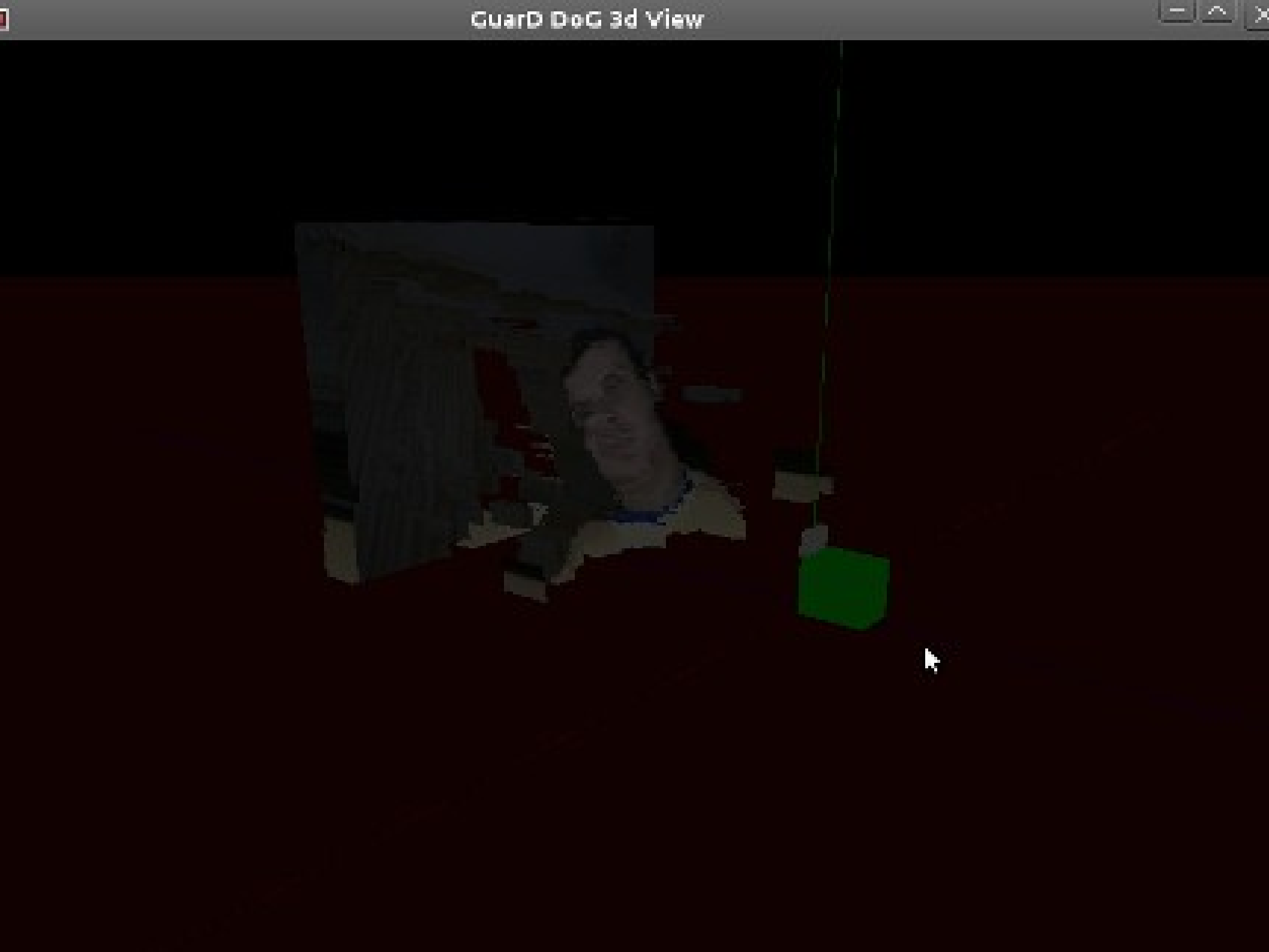
Visual Cortex

Ιδανικά :

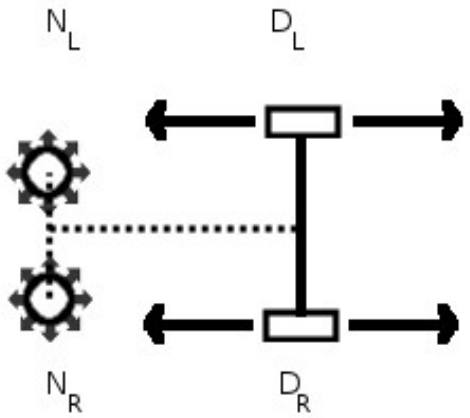
- Optical Flow tracking σε κάθε κάμερα
- Feature extraction
- Disparity Mapping
- Light/Shadow Depth estimation
- Facedetection

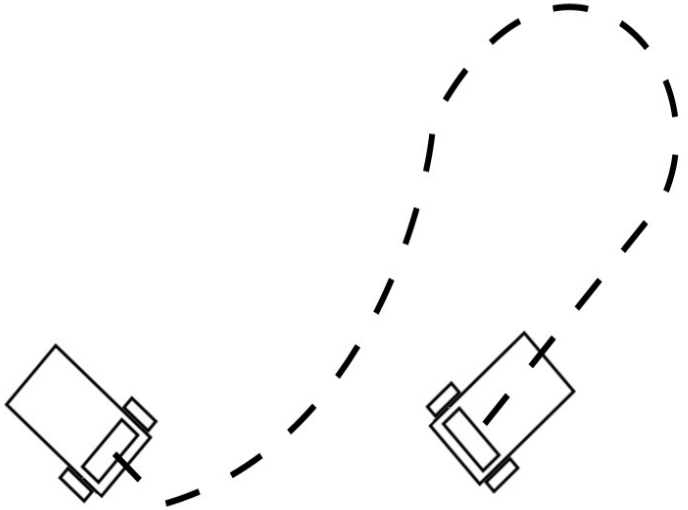
κτλ

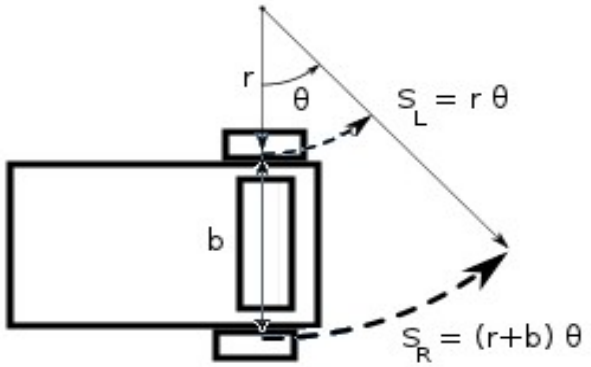


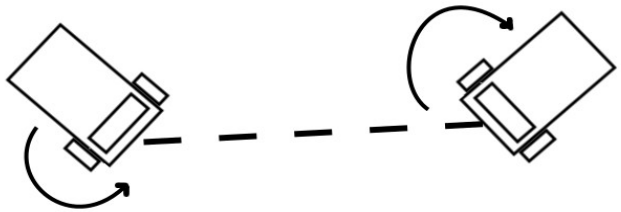

GuarD DoG 3d View

Driving with a Two Wheeled configuration

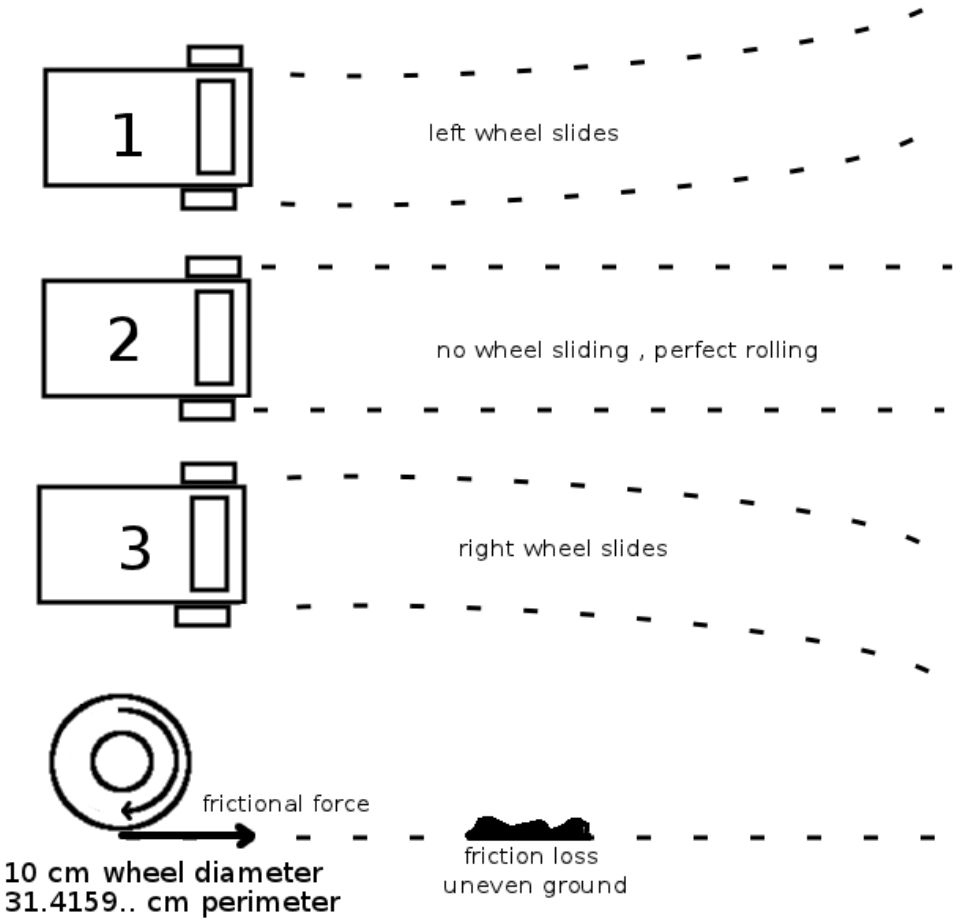








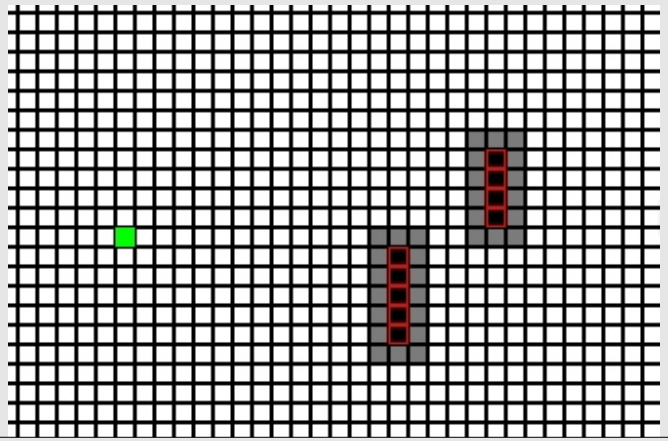
Driving with a Two Wheeled configuration



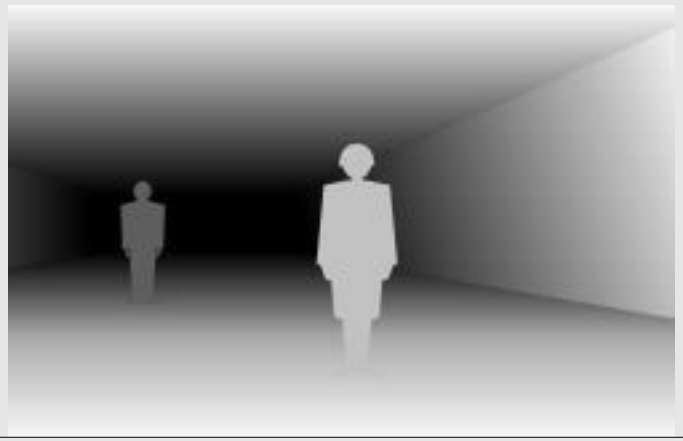
World Mapping

Προς το παρόν η υλοποίηση λαμβάνει απλά την θέση των encoders των μοτέρ (dead reckoning) 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!

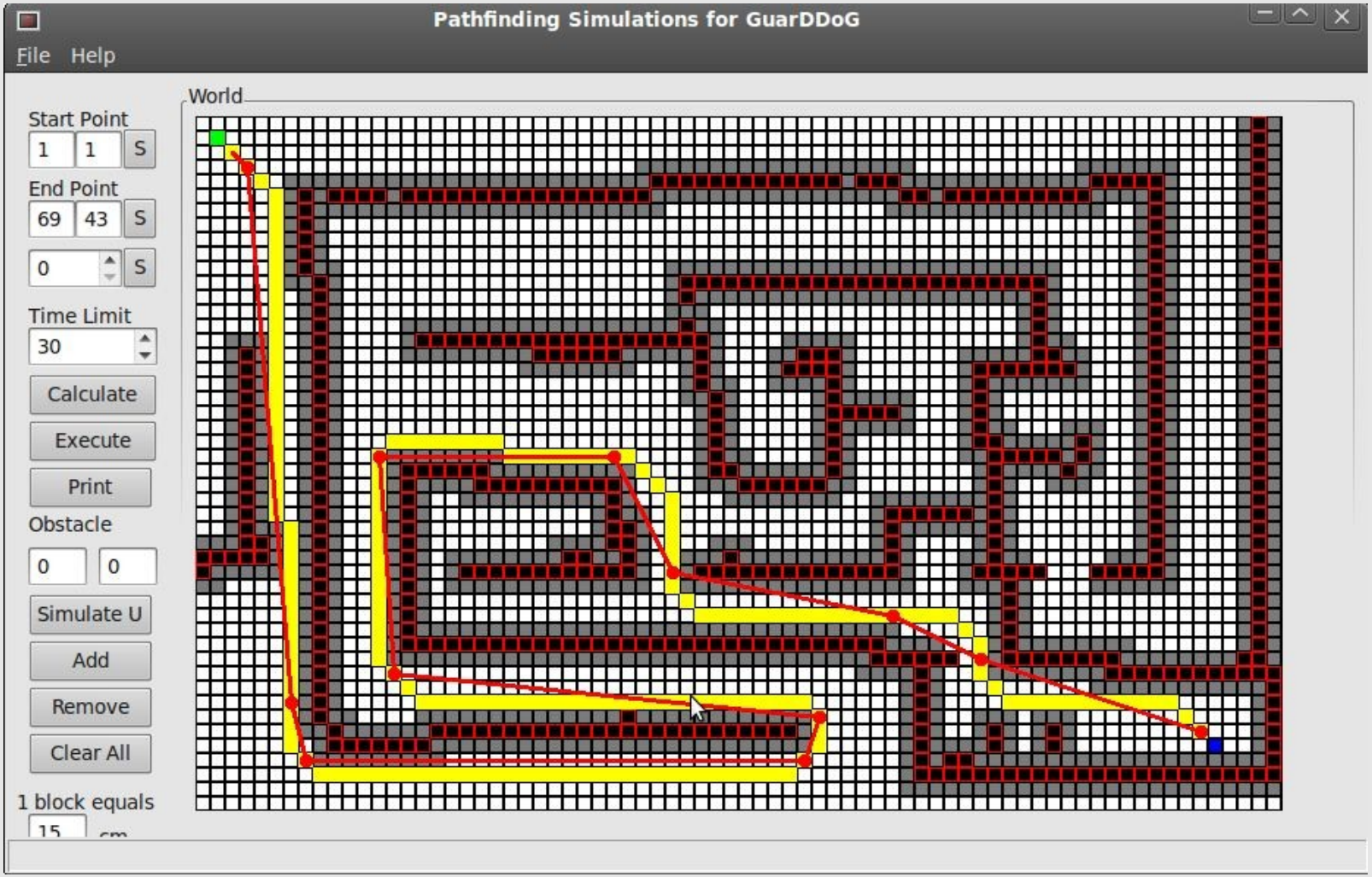


2D GuarddoG represenatation



Much better 3d representation

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) ;

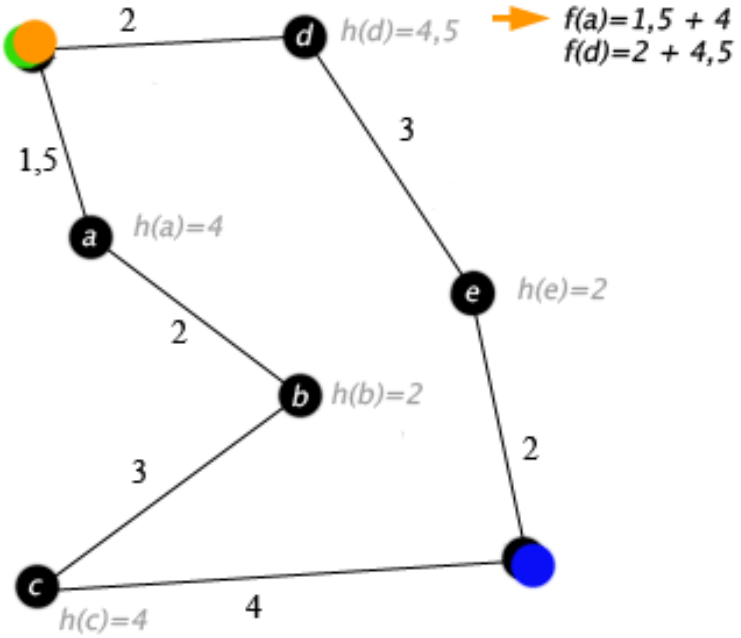
int GetRouteWaypoint(struct Map * themap,unsigned int agentnum,unsigned int count,unsigned int *x,unsigned int *y) ;

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

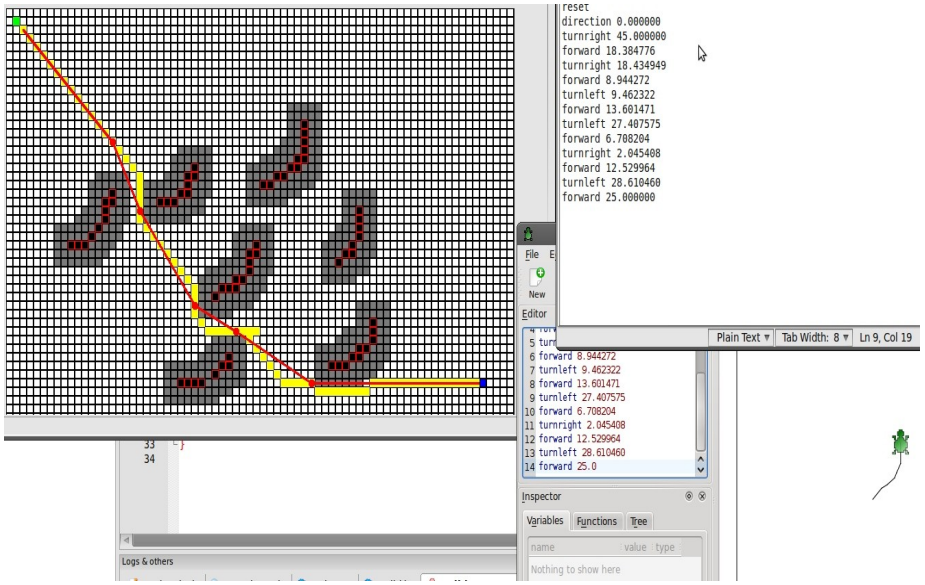
* Manhattan distance heuristic + στροφές γιατί κινούμαστε σε “πραγματικό” χώρο !

