Guard Dog Project

Ιστορική Αναδρομή

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

Τι είναι ένα ρομπότ ?

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

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

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

  Τηλεκατευθυνόμενο != Robot

GuarddoG Project
Ο Στόχος

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

- Οι υπολογισμοί πηγαίνουν

1 → 2 → 3

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 !
- Αλλά έχω κάνει και πάρα πολύ δουλειά ήδη..
- Τα standards είναι πολύ υψηλά για να θεωρήθει τελειωμένο..

Μεγάλο Project
=
Μεγάλη παρουσίαση (ελπίζω ενδιαφέρουσα)

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

φτιάξτε το δικό σας ! :)

Αρχιτεκτονική του Software

Το κάθε module έχει σαφώς καθορισμένη λειτουργία

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

Video Input
Visual Cortex
World Mapping
World3D
RVKnowledgebase
Motor HAL

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

Video Input

Πως αναπαριστάται μια εικόνα?

Χρήση ως εξής :

InitVideoInputs(2);
InitVideoFeed(“dev/video0”,320,240,3,1,0);
InitVideoFeed(“dev/video1”,320,240,3,1,0);
unsigned char * GetFrame(int webcam_id);

Video Input

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

picture = GetFrame(0);

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

Video Input

Small synchronization problems (μεταξύ των 2 feeds) λόγω ανυπαρξίας κάποιου hardware clock!

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

(USB controller / Webcam Driver κτλ)

Πρόβλημα #1

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

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

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 !

Θα εξηγήσω συνοπτικά τι περίπου κάνουν τα φίλτρα!

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

Κυρίως πρόβλημα Patch Matching!

Τι ταιριάζει πού ?

Visual Cortex

Τα ίδια αντικείμενα με :

* Ελαφρώς διαφορετική γωνία στον χώρο
* Στον άξονα του χρόνου
* Ανάλογα με τον φωτισμό
* Ηλεκτρομαγνητικό Θόρυβο στο CCD
* 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 το “εκμεταλλευτούμε” το context αυτό , ώστε να έχουμε κάποιο feasable αποτέλεσμα!

Visual Cortex

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

Συγκρίνουμε Patches , μετράμε τις αποστάσεις

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

136 – 62 = distance “**74**”

Visual Cortex

Εμπειρικές μετρήσεις

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

Visual Cortex

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

136 – 62 = distance “**74**”

Visual Cortex

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

Κάθε operation είναι 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

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 (όχι ακραία ταχύ :P) και η μεγαλύτερη εικόνα θα βοηθούσε πολύ περισσότερο για ένα ακριβές αποτέλεσμα

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 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..!
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	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	***10***	***123***	***165***	***200***	***165***	123	10	
X	10	123	165	200	165	123	10	

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

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

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

Άλλες τεχνικές
Visual Cortex

- 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!

Άλλες τεχνικές
Optical Flow

Άλλες τεχνικές
Optical Flow

Άλλες τεχνικές
Optical Flow

Άλλες τεχνικές
Hough transformation , lines

- Line Detection for tracking lines , and improving things

- Ακόμα περισσότερη υπολογιστική δουλειά για το GuarddoG
- Ακόμα καλύτερα Contours αντί για lines κ.ο.κ

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

- TODO list!

Visual Cortex

Ιδανικά :

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

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 ?

World3D

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

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

World3D

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

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

World 3D

Trackers
Moving stuff outside the window doesn't bother the system

World 3D

Reset
Pause
Trackers
Moving stuff outside the window doesn't bother the system

World 3D

World 3D

World 3D

World3D

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

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

2D GuarddoG represenatation

Much better 3d representation

Word3D

http://tinyurl.com/fundamentalmatrix

World Mapping

World Mapping

A * algorithm , with a twist !
LOGO output

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

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

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

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

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

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

int DeleteMap(struct Map * themap) ;
```

World Mapping

Λίγα λόγια για τον A*

A * algorithm

* Optimal

* Manhattan distance heuristic + στροφές 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

++ Compatibility
++ Rationality
++ Simultation easier
++Debugging easier
++Human control easier

```
reset
direction 0.000000
turnright 45.000000
forward 18.384776
turnright 18.434949
forward 8.944272
turnleft 9.462322
forward 13.601471
turnleft 27.407575
forward 6.708204
turnright 2.045408
forward 12.529964
turnleft 28.610460
forward 25.000000
```

Plain Text ▾ Tab Width: 8 ▾ Ln 9, Col 19

