Τεχνολογικό Εκπαιδευτικό Ίδρυμα Κρήτης

Σχολή Τεχνολογικών Εφαρμογών

Τμήμα Εφαρμοσμένης Πληροφορικής και Πολυμέσων

Π Τ Υ Χ Ι Α Κ Η Ε Ρ Γ Α Σ Ι Α

ΕΚΤΙΜΗΣΗ ΘΕΣΗΣ ΚΑΜΕΡΑΣ ΓΙΑ ΑΥΤΟΚΙΝΟΥΜΕΝΑ ΡΟΜΠΟΤ

ΔΑΝΙΗΛΙΔΗΣ ΙΩΑΝΝΗΣ Α.Μ. 1338

ΜΑΘΙΟΥΔΑΚΗΣ ΧΡΗΣΤΟΣ Α.Μ. 1487

ΕΠΙΒΛΕΠΩΝ ΚΑΘΗΓΗΤΗΣ :

ΠΑΛΑΜΑΣ ΓΕΩΡΓΙΟΣ

ΗΡΑΚΛΕΙΟ 2010

ΕΥΧΑΡΙΣΤΙΕΣ

Έχοντας ολοκληρώσει την πτυχιακή μας εργασία θα θέλαμε να ευχαριστήσουμε τον επιβλέπων καθηγητή Κ. Παλαμά Γεώργιο για την ευκαιρία που μας έδωσε να ασχοληθούμε με ένα τόσο σημαντικό αλλά και ενδιαφέρον θέμα.

Επίσης αφήνουμε τις πιο θερμές μας ευχαριστίες στις οικογένειές μας για την πολύ σημαντική ηθική αλλά και οικονομική τους στήριξη καθ' όλη την διάρκεια της εκπόνησης της εργασίας αλλά και γενικότερα των σπουδών μας.

ΠΕΡΙΕΧΟΜΕΝΑ

1. ΕΙΣΑΓΩΓΗ

1.1 Γενικά – Προσδιορισμός του προβλήματος

Ένα σημαντικό πρόβλημα στις περισσότερες ρομποτικές εφαρμογές είναι η εύρεση της θέσης του ρομπότ μέσα σε ένα χώρο. Πολλοί ερευνητές εστιάζουν στην εύρεση του διανύσματος θέσης με αισθητήρια αδράνειας και με άλλα αισθητήρια προσέγγισης. Τα τελευταία χρόνια, έχουν αναφερθεί συστήματα και αλγόριθμοι όπου υπολογίζεται η θέση μιας κάμερας από την αναγνώριση ενός γνωστού αντικειμένου.

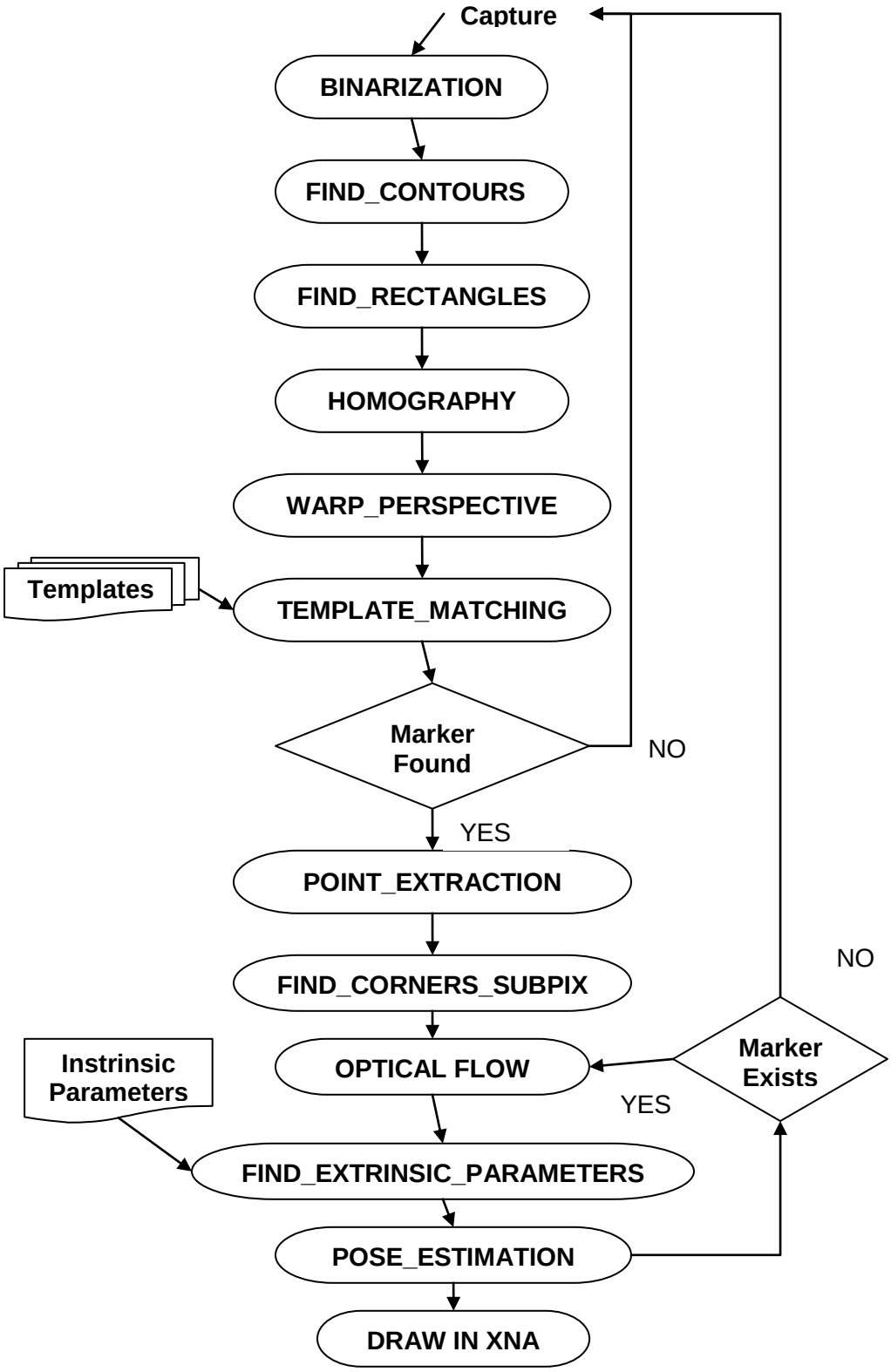
Στην παρούσα εργασία , η εύρεση του διανύσματος θέσης ενός αντικειμένου σε σχέση με μια κάμερα επιτυγχάνεται από την αναγνώριση του αντικειμένου στο επίπεδο της εικόνας και την προβολή του στον τρισδιάστατο χώρο. Η προβολή αυτή επιτυγχάνεται χρησιμοποιώντας τον πίνακα βαθμονόμησης της κάμερας και ενός πίνακα περιστροφής.

Το πρόβλημα που καλούμαστε λοιπόν να λύσουμε είναι να αναπτύξουμε ένα σύστημα τέτοιο ώστε, αφού τοποθετήσουμε ένα αυτόνομο όχημα/ρομπότ σε μια άγνωστη θέση, μέσα σε ένα εκ των προτέρων άγνωστο περιβάλλον, με ένα μόνο γνωστό σχέδιο - αντικείμενο (marker) να είναι ικανό να προσδιορίζει κάθε στιγμή την σχετική θέση του οχήματος μέσα στον χώρο.

1.2 Σκοπός της εργασίας

Σκοπός της παρούσας εργασίας είναι η ανάπτυξη ενός αξιόπιστου, αποδοτικού και φθηνού (μοναδική απαίτηση υλικού είναι η κατοχή μιας απλής web-camera και ενός υπολογιστή) συστήματος εκτίμησης θέσης της κάμερας ικανό να λειτουργήσει σε πραγματικό χρόνο και να είναι ανεκτικό σε εξωτερικές παρεμβολές . Για τον σκοπό αυτό, μας είναι αρκετό να μπορούμε κάθε χρονική στιγμή να γνωρίζουμε το *που βρίσκεται η κάμερα* (localization) σε σχέση με ένα συγκεκριμένο προκαθορισμένο σχέδιο - αντικείμενο (marker). Ο πραγματικός χρόνος εκτέλεσης είναι πολύ σημαντικός για εφαρμογές όπου έχουμε δυναμικά περιβάλλοντα και το αυτοκινούμενο όχημα/ρομπότ είναι αναγκαίο να καθορίσει την επόμενη κίνησή του.

1.3 Διαγραμματική Περιγραφή Αλγορίθμου



1.4 Υλικό – Λογισμικό

Η υλοποίηση της εργασίας πραγματοποιήθηκε σε γλώσσα προγραμματισμού C# με τη χρήση του IDE MS Visual C# 2008 Express Edition σε λειτουργικό περιβάλλον Windows XP - VISTA. Επίσης για την επεξεργασία των εικόνων, την απεικόνιση τους αλλά και την λειτουργία της κάμερας έγινε χρήση της βιβλιοθήκης Emgu CV η οποία είναι ένας wrapper της βιβλιοθήκης Intel OpenCV (Open Computer Vision Library) σε C# . Ακόμα έγινε χρήση της βιβλιοθήκης MS XNA Game Studio για την προβολή των αποτελεσμάτων της εκτίμησης θέσης της κάμερας σε 3d περιβάλλον .

Τέλος η εργασία υλοποιήθηκε σε 2 φορητούς υπολογιστές επεξεργαστικής ισχύος 2.26 GHz x2 και 1.8 GHz x2 με 2 Gb μνήμη έκαστος με ενσωματωμένες web-camera τεχνολογίας pinhole .

2. ΚΑΜΕΡΑ

2.1 Γενικά περί κάμερας

Για την κατανόηση του τρόπου δόμησης και λειτουργίας όλου του συστήματος που αναπτύχθηκε, κρίνεται σκόπιμο να παρουσιαστούν και αποσαφηνιστούν κάποιες βασικές αρχές λειτουργίας των καμερών, των συστημάτων συντεταγμένων τους και του τρόπου επεξεργασίας των εικόνων που λαμβάνονται από αυτές.

Η κάμερα είναι η συσκευή που χρησιμοποιείται για την καταγραφή εικόνων που περιγράφουν το περιβάλλον το οποίο μας ενδιαφέρει. Στην ουσία είναι ένα μέσο καταγραφής από τον 3D χώρο στην δισδιάστατη εικόνα. Το μοντέλο της κάμερας που χρησιμοποιήθηκε για την λήψη των εικόνων είναι το βασικό μοντέλο Pinhole κάμερας. Το μοντέλο αυτό εκτιμάται ότι είναι το πιο απλό και ιδανικό μοντέλο για την περιγραφή των λειτουργιών μιας κάμερας.

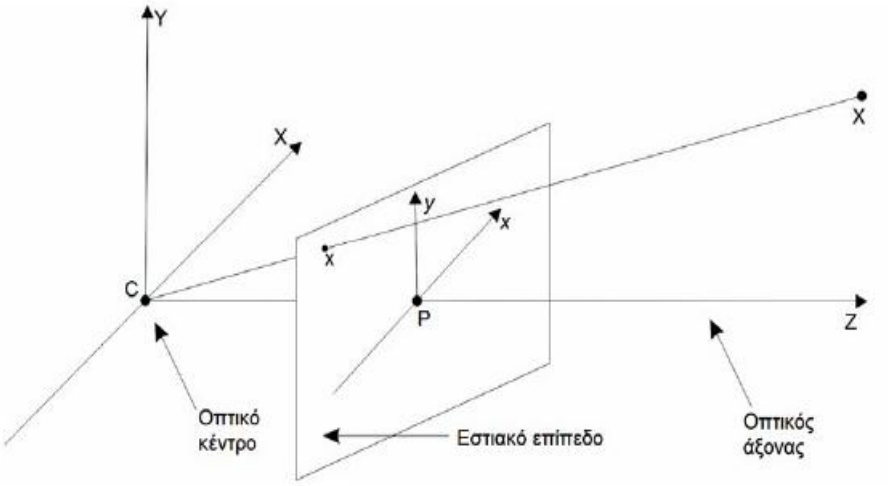
Βασικό μοντέλο Pinhole κάμερας :

Η κάμερα έχει μια οπή από την οποία διέρχεται το φως και δημιουργεί μια ανεστραμμένη εικόνα (ανεστραμμένο αντικείμενο) στην επιφάνεια της κάμερας που βρίσκεται σε απόσταση f από την οπή. Για λόγους απλότητας θεωρούμε ότι η επιφάνεια της κάμερας βρίσκεται ανάμεσα στην οπή και το αντικείμενο, όπως φαίνεται στην εικόνα 2.1. Η εικόνα που αποτυπώνεται πάνω στην επιφάνεια της κάμερας δεν είναι πλέον ανεστραμμένη.

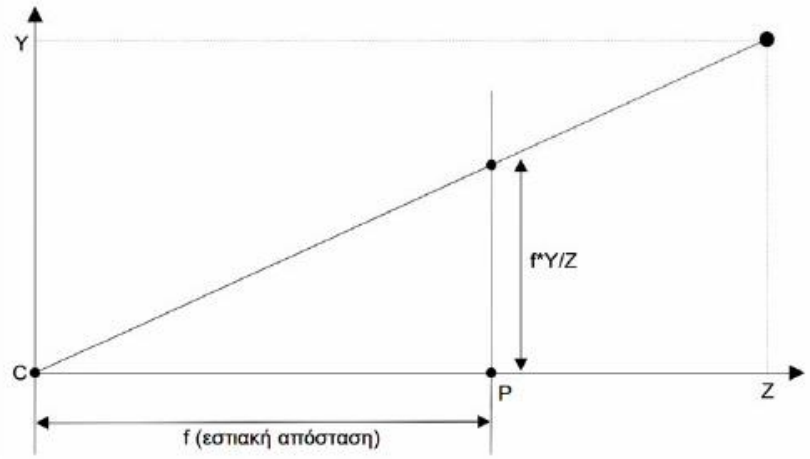
Κεντρική προβολή (central projection) :

Η εικόνα 2.1 απεικονίζει μια κάμερα που είναι τοποθετημένη στην αρχή του καρτεσιανού συστήματος συντεταγμένων. Το κέντρο του συστήματος C ονομάζεται οπτικό κέντρο ή κέντρο της κάμερας (camera center). Θεωρούμε ένα επίπεδο σε απόσταση f από το κέντρο C, όπου είναι το επίπεδο στο οποίο προβάλλονται τα σημεία του χώρου. Το επίπεδο αυτό ονομάζεται εστιακό επίπεδο (image plane) και η απόστασή του f από το κέντρο C ονομάζεται εστιακή απόσταση (focal length). Ουσιαστικά λοιπόν, το κέντρο του εστιακού επιπέδου P θεωρούμε ότι είναι η προβολή του σημείου C πάνω

στο εστιακό επίπεδο. Επίσης, η ημιευθεία που ξεκινά από το σημείο C και εκτείνεται στο άπειρο ενώ τέμνει κάθετα το εστιακό επίπεδο ονομάζεται οπτικός άξονας (principal axis).