* 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:

$$| h(x) - h^*(x) | = O(\log h^*(x))$$

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 */

Turnleft 20

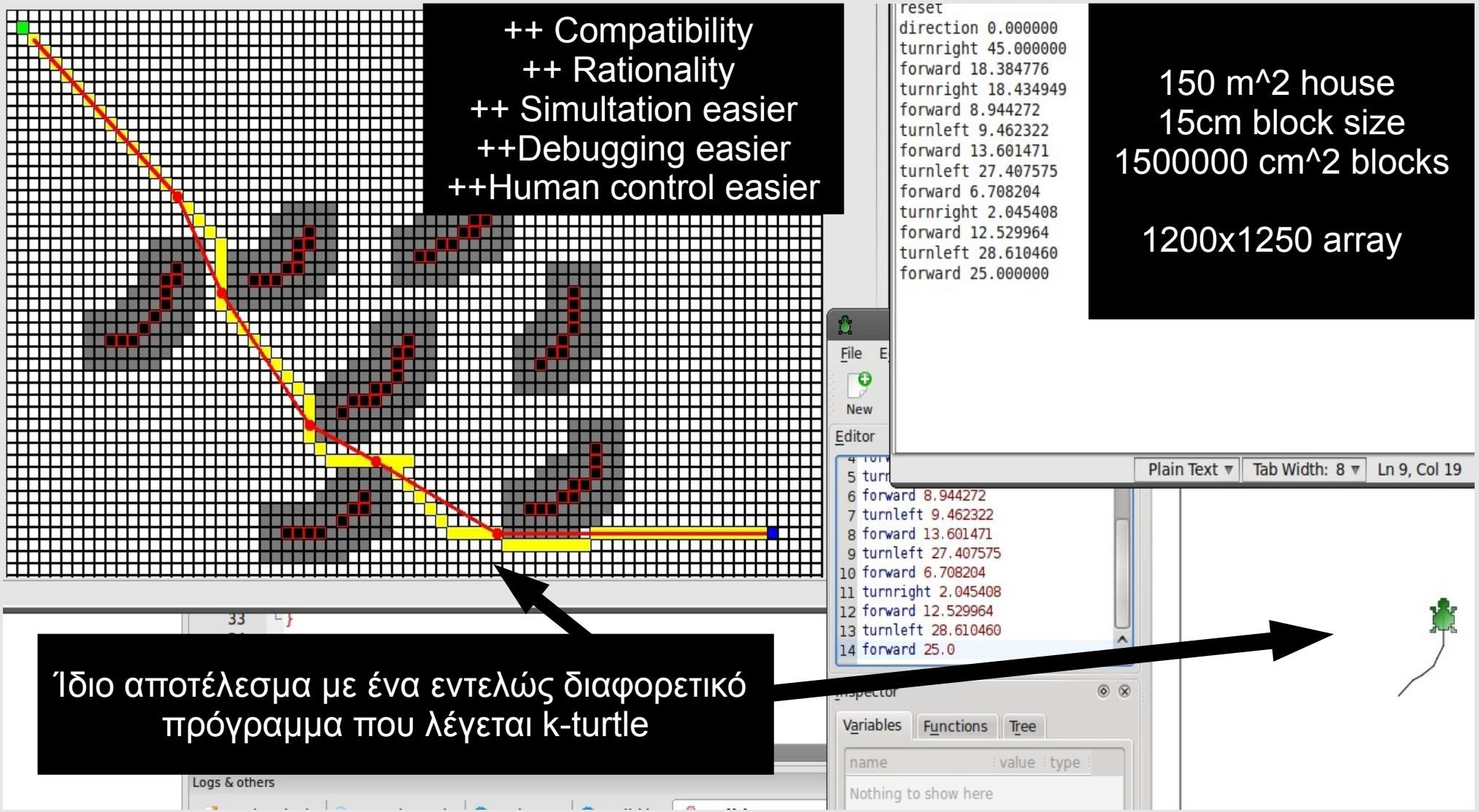
Backward 5

Κτλ κτλ..

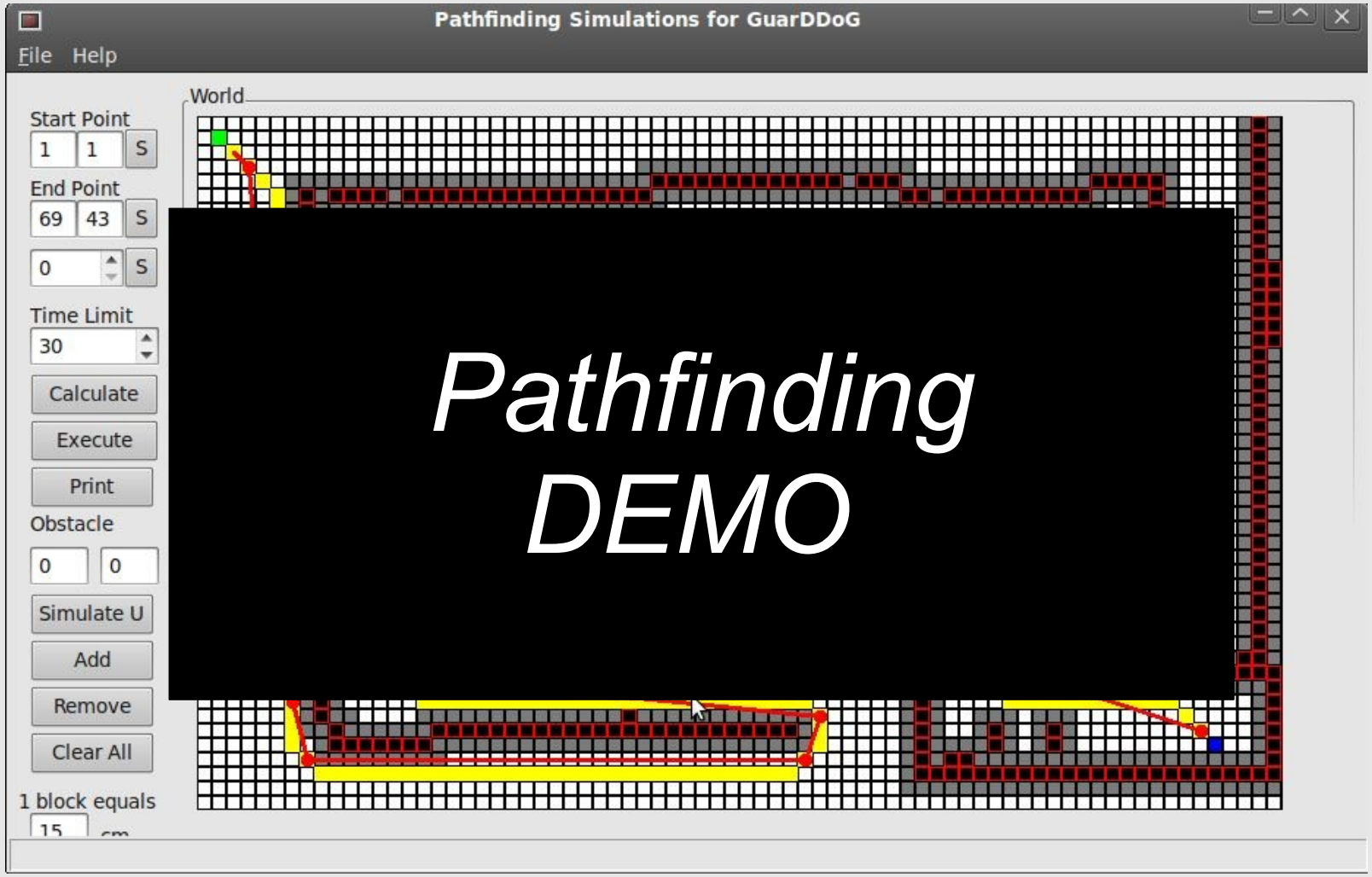
Σημείο 0 στον χάρτη θεωρείται η άκρη πάνω πάνω και αριστερά του χώρου

Σε κάθε κίνηση σημείο 0 θεωρείται η αρχική θέση της κίνησης , από εκεί προστίθεται στο στίγμα του GuarddoG

World Mapping
LOGO Output



World Mapping



SLAM

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 SLAM

Ο σωστός τρόπος για να γίνει ονομάζεται **SLAM(Simultaneous localization and mapping)**
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

SLAM

The method is an estimation of the Bayesian filtering problem where we try to approximate the probability of a point X given n sensor readings , or $P(x_n|Z^n)$Prediction phase (using only motion data) uses

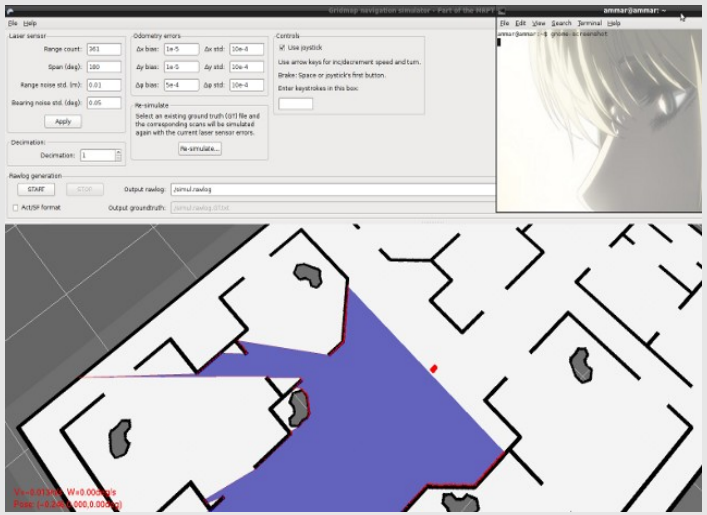
$$P(x_n|Z^{n-1}) and\ P(x_n|x_{n-1}, U_{n-1})$$

obtained by integration

$$P(x_n|Z^{n-1}) = \int P(x_n|x_{n-1}, U_{n-1})\, P(x_n|Z^{n-1})\, dx_{n-1}$$

The Update phase utilizes the sensory input Z to produce

$$P(x_n|Z^n) = \frac{P(z_n|x_n)\, P(x_n|Z^{n-1})}{P(z_n|Z^{n-1})}$$

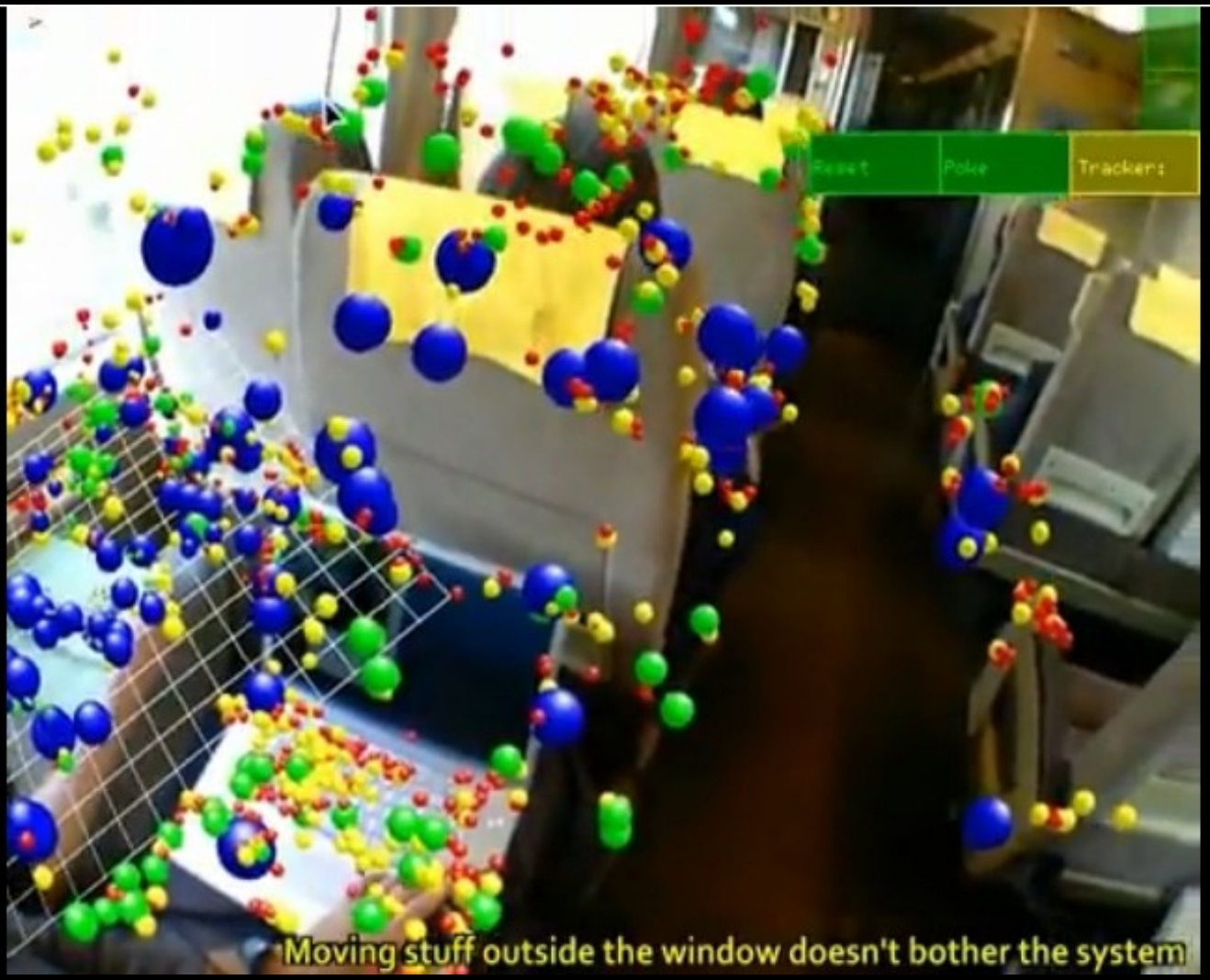


SLAM

Monte Carlo Localization Algorithm

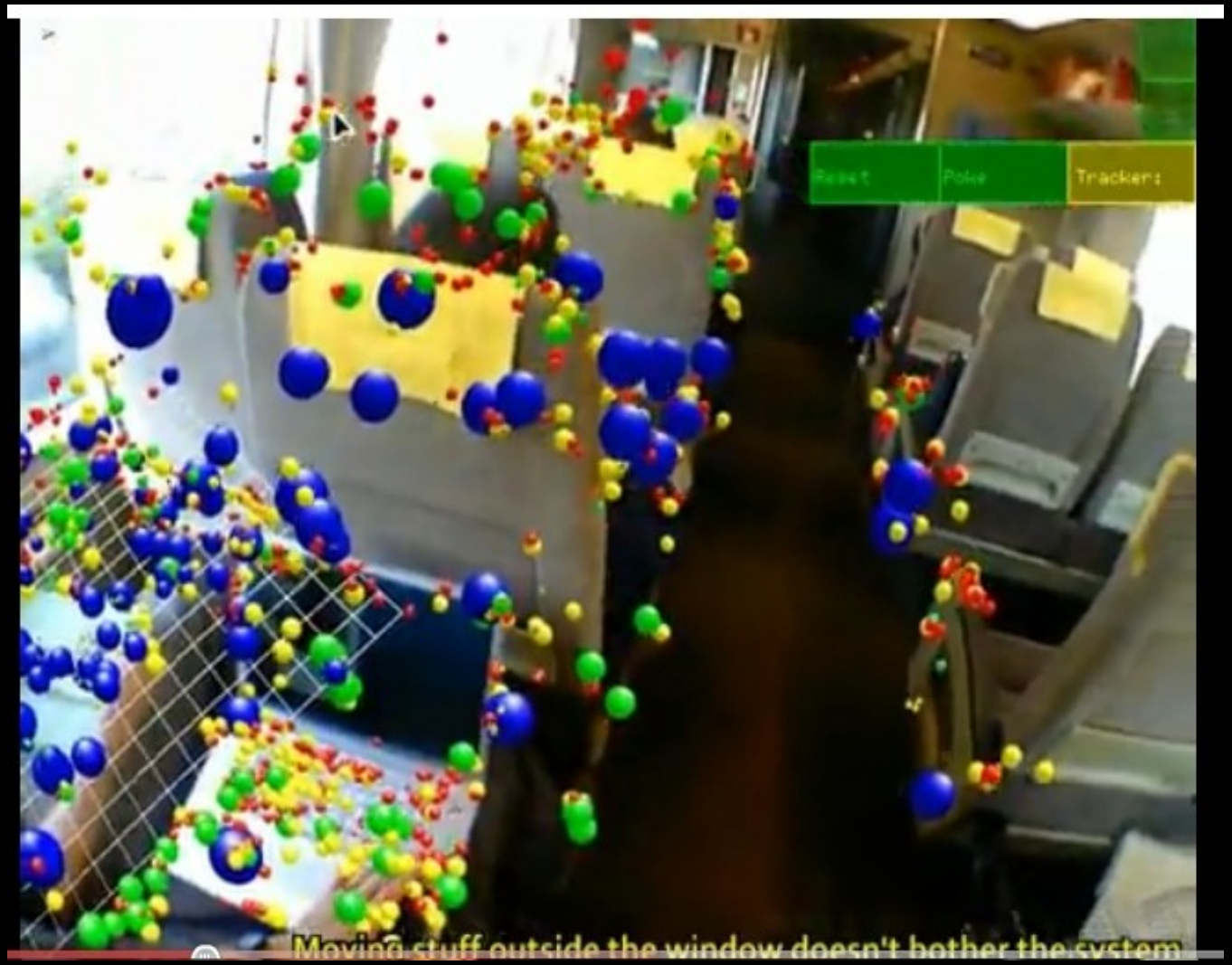
```
    input:
        Distance Ut
        Sensor reading Zt
        Sample set St={(Xt(i),Wt(i))|i=1,...,n}

  //PREDICTION PHASE
    for (i=1; i<n; ++i)  // Update the current set of samples
       {
         Xt  = updateDist(Xt, Ut) // Compute new location using motion model
         Wt(i) = prob(Zt|Xt(i))   // Compute new weighted probability
       }
  //UPDATE PHASE
    St+1  =  null
    for (i=1; i<n; ++i) // Resample to get the next generation of samples
      {
         Sample an index j from the distribution given by the weights in St
         Add (Xt(j), Wt(j)) to St+1  // Add sample j to the set of new samples
      }
    return St+1
```