```
5 turn
6 forward 8.944272
7 turnleft 9.462322
8 forward 13.601471
9 turnleft 27.407575
10 forward 6.708204
11 turnright 2.045408
12 forward 12.529964
13 turnleft 28.610460
14 forward 25.0
```

Ίδιο αποτέλεσμα με ένα εντελώς διαφορετικό πρόγραμμα που λέγεται k-turtle

World Mapping

Πρόβλημα #3

Πώς μπορούμε να γράψουμε προγράμματα που να κινούν μοτέρ ?

Πώς μπορεί να φτιάξουμε από μόνοι μας μια “συσκευή” ?

Πώς συνδέεται το hardware με το software ?

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
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

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

High level εντολές και ενοποιημένο pipelining για την επεξεργασία τους

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

RV Knowledge base coupled with internet databases

RV Knowledge base coupled with internet databases

GuarddoG in numbers

- 4+ χρόνια (p3**04**0023)
- 3 complete rewrites
- Περίπου 639 euro construction cost το συγκεκριμένο prototype
- C 58% , C++ 39% BAShell 1% , 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

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

Bank Balance = - 60 euro

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

WinXP Embedded

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

αλλά..

WinXP Embedded

WinXP Embedded

No Offence.. :P

Windows Γενικά

- Για να βάλω text to speech

  WinXP/Embedded έως SAPI 4.1

  Vista έως SAPI 5.0

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

Περιορισμοί , περιορισμοί περιορισμοί ...

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

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

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

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

Εν το μεταξύ , για να επανέλθω ..

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

GuarddoG mk2

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

GuarddoG mk2 -> mk3

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

Αλλαγές Hardware -> Αλλαγές Software

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

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

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

FOSS and Contributions

Μπορείτε να :

- Κατεβάσετε
- Χρησιμοποιήσετε
- Μελετήσετε
- Βελτιώσετε

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

τον κώδικα !

Getting started , Checklist

Για το Vision κομμάτι , χρειάζεται:

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

  (Logitech UVC driver ++)

Getting Started , Testing Movement

Για “κίνηση” :

- Arduino Duemillenove
- MD25 Motor Kit
- USB 2 I2C

και άλλες μικρές αγορές ..

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

Replicating GuarddoG

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

Κάποιος μπορεί να έχει το δικό του GuarddoG!

Grand Future Plans!

- Προσοπική φιλοδοξία

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

- DARPA grand challenge style

FAQ

- Τι σχέση έχουν αυτά με την ΑΣΟΕΕ ?

Πολύ λίγη , αλλά αν είναι να ασχοληθώ με για όλη την υπόλοιπη ζωή μου με SAP , SQL λογιστικές εφαρμογές και να φτιάχνω websites , ευχαριστώ δεν θέλω.. καλύτερα να ανοίξω σουβλατζίδικο!

- Τόση δουλειά και την δίνεις OpenSource ?

ΝΑΙ , 15000 γραμμές δεν είναι τίποτα μπροστά στον linux kernel πχ

- Στην αρχή είπες ότι είναι πολύ εύκολο!

Υπάρχει τόση πολύ δουλειά που πλέον είναι έτοιμη που ναι , είναι εύκολο!

- 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..
Επειδή δεν βγάζω ποσοστά από τις πωλήσεις windows δεν έχω κανένα λόγο για pro-windows bias.. Αντίστοιχα από την εμπειρία μου έχω κάθε λόγο για pro-foss bias!

TODO – Contributions Wishlist

Computer Vision / Linear Algebra

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

SLAM efficient implementation

TODO - Contributions Wishlist

etc..

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

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

TODO – Physical things to do

New Plexiglass lasercut ,CAD chassis !!!!
Better cameras
More processing power
etc..

GuarddoG Repository!

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

FOSS Aueb !

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

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

I am **AmmarkoV**
http://ammarkov.ath.cx

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

Suggested reading
for computer vision

+ μάθηματα

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

Suggested reading
for AI

+ μάθημα
Τεχνητής νοημοσύνης

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

WxWidgets

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

Suggested Reading για GUIs

That's all Folks!

but..

I ' LL BE BACK!