Εικόνα 2.1 : Γεωμετρία Pinhole κάμερας , άξονες X,Y,Z.



Εικόνα 2.2 : Γεωμετρία Pinhole κάμερας , άξονες Y,Z.

Για να εξηγηθεί το πως τα σημεία του ευκλείδειου τρισδιάστατου χώρου απεικονίζονται στον δυσδιάστατο χώρο (πάνω στο εστιακό επίπεδο), εστιάζουμε την προσοχή μας στην εικόνα 2.2. Εδώ απεικονίζεται το παραπάνω περιγραφόμενο σύστημα παρατηρώντας το από πλάγια, δηλαδή από το επίπεδο YZ. Από τα όμοια τρίγωνα που σχηματίζονται μπορούμε εύκολα να καταλάβουμε ότι οποιοδήποτε σημείο του χώρου με συντεταγμένες $(X,Y,Z)^{\intercal}$προβάλλεται πάνω στο εστιακό επίπεδο στο σημείο $(fX\ /\ Z, fY\ /\ Z)$. Ισχύει δηλαδή η σχέση:

$$(X,Y,Z)^{\intercal} \rightarrow \left(f\frac{X}{Z}, f\frac{Y}{Z}\right)^{T} \qquad (2.1)$$

όπου στο εστιακό επίπεδο η τρίτη διάσταση αγνοείται, αφού είναι σταθερή για όλα τα σημεία του.

Για την καλύτερη επεξεργασία και διαχείριση των σημείων στο εστιακό επίπεδο αλλά και γενικότερα στον χώρο , χρησιμοποιούμε ομογενείς συντεταγμένες :

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.2)$$

Ο πίνακας $P = \begin{bmatrix} f_x & 0 & 0 & 0 \\ 0 & f_y & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$ ονομάζεται πίνακας προβολής (camera projection matrix) και μπορεί να γραφτεί ως $P - diag(f_x, f_y, 1)[I|0]$ όπου $diag(f_x, f_y, 1)$ είναι διαγώνιος πίνακας και το $[I|0]$ αναπαριστά πίνακα που χωρίζεται σε ένα μοναδιαίο πίνακα διάστασης 3x3 και έναν μηδενικό πίνακα στήλη διάστασης 3x1 (διάνυσμα). Αν θεωρήσουμε ένα σημείο X του τρισδιάστατου χώρου, όπου η αναπαράστασή του σε ομογενείς συντεταγμένες είναι $(X,Y,Z,1)^{T}$, τότε αυτό το σημείο προβάλλεται στην εστιακή εικόνα στο σημείο x, σύμφωνα με την εξίσωση $X = PX$.

.

Προβολή με μετατοπισμένο οπτικό κέντρο (principal point offset) :

Όλα τα παραπάνω ισχύουν στην περίπτωση της «κεντρικής προβολής» (central projection) όπου το κέντρο της εικόνας θεωρείται ότι έχει συντεταγμένες $p = (0,0,f)^T$. Αν θεωρήσουμε το κέντρο της εικόνας στο πραγματικό κέντρο του εστιακού επιπέδου, δηλαδή $p = (p_x, p_y, f)^T$ όπως φαίνεται στην εικόνα 2.3. Στην περίπτωση αυτή έχουμε «προβολή με μετατοπισμένο οπτικό κέντρο» (principal point offset) και οι εξισώσεις 2.1 και 2.2 γίνονται αντίστοιχα :

$$(X,Y,Z)^T \rightarrow \left(f\frac{X}{Z} + p_x\ , f\frac{Y}{Z} + p_y\right)^T \quad (2.3)$$

Και

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \rightarrow \begin{pmatrix} fX + Zp_x \\ fY + Zp_y \\ Z \end{pmatrix} = \begin{bmatrix} f_x & 0 & p_x & 0 \\ 0 & f_y & p_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \quad (2.4)$$

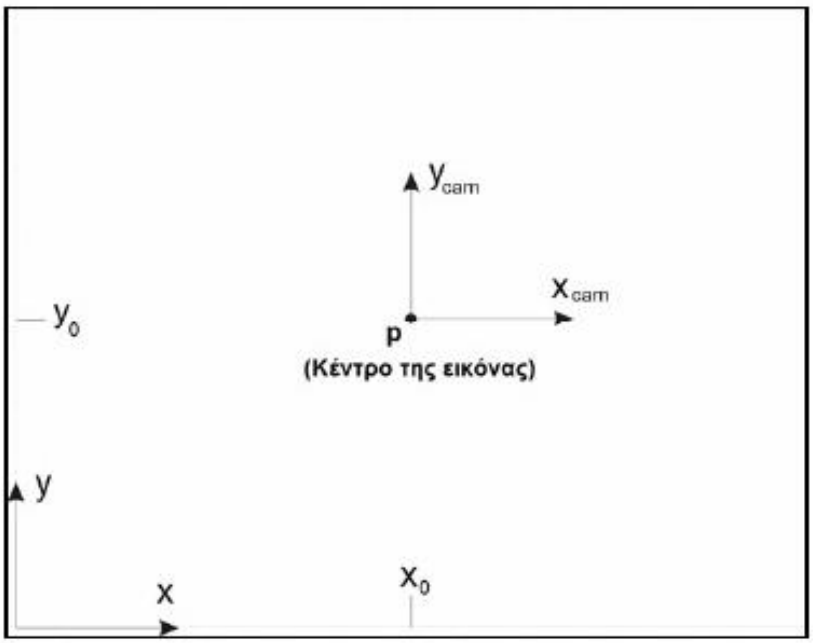
Όπου

$$K = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_x & p_y \\ 0 & 0 & 1 \end{bmatrix} \quad (2.5)$$

όπου K είναι ο πίνακας βαθμονόμησης της κάμερας (camera calibration matrix) και ισχύει ότι :

$$x = K[I|0]x_{cam} \quad (2.6)$$

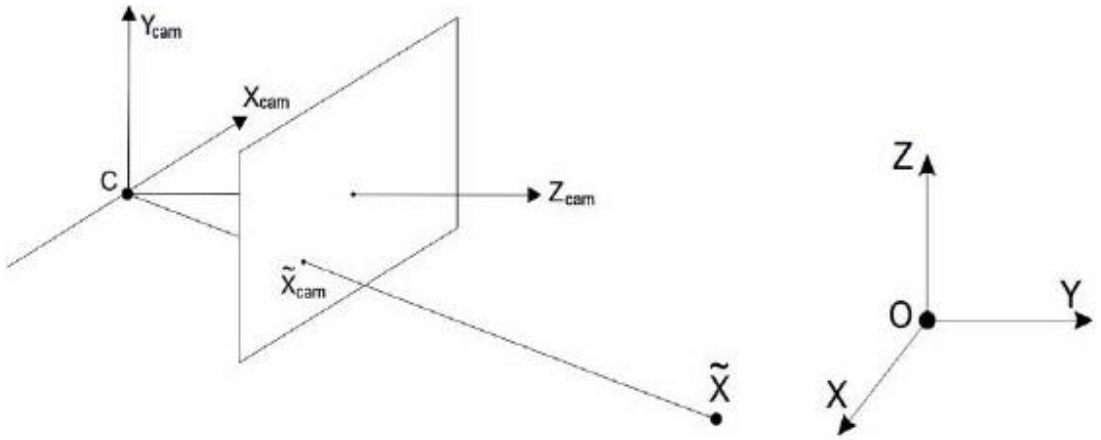
Το νέο σύστημα συντεταγμένων που έχει κέντρο της εικόνας το σημείο ονομάζεται «σύστημα συντεταγμένων της κάμερας» (camera coordinate system).



Εικόνα 2.3 : Εικόνα με μετατοπισμένο οπτικό κέντρο.

Περιστροφή και μεταφορά κάμερας (camera rotation and translation) :

Τα σημεία στον 3D χώρο εκφράζονται γενικά στο παγκόσμιο σύστημα συντεταγμένων (world coordinate system). Το σύστημα συντεταγμένων της κάμερας και το παγκόσμιο σύστημα συντεταγμένων συνδέονται μέσω περιστροφής (rotation) και μετατόπισης (translation).



Εικόνα 2.4 : Ο ευκλείδειος μετασχηματισμός μεταξύ συντεταγμένων κάμερας και παγκόσμιου συστήματος συντεταγμένων.

Αν θεωρήσουμε $\overset{o}{X} = (x_s, y_s, z_s)^T$ ένα σημείο του χώρου στο παγκόσμιο σύστημα συντεταγμένων, τότε από την εικόνα 2.4 παρατηρούμε ότι το ίδιο σημείο στο σύστημα συντεταγμένων της κάμερας θα είναι $\overset{o}{X}_{cam} = R\left(\overset{o}{X} - \overset{o}{C} \right)$ όπου $\overset{o}{C}$ είναι το διάνυσμα OC δηλαδή οι συντεταγμένες του κέντρου της κάμερας εκφρασμένες στο παγκόσμιο σύστημα συντεταγμένων και R είναι ο 3x3 πίνακας στροφής που εκφράζει τον προσανατολισμό του συστήματος συντεταγμένων της κάμερας ως προς το παγκόσμιο σύστημα συντεταγμένων. Εκφράζοντας την παραπάνω εξίσωση σε ομογενείς συντεταγμένες έχουμε :

$$X_{cam} = \begin{bmatrix} R & -R\,\overset{o}{C} \\ 0 & 1 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} R & -R\,\overset{o}{C} \\ 0 & 1 \end{bmatrix} X \qquad (2.7)$$

Από τις (2.5) και (2.6) προκύπτει η σχέση :

$$x = KR\left[I \mid -\overset{o}{C} \right] X \qquad (2.8)$$

που δείχνει την σχέση ενός σημείου X του χώρου εκφρασμένο στο παγκόσμιο σύστημα συντεταγμένων και του ίδιου σημείου x εκφρασμένο στο σύστημα συντεταγμένων της κάμερας. Ο πίνακας βαθμονόμησης K περιέχει τις εσωτερικές παραμέτρους (internal camera parameters) της κάμερας (εστιακή απόσταση *f*, συντεταγμένες κέντρου του εστιακού επιπέδου p_x και p_y) ενώ οι παράμετροι R και $\overset{o}{C}$ ονομάζονται εξωτερικές παράμετροι της κάμερας (external camera parameters) και σχετίζουν τη θέση και τον προσανατολισμό της κάμερας ως προς το παγκόσμιο σύστημα συντεταγμένων.

2.2 Βαθμονόμηση Κάμερας (Camera Calibration)

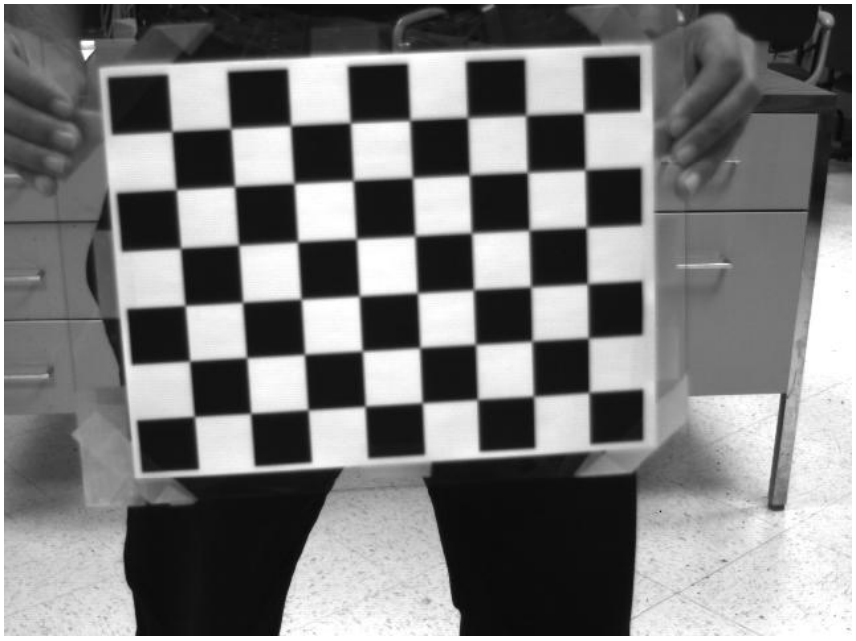
Η διαδικασία της βαθμονόμησης της κάμερας (camera calibration) ουσιαστικά οδηγεί στην εύρεση του πίνακα βαθμονόμησης δηλαδή στην εύρεση των εσωτερικών παραμέτρων της κάμερας που χρησιμοποιήθηκε.