World 3D SLAM



MRPT Project Screenshot

World 3D SLAM



MRPT Project Screenshot

World 3D SLAM

MRPT Project Screenshot

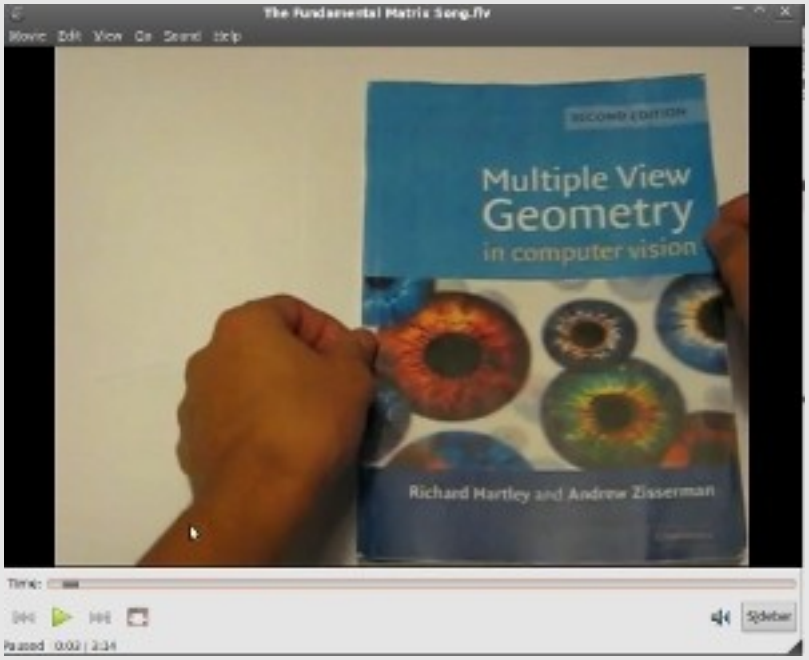
World 3D SLAM

MRPT Project Screenshot

World 3D SLAM

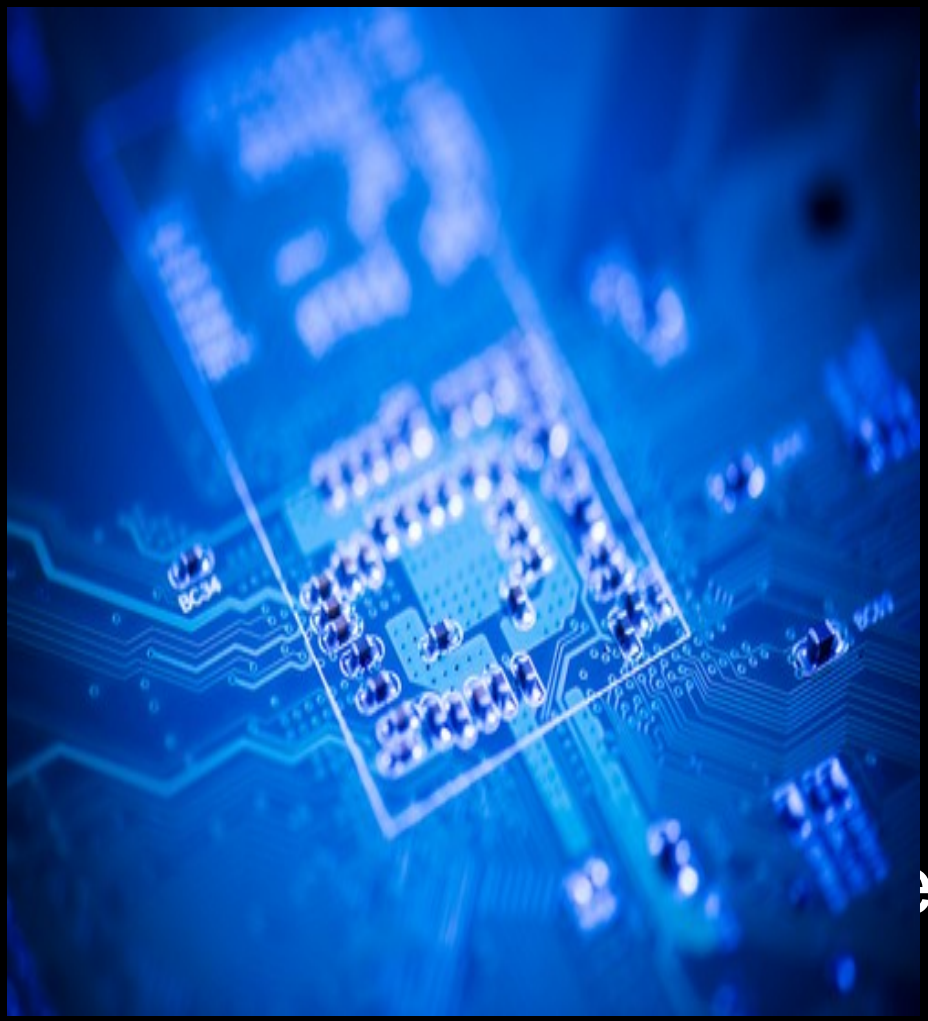
MRPT Project Screenshot

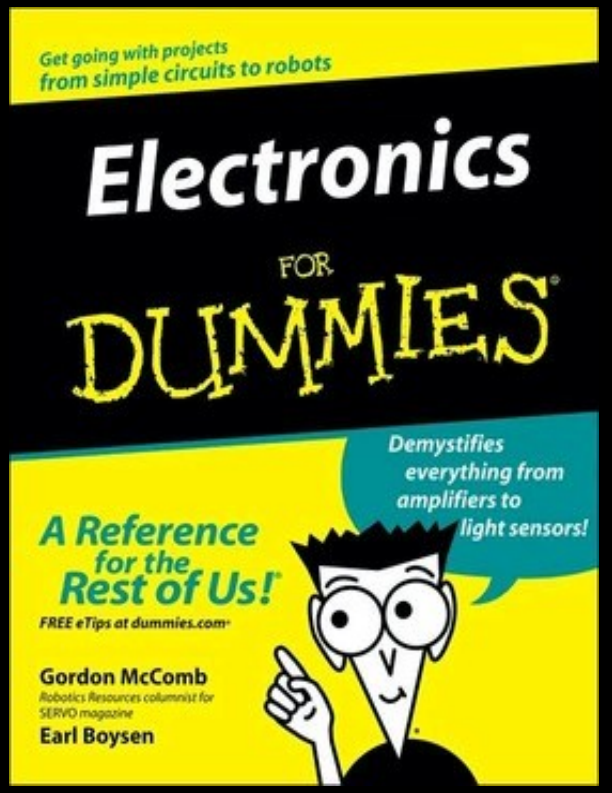
Word3D SLAM



http://tinyurl.com/fundamentalmatrix

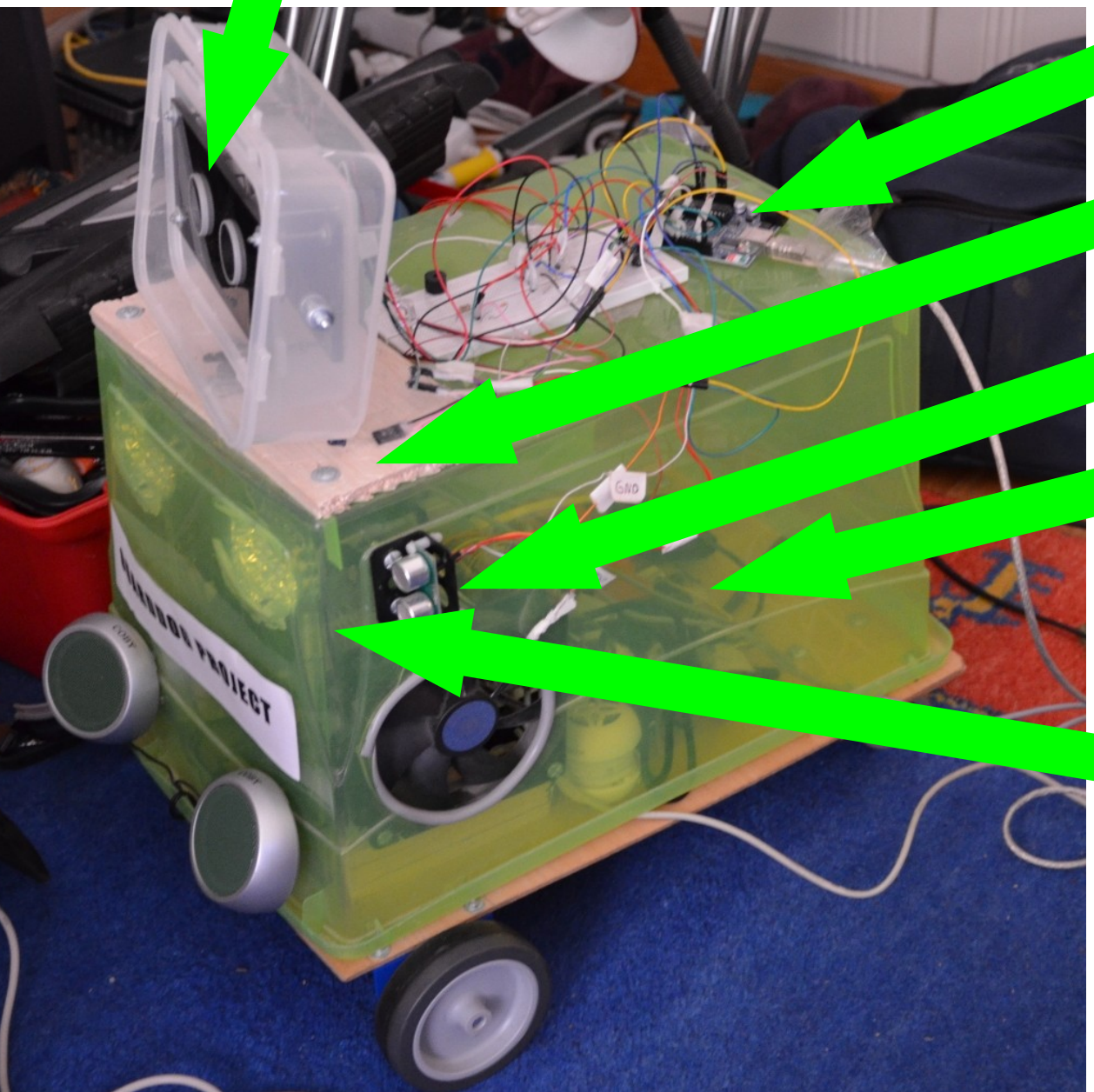
Πρόβλημα #3




Get going with projects
from simple circuits to robots
Electronics
FOR
DUMMIES
Demystifies
everything from
amplifiers to
light sensors!
A Reference
for the
Rest of Us!
FREE eTips at dummies.com
Gordon McComb
Robotics Resources columnist for
SERVO magazine
Earl Boysen

Under the hood..!

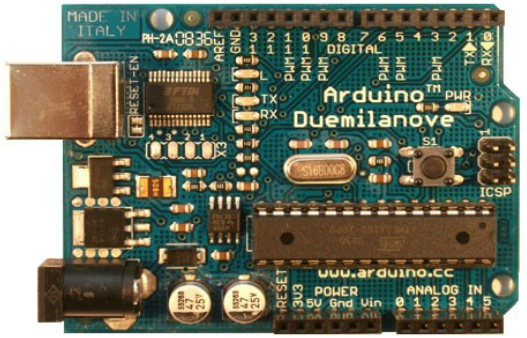
Over the hood ..!



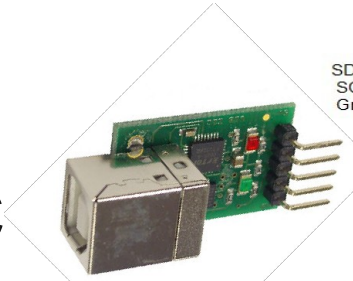
Motor HAL

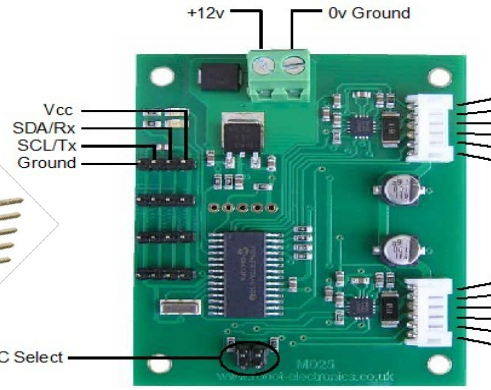
MotorHAL

Arduino



MD23
Via USB 2 I2C

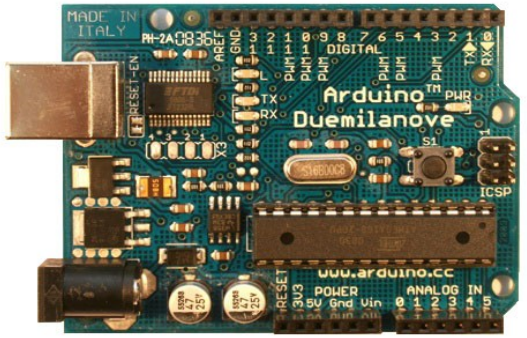




Mindstorm

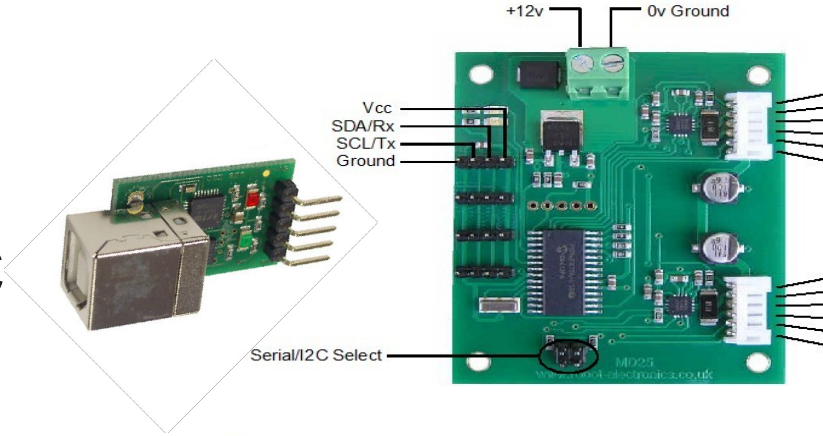
Motor HAL

Arduino



MotorHAL

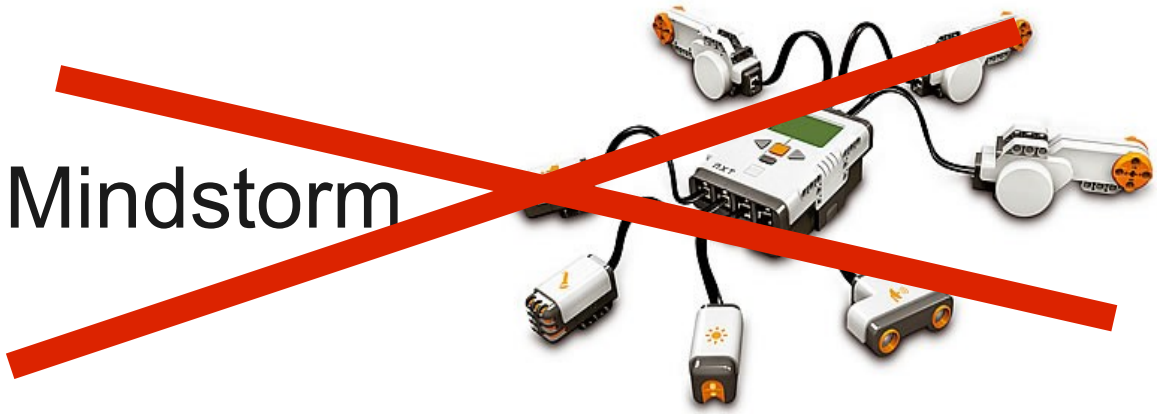
MD23
Via USB 2 I2C



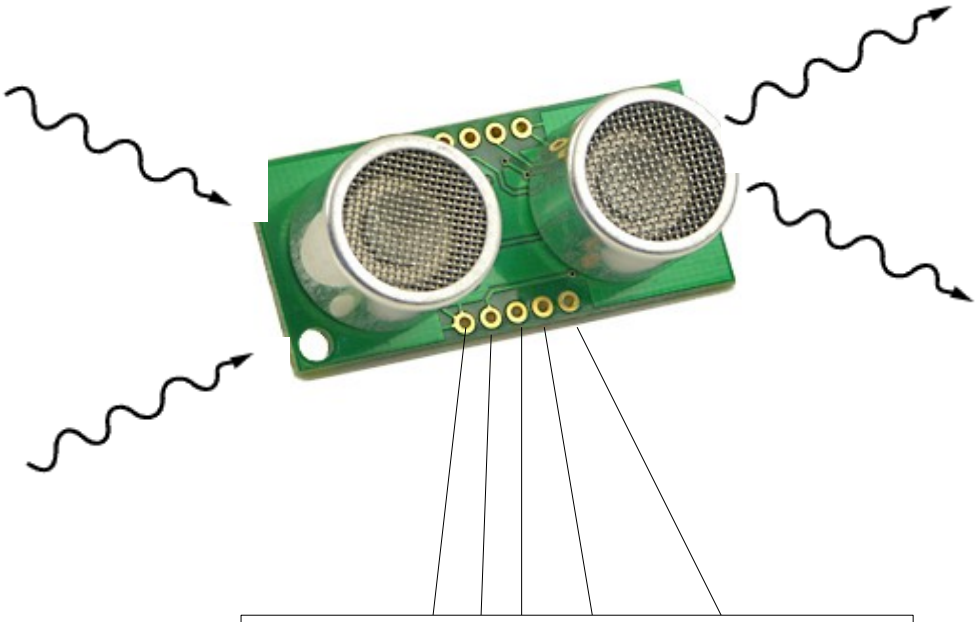
Παρ`όλα αυτά πολύ καλή ιδέα για μια αρχή στην επικοινωνία με microcontrollers κτλ

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

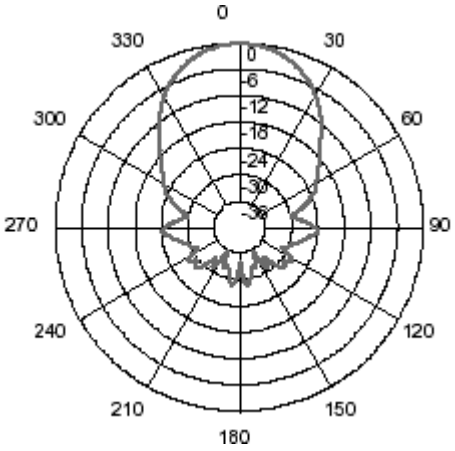
Mindstorm



Motor HAL
Ultrasonic Sensors



5V , Out , Trigger , Null , GND



Frequency	40kHz
Max Range	4 meters
Min Range	3 centimeters
Input Trigger	10uSec minimum, TTL level pulse
Echo Pulse	Positive TTL level signal, proportional to range

Motor HAL
Accelerometer Sensors

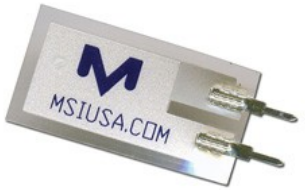
5V , Out X , Out Y , Trigger , Clock , GND

* Measures ±3 g on each axis
* Simple pulse output of g-force for each axis
* Convenient 6-pin 0.1" spacing DIP module
* Analog output fo temperature (Tout pin)
* Low current at 3.3 or 5 V operation: less than 4 mA at 5 VDC

Sample Applications:

* Dual-axis tilt sensing for autonomous robotics applications
* Single-axis rotational position sensing
* Movement/Lack-of-movement sensing for alarm systems
* R/C hobby projects such as autopilots

MotorHAL
Piezo strips (bump / vibration)

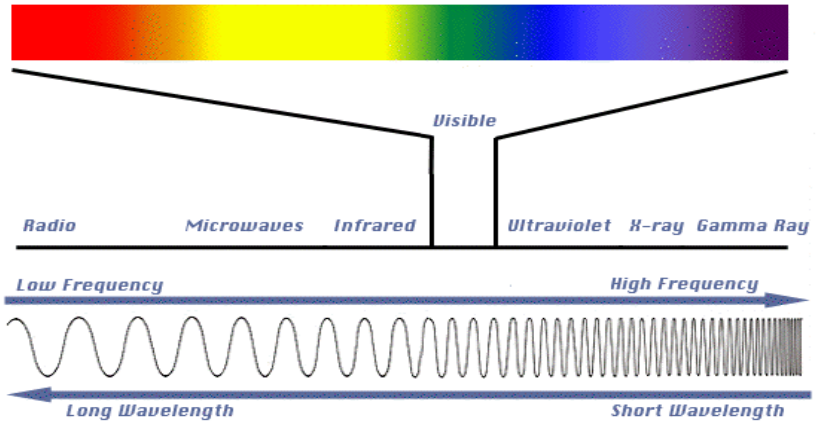


* Power Requirements: N/A
* Communication: Analog (Up To ~70 VDC; Sensitivity 50 mV/g)
* Dimensions: .98 x .52 in (25 x 13 mm)
* Operating Temperature: +32 to +158 °F (0 to +70 °C)

Example Applications:

* Flexible Switch
* Vibration Sensor
* Alarm System Sensor
* Product Damage/Shock Detector

MotorHAL
Infrared



MotorHAL
Motors.. !

RD-01/02 Step Motors
with encoders

Futaba Servos
(Continuous/normal) rotation

Motor HAL
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 HAL
Abstraction Layer

Τι κάνει ?

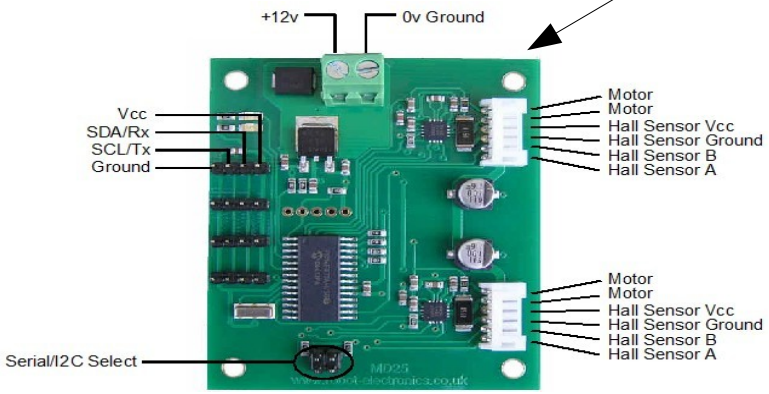
```
RobotInit("/dev/ttyUSB0","/dev/ttyUSB1");
while ( ( RobotGetUltrasonic(0)>100 ) && ( RobotGetUltrasonic(1)>100) )
 {
   RobotMove(255,360); /*Move full speed until wheel turns 360 degrees */
 }
RobotStopMovement();
RobotClose();
```

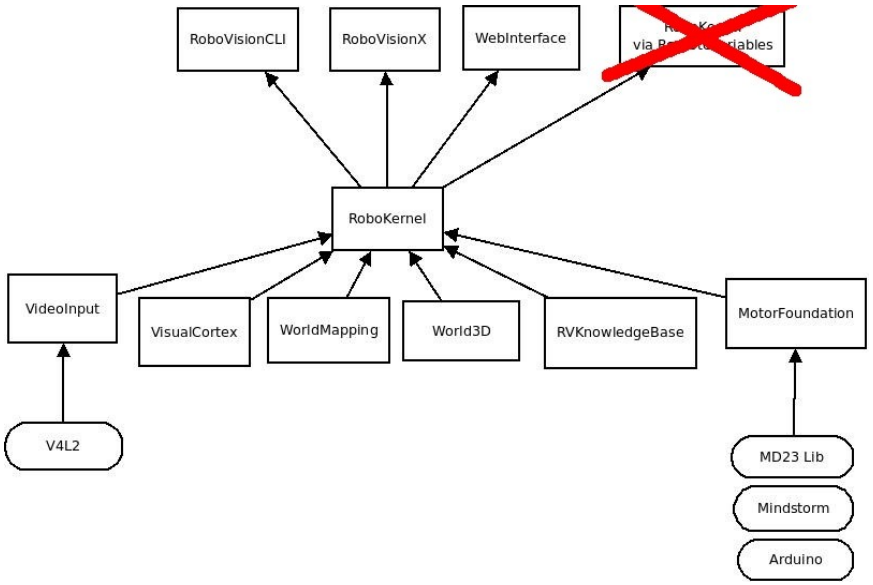
Όσο οι υπέρηχοι δεν πιάνουν εμπόδιο πιο κοντά από 100cm
γυρίζει τις ρόδες 360 μοίρες
Αν υπάρχει εμπόδιο , σταματάει και κλείνει την επικοινωνία με τα μοτέρ..

Motor HAL

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..



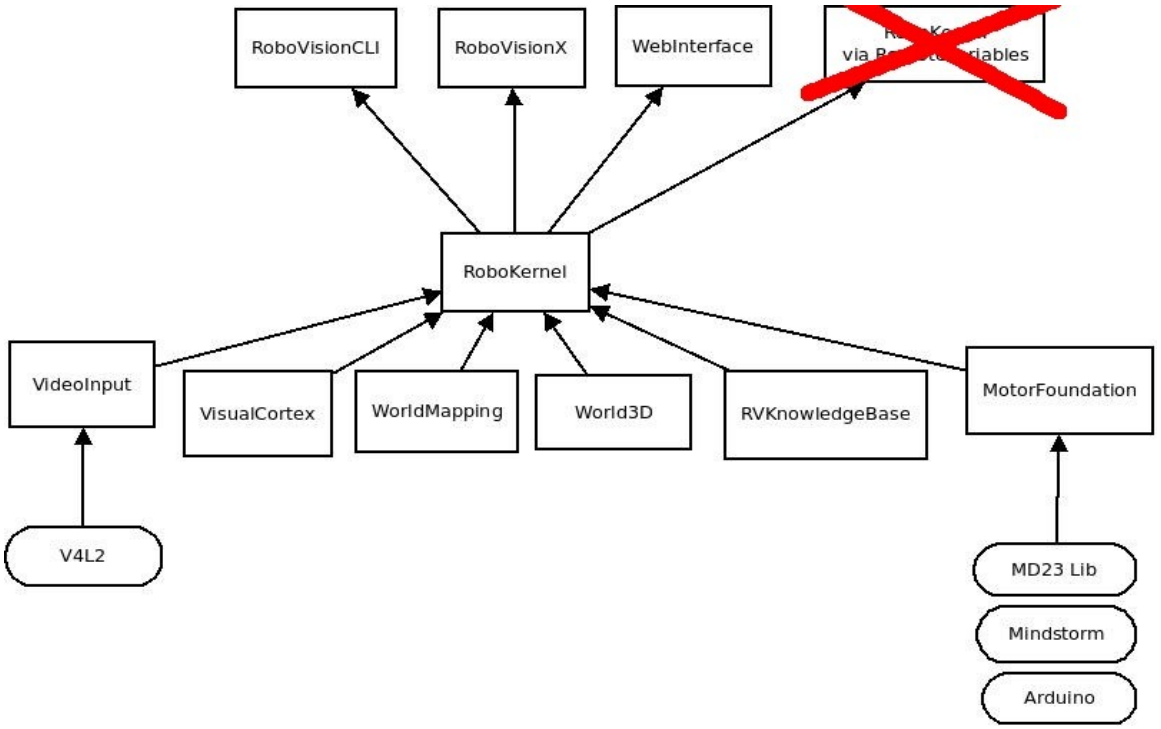


RV knowledge base

προσθέτοντας νοημοσύνη..

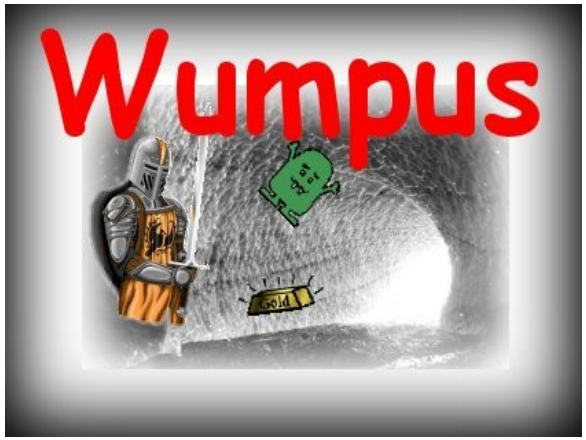
Στόχος , κάτι σαν το http://openmind.media.mit.edu/

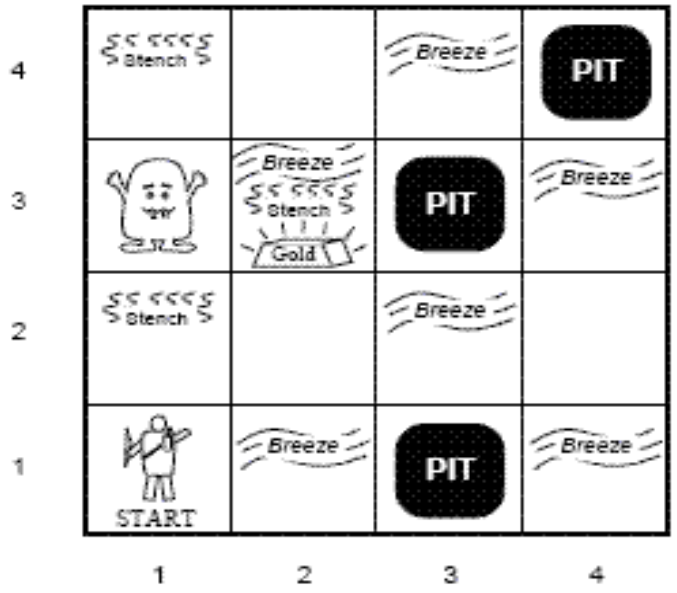
High level οντότητες , εντολές , ιεραρχίες και ενοποιημένο pipelining για την επεξεργασία τους



RV knowledge base

First order logic fits nicely , guarddog lives in a wumpus like world

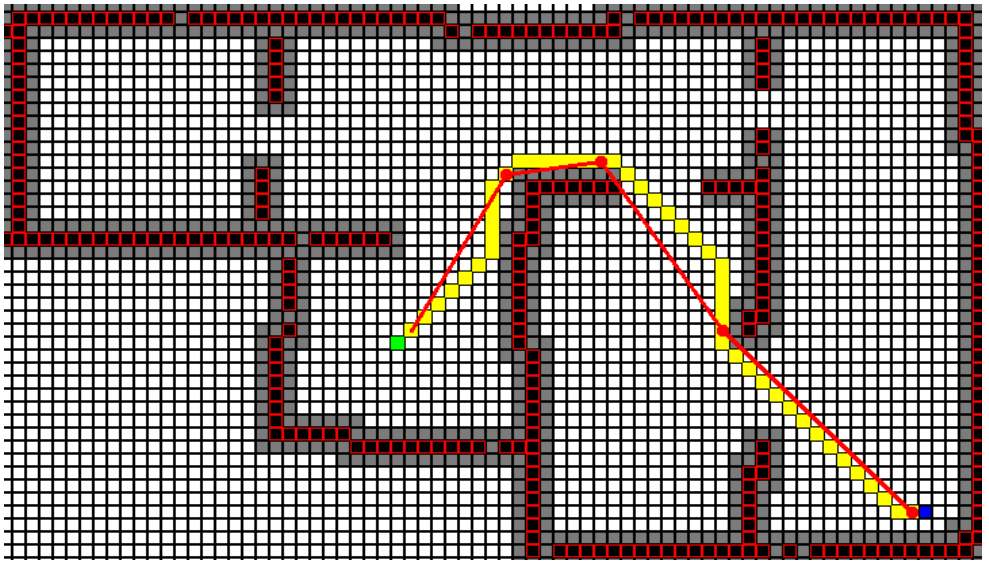




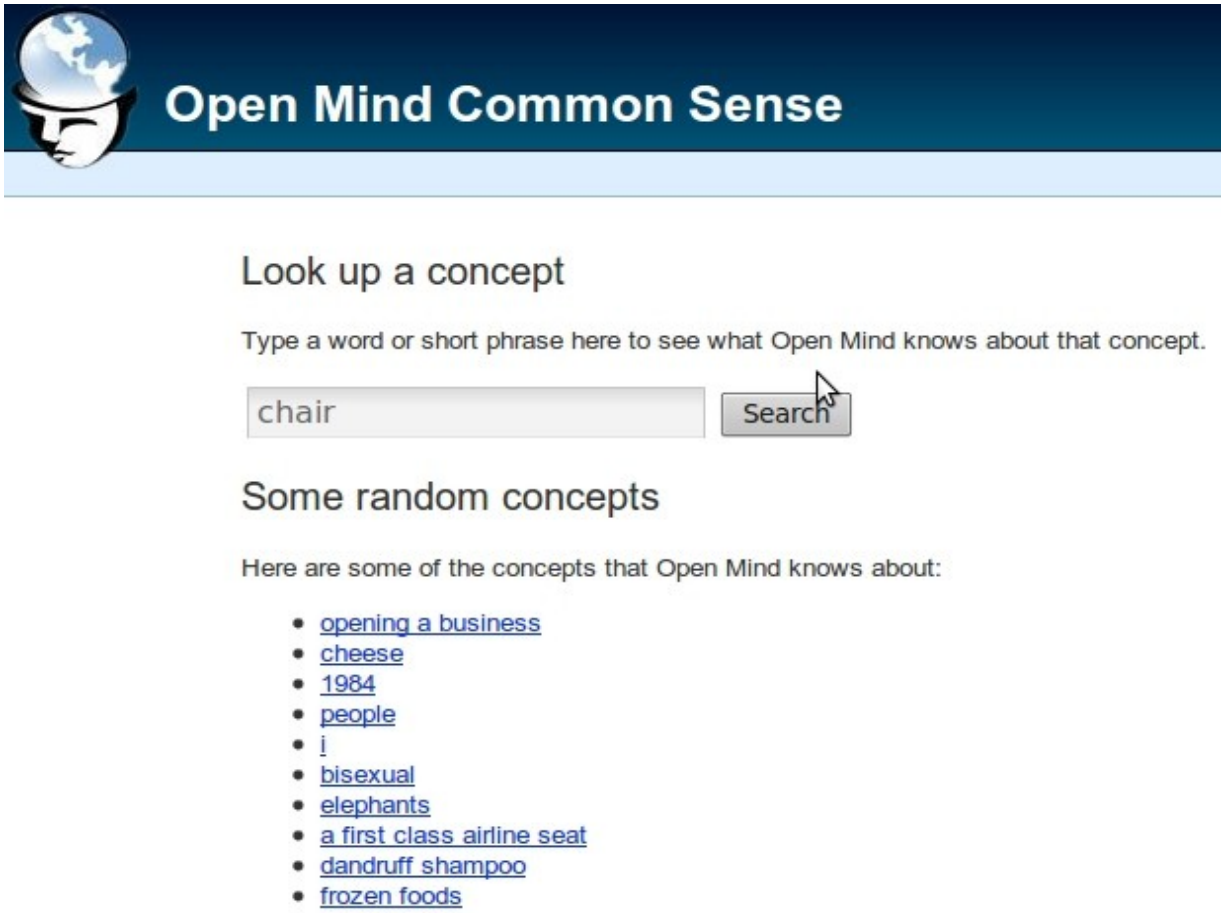
Τρόπος επικοινωνίας “κοντά στον άνθρωπο”.

Εύκολο να προγραμματιστούν συμπεριφορές όπως έλεγχος δωματίων , επαναφόρτιση κτλ .

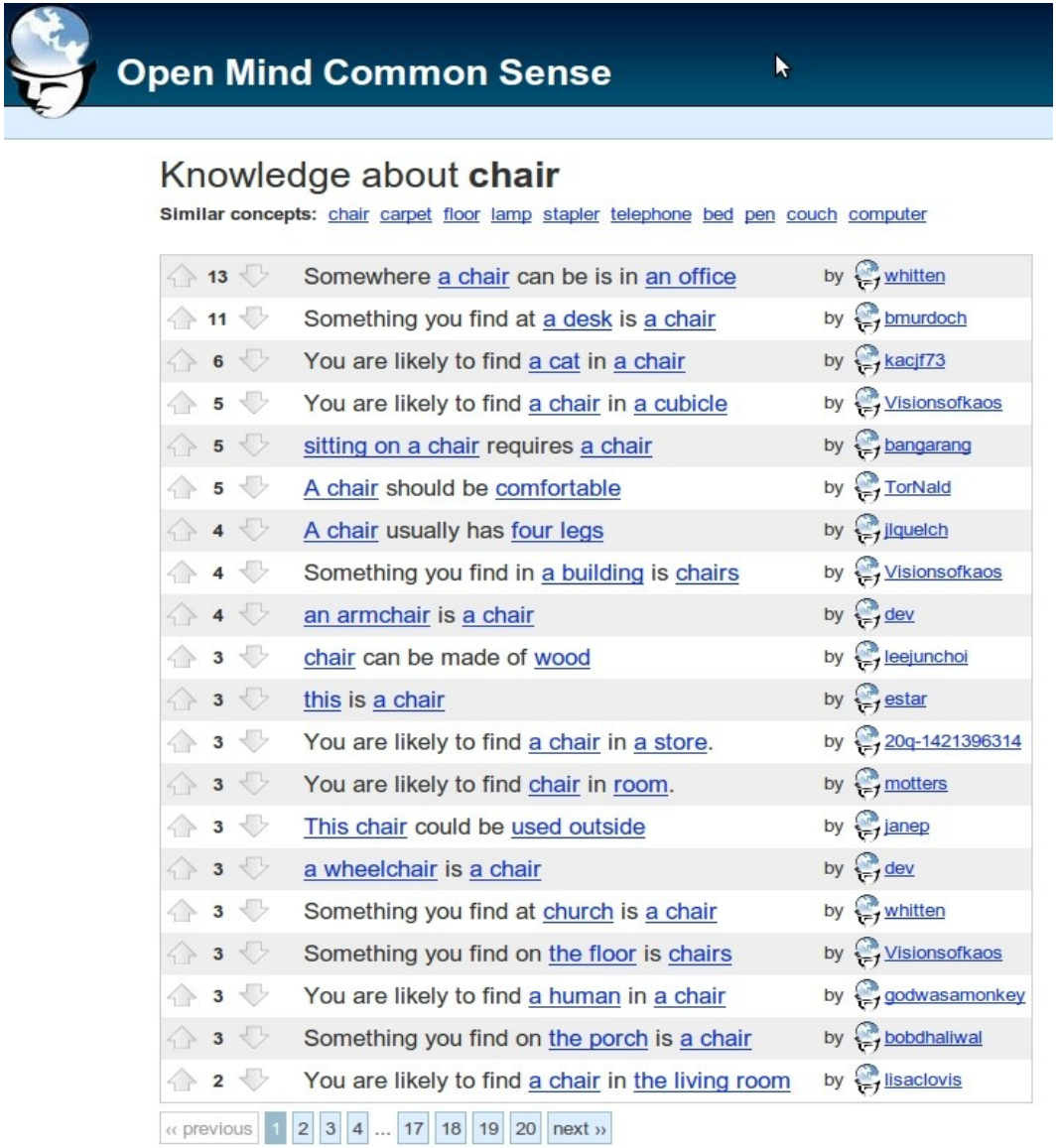
Ωστόσο δεν είναι το target του project οπότε προς το παρόν είναι επίσης stub



Open Mind


Open Mind Common Sense
Look up a concept
Type a word or short phrase here to see what Open Mind knows about that concept.
chair
Search
Some random concepts
Here are some of the concepts that Open Mind knows about:
opening a business
cheese
1984
people
i
bisexual
elephants
a first class airline seat
dandruff shampoo
frozen foods

Open Mind


Open Mind Common Sense
Knowledge about chair
Similar concepts: chair carpet floor lamp stapler telephone bed pen couch computer
13 Somewhere a chair can be is in an office by whitten
11 Something you find at a desk is a chair by bmurdoch
6 You are likely to find a cat in a chair by kacjf73
5 You are likely to find a chair in a cubicle by Visionsofkaos
5 sitting on a chair requires a chair by bangarang
5 A chair should be comfortable by TorNald
4 A chair usually has four legs by jlquelch
4 Something you find in a building is chairs by Visionsofkaos
4 an armchair is a chair by dev
3 chair can be made of wood by leejunchoi
3 this is a chair by estar
3 You are likely to find a chair in a store. by 20q-1421396314
3 You are likely to find chair in room. by motters
3 This chair could be used outside by janep
3 a wheelchair is a chair by dev
3 Something you find at church is a chair by whitten
3 Something you find on the floor is chairs by Visionsofkaos
3 You are likely to find a human in a chair by godwasamonkey
3 Something you find on the porch is a chair by bobdhaliwal
2 You are likely to find a chair in the living room by lisaclovis
« previous 1 2 3 4 ... 17 18 19 20 next »

RV Knowledge base coupled with object recognition

RV Knowledge base coupled with object recognition

RV Knowledge base coupled with object recognition

RV Knowledge base coupled with object recognition

RV Knowledge base coupled with internet databases

WolframAlpha computational... knowledge engine

what is a volcano

Input interpretation: *Mathematica* form

volcano (English word)

Definitions:

1 noun a fissure in the earth's crust (or in the surface of some other planet) through which molten lava and gases erupt

2 noun a mountain formed by volcanic material

American pronunciation:

volk'eynoh (IPA: volk'eɪnoʊ)

Hyphenation:

vol-ca-no (7 letters | 3 syllables)

First known use in English:

1613 (European Renaissance | Jacobean Era) (397 years ago)

Word origins:

Italian | Latin | French

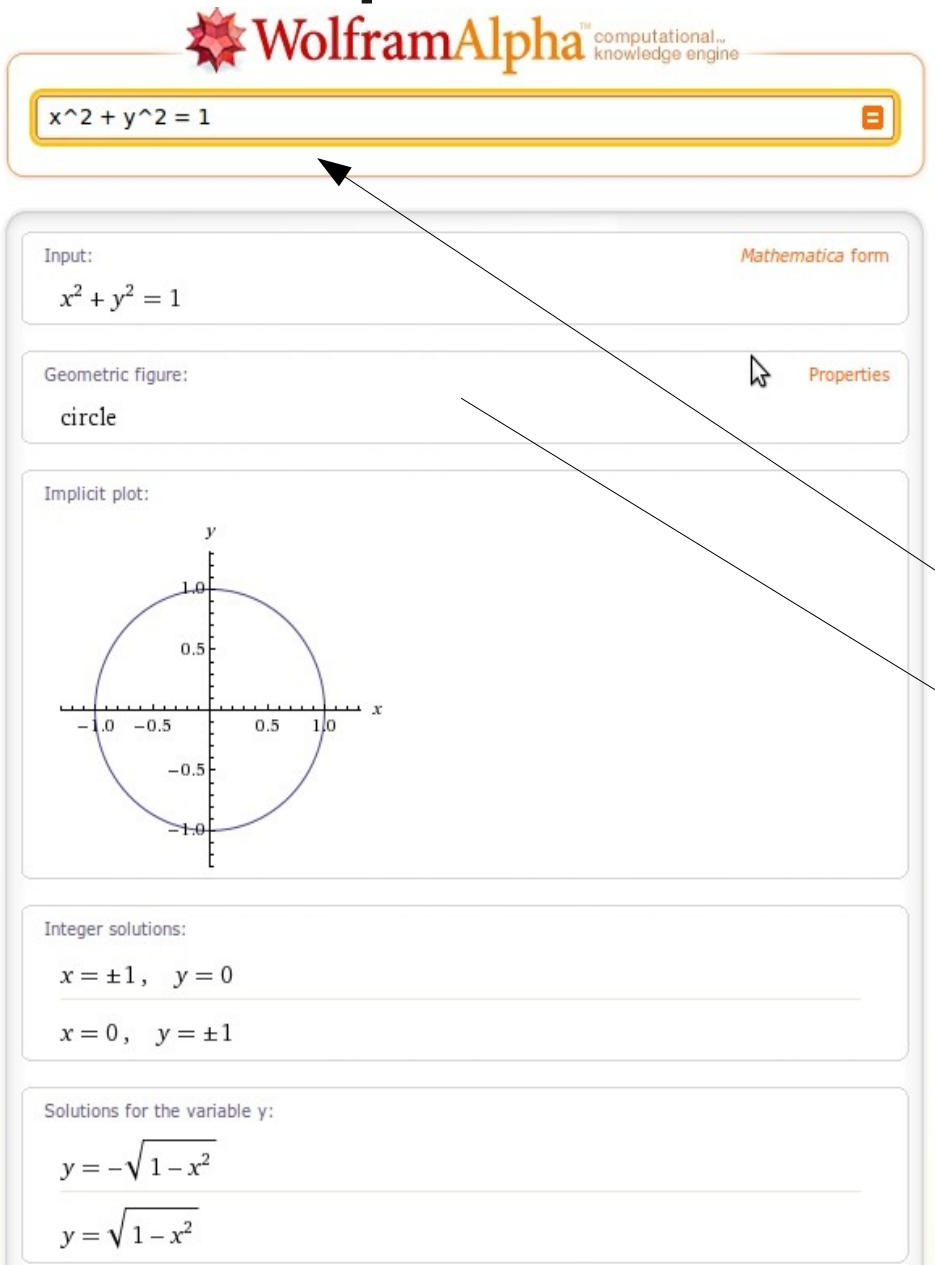
Inflected form:

volcanoes

What is a volcano?

A fissure in the earth`s crust (or in the surface of some other planet) through which molten lava and gases erupt

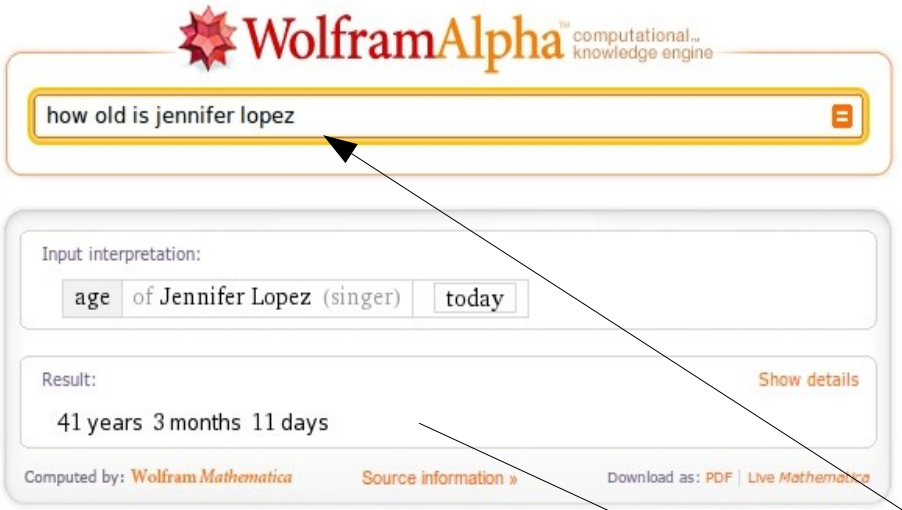
RV Knowledge base coupled with internet databases



Calculate x^2 + y^2 = 1

A circle!