Στη παρορούσα εργασία χρησιμοποιήθηκε η συνάρτηση CalibrateCamera() της βιβλιοθήκης Emgu Cv για την εύρεση του πίνακα βαθμονόμησης της κάμερας (εσωτερικοί παράμετροι - intrinsic parameters). Αναλυτικότερα , χρησιμοποιείται μία κόλλα Α4 που περιέχει λευκά και μαύρα τετράγωνα σε διάταξη σκακιέρας προκαθορισμένου αριθμού γραμμών – στηλών(βλέπε εικόνα 2.5) (μεγαλύτερη από 4x4 τετράγωνα ή αλλιώς 3x3 εσωτερικές γωνίες). Δημιουργείται ένας πίνακας με σημεία που απεικονίζουν τις εσωτερικές γωνίες μιας σκακιέρας (εικονικά σημεία)(βλέπε εικόνα 2.6). Στη συνέχεια φωτογραφίζεται η σκακιέρα από διάφορες οπτικές γωνίες (βλέπε εικόνα 2.7)(όσο περισσότερες τόσο πιο ακριβής θα είναι ο πίνακας βαθμονόμησης). Σε κάθε λήψη ανιχνεύονται οι εσωτερικές της γωνίες (βλέπε εικόνα 2.8) και αποθηκεύονται σε ένα άλλο πίνακα. Για τον σκοπό αυτό γίνεται χρήση μίας άλλης συνάρτησης , η FindChessboardCorners()(θα αναλυθεί εκτενέστερα σε επόμενο κεφάλαιο). Τέλος για να βρεθούν οι εσωτερικές παράμετροι αντιστοιχίζονται τα πραγματικά σημεία από όλες τις λήψεις της σκακιέρας με τα εικονικά.

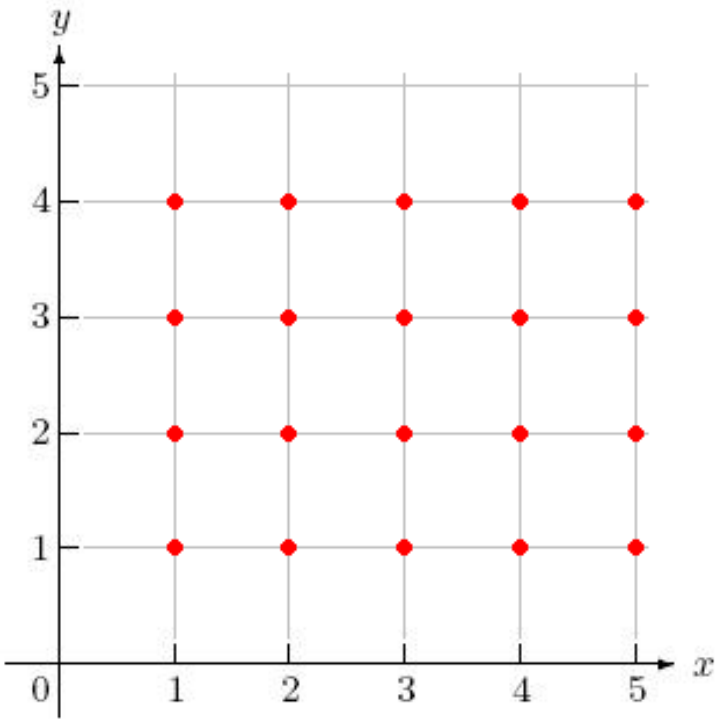
Ενδεικτικά αποτελέσματα Camera Calibration

Focal Length: fc = [670.480 676.873]
Principal point: cc = [371.687 271.948]
Distortion: kc = [0.258703 -1.533815 0.008192 0.025492]
Camera Matrix = [670.480 0 371.687; 0 676.873 271.948; 0 0 1]

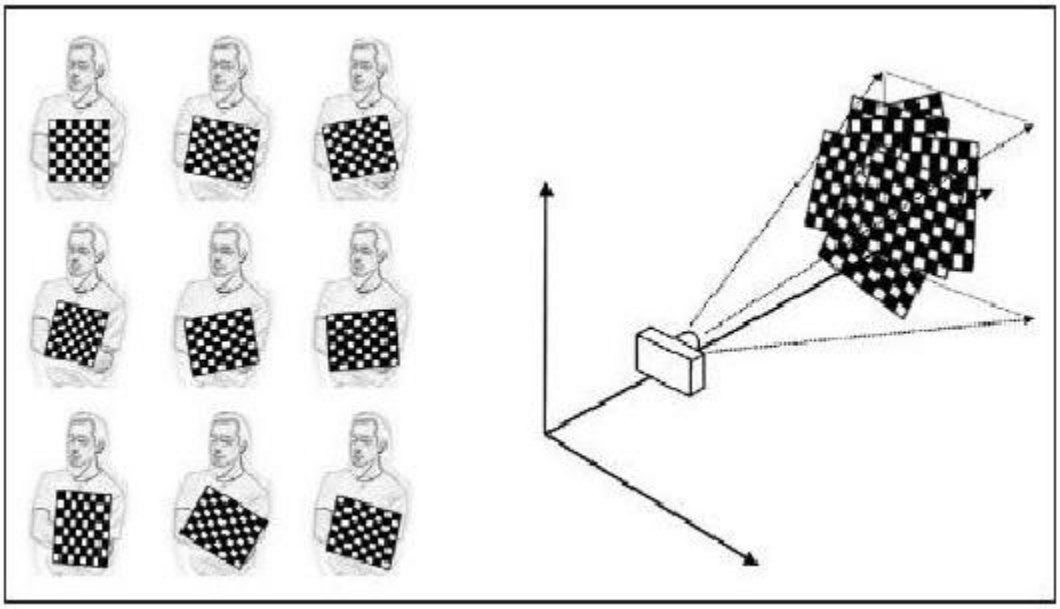
Βρέθηκε δηλαδή η εστιακή απόσταση (fc), το κέντρο του εστιακού επιπέδου (cc) καθώς επίσης και οι συντελεστές παραμόρφωσης (kc).



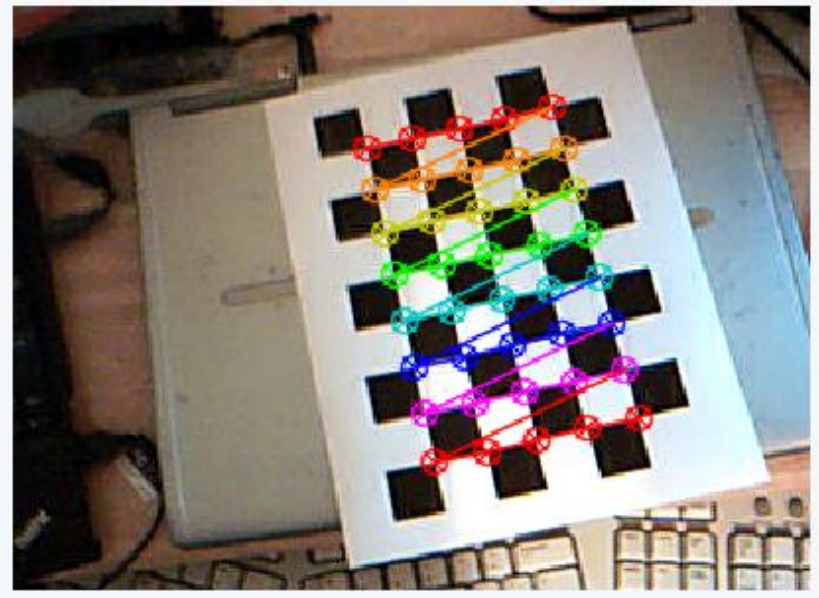
Εικόνα 2.5 : Σκακιέρα διαστάσεων 9x7 τετραγώνων.



Εικόνα 2.6 : Σημεία που απεικονίζουν τις εσωτερικές γωνίες μιας σκακιέρας.



Εικόνα 2.7 : Εικόνες της σκακιέρας τραβηγμένες από διαφορετικές οπτικές γωνίες.



Εικόνα 2.8 : Εικόνα όπου έχουν επισημανθεί οι εσωτερικές γωνίες της σκακιέρας.

Επεξήγηση μεθόδου *CalibrateCamera( )* :

```
public static void CalibrateCamera (
        MCvPoint3D32f [ ] [ ] objectPoints,
        PointF [ ] [ ] imagePoints,
        Size imageSize,
        IntrinsicCameraParameters intrinsicParam,
        CALIB_TYPE flags,
        out ExtrinsicCameraParameters[] extrinsicParams
)
```

όπου : objectPoints

Ένας πίνακας σημείων που απεικονίζουν την σκακιέρα (εικονική σκακιέρα)(βλέπε εικόνα 2.6).

imagePoints

Ένας πίνακας των συντεταγμένων των γωνιών της σκακιέρας που έχουν ανιχνευθεί μέσα σε κάθε λήψη της (βλέπε εικόνα 2.8).

imageSize

Το πλήθος των εσωτερικών γωνιών της σκακιέρας.

intrinsicParam

Ένας πίνακας για την επιστροφή των αποτελεσμάτων των εσωτερικών παραμέτρων της κάμερας.

flags

Μέθοδος βαθμονόμησης που θα ακολουθηθεί.

extrinsicParams

Ένας πίνακας για την επιστροφή των αποτελεσμάτων των εξωτερικών παραμέτρων της κάμερας για κάθε λήψη.

2.3 Χρήση κάμερας

Όλη η εργασία στηρίζεται στην λήψη φωτογραφιών και στην επεξεργασία τους. Ένας από τους βασικούς στόχους είναι η λειτουργία σε πραγματικό χρόνο. Αυτό επιτυγχάνεται εν μέρει από την ταχύτητα της κάμερας που χρησιμοποιείται. Για να δούμε video σε πραγματικό χρόνο η κάμερα πρέπει να λειτουργεί σε τουλάχιστον 20-25 frames per second (fps). Αυτό σημαίνει πως πρέπει να γίνονται τουλάχιστον 20-25 λήψεις φωτογραφιών το δευτερόλεπτο και στην περίπτωσή μας να γίνεται και η επεξεργασία τους σε αυτόν τον χρόνο. Πρακτικά το Emgu Cv που χρησιμοποιήθηκε λειτουργεί μέχρι 30 fps . Για την λήψη των εικόνων χρησιμοποιείται η συνάρτηση Capture.QueryFrame() .

3. ΑΝΑΓΝΩΡΙΣΗ MARKER

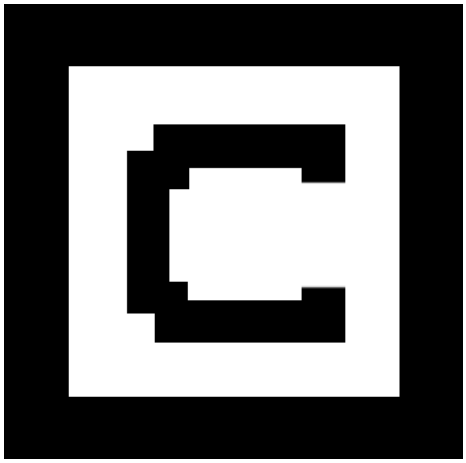
3.1 Γενικά

Ένα μεγάλο πρόβλημα που καλούνται να λύσουν σήμερα οι ερευνητές είναι ο προσδιορισμός της θέσης του ρομπότ σε ένα χώρο. Στην παρούσα εργασία το ρομπότ για να αντιληφθεί την θέση του στον χώρο χρειάζεται κάποια προκαθορισμένα αντικείμενα – σχέδια γνωστά σε αυτό έτσι ώστε μετά από κάποιες διαδικασίες να υπολογίσει την θέση του σε σχέση με αυτά. Τα αντικείμενα – σχέδια αυτά ονομάζονται markers.

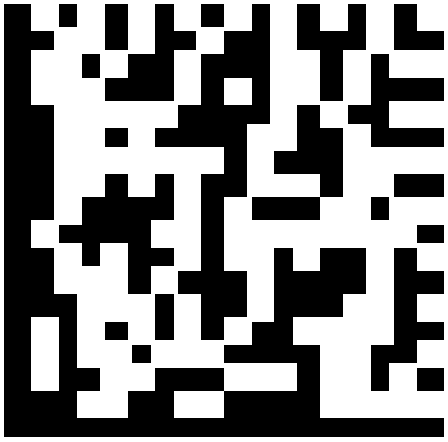
Τι είναι marker.

Marker είναι ένα αντικείμενο - σχέδιο προκαθορισμένο και μοναδικό. Αυτό σημαίνει πως πρέπει να έχει αρχικά συγκεκριμένο σχήμα (τετράγωνο ,ορθογώνιο ,τρίγωνο , κυκλικό) αλλά το πιο σύνηθες είναι το τετράγωνο. Πρέπει να έχει έντονες χρωματικές εναλλαγές (συνήθως είναι ασπρόμαυρα) έτσι ώστε με την μετατροπή της εικόνας σε ασπρόμαυρη να είναι ξεκάθαρα τα σκούρα σημεία από τα ανοιχτόχρωμα (thresholding). Ακόμη σημαντικό ρόλο παίζει και το περίγραμμα που πρέπει να είναι σταθερό σε όλα τα markers που χρησιμοποιούνται σε ένα project.Αυτό συμβαίνει γιατί πρώτα αναγνωρίζεται το εξωτερικό περίγραμμα και στην συνέχεια ερευνάται το εσωτερικό περιεχόμενο. Το εσωτερικό περιεχόμενο μπορεί να ποικίλει ανάλογα το είδος του marker. Ανάλογα με το περιεχόμενο αλλάζει και ο τρόπος αναγνώρισής του κάθε marker.

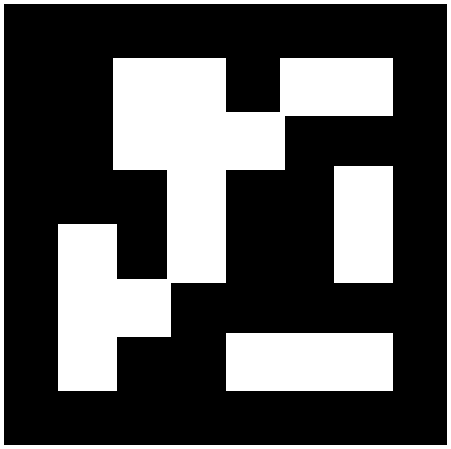
Τα πιο συνήθη markers έχουν τις παρακάτω μορφές:



Εικόνα 3.1 : Template marker

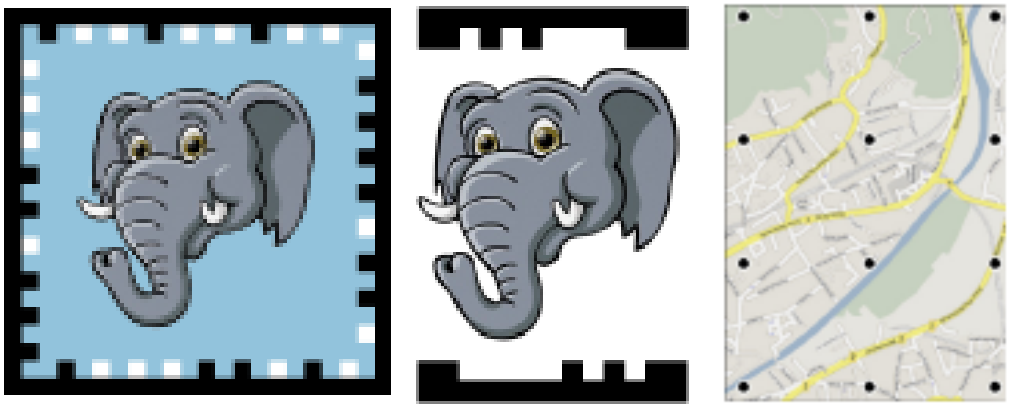


Εικόνα 3.2 : DataMatrix marker



Εικόνα 3.3 : ID marker

Εναλλακτικές μορφές markers είναι οι παρακάτω :



Εικόνα 3.4 :Frame marker - Split markers - Dot markers(από αριστερά προς τα δεξιά).

3.2 Περιγραφή Αλγορίθμου Αναγνώρισης Marker

Στην προηγούμενη ενότητα αναφέρθηκε μια γενική εικόνα για το τι είναι marker και κάποια γενικά τους χαρακτηριστικά. Ο κάθε marker , ανάλογα με το είδος του , χρειάζεται και ένα διαφορετικό τρόπο να ανιχνευτεί και να αναγνωριστεί. Στην παρούσα εργασία γίνεται χρήση ενός template marker , οπότε παρακάτω θα αναλυθεί ένας προτεινόμενος αλγόριθμος για την αναγνώριση ενός τέτοιου marker με την χρήση της βιβλιοθήκης Emgu Cv.

Μια τέλεια αναγνώριση marker θα πρέπει να είναι όσο το δυνατό πιο άμεση , ανεκτική σε πιθανές παραμορφώσεις του marker , να μην επηρεάζεται από εξωτερικές παρεμβολές(π.χ. χαμηλή φωτεινότητα) και να είναι όσο το δυνατόν πιο ακριβής στα αποτελέσματά της. Ο αλγόριθμος που δημιουργήθηκε προσπαθεί να καλύψει όλες τις παραπάνω απαιτήσεις και κατά ένα μεγάλο ποσοστό επιτυγχάνεται.

Σύντομη περιγραφή αλγορίθμου αναγνώρισης marker .

(Τα παρακάτω αναλύονται εκτενέστερα στην συνέχεια της ενότητας).

- Λήψη εικόνας από την κάμερα.
- Μετατροπή εικόνας σε ασπρόμαυρη(Adaptive Threshold).
- Ανίχνευση περιγραμμάτων στην εικόνα.
- Αναγνώριση τετραγώνων – ορθογωνίων.
- Φιλτράρισμα τετραγώνων με σχετικά κριτήρια.
- Εύρεση ομογραφίας για κάθε τετράγωνο.
- Αναδίπλωση – περιστροφή (Warp Perspective) και τροποποίηση της εικόνας(κατάλληλα για σύγκριση).
- Σύγκριση εικόνων με templates.
- Αναγνώριση marker και εκτίμηση προσανατολισμού.
- Εύρεση σημείων για track (Αναλύεται σε επόμενη ενότητα).

3.3 Μετατροπή εικόνας σε ασπρόμαυρη (Adaptive Threshold)

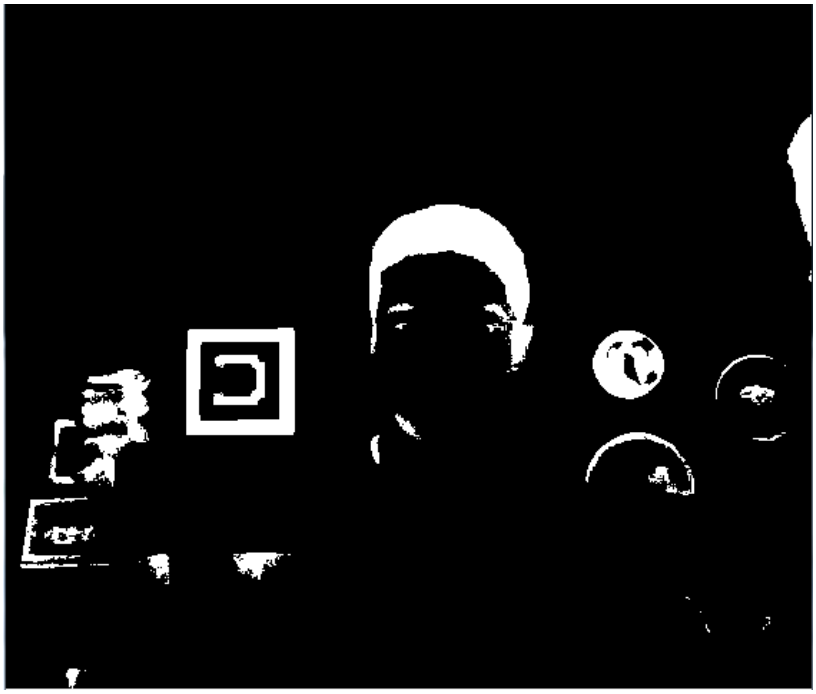
Στην παρούσα φάση καλούμαστε να αναγνωρίσουμε ένα marker μέσα σε ένα χώρο. Για το σκοπό αυτό χρειάζονται ένα η παραπάνω στιγμιότυπα του χώρου. Για αυτό πραγματοποιούνται λήψεις εικόνων από την κάμερα. Οι εικόνες αυτές αρχικά είναι έγχρωμες με υψηλή ανάλυση (βλέπε εικόνα 3.5) , γεγονός που τις καθιστά δύσχρηστες για επεξεργασία.

Σε άλλες μεθόδους αναγνώρισης ίσως αυτό να ήταν απόλυτα χρήσιμο , όμως στην περίπτωση μας δεν είναι αναγκαίο, αντιθέτως , καθυστερεί την όλη διαδικασία. Για τον λόγο αυτό , αρχικά , η κάθε εικόνα μετατρέπεται σε αποχρώσεις του γκρι (Grayscale) (βλέπε εικόνα 3.6). Στην συνέχεια η grayscale εικόνα μετατρέπεται σε ασπρόμαυρη (βλέπε εικόνα 3.7) , περνώντας από την διαδικασία του Adaptive Threshold (η μέθοδος αναλύεται παρακάτω) .Χρησιμοποιείται Adaptive ή αλλιώς δυναμικό threshold και όχι απλό για να μην υπάρχουν αλλοιώσεις ή παραμορφώσεις στην ασπρόμαυρη εικόνα λόγο φωτεινότητας και ανακλάσεων στην αρχική εικόνα. Σύμφωνα με την μέθοδο αυτή η τιμή του κατωφλιού επιλέγεται διαφορετική για κάθε pixel σε κάθε τμήμα της εικόνας σε αντίθεση με την απλή κατωφλίωση όπου όλα τα pixel έχουν σταθερό κατώφλι.

Μια ασπρόμαυρη εικόνα περιέχει πολύ μικρότερο μέγεθος πληροφορίας σε σχέση με οποιοδήποτε άλλο είδος εικόνας. Αυτό σημαίνει πως η επεξεργασία της γίνεται πολύ πιο εύκολα και γρήγορα , πράγμα πολύ καίριο στην παρούσα εργασία. Επίσης γίνονται πολύ πιο ευδιάκριτες οι γωνίες και οι ευθείες μέσα στην εικόνα. Ακόμη πρέπει να επισημανθεί ότι για τις ανάγκες της εργασίας χρησιμοποιήθηκε διαδικασία αντιστροφής άσπρου – μαύρου. Σε κάθε άλλη περίπτωση θα μπορούσε να χρησιμοποιηθεί Adaptive threshold στην αρχική έγχρωμη εικόνα , στην συγκεκριμένη περίπτωση όμως γίνεται μετατροπή της εικόνας σε αποχρώσεις του γκρι (Grayscale) ,λόγο περιορισμού του Emgu Cv.

Εικόνα 3.5 : Έγχρωμη εικόνα.

Εικόνα 3.6 : Grayscale εικόνα.



Εικόνα 3.7 : Ασπρόμαυρη εικόνα αντεστραμμένων χρωμάτων(color inverted).

Επεξήγηση μεθόδου *cvAdaptiveThreshold( )* :

```
public static void cvAdaptiveThreshold(
        IntPtr src,
        IntPtr dst,
        double maxValue,
        ADAPTIVE_THRESHOLD_TYPE adaptiveType,
        THRESH thresholdType,
        int blockSize,
        double param1
)
```

όπου : src

Η εικόνα που εισάγεται έτσι ώστε να μετατραπεί σε ασπρόμαυρη.

dst

Η εικόνα που αποθηκεύεται το αποτέλεσμα της επεξεργασίας.
(Πρέπει να είναι ίδιου τύπου όπως η εικόνα src).

maxValue

Η μέγιστη τιμή που μπορεί να πάρει το κατώφλι .Χρησιμοποιείται με το CV_THRESH_BINARY και CV_THRESH_BINARY_INV είδη κατωφλιών.

adaptiveType

Είδος δυναμικής κατωφλίωσης.

thresholdType

Είδος κατωφλίωσης. Πρέπει να είναι ένα από τα: CV_THRESH_BINARY, CV_THRESH_BINARY_INV

blockSize

Ο αριθμός των γειτονικών pixel που λαμβάνονται υπόψη για την εύρεση του σωστού κατωφλίου.

param1

Παράμετρος που χρησιμοποιείται μόνο όταν το είδος του δυναμικού κατωφλίου είναι το CV_ADAPTIVE_THRESH_MEAN_C. Μπορεί να είναι και αρνητικό.

3.4 Εύρεση τετραγώνων – Ορθογωνίων

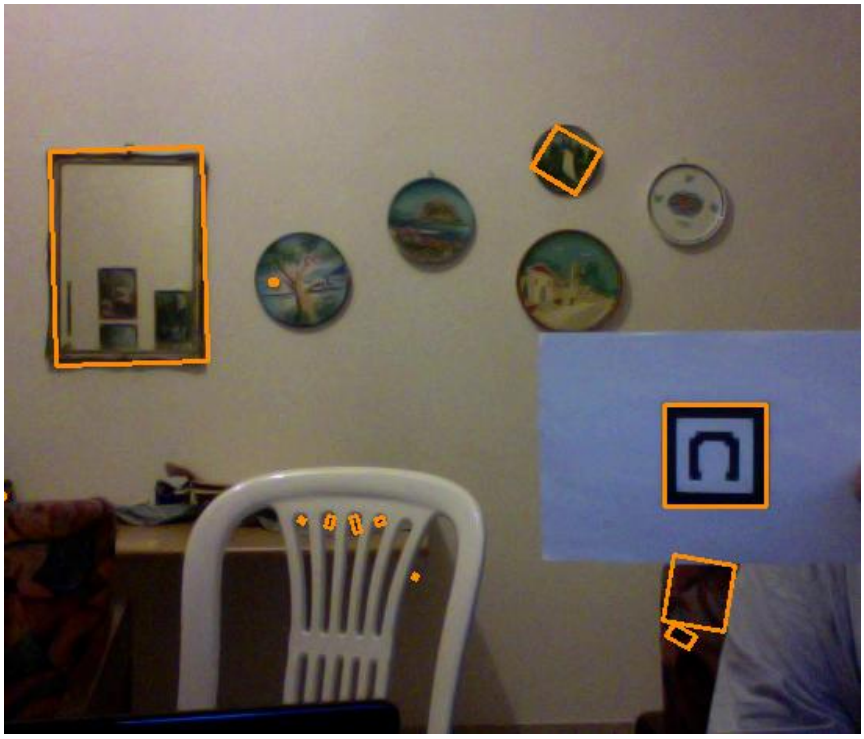
Στην προηγούμενη υποενότητα αναφέρθηκε πώς μπορεί μια έγχρωμη εικόνα , να μετατραπεί σε ασπρόμαυρη. Ο βασικότερος λόγος που χρειάζεται να γίνει αυτό είναι για να εξαλειφτεί μεγάλο μέρος άχρηστης πληροφορίας – λεπτομέρειες της εικόνας έτσι ώστε να τονιστούν οι βασικές πληροφορίες που χρειάζονται για την εύρεση του marker.

Ο marker που χρησιμοποιήθηκε είναι είδους template marker και έχει κάποια βασικά χαρακτηριστικά. Αρχικά είναι ασπρόμαυρος , έχει ένα παχύ μαύρο τετράγωνο περίγραμμα. Το εσωτερικό του περιγράμματος είναι λευκό και περιέχει ένα μαύρο μοναδικό σχέδιο (βλέπε εικόνα 3.1) . Η αναγνώριση βάσει αυτών των χαρακτηριστικών του marker γίνεται βηματικά. Στην υποενότητα αυτή θα αναλυθεί η διαδικασία με την οποία αρχικά αναγνωρίζεται ο marker σαν τετράγωνο και γενικότερα όλα τα τετράγωνα της εικόνας.