RV Knowledge base coupled with internet databases



How old is Jennifer Lopez?

41 years 3 months 11 days

Energy Issues

Αυτονομία , και άλλα..

Energy Issues

Chemistry	Cell Voltage	Mj/Kg	Comments
NiCd	1.2	0.14	$
Lead acid	2.1	0.14	$$
NiMH	1.2	0.36	$
Lithium - ion	3.6	0.46	$$$$

* Ανοικτά προβλήματα χημείας/φυσικής
* Τα κομμάτια που χρησιμοποιώ (motherboard , motors , κτλ) είναι “οικονομικά” σε ρεύμα
* Το Guarddog χρησιμοποιεί NiMH 12V με περίπου 20-30mins αυτονομία
* Πρακτικά δουλεύει με 220V
* Είναι παρα πολυ ακριβός ο πειραματισμός με πηγές ενέργειας.. **150 euro για NiMH**

Με την τρέχουσα τεχνολογία θα πρέπει να υπάρχει κάπου μια βάση φόρτισης και σε κάθε περιπολία να επιστρέφει και να φορτίζει..
Κατα αυτό τον τρόπο θα κρατά υψηλή τάση και θα έχουμε όσο το δυνατόν μικρότερο χρόνο μεταξύ 2 πλήρων φορτίσεων , την τάση της μπαταρίας μπορούμε να την δούμε μέσω ACPI σε linux (sensors)

Κατασκευαστικά θέματα

Στον άυλο κόσμο του Software κάνουμε κάτι backup , copy , recompile , batch run

Στον φυσικό ,υλικό κόσμο αν μια τρύπα ανοίξει λάθος σε ένα πλαστικό , μπορεί να χρειαστεί να ξαναγίνει όλη η κατασκευή από την αρχή..! Χρειάζεται πολύ ακριβό εργαστήριο για R&D

Θέματα στήριξης , η κεφαλή 2 αξόνων , η καλωδίωση και άλλα είναι πολύ πιο χρονοβόρα στην κατασκευή από όσο μπορεί κάποιος να φανταστεί..

GuarddoG in numbers

- 4+ χρόνια (*p3**04**0023 , 6.4μο*)
- 3 complete rewrites
- Περίπου 639 euro construction cost το συγκεκριμένο prototype
- C 58% , C++ 38% BAShell 2% , Arduino C 1% , PHP 1%
- Και δεν είναι έτοιμο ακόμα..

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

(!) Without batteries (!)

Commercial platforms






NXT Mindstorm – €300+ (no cameras)

Rovio – €200 +
Spykee – €350+

AIBO – €2000+

Nao Bot - €10000+

Papero - €30000+

(estimation από Internet search)

Papero almost like guarddog , costs 30000 euro :P
Pentium M 1.6 GHz processor, 512 MB Ram, 40 GB HD, USB 2.0, microphones, And Dual CCD cameras for eyes and functional plastic body.

About the cost

Large Scale Production
is much much cheaper, especially made in China..

Οικονομίες κλίμακας..

Επίσης λογικά , το να αγοράσει πολύς κόσμος κάποια από τα ανταλλακτικά του σε μια εκδοχή που δεν έχει τόσο critical εφαρμογή (ασφάλεια) είναι καλύτερη ιδέα προς το παρόν..

Αντίστοιχα στατικά συστήματα ασφαλείας είναι πολύ πιο φθηνά ,αλλά μπορεί να είναι ασύμβατα με ένα κινούμενο αντικείμενο σε έναν χώρο που ελέγχεται με ανιχνευτή κίνησης πχ

Πρώτα βήματα GuarddoG mk1

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

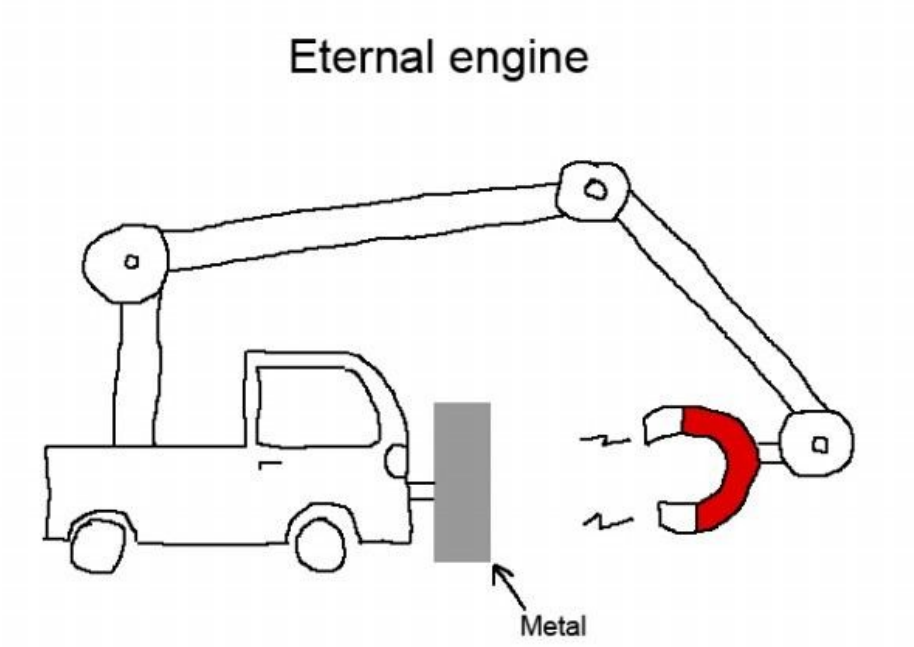
Κακό Calibration , ακτίνα όσο το καλώδιο USB (+ το USB hub)

Trials & Errors

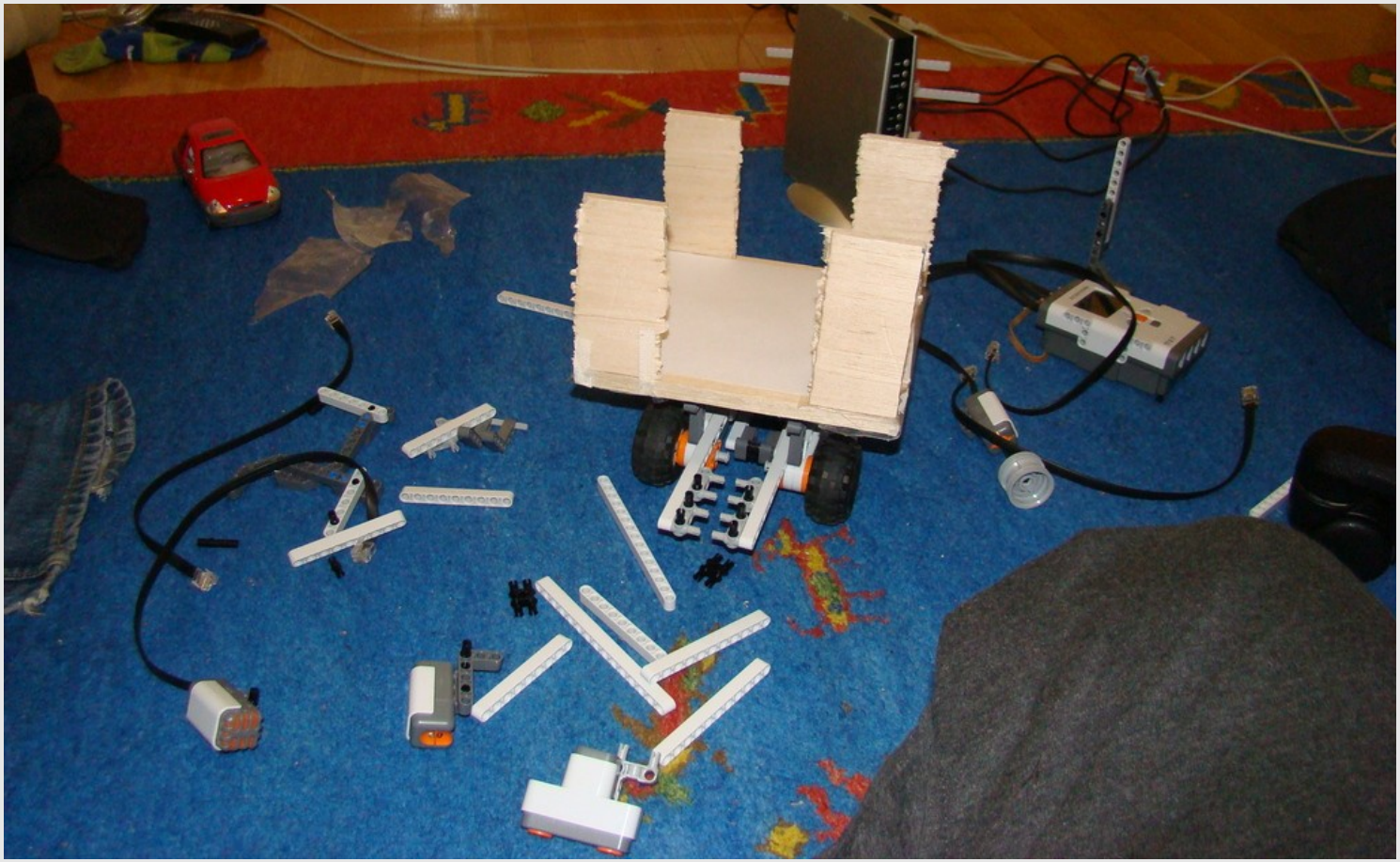
Πολλές ιδέες πολλές αποτυχίες..

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

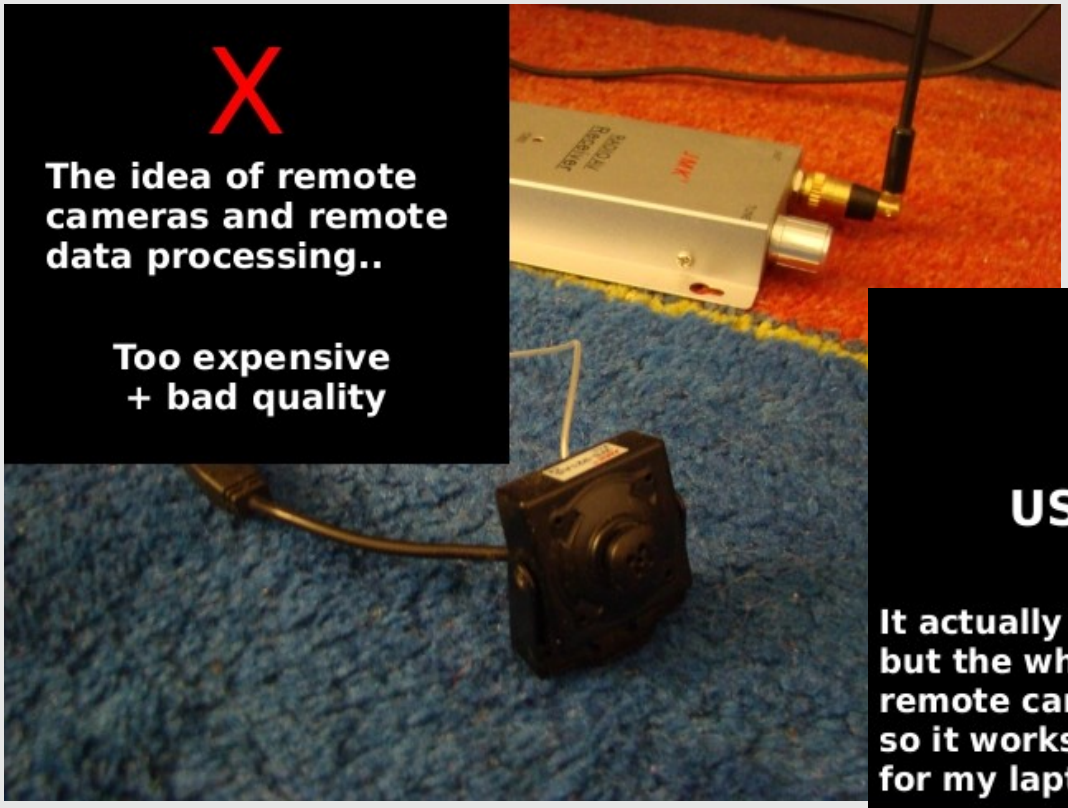
Σχεδόν τίποτα δεν δούλεψε όπως το σχεδίαζα στην αρχή ..



Balsa , όχι καλές στατικές ιδιότητες



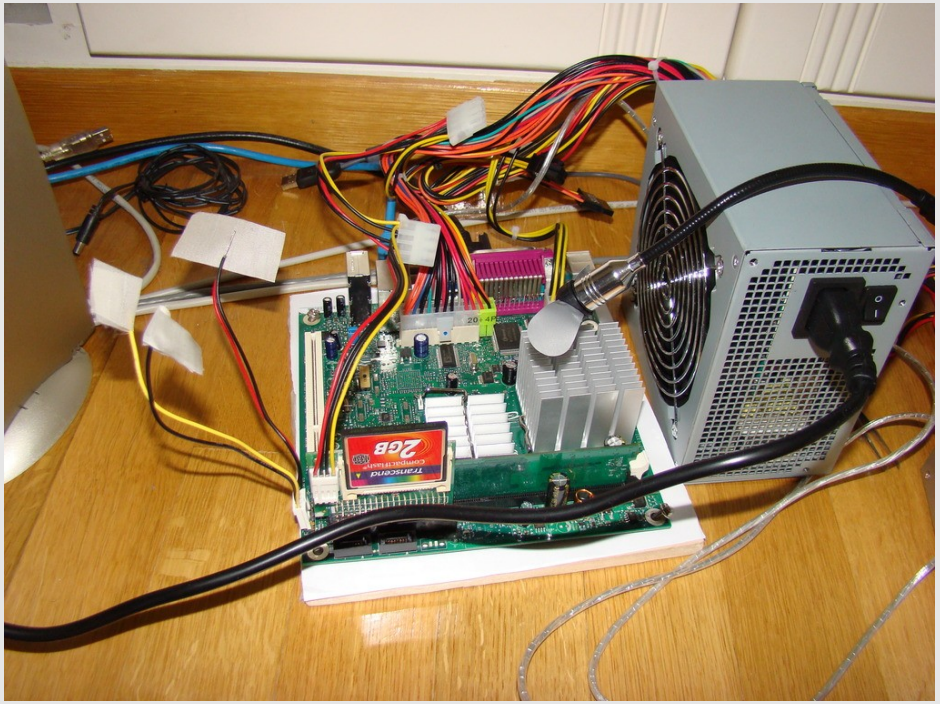
Wireless Cameras



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

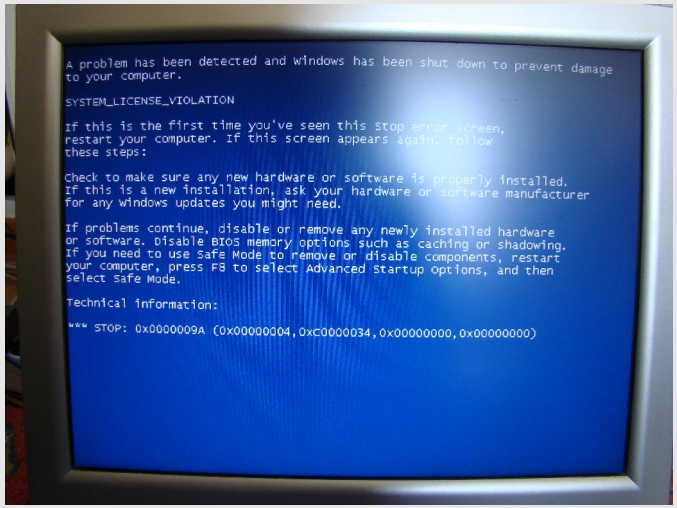
- Lock in , πολλά πράματα που είχα ήδη φτιάξει με DirectX
- Για χαμηλή κατανάλωση ρεύματος και αντοχή στην κίνηση θα πρέπει το PC να λειτουργεί χωρίς σκληρό δίσκο
- WinXP thrashed to death my CF card in 4 hours

WinXP Embedded

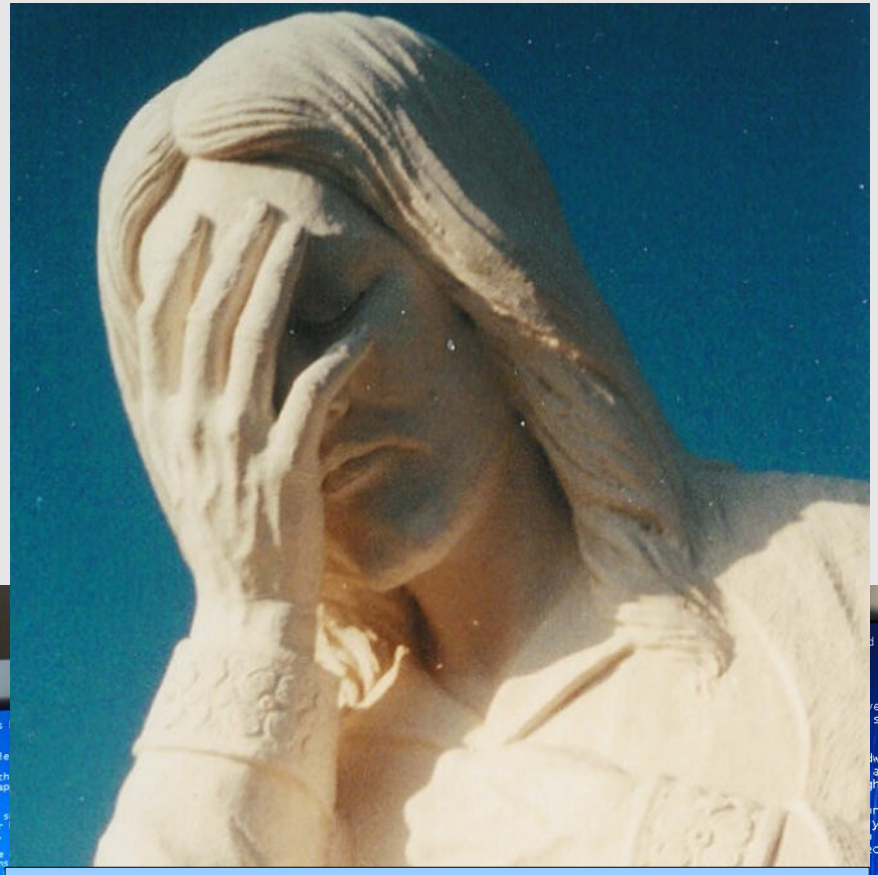
- OK με την (καινούργια) CF
- Binaries “Συμβατά” από WinXP
- Οι drivers για wifi κτλ μετά απο πολλά updates , *.inf hacks κτλ κτλ δούλεψαν

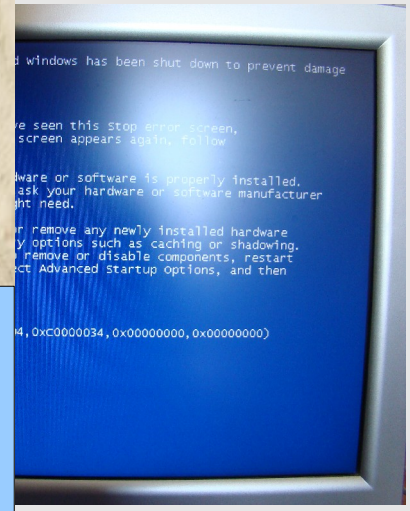
αλλά..