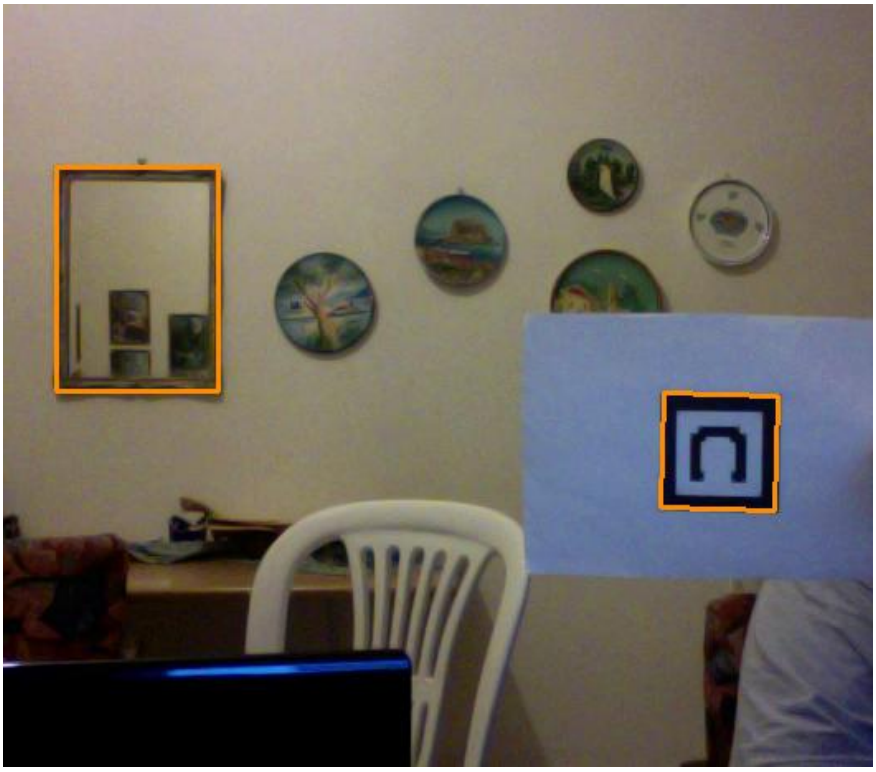
Για να ξεκινήσει η διαδικασία επεξήγησης της εύρεσης τετραγώνων , θεωρείται σκόπιμο να επεξηγηθεί ο ορισμός contour (περίγραμμα – σχήμα) όπου στην συνέχεια θα αναφέρεται με αυτό το όνομα. Contour στην γλώσσα του computer vision είναι μια λίστα από σημεία που απεικονίζουν με τον ένα ή τον άλλο τρόπο μια καμπύλη. Στο Emgu Cv τα contours αναπαριστώνται από συνέχειες σημείων όπου η κάθε μια από αυτές περιέχει πληροφορίες για το επόμενο σημείο στην καμπύλη. Σύμφωνα με τον παραπάνω ορισμό μπορούμε να συμπεράνουμε πως ένα τετράγωνο – ορθογώνιο είναι μια κλειστή συνέχεια σημείων με τέσσερις κορυφές , αντίστοιχα εάν θέλαμε να βρούμε ένα τρίγωνο θα αναζητούσαμε τρεις κορυφές κλπ.

Για την εύρεση των contours μέσα σε μια εικόνα χρησιμοποιείται η μέθοδος FindContours πάνω στην ασπρόμαυρη εικόνα, παραμετροποιημένη ούτος ώστε να αποθηκεύονται μονό τα εξωτερικά contours. Αυτό σημαίνει ότι εάν βρεθεί ένα contour να περιέχεται μέσα σε ένα άλλο ,τότε αυτό αγνοείται. Στη συνέχεια σε πρώτη φάση ελέγχεται αν το μέγεθος του κάθε contour είναι ικανοποιητικό και θα μπορούσε να είναι ο marker. Με τον τρόπο αυτό αποκλείονται contours πολύ μικρού ή πολύ μεγάλου μεγέθους, τα οποία μπορεί να οφείλονται σε θόρυβο ή αντικείμενα στον χώρο τα οποία μπορεί να έχουν παρόμοιο σχήμα με τον marker (βλέπε εικόνα 3.8) (Παράθυρα , κάδρα, πόρτες , κλπ).

Σε δεύτερη φάση ελέγχονται τα contours τα οποία έχουν βρεθεί , για το αν έχουν τέσσερις κορυφές , με τον τρόπο αυτό φιλτράρονται τα contours και αποθηκεύονται αυτά που πλησιάζουν το σχήμα ενός τετραγώνου. Στο σημείο αυτό αξίζει να σημειωθεί πως στην περίπτωση που είχαμε ένα τριγωνικού ή n- γωνικού σχήματος marker , θα ψάχναμε και τον αντίστοιχο αριθμό κορυφών. Ακόμα υπάρχει ένας έλεγχος που θα μπορούσε να πραγματοποιηθεί για την αυστηρή ανίχνευση ενός τετραγώνου, θα ελεγχόταν όλες του οι ακμές αν είναι ίσες καθώς και οι γωνίες μεταξύ τους να είναι ορθές. Όμως αυτό στην παρούσα εργασία δεν θα μπορούσε να λειτουργήσει αφού ο marker πολλές φορές ανάλογα με την οπτική γωνία παραμορφώνεται. Τέλος αποθηκεύονται σε μια λίστα, σαν ορθογώνια τα contours που βρέθηκαν και επαληθεύουν όλα τα παραπάνω κριτήρια(βλέπε εικόνα 3.9).



Εικόνα 3.8 : Παράδειγμα εύρεσης τετραγώνων – ορθογωνίων (χωρίς φιλτράρισμα).



Εικόνα 3.9 : Παράδειγμα εύρεσης τετραγώνων – ορθογωνίων (με φιλτράρισμα).

Επεξήγηση μεθόδου *FindContours( )* :

```
public Contour<Point> FindContours(
        CHAIN_APPROX_METHOD method,
        RETR_TYPE type,
        MemStorage stor
)
```

Όπου : method
Το είδος της προσέγγισης των contours.

type
Ο τρόπος με τον οποίο αποθηκεύονται τα contours.

stor
Λίστα αποθήκευσης των contours.

Return Value
Επιστρέφει contours εάν υπάρχουν , null εάν δεν βρεθούν.

3.5 Εύρεση ομογραφίας

Η ομογραφία είναι μια έννοια της μαθηματικής επιστήμης της γεωμετρίας. Ορίζεται ως η σχέση μεταξύ 2 εικόνων, τέτοια που κάθε δεδομένο σημείο στης μία εικόνα αντιστοιχεί σε ένα μοναδικό σημείο της άλλης και αντιστρόφως.

Στην όραση υπολογιστών (computer vision) ομογραφία ορίζεται σαν μια προοπτική προβολή από ένα επίπεδο σε ένα άλλο. Η προβολή σε μια δισδιάστατη επίπεδη επιφάνεια των σημείων της εικόνας που λαμβάνεται από την κάμερα είναι ένα παράδειγμα ομογραφίας. Είναι δυνατόν να εκφραστεί αυτή η προβολή με πολλαπλασιασμούς πινάκων, εάν χρησιμοποιήσουμε ομογενές σύστημα συντεταγμένων για να εκφράσουμε το σημείο Q της φωτογραφίας και το σημείο q στο επίπεδο , που προβλήθηκε το Q. Εάν ορίσουμε :

$$\tilde{Q} = [X\ Y\ Z\ 1]^T \qquad (3.1)$$

$$\tilde{q} = [x\ y\ z\ ]^T \qquad (3.2)$$

Τότε μπορούμε να εκφράσουμε την ομογραφία ως εξής :

$$\tilde{q} = sH\tilde{Q} \qquad (3.3)$$

Εδώ παρουσιάστηκε η παράμετρος s , η οποία είναι ένας αυθαίρετος παράγοντας κλίμακας.(προορίζεται για να καταστήσει σαφές ότι η ομογραφία ορίζεται μόνο ως προς αυτόν τον παράγοντα) . Είναι άμεσα συνδεδεμένο με το H και εξηγείται αμέσως παρακάτω.

Με λίγη γεωμετρία και μερική άλγεβρα πινάκων , μπορεί να λυθεί ο πίνακας μετασχηματισμού. Η πιο σημαντική παρατήρηση είναι ότι το H έχει δυο μέρη: Τον φυσικό μετασχηματισμό , ο οποίος ουσιαστικά εντοπίζει το επίπεδο του αντικειμένου (object plane) που βλέπουμε και την προβολή η οποία παρουσιάζει τον πίνακα των εσωτερικών παραμέτρων της κάμερας (βλέπε εικόνα 3.10).Όσον αφορά το κομμάτι του φυσικού μετασχηματισμού, είναι το άθροισμα των αποτελεσμάτων κάποιας περιστροφής R και της μετατόπισης t που σχετίζει το επίπεδο που βλέπουμε, στο επίπεδο της εικόνας. Επειδή όμως εργαζόμαστε σε ένα ομογενές σύστημα συντεταγμένων μπορούμε να τα συνδέσουμε σε ένα ενιαίο πίνακα ως εξής :

$$W = [R\ t] \quad (3.4)$$

Όπου $W = [R\ t]$ είναι ένας 3x4 πίνακας όπου οι τρείς πρώτες στήλες περιλαμβάνουν τις εννέα τιμές του R και η τελευταία τα τρία στοιχεία του t. Στην συνέχεια ο πίνακας M(πίνακας εσωτερικών παραμέτρων της κάμερας) πολλαπλασιάζεται με το $W\tilde{Q}$. Αυτό αποδίδεται ως :

$$\tilde{q} = sMW\tilde{Q} \quad \text{όπου} \quad M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \quad (3.5)$$

Φαίνεται σαν να έχουμε τελειώσει, αλλά στην πράξη το ενδιαφέρον μας δεν είναι στην συντεταγμένη $\tilde{Q}$ η οποία ορίζεται για όλο τον χώρο , αλλά για μια συντεταγμένη $\widetilde{Q'}$, η οποία ορίζεται μόνο στο επίπεδο το οποίο κοιτάμε. Αυτό μας επιτρέπει μια μικρή απλοποίηση.

Χωρίς να θέλουμε να γενικεύσουμε, μπορούμε να διαλέξουμε και να ορίσουμε το επίπεδο του αντικειμένου ούτος ώστε $Z = 0$. Το κάνουμε αυτό γιατί αν παράλληλα χωρίσουμε τον πίνακα περιστροφής (Rotation matrix) σε τρείς μονοδιάστατους πίνακες(ανά στήλη) $R = [r_1\ r_2\ r_3]$. Τότε ο ένας από τους πίνακες δεν χρειάζεται. Ποιο συγκεκριμένα :

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = sM[r_1\ r_2\ r_3\ t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = sM[r_1\ r_2\ t]\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.6)$$

Ο πίνακας ομογραφίας H ο οποίος απεικονίζει τα σημεία ενός επίπεδου αντικειμένου στην κάμερα, στην συνέχεια περιγράφεται από την σχέση $H = sM[r_1\ r_2\ t]$, όπου :

$$\tilde{q} = sH\tilde{Q}' \quad (3.7)$$

Παρατηρείται ότι ο H είναι τώρα ένας 3x4 πίνακας.

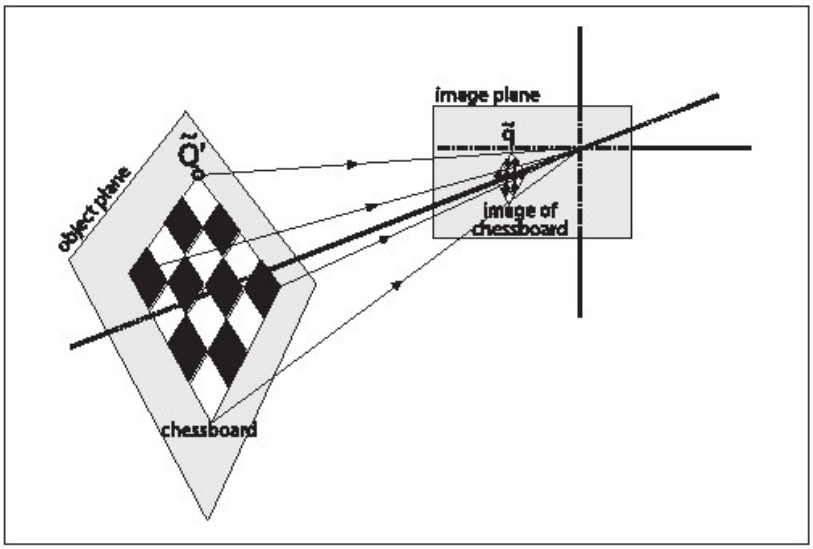
Το Emgu Cv χρησιμοποιεί τις προηγούμενες εξισώσεις για να υπολογίσει τον πίνακα ομογραφίας. Χρησιμοποιεί πολλαπλές εικόνες του ίδιου αντικειμένου για να υπολογίσει τους ξεχωριστούς πίνακες περιστροφής και μετατόπισης (Rotation και Translation) για κάθε οπτική γωνία καθώς και τις εσωτερικές παραμέτρους της κάμερας (οι οποίες είναι ίδιες για κάθε οπτική γωνία). Όπως είναι γνωστό, η περιστροφή ορίζεται από τρείς γωνίες και η θέση από τρείς μετατοπίσεις. Για τον λόγο αυτό έχουμε έξι αγνώστους για κάθε οπτική γωνία. Αυτό δεν είναι πρόβλημα επειδή ένα γνωστό επίπεδο αντικείμενο ,(όπως η σκακιέρα) μας δίνει οχτώ εξισώσεις – Αυτό γιατί η απεικόνιση ενός τετραγώνου μέσα σε ένα τετράπλευρο μπορεί να περιγραφεί από τέσσερα (x,y) σημεία.

Κάθε καινούργια εικόνα μας δίνει οχτώ εξισώσεις , με κόστος έξι νέους εξωτερικούς αγνώστους, οπότε αν δοθούν αρκετές εικόνες, είμαστε σε θέση να υπολογίσουμε οποιοδήποτε αριθμό εσωτερικών αγνώστων.
Ο πίνακας ομογραφίας H σχετίζει τις θέσεις των σημείων μιας εικόνας, με τα σημεία μιας εικόνας προορισμού με τις ακόλουθες απλές εξισώσεις :

$$P_{dst} = Hp_{src} \ , \ P_{src} = H^{-1}p_{dst}$$

$$P_{dst} = \begin{bmatrix} x_{dst} \\ y_{dst} \\ 1 \end{bmatrix}, \ P_{src} = \begin{bmatrix} x_{src} \\ y_{src} \\ 1 \end{bmatrix}$$

Αξίζει να σημειωθεί ότι μπορούμε να υπολογίσουμε το H χωρίς να γνωρίζουμε τίποτα για τις εσωτερικές παραμέτρους τις κάμερας. Αντιθέτως ο υπολογισμός πολλαπλών ομογραφιών από πολλαπλές λήψεις, είναι μια μέθοδος που χρησιμοποιεί το Emgu Cv για να βρει τις εσωτερικές παραμέτρους της κάμερας.
Στην παρούσα εργασία χρησιμοποιείται η συνάρτηση FindHomography( ) όπου δέχεται σαν είσοδο μια λίστα από εικονικά σημεία και άλλη μια λίστα από πραγματικά ενός τετραγώνου και μας επιστρέφει τον καλύτερο δυνατό πίνακα ομογραφίας. Χρειάζεται τουλάχιστον τέσσερα σημεία για να βρεθεί ο πίνακας ομογραφίας αλλά όσο περισσότερα σημεία χρησιμοποιούνται, τόσο καλλίτερο θα είναι το αποτέλεσμα. Ο λόγος για τον οποίο υπολογίζεται ο πίνακας ομογραφίας θα εξηγηθεί στην επόμενη υποενότητα.



Εικόνα 3.10 : Παράδειγμα απεικόνισης ομογραφίας – αντιστοίχηση σημείων εικόνας σε επίπεδο.

Επεξήγηση μεθόδου *FindHomography( )* :

```
public static HomographyMatrix FindHomography(
                    Matrix<float> srcPoints,
                    Matrix<float> dstPoints,
                    HOMOGRAPHY_METHOD method,
                    double ransacReprojThreshold
)
```

Όπου : srcPoints
Σημεία στην αρχική εικόνα.

dstPoints
Σημεία ενός επιπέδου , όπου αντιστοιχίζονται με τα srcPoints.

Method
Μέθοδος που χρησιμοποιείται για να βρεθεί η ομογραφία.

ransacReprojThreshold
Το μέγιστο επιτρεπόμενο όριο λάθους της μεθόδου RANSAC. Χρησιμοποιείται μόνο με την μέθοδο ομογραφίας RANSAC.

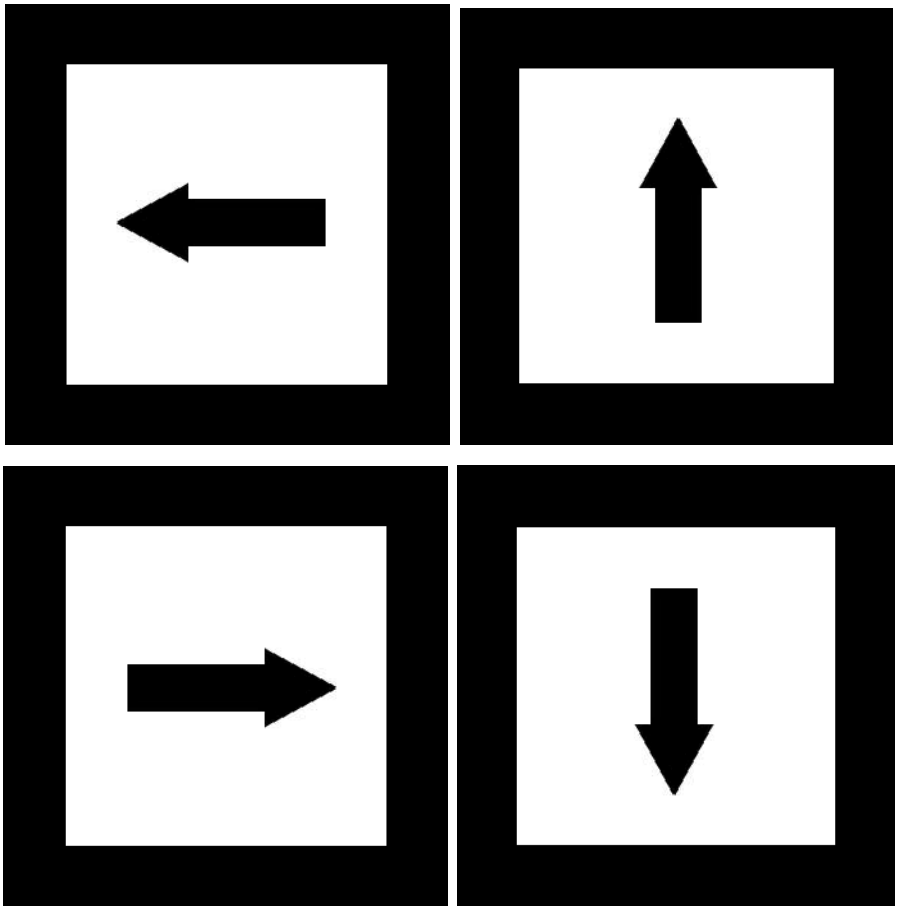
Return Value
Επιστρέφει ένα 3x3 πίνακα ομογραφίας εάν υπάρχει , null εάν δεν βρεθεί.

3.6 Προσαρμογή Προοπτικής (Warp Perspective)

Η αναγνώριση του marker πρέπει να είναι όσο το δυνατόν πιο ακριβής. Για το λόγο αυτό , τα τετράγωνα – πιθανοί markers που αναφέρθηκαν σε προηγούμενη υποενότητα πρέπει να μετασχηματιστούν κατάλληλα , ούτως ώστε να ελεγχθούν εκτενέστερα για το εάν τελικά είναι ο marker που αναζητάται ή όχι. Στην παρούσα φάση αναλύεται το πώς μετασχηματίζεται-προσαρμόζεται το κάθε τετράγωνο , για να είναι έτοιμο για περεταίρω έλεγχο.

Αρχικά βρίσκεται ο πίνακας ομογραφίας του κάθε τετραγώνου όπου αντιστοιχίζονται οι τέσσερεις γωνίες του με τέσσερα σημεία (που απεικονίζουν τις γωνίες ενός τετραγώνου) σε ένα άλλο επίπεδο (βλέπε υποενότητα 3.5). Έχοντας βρει αυτά εξασφαλίζουμε ότι η προβολή του κάθε τετραγώνου θα γίνεται σε ένα συγκεκριμένο επίπεδο με συγκεκριμένες διστάσεις και συγκεκριμένο προσανατολισμό. Αυτό σημαίνει πως κάθε τετράγωνο μπορεί να βρεθεί μόνο σε τέσσερεις πιθανούς προσανατολισμούς (θέσεις). Για παράδειγμα , έστω έχουμε ένα τετράγωνο που στο εσωτερικό του απεικονίζεται ένα βέλος. Η μύτη του δείχνει στο κέντρο μίας πλευράς του τετραγώνου. Το τετράγωνο αυτό μπορεί στην συνέχεια να βρεθεί σε τέσσερεις καταστάσεις (προσανατολισμούς) , δηλαδή , το βέλος μπορεί να ‘κοιτάζει ‘ μόνο πάνω, κάτω , δεξιά ή αριστερά καθώς περιστρέφεται το τετράγωνο.(βλέπε εικόνα 3.11). Εδώ αξίζει να σημειωθεί πως ο προσανατολισμός του τετραγώνου αλλάζει κάθε 90 + 1 μοίρες περιστροφής, δηλαδή αν το βέλος του παραδείγματος βλέπει προς τα πάνω και το περιστρέψουμε μία μοίρα δεξιά τότε η απεικόνισή του θα είναι να ‘κοιτάζει’ δεξιά.

Η εμφάνιση του τετραγώνου μετατοπισμένο και στο κατάλληλο μέγεθος σύμφωνα με τον πίνακα ομογραφίας γίνεται με την συνάρτηση WarpPerspective() του Emgu Cv.



Εικόνα 3.11 : Τέσσερεις καταστάσεις που μπορεί να βρεθεί το τετράγωνο μετά από την ομογραφία.

Εικόνα 3.12 : Παράδειγμα warp perspective.

Επεξήγηση μεθόδου WarpPerspective*( )* :

```
public Image<TColor, TDepth> WarpPerspective<TMapDepth>(
        Matrix<TMapDepth> mapMatrix,
        INTER interpolationType,
        WARP warpType,
        TColor backgroundColor
)
```

Όπου : mapMatrix
Ο πίνακας ομογραφίας.

interpolationType
Τύπος αναπροσαρμογής της εικόνας.

warpType
Τύπος προσαρμογής προοπτικής.

backgroundColor
Χρώμα που χρησιμοποιείται για να γεμίσει κενά σημεία.

3.7 Σύγκριση τετραγώνων με template

Στις προηγούμενες υποενότητες αναφέρθηκαν αναλυτικά όλες οι διαδικασίες που πρέπει να ακολουθηθούν, ούτως ώστε στο σημείο αυτό να είναι δυνατόν, με μια τελευταία απλή σύγκριση – αφαίρεση να αναγνωριστεί εάν τα τετράγωνα που βρέθηκαν τελικά , είναι ο marker που αναζητείται ή όχι.

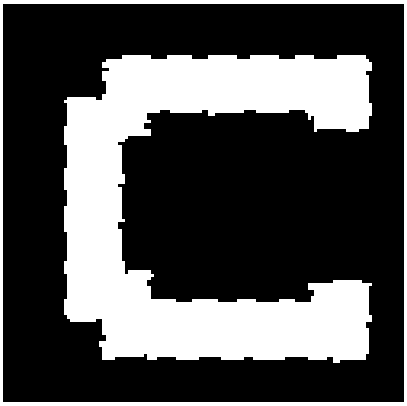
Συνοπτικά , γίνεται λήψη μιας εικόνας μέσω της κάμερας και μετατρέπεται σε ασπρόμαυρη. Στην συνέχεια γίνεται ανίχνευση contours και έπειτα από μια διαδικασία , που αναλύθηκε παραπάνω, αναγνωρίζονται όλα τα τετράγωνα. Μέσω ενός ακόμα φιλτραρίσματος απορρίπτονται τα ποιο ακατάλληλα. Τελικά καταλήγουμε σε μια διαδικασία καθοριστική για το τελικό στάδιο αναγνώρισης του marker. Αυτή είναι η εύρεση της ομογραφίας κάθε τετραγώνου και η προσαρμογή του τετραγώνου αυτού σ ε μια συγκεκριμένη προοπτική και ένα συγκεκριμένο μέγεθος.

Μέχρι στιγμής υπάρχει η εικόνα ενός τετραγώνου προκαθορισμένου μεγέθους και με όσο το δυνατόν πιο σταθερή προοπτική. Από την άλλη μεριά υπάρχουν εικόνες – templates, που απεικονίζουν τον – τους marker-s σαν πρότυπα με συγκεκριμένο μέγεθος και προοπτική και στις τέσσερις καταστάσεις (βλέπε εικόνα 3.11) πού μπορεί να βρεθεί. Βάσει του μεγέθους τους και της προοπτικής τους έγιναν και οι παραπάνω διαδικασίες στα τετράγωνα που ανιχνεύθηκαν, έτσι ώστε να έχουν κοινά χαρακτηριστικά, προκειμένου να γίνει ακριβέστερη η σύγκριση τους. Έχοντας λοιπόν τα πρότυπα εικόνας – marker (templates) και τετράγωνα ίδιου μεγέθους (πιθανοί markers) δεν είναι δύσκολο να γίνει απλά μια αφαίρεση των δύο εικόνων. Η αφαίρεση γίνεται με την συνάρτηση AbsDiff( ) του Emgu Cv. Στην συνέχεια καταμετρώνται τα διαφορετικά εικονοστοιχεία (pixels) μεταξύ τους. Λόγω θορύβου αυτά ποτέ δεν θα μπορούσαν να είναι 0, όμως μετά από πολλές εργαστηριακές δοκιμές βρέθηκε η μέγιστη επιτρεπόμενη απόκλιση , ούτως ώστε να αναδειχτεί ένα τετράγωνο σαν marker. Αξίζει να σημειωθεί πως η απόκλιση, εξαρτάται από πολλούς παράγοντες, όπως το μέγεθος του τετραγώνου ή το μέγεθος του σχεδίου που περιέχεται στο τετράγωνο. Ο αριθμός των εικονοστοιχείων (pixels) εξαρτάται από το μέγεθος της εικόνας , δηλαδή όσο μεγαλύτερη είναι η εικόνα τόσο περισσότερα εικονοστοιχεία θα έχει. Άρα αυτό ισούται με μεγαλύτερη απόκλιση. Για τον ίδιο ακριβώς λόγο, όσο μεγαλύτερο το σχέδιο μέσα στο τετράγωνο τόσο μεγαλύτερη και η απόκλιση. Οπότε κάθε marker που χρησιμοποιείται για συγκεκριμένο σκοπό, θα πρέπει να έχει το ίδιο μέγεθος τετραγώνου, αλλά και το σχέδιο στο εσωτερικό του να είναι ταυτόχρονα μοναδικό και να έχει παρεμφερή μέγεθος.