WinXP Embedded



WinXP Embedded





No Offence.. :P

Windows Γενικά

- Για να βάλω text to speech

  WinXP/Embedded έως SAPI 5.1

  Vista έως SAPI 5.3

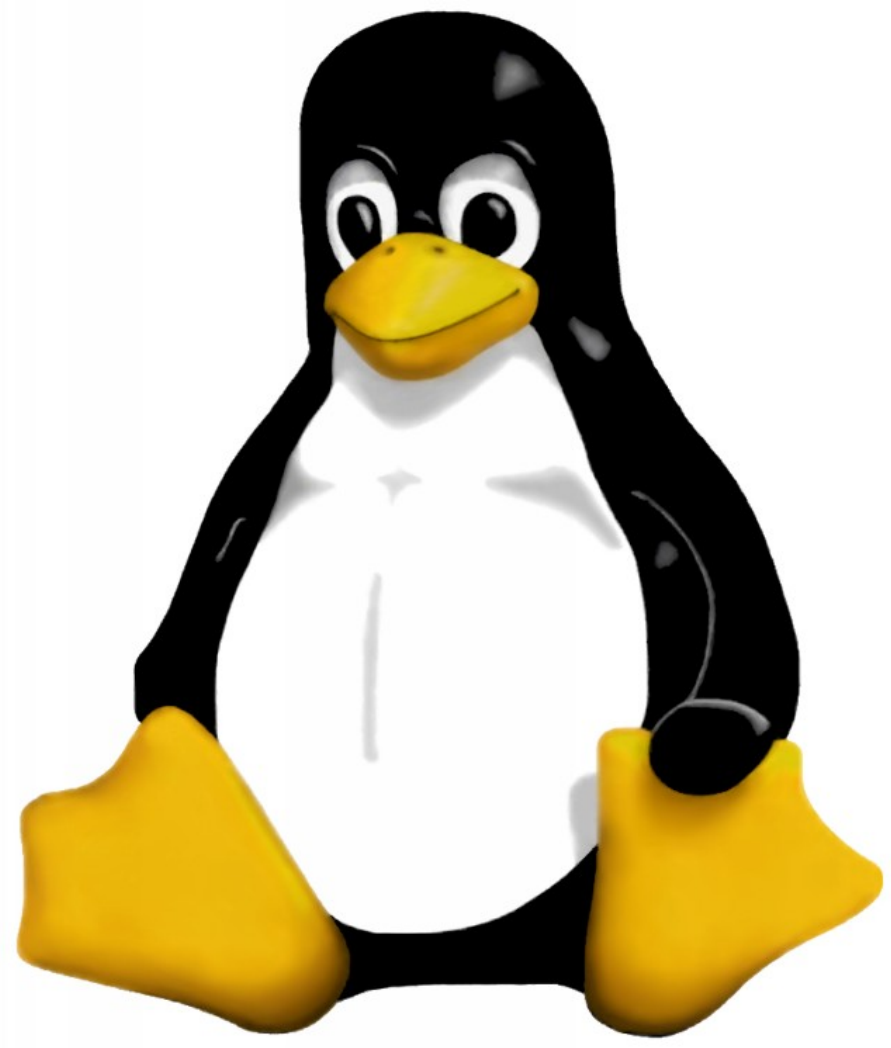
  Win 7 έως SAPI 5.4
- 50+ euro per guarddog license..
- No support for non Intel cpus (ARM i.e.)
- Cannot be used on a headless configuration

- 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 ...
by design

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”);

Windows is money orientated

Windows : Εμπορικό προϊόν,από εταιρεία,profit oriented!
Οι άνθρωποι που το έχουν φτιάξει και το προωθούν πληρώνονται , αν δεν είχαν οικονομικό κέρδος δεν θα ασχολούνταν.. Ο Bill Gates δεν είναι τυχαία ο πλουσιότερος άνθρωπος στις ΗΠΑ και 2ος στον κόσμο..

Linux : Δωρεάν προϊόν (GPL) , από όποιον θέλει να ασχοληθεί, προφανώς με στόχο την ποιότητα και το να δουλεύει , δεν πρόκειται να σε πληρώσει κανείς επειδή έφτιαξες το X feature ,κανείς δεν θα σου ζητήσει λεφτά,κανείς δεν θα πάρει λεφτά! Τα λεφτά δεν έχουν σχέση..
Οι άνθρωποι που το έχουν φτιάξει το κάνουν κυρίως για την χαρά και την “δόξα” του να γράφεις κάτι καλό και να μοιράζεσαι την γνώση

Δεν έχω να κερδίσω τίποτα που σας το λέω..
Απλά είστε Computer Scientists και θά πρεπε να το ξέρετε..

Science != Money

GuarddoG is built using only GPL software

No animals were harmed in the making of this robot..

The whole software stack costs 0 €
No lock in to any hardware vendor

Forbes

Search news, busines

Gallery: 28 Dirt-Cheap ETFs

Αν δεν με πιστεύετε.. :P

World's Most Powerful People

#9 Sonia Gandhi

Bill Gates

$54 B

Net Worth Calculated September 2010

335 shares 122 tweets

Share Tweet

+ Follow Bill Gates 207

Age: **55**

Title: **Co-Chair**

Organization: **Bill & Melinda Gates Foundation**

Source: **Microsoft, self-made**

Residence: **Medina, WA**

Country of citizenship: **United States**

Education: **Dropout, Harvard University**

Marital Status: **Married**

Children: **3**

Powerful People
#10

Forbes 400
#1

World's Billionaires
#2

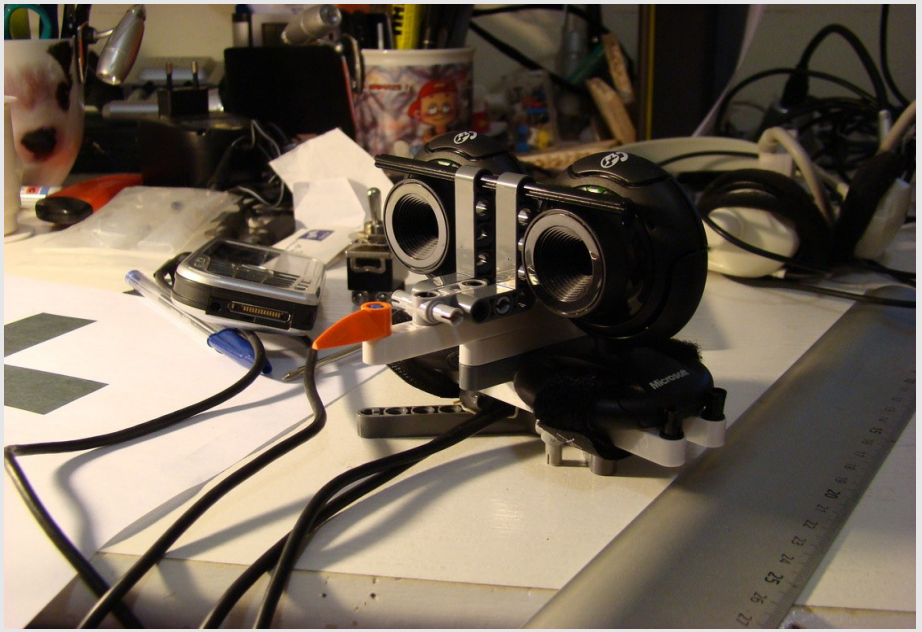
Εν το μεταξύ , για να επανέλθω ..

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

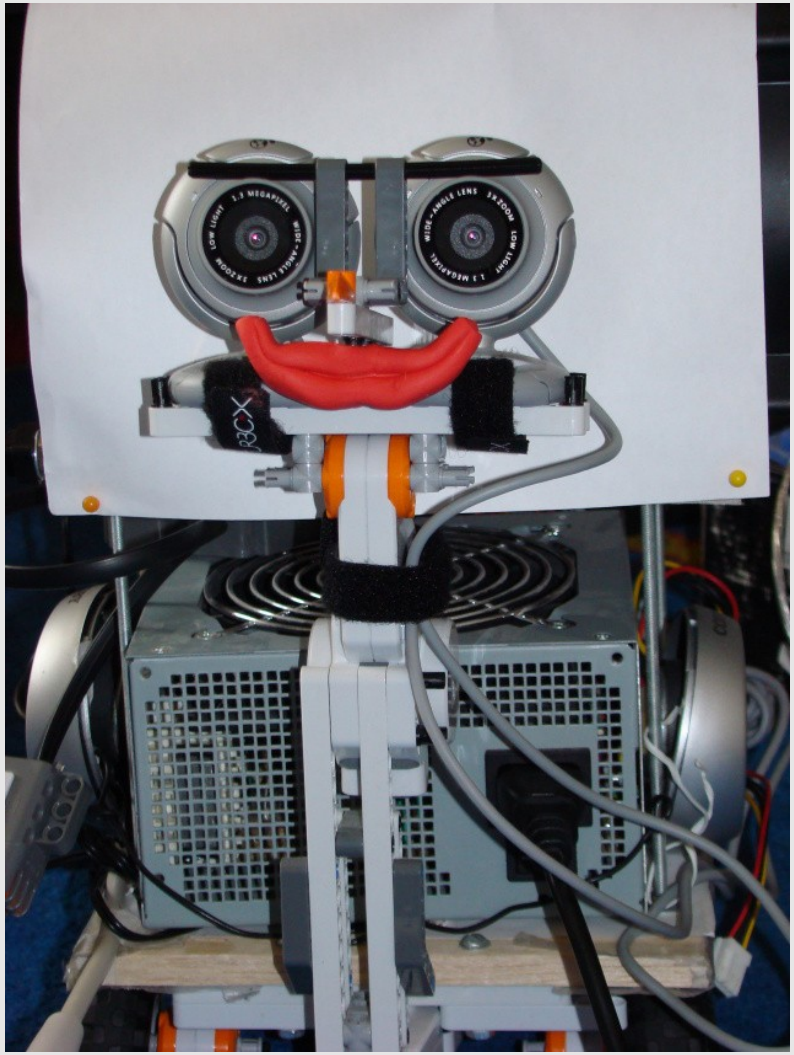
Κακό Calibration , ακτίνα όσο το καλώδιο USB (+ το USB hub)

Πρώτα βήματα GuarddoG mk1

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



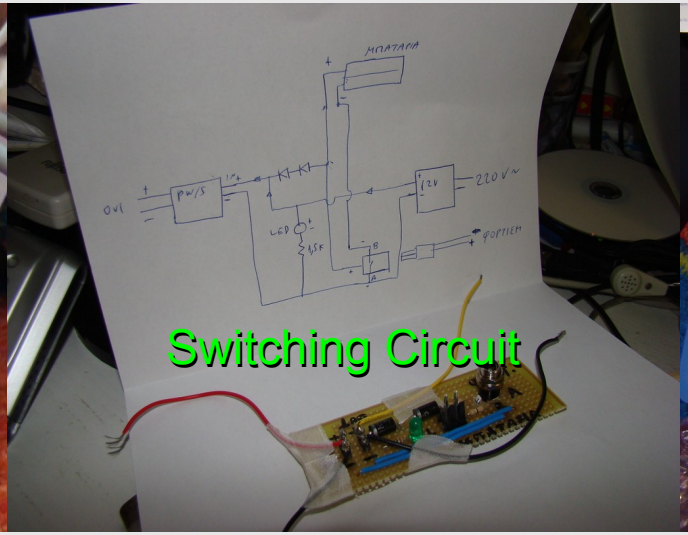
GuarddoG mk2



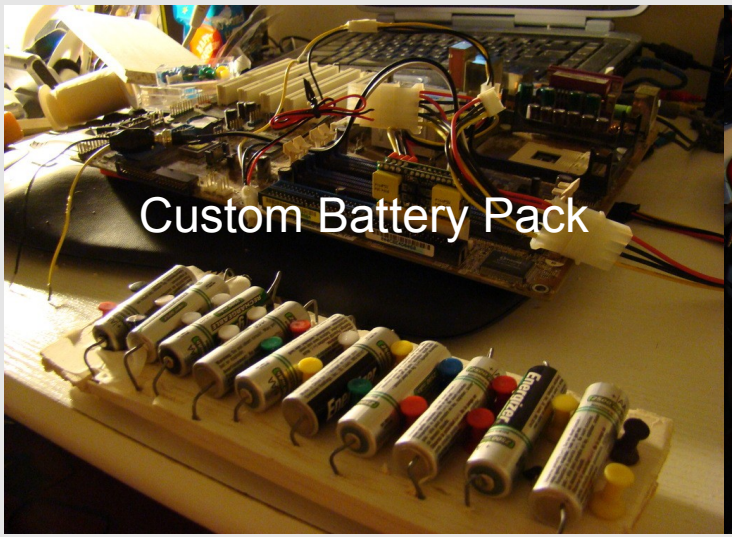
- Βαρύ τροφοδοτικό
- Γενικά μεγάλο βάρος για τα mindstorm motors
- Κακό alignment καμερών
- Software σε πρώιμο “μονοκόμματο” στάδιο

GuarddoG mk2 -> mk3


Γέμισμα τροχών με βαμβάκι


Switching Circuit

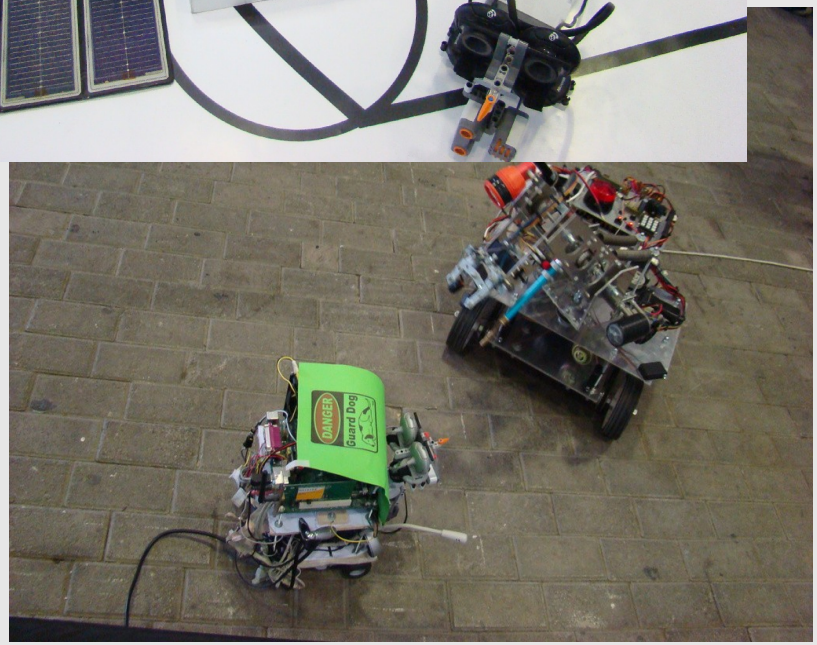

Pico PSU


Custom Battery Pack


Better Battery Pack


Even better battery pack

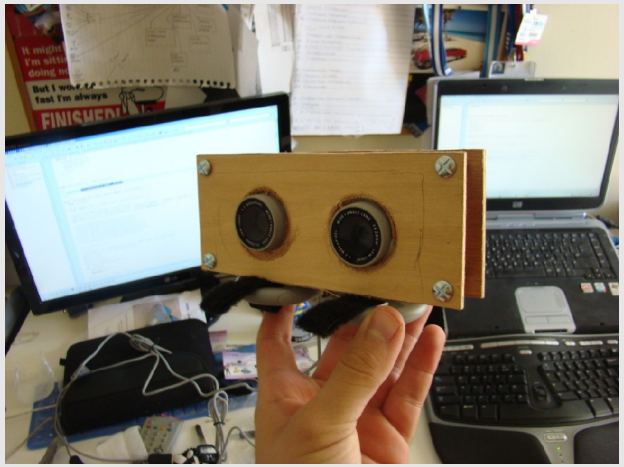
GuarddoG mk3

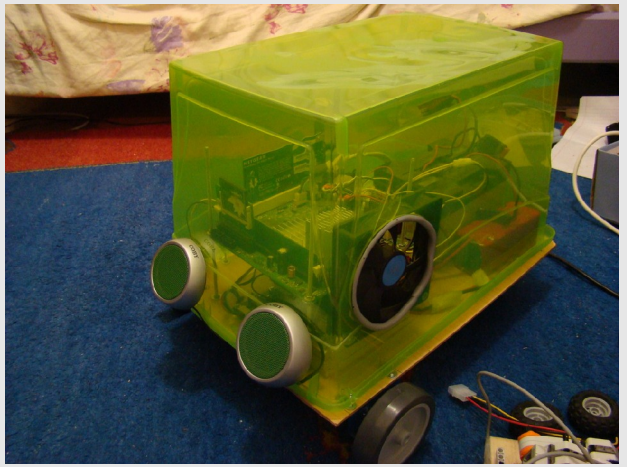


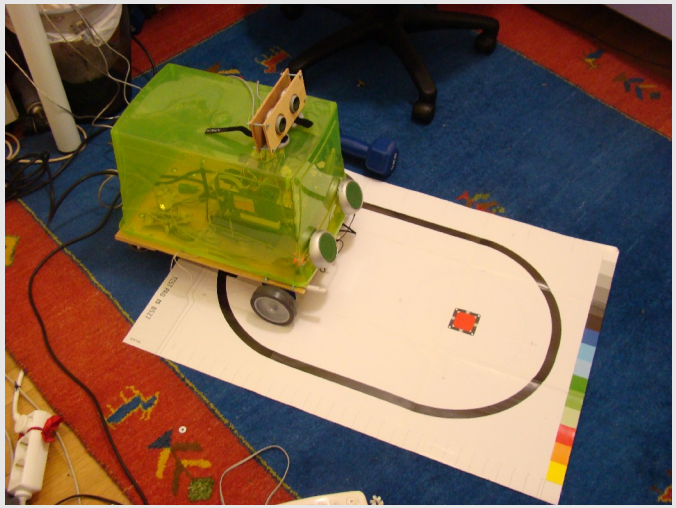
- Βραβεύση στην Athens Digital Week 2008 , στο Robotics κομμάτι

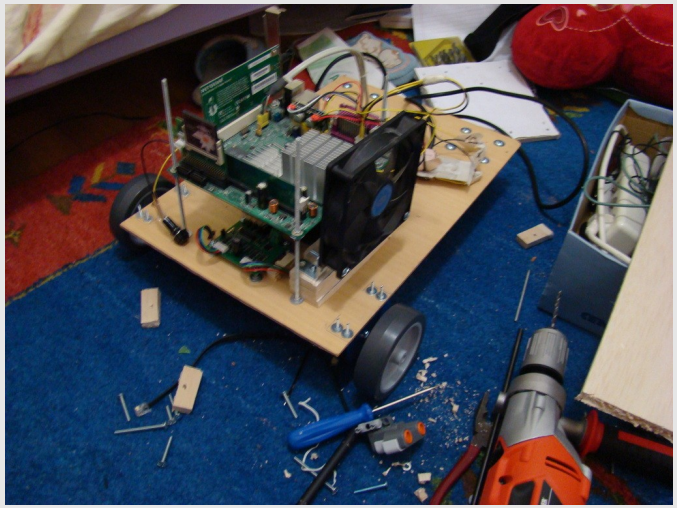
  με το extra budget απόφαση για remake from scratch όλου του project με πολύ υψηλότερα standards :)

GuarddoG mk4 (building)









GuarddoG mk4

Βράβευση στην Athens Digital Week 2010

GuarddoG mk4

Παρουσία στην Διεθνή έκθεση Θεσσαλονίκης 2011

Media Coverage

(30 seconds each ;P)

Κοινότητα – A. Qammaz

Συνέντευξη: Ammar Qammaz

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

Progress List
12/2011

Module	Progress	Problems Pending
Video Input	90 %	Hardware Sync , new body
Image Processing	90%	Disparity Mapping Fine tune , Code Quality
2D Path Planning	99%	-
3D SLAM	40%	Work required for stable tracking and 3d path planning
Hardware	80%	Plexiglass Frame , cabling problems , newer hardware
Supporting Framework (OS etc)	80%	Packaging issues
RV Knowledge Base	10%	Stub

Feature List !

- **Disparity Mapping**
- **Face Detection**
- **Stationary Guard mode**
- **Path Planning**
- **Camera Pose tracking**
- **Physical Movement , Sensor Input**
- **Performance Monitoring via GnuPlot**
- HTTP Web Interface
- GUI / CLI Input
- Act as wifi AP 802.11b
- Control via GSM/SMS
- Control via Joystick
- Control via IRC
- Text To Speech
- Scripts (music , sound , notify etc)

TODO List !

- SLAM
- Stable Pose Tracking
- Proper Odometry
- Speech To Text
- Battery Power Supply
- Better Hardware
- Physical Build

- Actual Target :)

TODO – Contributions Wishlist

Computer Vision / Linear Algebra

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

SLAM efficient implementation

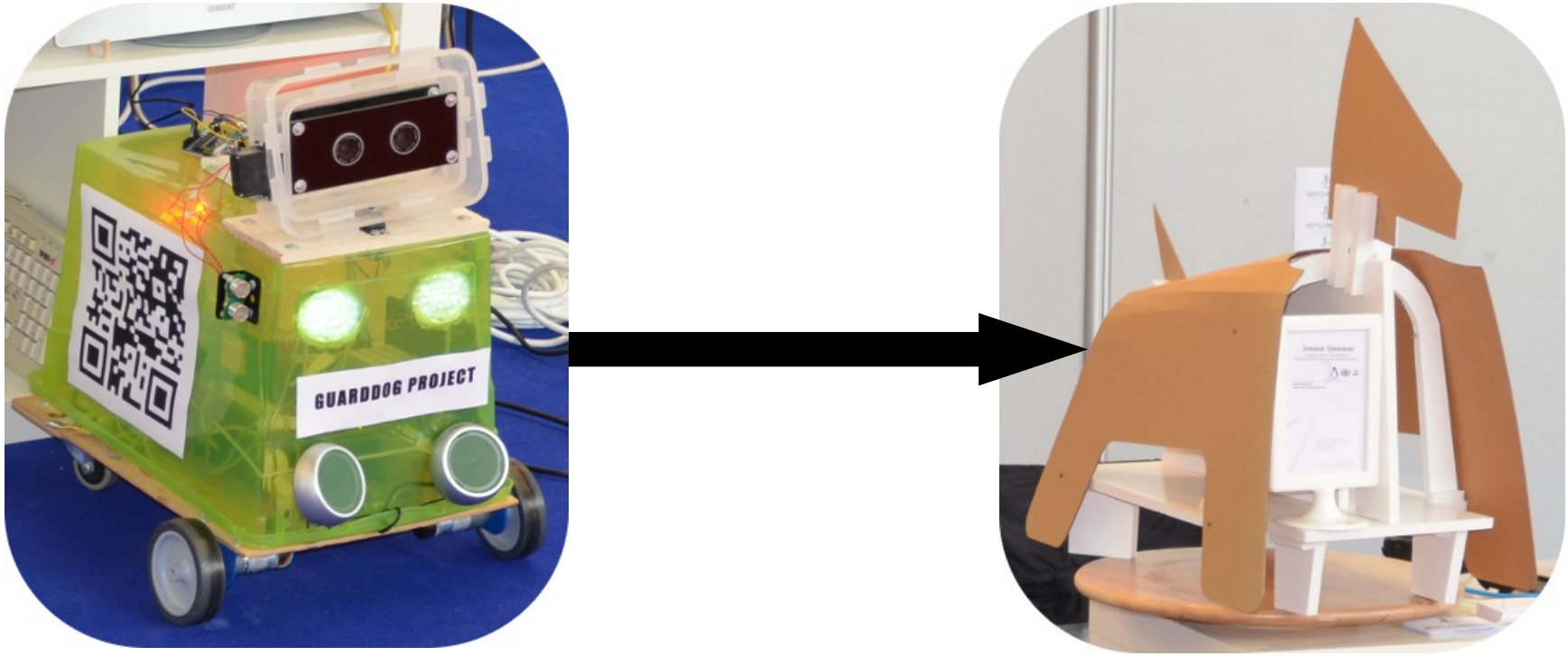
TODO - Contributions Wishlist

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 ,CAD chassis !!!!
Better cameras
More processing power , hardware is 5yrs old
etc..

FAQ

- Τι σχέση έχουν αυτά με πτυχιακή στην ΑΣΟΕΕ ?

Όχι παρα πολύ, αλλά αν είναι να ασχοληθώ για όλη την υπόλοιπη ζωή μου με SAP , SQL , λογιστικές εφαρμογές και να φτιάχνω websites , ευχαριστώ δεν θέλω.. καλύτερο cost/benefit να ανοίξω σουβλατζίδικο ή καφετέρια στην Ελλάδα :) !

- Τόση δουλειά και την δίνεις OpenSource ?

ΝΑΙ , 25000 γραμμές δεν είναι τίποτα μπροστά στον linux kernel και τις διάφορες GPL βιβλιοθήκες του πχ

- Στην αρχή είπες ότι είναι πολύ εύκολο!

Υπάρχει τόση πολύ δουλειά που πλέον είναι έτοιμη που ναι , είναι εύκολο! Το internet είναι τρομερό εργαλείο!

- Πότε θα είναι έτοιμο ?

Όταν είναι έτοιμο!

- Κάποιος μπορεί να τα κάνει με την Χ πλατφόρμα (e.g. Windows)

Ναι , κάποιος θα μπορούσε να το κάνει σε DOS ή Windows 95 επίσης.. Δεν βγάζω ποσοστά από πωλήσεις και δεν έχω κανένα λόγο για pro-windows bias.. Αντίστοιχα από την εμπειρία μου έχω κάθε λόγο για pro-foss bias!

- Με τι funds το κατασκεύασες?

Χρήματα από part-time jobs , websites κτλ + γονεϊκό sponsoring!

- Μπορεί να έχει εμπορικό μέλλον στην Ελλάδα κάτι παρόμοιο ?

Γιατί όχι ?

FOSS , GPL and and Contributions

Μπορείτε να :

- Κατεβάσετε
- Βελτιώσετε
- Μελετήσετε
- Χρησιμοποιήσετε

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

τον κώδικα !

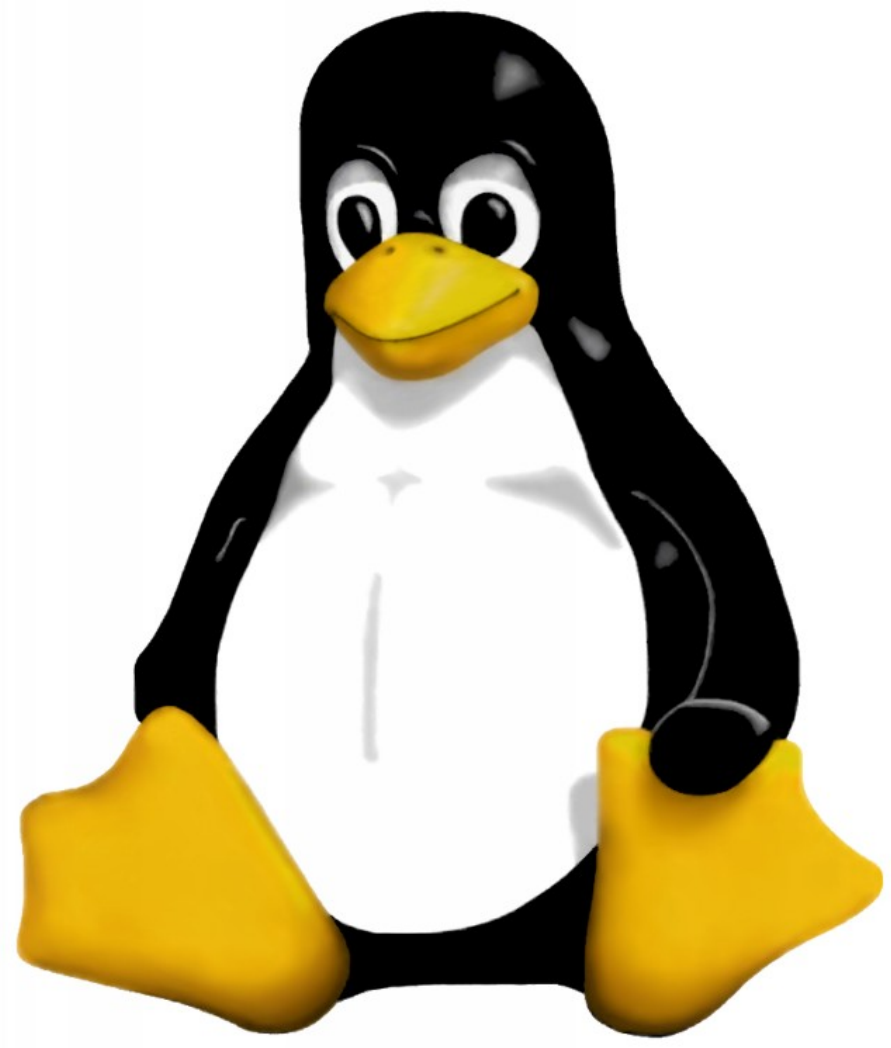
Ακόμα και για εμπορική εφαρμογή , αρκεί τα κομμάτια που παραλάβατε ανοικτά να τα διανέμετε σαν ανοικτό λογισμικό :)

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 ++)

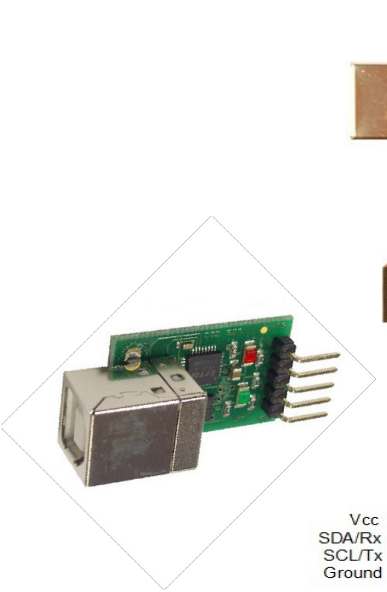


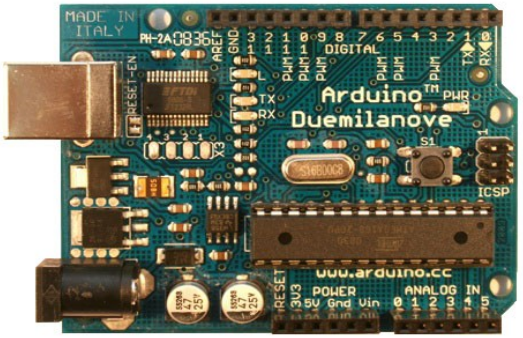
Getting Started , Testing Movement

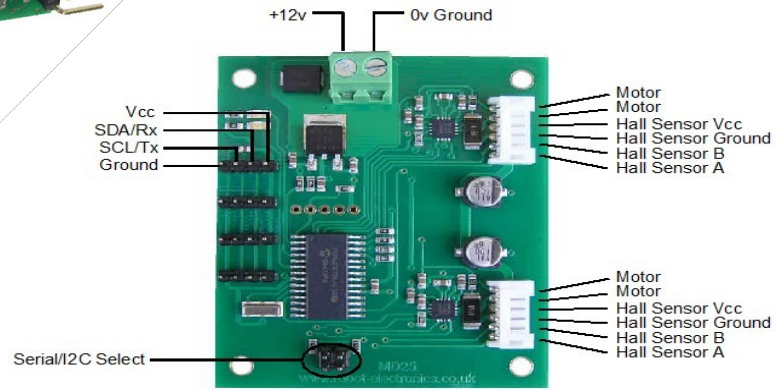
Για “κίνηση” :

- 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 και να το προγραμματίσει να κάνει ο,τι αυτός/η θέλει !

Future Plans!

- Προσωπική φιλοδοξία

  Χρήση του Vision κομματιού σε ένα αυτοκίνητο για ένα τηλεκατευθυνόμενο / “ρομποτικό” αυτοκίνητο!

- DARPA grand challenge style

Grand Cooperative Driving Challenge

GCDC.net

GuarddoG Repository!

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

http://ammar.gr/gddg

http://ammar.gr/

FOSS Aueb !

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

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

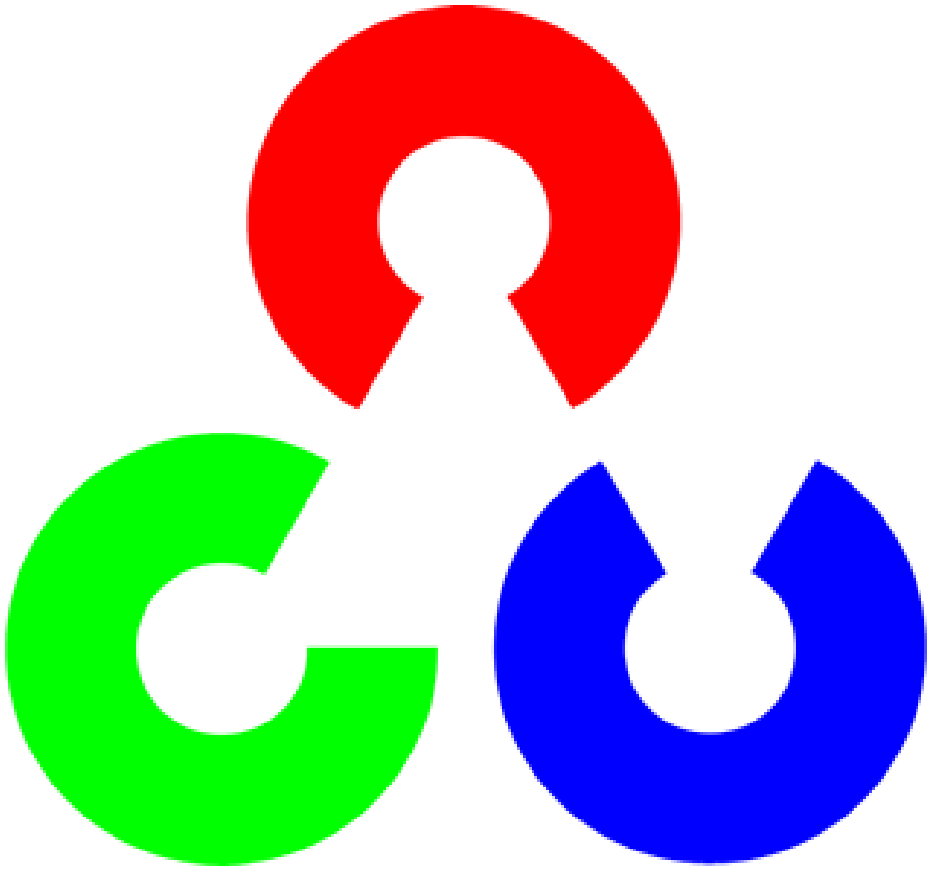
I am **AmmarkoV**
http://ammar.gr

Linux – Ubuntu for example

- FOSS
- Easy to download
- Easy to install
- Takes a while to get used to
- Wine and VMs for windows compatibility
- Big community

http://www.ubuntu.com/desktop/get-ubuntu/download

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

http://opencv.willowgarage.com/

sudo apt-get install opencv-doc libcv-dev libhighgui-dev libcvaux-dev

AR Toolkit

* Single camera position/orientation tracking.

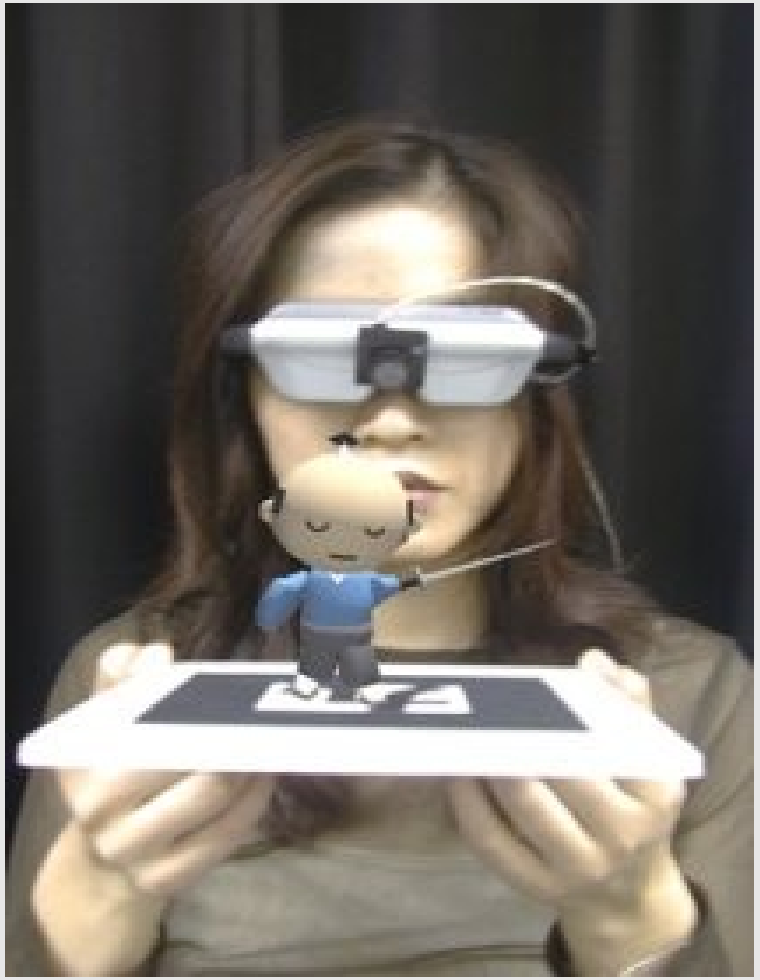
* Tracking code that uses simple black squares.

* The ability to use any square marker patterns.

* Easy camera calibration code.

* Fast enough for real time AR applications.

* Free and open source.



http://www.hitl.washington.edu/artoolkit/

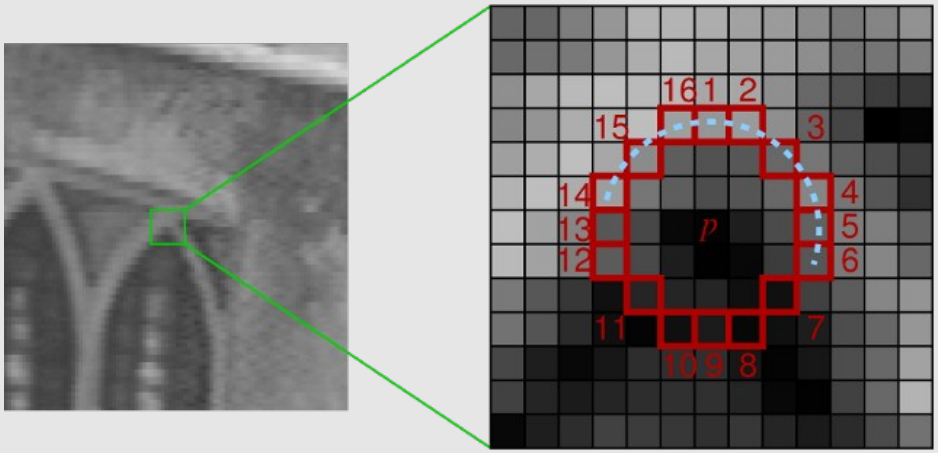
svn co https://artoolkit.svn.sourceforge.net/svnroot/artoolkit artoolkit

OpenSURF

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

http://www.chrisevansdev.com/computer-vision-opensurf.html
svn checkout http://opensurf1.googlecode.com/svn/trunk/ opensurf1-read-only

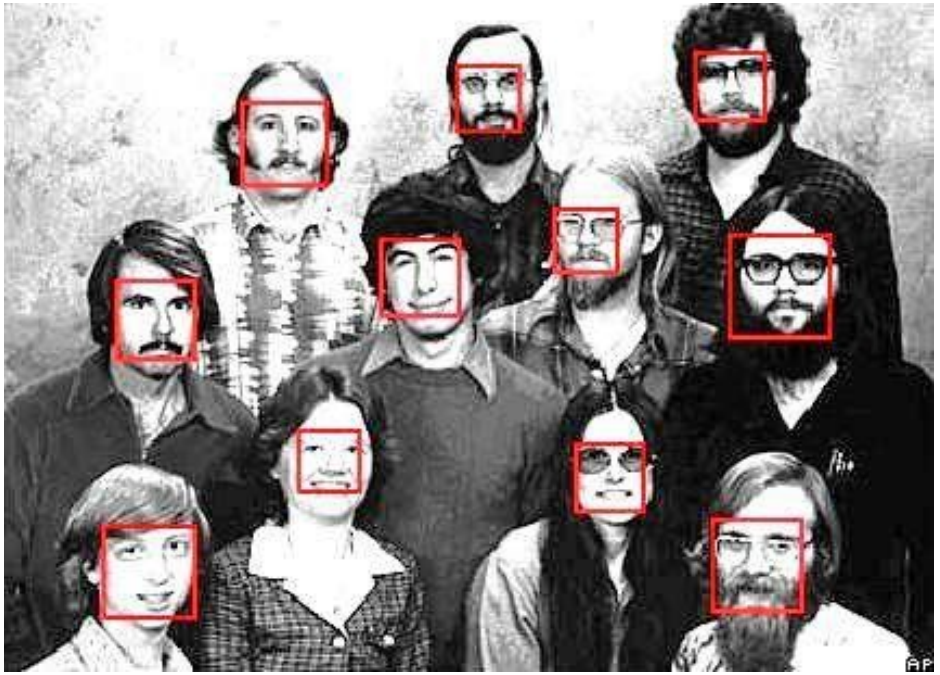
FAST Corner Detection



- Fast
- BSD license
- Compact
- Cross Platform

http://www.edwardrosten.com/work/fast.html

FD lib



- The library is free to use for non-commercial and research purposes and, of course, comes with no warranty. If you would like to use it in a commercial application, please contact Dr. Bernd Ctortecka at Max-Planck-Innovation GmbH.
- W. Kienzle, G. Bakir, M. Franz and B. Scholkopf: Face Detection - Efficient and Rank Deficient. In: Advances in Neural Information Processing Systems 17, pg. 673-680, 2005.

http://www.kyb.mpg.de/bs/people/kienzle/facedemo/facedemo.htm

Gnu Scientific Library

GNU Operating System

Philosophy | Licenses | Education | Downloads | Documentation | Help GNU | Join the FSF!

Why GNU/Linux? Search

GSL - GNU Scientific Library

Introduction

The GNU Scientific Library (GSL) is a numerical library for C and C++ programmers. It is free software under the GNU General Public License.

The library provides a wide range of mathematical routines such as random number generators, special functions and least-squares fitting. There are over 1000 functions in total with an extensive test suite.