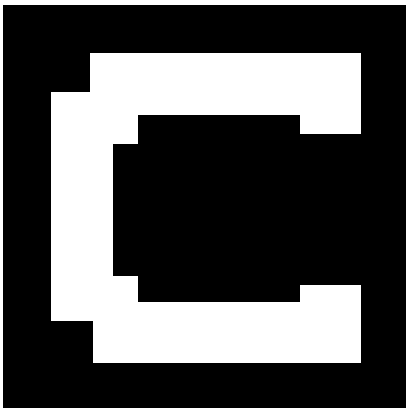
Στο σημείο αυτό μπορεί κάποιος να αναρωτάτε, γιατί πολύ απλά από την στιγμή που βρέθηκαν τα τετράγωνα, δεν τέθηκαν σαν σημεία ενδιαφέροντος και να γίνει απευθείας η αφαίρεση των εικόνων. Η προφανής απάντηση είναι ότι, αυτό δεν θα μπορούσε να γίνει σε καμία περίπτωση γιατί θα έπρεπε να υπάρχει αποθηκευμένη η εικόνα πρότυπο του marker σε όλες τις πιθανές προοπτικές καθώς και σε όλες τις θέσεις περιστροφής (πράγμα αδύνατον). Ακόμα και αν συνέβαινε αυτό, εκτός του ότι θα καθυστερούσε υπερβολικά τον αλγόριθμο ανίχνευσης του marker, ο θόρυβος θα ήταν υπερβολικά πολύς , οπότε και τα αποτελέσματα θα ήταν αμφιλεγόμενα – ανακριβή.

Τελικά αφού υπολογιστεί ο αριθμός των διαφορετικών pixel, αν αυτά είναι κάτω από το επιτρεπόμενο όριο, τότε το τετράγωνο αυτό σαν ένας marker με προσανατολισμό ανάλογο με ποιο από τα τέσσερα templates του συγκεκριμένου marker έχει ταυτιστεί. Η κρισιμότητα του προσανατολισμού θα επεξηγηθεί στη επόμενη ενότητα.

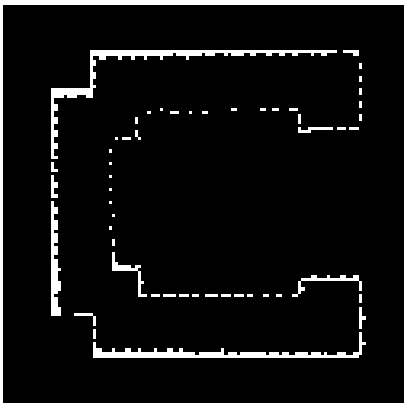
Εικόνα 3.13 : Εικόνα με μαρκαρισμένο το τετράγωνο που βρέθηκε στον χώρο.



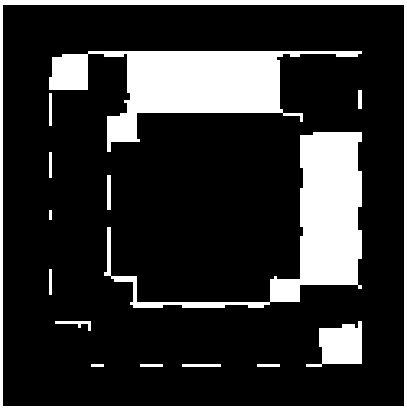
Εικόνα 3.14 : Εικόνα με ομογραφία και warp perspective του παραπάνω τετραγώνου.



Εικόνα 3.15 : Εικόνα ενός template.



Εικόνα 3.16 : Εικόνα με το αποτέλεσμα της αφαίρεσης των δυο παραπάνω εικόνων. Τα λευκά pixel είναι η διαφορά μεταξύ τους. Είναι μια πετυχημένη ανίχνευση.



Εικόνα 3.17 : Εικόνα με το αποτέλεσμα της αφαίρεσης του παραπάνω template με μια άλλη εικόνα ομογραφίας. Η διαφορά τους είναι πολύ μεγάλη οπότε η ανίχνευση κρίνεται ανεπιτυχής.

4. ΑΝΙΧΝΕΥΣΗ ΧΑΡΑΚΤΗΡΙΣΤΙΚΩΝ ΣΗΜΕΙΩΝ

4.1 Γενικά

Η ανίχνευση γωνιών ή γενικότερα η ανίχνευση σημείων ενδιαφέροντος είναι μια προσέγγιση που χρησιμοποιείται στα συστήματα όρασης υπολογιστών για να μπορούμε να εξαγάγουμε ορισμένα είδη χαρακτηριστικών γνωρισμάτων έτσι ώστε να βγάλουμε χρήσιμα συμπεράσματα από μια εικόνα. Τα σημεία γωνιών είναι σημεία της εικόνας, τα οποία συνήθως αντιστοιχούν σε γωνίες αντικειμένων. Η ιδιαιτερότητα αυτών των σημείων είναι ότι μπορούν να αντιστοιχηθούν με μοναδικό τρόπο σε μια ακολουθία εικόνων.

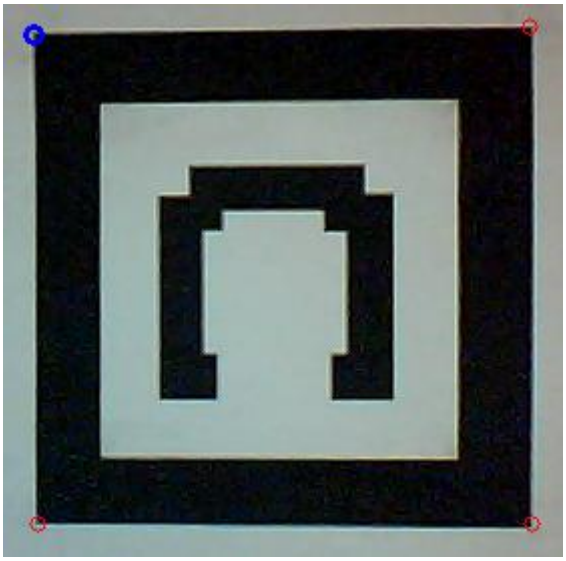
Οι αλγόριθμοι ανίχνευσης σημείων μπορούν να ταξινομηθούν σε δύο κατηγορίες. Η πρώτη κατηγορία αφορά αλγορίθμους, οι οποίοι αρχικά εξάγουν ακμές και στη συνέχεια εντοπίζουν σημεία πάνω στις ακμές τα οποία έχουν μέγιστη καμπυλότητα, ενώ οι αλγόριθμοι της δεύτερης κατηγορίας αναζητούν σημεία όπου οι ακμές τέμνονται. Η ανίχνευση σημείων χρησιμοποιείται συχνά στην ανίχνευση κίνησης, στην παρακολούθηση τροχιάς κινητού μέσου και στην αναγνώριση αντικειμένου.

4.2 Εξαγωγή χαρακτηριστικών σημείων και προσανατολισμός από αναγνωρισμένο marker

Όπως αναφέρθηκε στην προηγούμενη ενότητα, για να βρεθεί ένας marker πρέπει πρώτα να αναγνωριστεί το εξωτερικό του τετράγωνο. Αφού λοιπόν επιβεβαιωθεί σαν marker, πολύ απλά μπορούν να χρησιμοποιηθούν οι τέσσερις γωνίες του τετραγώνου, σαν χαρακτηριστικά σημεία και να χρησιμοποιηθούν στην συνέχεια. Καθοριστικό όμως ρόλο παίζει και ο προσανατολισμός του marker. Έχοντας βρει τέσσερα σημεία, δεν μπορούμε να αντιληφθούμε εάν ο marker είναι σε ορθή ή άλλη θέση. Όπως έχει προαναφερθεί, για την αναγνώριση του marker, υπάρχουν τέσσερα διαφορετικά templates σε τέσσερις διαφορετικές περιστραμένες θέσεις (βλέπε εικόνα 3.11).

Έχοντας λοιπόν βρει με πιο από τα τέσσερα templates έχει ταυτοποιηθεί σαν marker, αντιστοιχίζοντας και τις τέσσερις γωνίες του τετραγώνου με ανάλογη σειρά. Για παράδειγμα αν το template με το βέλος έχει σαν αρχική θέση, τη θέση όπου το βέλος κοιτάζει πάνω και ταυτιστεί με ένα τετράγωνο (πιθανό marker), τότε οι γωνίες που θα εξαχθούν, θα έχουν σωστή σειρά, ξεκινώντας με τις δυο πάνω από τα αριστερά και στην συνέχεια τις δυο κάτω πάλι από αριστερά. Αντίθετα αν ταυτιστεί με το template οπού το βέλος κοιτάζει δεξιά, θα πρέπει να μεταφερθούν οι γωνίες στην σωστή σειρά, δηλαδή η πρώτη γωνία να είναι η πάνω δεξιά, η δεύτερη κάτω δεξιά και ούτω κάθε εξής.

Όλη αυτή η διαδικασία γίνεται με απώτερο σκοπό, την εκτίμηση της θέσης της κάμερας (pose estimation) σε σχέση με τον marker, όπου είναι και ο σκοπός της εργασίας. Ουσιαστικά όμως η εκτίμηση γίνεται βάσει των σημείων που προαναφέρθηκαν και στην υπόλοιπη διαδικασία θα γίνει προσπάθεια ακολούθησης των σημείων αυτών.



Εικόνα 4.1 : Εικόνα που έχουν ανιχνευθεί οι τέσσερις γωνίες του τετραγώνου.

4.3 Αλγόριθμοι ανίχνευσης σημείων

4.3.1 Αλγόριθμος Harris για ανίχνευση σημείων

Ο ανιχνευτής σημείων ή ανιχνευτής γωνιών (Corner Detector) που επινοήθηκε από τους Harris και Stephens το 1988 χρησιμοποιείται ευρύτατα σε εφαρμογές που περιλαμβάνουν κίνηση. Ο ανιχνευτής Harris είναι ιδιαίτερα δημοφιλής λόγω της αποτελεσματικότητας του σε εφαρμογές όπου οι εικόνες περιέχουν θόρυβο και οι συνθήκες φωτισμού αλλάζουν. Ο ανιχνευτής γωνιών Harris είναι βασισμένος στην τοπική συνάρτηση αυτοσυσχέτισης (local autocorrelation function) ενός σήματος, η οποία μετρά τις τοπικές αλλαγές του σήματος. Ειδικότερα ο αλγόριθμος εξετάζει ελάχιστες και μέγιστες τιμές των ιδιοτιμών α και β του πίνακα αυτοσυσχέτισης της εικόνας. Ο πίνακας αυτοσυσχέτισης είναι:

$$A = \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} \qquad (4.1)$$

όπου I_x και I_y είναι οι παράγωγοι της εικόνας στις κατευθύνσεις x και y αντίστοιχα. Με την ανάλυση των ιδιοτιμών του πίνακα A μπορούμε να καταλήξουμε στα εξής συμπεράσματα:

i. Εάν και οι δύο ιδιοτιμές έχουν μικρή τιμή (σχεδόν μηδενική)τότε δεν υπάρχει κανένα χαρακτηριστικό ενδιαφέροντος στο συγκεκριμένο Pixel (x,y).

ii. Εάν η μια ιδιοτιμή είναι μικρή (σχεδόν μηδενική) και η άλλη έχει μεγάλη θετική τιμή τότε μια ακμή εντοπίζεται.

iii. Εάν και οι δύο ιδιοτιμές έχουν ευδιάκριτα μεγάλες θετικές τιμές, τότε μια γωνία εντοπίζεται.

Ο Harris και Stephens παρατήρησαν ότι ο ακριβής υπολογισμός των ιδιοτιμών είναι υπολογιστικά ασύμφορος, δεδομένου ότι απαιτείται ο υπολογισμός μια τετραγωνικής ρίζας. Για να αποφύγουν το υπολογιστικό κόστος προτείνουν τη χρήση ενός μεγέθους που περιλαμβάνει την ορίζουσα και το ίχνος της μήτρας αυτοσυσχέτισης

$$R = det(A) - k\big(tr(A)\big)^2 = \alpha\beta + k(\alpha + \beta)^2 \qquad (4.2)$$

όπου α και β είναι οι ιδιοτιμές της μήτρας A (Σχέση 2.1), όπου το *k* είναι μια παράμετρος ευαισθησίας που μπορεί να καθοριστεί εμπειρικά, αν κι έχει παρατηρηθεί ότι η τιμή του κυμαίνεται από 0.04 έως 0.15. Να σημειωθεί ότι αν κι οι δύο ιδιοτιμές είναι μεγάλες το αποτέλεσμα είναι το μέγεθος R να είναι πολύ μεγάλο. Για τον παραπάνω λόγο έχει παρατηρηθεί ότι ο αλγόριθμος ανίχνευσης σημείων Harris είναι πολύ ευαίσθητος στις μεταβολές της αντίθεσης της εικόνας, αλλά ταυτόχρονα κάνει πολύ δύσκολη την επιλογή των κατωφλίων.

Για να αποφύγουμε την χρήση της παραμέτρου *k* στην εξίσωση του R, ο προσδιορισμός της τιμής του οποίου πολλές φορές δημιουργεί προβλήματα, ο Noble (1989) παρουσίασε ένα εναλλακτικό μέγεθος

$$R = \frac{det(A)}{tr(A)} \qquad (4.3)$$

<u>Στάδια υλοποίησης του αλγορίθμου ανίχνευσης Harris:</u>

i. Αρχικά υπολογίζεται η πρώτη παράγωγος κατά τη διεύθυνση x και y με χρήση φίλτρων 3x3 (Prewitt, Sobel κ.ο.κ).