The complete range of subject areas covered by the library includes,

Complex Numbers	Roots of Polynomials
Special Functions	Vectors and Matrices
Permutations	Sorting
BLAS Support	Linear Algebra
Eigensystems	Fast Fourier Transforms
Quadrature	Random Numbers
Quasi-Random Sequences	Random Distributions
Statistics	Histograms
N-Tuples	Monte Carlo Integration
Simulated Annealing	Differential Equations
Interpolation	Numerical Differentiation
Chebyshev Approximation	Series Acceleration

http://www.gnu.org/s/gsl/

Festival

- Easy TTS
- Greek translation is closed-source (περίεργο ?) :P
- Kind of old but good enough results
- Univeristy of Endiburgh

http://www.cstr.ed.ac.uk/projects/festival/
sudo apt-get install festival

CMU Sphinx

- Speech to text
- Only english
- Havent really used it yet :P
- Carnegie Mellon University

http://cmusphinx.sourceforge.net/

Sudo apt-get install libshpinxbase-dev libsphinx2-dev

ROS

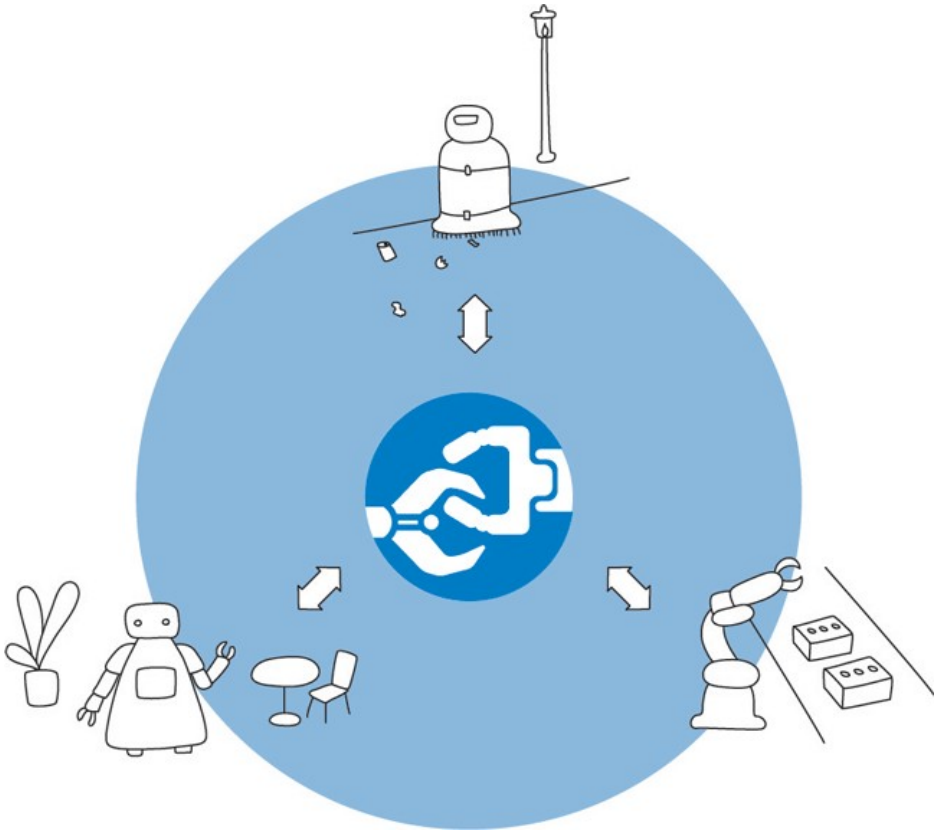
- ROS is an open-source, meta-operating system for your robot. It provides the services you would expect from an operating system, including hardware abstraction, low-level device control, implementation of commonly-used functionality, message-passing between processes, and package management. It also provides tools and libraries for obtaining, building, writing, and running code across multiple computers. ROS is similar in some respects to 'robot frameworks,' such as Player, YARP, Orocos, CARMEN, Orca, MOOS, and Microsoft Robotics Studio.
- ROS currently only runs on Unix-based platforms. Software for ROS is primarily tested on Ubuntu and Mac OS X systems, though the ROS community has been contributing support for Fedora, Gentoo, Arch Linux and other Linux platforms.

Something like my robovision :)

http://www.ros.org/wiki/ROS/

RoboEarth

A Worldwide Web for Robots



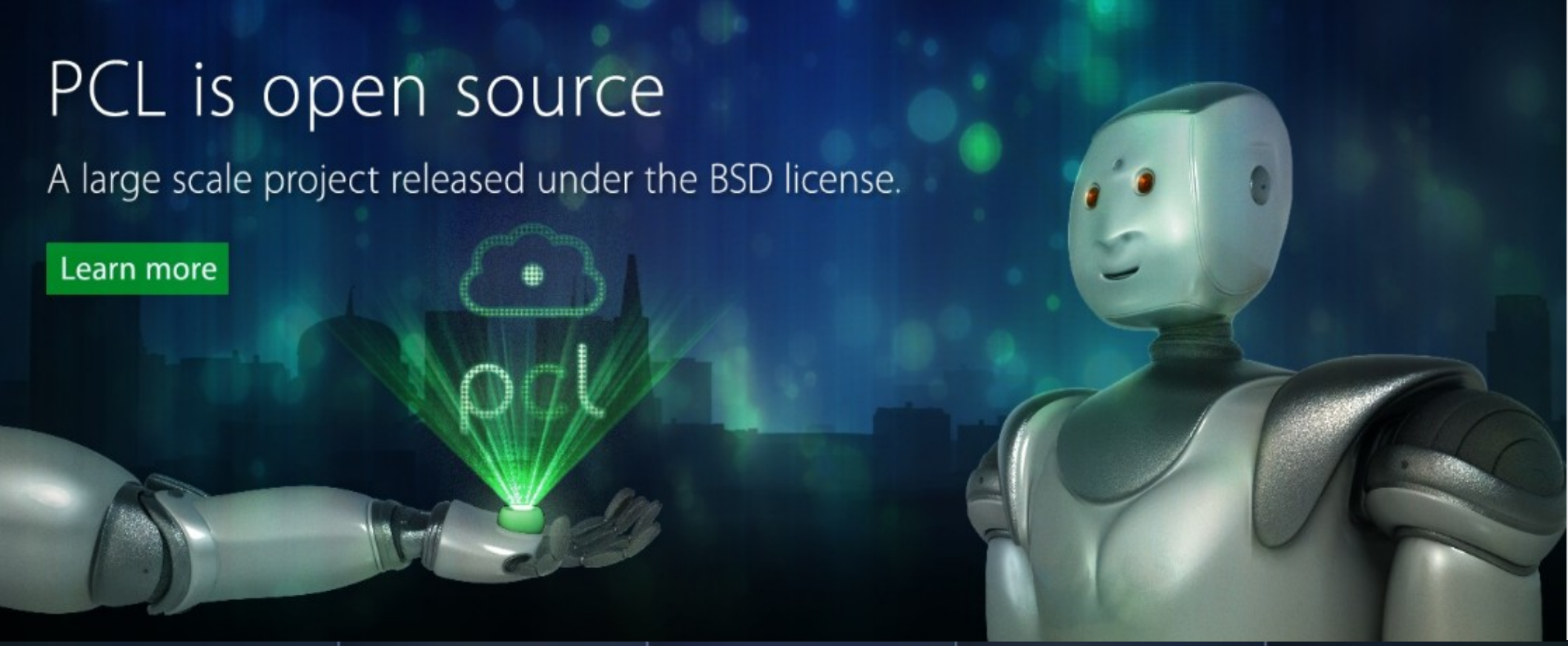
RoboEarth will include everything needed to close the loop from robot to RoboEarth to robot. The RoboEarth World-Wide-Web style database will be implemented on a Server with Internet and Intranet functionality. It stores information required for object recognition (e.g., images, object models), navigation (e.g., maps, world models), tasks (e.g., action recipes, manipulation strategies) and hosts intelligent services (e.g., image annotation, offline learning).

http://www.roboearth.org/

PCL is open source

A large scale project released under the BSD license.

Learn more

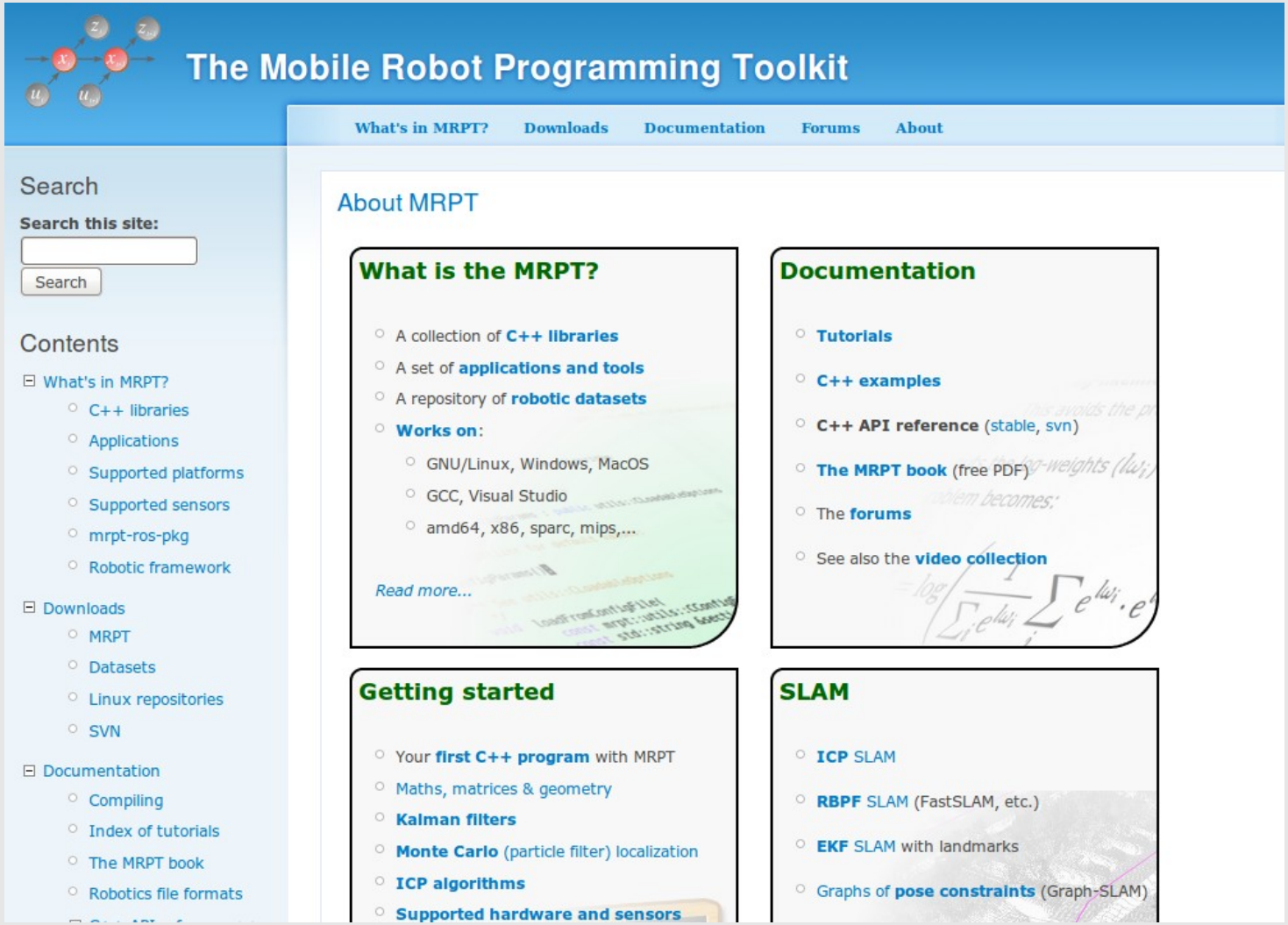


Initial point cloud data | Filtering | Segmentation | Surface reconstruction | Model fitting

The Point Cloud Library (or PCL) is a large scale, open project [1] for point cloud processing. The PCL framework contains numerous state-of-the art algorithms including filtering, feature estimation, surface reconstruction, registration, model fitting and segmentation. These algorithms can be used, for example, to filter outliers from noisy data, stitch 3D point clouds together, segment relevant parts of a scene, extract keypoints and compute descriptors to recognize objects in the world based on their geometric appearance, and create surfaces from point clouds and visualize them -- to name a few.

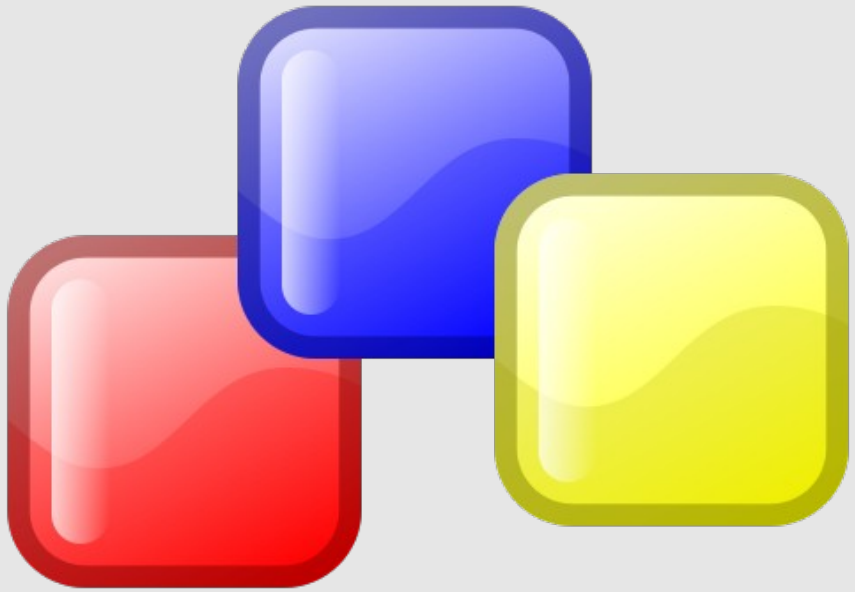
http://pointclouds.org/

Mobile Robot Programing Toolkit

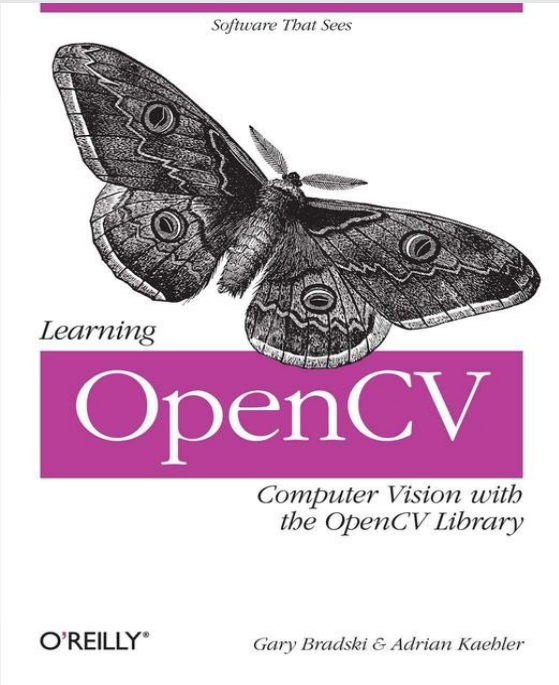


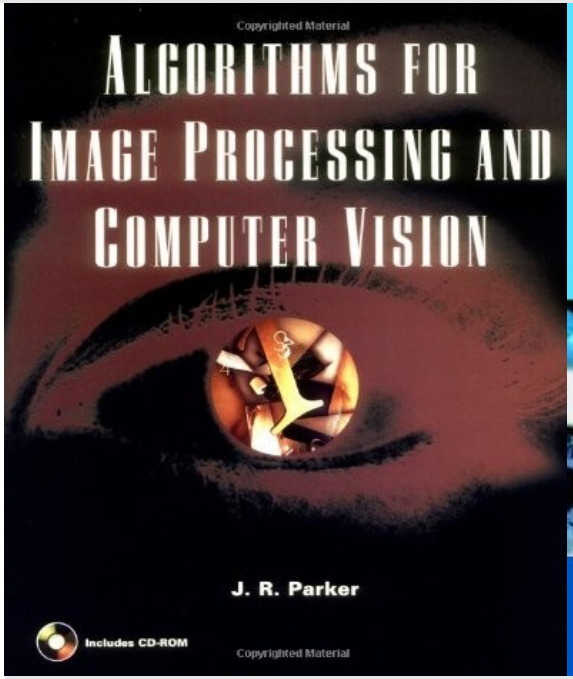
WxWidgets

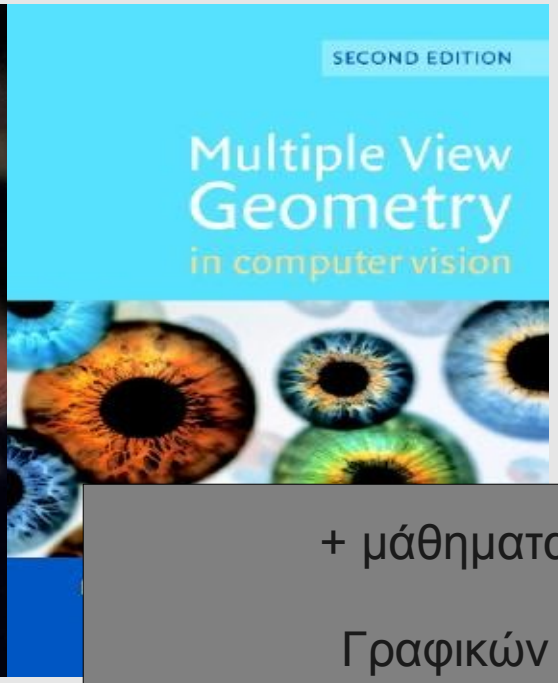
- Crossplatform
- Native Controls
- Easy
- Object Oriented in a good way :)
- Πολλές παρατρεχάμενες libs



Suggested reading
for computer vision



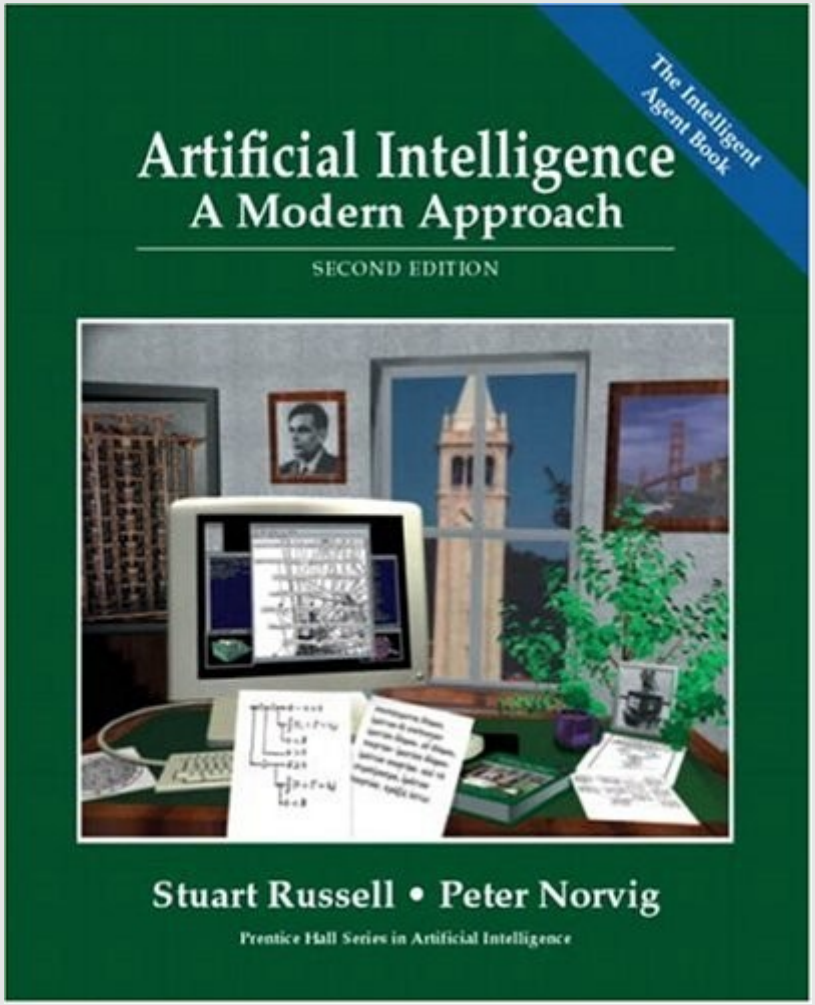




+ μάθηματα

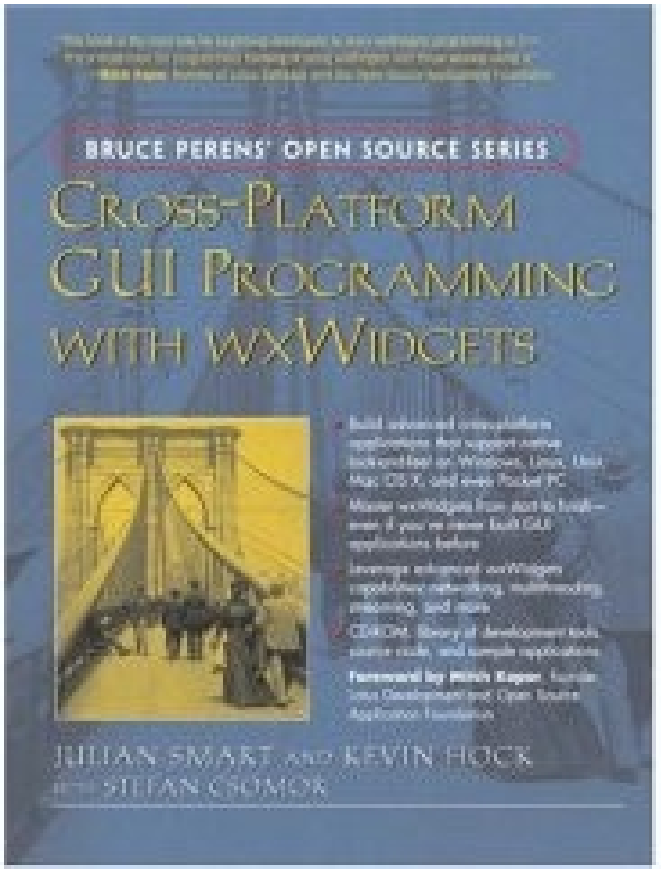
Γραφικών
Επικοινωνία Α/Υ
Τεχνολογία Πολυμέσων
(μακάρι και image processing)

Suggested reading
for AI



+ μάθημα
Τεχνητής νοημοσύνης

Suggested Reading για GUIs




That's all Folks!

but..
I ' LL BE BACK!

THANK YOU FOR YOUR

ATTENTION