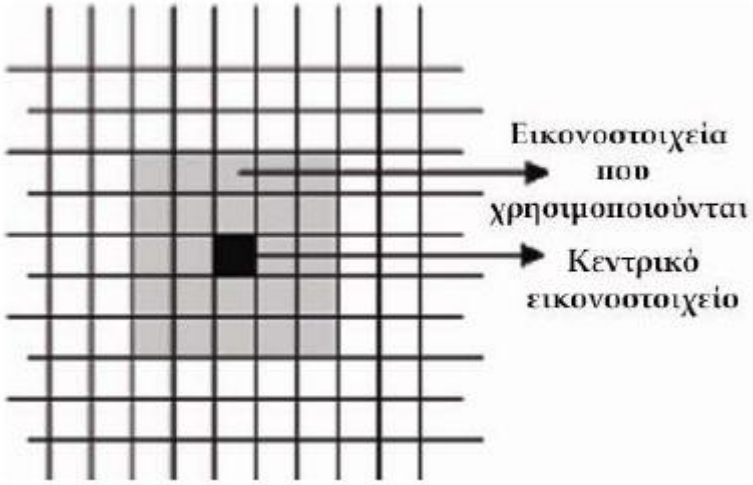
ii. Στη συνέχεια υπολογίζονται τα $I_x{}^2$, $I_y{}^2$, $I_x I_y$.

iii. Έπειτα, με χρήση ενός Gaussian φίλτρου εξομάλυνσης, υπολογίζονται οι δειγματοληπτικοί μέσοι $\langle I_x{}^2 \rangle$, $\langle I_y{}^2 \rangle$, $\langle I_x I_y \rangle$.

iv. Στο τέλος υπολογίζεται η γωνιακή τιμή από τη σχέση:

$$Cornerness\ Value\ (C) = \frac{((I_x)^2)+((I_y)^2)}{((I_x)^2)((I_y)^2)-((I_x I_y)^2)} \qquad (4.4)$$

Να επισημάνουμε ότι ένα pixel θεωρείται ως σημείο στη περίπτωση που η γωνιακή του τιμή βρίσκεται κάτω από ένα συγκεκριμένο κατώφλι (threshold). Καλό σημείο ορίζεται ως το Pixel που έχει μικρή γωνιακή τιμή, ενώ ως βέλτιστο το σημείο με την ελάχιστη γωνιακή τιμή. Ο υπολογισμός της γωνιακής τιμής για κάθε pixel προσδιορίζεται μετά την επιλογή μιας περιοχής γύρω από αυτό.



Εικόνα 4.2: Περιοχή εικονοστοιχείων που χρησιμοποιούνται για τον υπολογισμό του cornerless value.

4.3.2 Αλγόριθμος Kitchen-Rosenfeld για Ανίχνευση Σημείων

Ο αλγόριθμος Kitchen-Rosenfeld αποτελεί την πρώτη υλοποίηση αλγορίθμου εξαγωγής ακμών, ο οποίος αποτελεί σημείο αναφοράς για όλους τους μελλοντικούς αλγορίθμους εξαγωγής ακμών. Σε αυτή τη μέθοδο η γωνιακή τιμή υπολογίζεται από την παρακάτω σχέση:

$$Cornerness\ Value\ (C) = \frac{I_{XX}(I_y)^2+I_{yy}(I_x)^2-2I_{xy}I_xI_y}{(I_x)^2+(I_y)^2} \qquad (4.5)$$

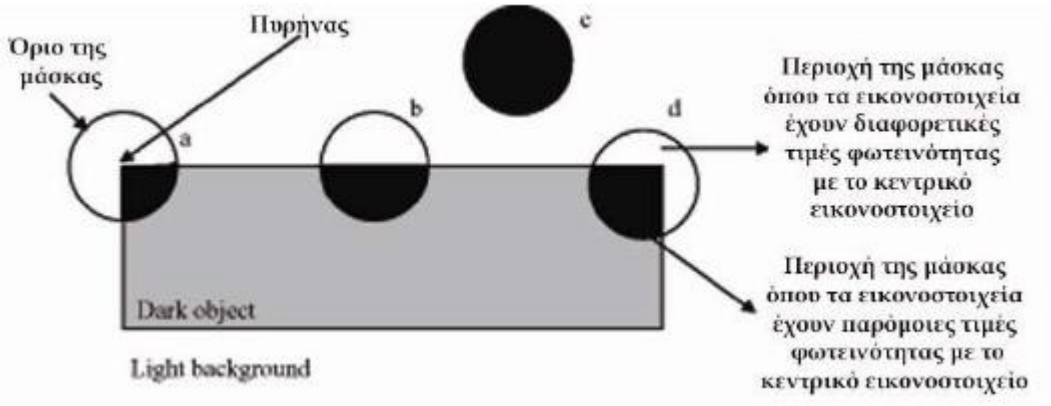
όπου I είναι η διαβάθμιση του γκρι, I_x είναι η πρώτη παράγωγος της εικόνας κατά τη διεύθυνση x, I_{XX} είναι η δεύτερη παράγωγος κατά τη διεύθυνση του x, I_y είναι η πρώτη παράγωγος κατά τη διεύθυνση του y, I_{yy} είναι η δεύτερη παράγωγος κατά τη διεύθυνση του y.

Ο αλγόριθμος λειτουργεί ως εξής:

Αν η τιμή του C είναι μέσα στα όρια που θέτουμε, τότε ο συγκεκριμένος τόπος στην εικόνα θεωρείται ότι αποτελεί σημείο. Έχει διαπιστωθεί επίσης ότι όσο μικρότερη είναι η τιμή του C, τόσο ισχυρότερη είναι η πιθανότητα ύπαρξης σημείου.

4.3.3 Αλγόριθμος SUSAN για Ανίχνευση Σημείων

Ο ανιχνευτής σημείων SUSAN (Smallest Univalue Segment Assimilation Nucleus) πέρα από το pixel που εξετάζεται χρησιμοποιεί μια κυκλική μάσκα, αποφεύγοντας έτσι τη χρήση παραγώγων. Για αυτό τον λόγο ο αλγόριθμος που αναπτύχθηκε από τον Smith δεν απαιτεί τη χρήση φίλτρων εξομάλυνσης. Κάθε pixel θεωρείται ως το κέντρο της κυκλικής μάσκας (Εικόνα 4.2) και στην συνέχεια συγκρίνονται όλα τα pixels που βρίσκονται μέσα στην κυκλική μάσκα με το κεντρικό pixel. Όλα τα pixels με παρόμοιες τιμές φωτεινότητας θεωρείται ότι ανήκουν στην ίδια δομική μονάδα στην εικόνα. Βρίσκοντας στο τέλος το τοπικό ελάχιστο σε μια τέτοια περιοχή εντοπίζεται το ακριβές pixel που αποτελεί σημείο.



Εικόνα 4.3 : Αναπαράσταση κυκλικής μάσκας του αλγορίθμου SUSAN

Κάθε pixel συγκρίνεται με τον πυρήνα χρησιμοποιώντας τη σχέση:

$$C(r, r_0) = 100e - \left(\frac{I(r)-I(r_0)}{t}\right)^6 \qquad (4.6)$$

Όπου r_0 είναι η θέση του κεντρικού pixel, r είναι η θέση όλων των άλλων εικονοστοιχείων της μάσκας, $I(r)$ είναι η τιμή φωτεινότητας του κάθε pixel και t μία παράμετρος που ονομάζεται κατώφλι της απόλυτης διαφοράς τιμών φωτεινότητας (brightness difference threshold).

4.3.4 Αλγόριθμος Kanade-Lucas-Tomasi (KLT) για Ανίχνευση Σημείων

Ο ανιχνευτής γωνιών KLT είναι έντονα βασισμένος στον ανιχνευτή γωνιών Harris. Οι συντάκτες του αλγορίθμου αποδεικνύουν ότι το μέγεθος $R' = min(\alpha, \beta)$ (α και β ορίστηκαν παραπάνω) είναι ένα καλύτερο μέτρο της δύναμης γωνίας από το μέγεθος R. Ο αλγόριθμος αυτός λειτουργεί συγκρίνοντας ένα μικρό κομμάτι εικόνας από δύο συνεχόμενα frames μιας ακολουθίας εικόνων, υποθέτοντας ότι οι εικόνες που έχουν μικρή διαφορά λήψης είναι σχετιζόμενες σε μεγάλο βαθμό. Ο αλγόριθμος αυτός χρησιμοποιείται σε εφαρμογές optical motion flow. Ένα σημαντικό πρόβλημα στον εντοπισμό της μετατόπισης d ενός σημείου από ένα καρέ στο επόμενο είναι ότι ένα μεμονωμένο pixel δεν μπορεί να αποτυπωθεί αξιόπιστα ως ίχνος, έκτος εάν έχει ένα πολύ διακριτό χαρακτηριστικό σε σχέση με τα γειτονικά του. Αυτό συμβαίνει εξαιτίας της εικόνας που προκύπτει λόγω αφαίρεσης του θορύβου με τη μέθοδο της λείανσης.

Εξαιτίας αυτών των προβλημάτων, η μέθοδος KLT δεν εντοπίζει ένα μεμονωμένο pixel αλλά παράθυρα με pixels τα οποία εξετάζονται ώστε να περιέχουν επαρκή πληροφορία. Η χρήση παραθύρων μικρού μεγέθους θεωρείται σημαντική, γιατί μέσα σε μια μικρή περιοχή μπορεί να παρατηρηθεί μονό μικρή ποσότητα αλλαγών. Κάθε ανακολουθία ανάμεσα σε διαδοχικά παράθυρα που δεν μπορεί να εξηγηθεί με translation θεωρείται λάθος και το διάνυσμα μετατόπισης επιλέγεται για να ελαχιστοποιήσει το υπόλοιπο λάθος.

4.3.5 Αλγόριθμος Moravec για Ανίχνευση Σημείων

Ο αλγόριθμος Moravec, είναι από τους παλαιότερους αλγορίθμους ανίχνευσης γωνίας, ο οποίος προσδιορίζει μια γωνία με σκοπό να είναι ένα σημείο με χαμηλή ομοιότητα. Η μέθοδος αυτή εξετάζει κάθε pixel στην εικόνα για διαπιστωθεί αν μια γωνία είναι παρούσα, στην συνέχεια μελετώντας κατά πόσο παρόμοιο είναι ένα κεντραρισμένο patch(περιοχή εικονοστοιχείων) ενός συγκεκριμένου pixel με τα γειτονικά του (κατά μεγάλο μέρος επικαλυπτόμενα patches). Η ομοιότητα μετράται υπολογίζοντας το άθροισμα των τετραγωνικών διαφορών μεταξύ δύο patches (SSD- sum of squared differences). Όσο μικρότερος είναι ο αριθμός που προκύπτει τόσο μεγαλύτερη είναι η ομοιότητα. Επομένως αντιλαμβανόμαστε ότι εάν το pixel είναι σε μια περιοχή υψηλής έντασης τότε όλα τα γειτονικά pixel θα είναι παρόμοια.

<u>Η γενικότερη λογική του αλγορίθμου είναι η εξής:</u>

i. Εάν το pixel είναι σε μια ακμή τότε τα κοντινά patches σε διεύθυνση κάθετη στην ακμή θα φαίνονται εντελώς διαφορετικά, αλλά τα κοντινά patches σε διεύθυνση παράλληλη στην ακμή θα παρουσιάζουν μια πολύ μικρή μεταβολή.

ii. Εάν το pixel βρίσκεται σε περιοχή με ποικιλομορφία σε όλες τις διευθύνσεις, τότε κανένα από τα γειτονικά patches δεν θα είναι παρόμοιο.

iii. Η αντοχή της γωνίας καθορίζεται ως το μικρότερο SSD ανάμεσα στο pixel και στα γειτονικά του (οριζόντια, κατακόρυφα και στις δύο διαγώνιους).

iv. Τελικά αν το μέγεθος αυτό είναι το τοπικό μέγιστο τότε το χαρακτηριστικό του ενδιαφέροντος έχει προσδιοριστεί.

Όπως έχει επισημανθεί από τον Moravec, ένα από τα κύρια προβλήματα αυτής της μεθόδου είναι ότι δεν είναι ισοτροπική. Δηλαδή αν υπάρχει μια ακμή που δεν είναι στην διεύθυνση των γειτονικών, τότε δεν θα εντοπιστεί σαν σημείο ενδιαφέροντος.

4.3.6 Αλγόριθμος Trajkovic –Hedley για Ανίχνευση Ακμών

Η μέθοδος Trajkovic-Hedley λειτουργεί με παρόμοιο τρόπο όπως ο αλγόριθμος SUSAN. Εξετάζει, δηλαδή, απ' ευθείας εάν το patch κάτω από ένα pixel είναι παρόμοιο (self-similar) μελετώντας τα γειτονικά του pixels. Το $\vec{c}$ είναι το pixel που μας ενδιαφέρει και $\vec{p} \in P$ είναι το σημείο ενός κύκλου με ακτίνα P με επίκεντρο το $\vec{c}$. Ενώ το σημείο $\vec{p}'$ είναι το σημείο απέναντι του $\vec{p}$ πάνω στη διάμετρο.
Η συνάρτηση απόκρισης είναι:

$$r(\vec{c}) = min_{\vec{p} \in P}\left(I(\vec{p}) - I(\vec{c})\right)^2 + \left(I(\vec{p}') - I(\vec{c})\right)^2 \qquad (4.7)$$

Το παραπάνω μέγεθος θα είναι μεγάλο όταν δεν υπάρχει διεύθυνση στην οποία το κεντρικό pixel είναι παρόμοιο με δύο γειτονικά pixels κατά μήκος της διαμέτρου. Το P είναι ένας διακεκριμένος κύκλος. Για να δοθεί μια πιο ισοτροπική απόκριση στις ενδιάμεσες διαμέτρους χρησιμοποιείται η παρεμβολή. Εφόσον κάθε υπολογισμός δίνει ένα άνω όριο στο όριο, εξετάζονται οι οριζόντιες και οι κατακόρυφες διευθύνσεις ώστε να αποφασίσουμε αν αξίζει τελικά να προχωρήσουμε στον πλήρη υπολογισμό του *c*.

4.3.7 Αλγόριθμος Wang-Brady για Ανίχνευση Σημείων

Ο αλγόριθμος Wang-Brady θεωρεί την εικόνα ως μια επιφάνεια και ψάχνει σε αυτή τις θέσεις όπου παρουσιάζεται μεγάλη κυρτότητα κατά μήκος της ακμής. Δηλαδή, ο αλγόριθμος ψάχνει τις θέσεις όπου η ακμή αλλάζει την κατεύθυνση γρήγορα. Το αποτέλεσμα γωνιών δίνεται από τη σχέση:

$$C = \nabla^2 I - c|\nabla I|^2 \qquad (4.8)$$

όπου η παράμετρος C δείχνει πόσο edge-phobic είναι ο ανιχνευτής. Να σημειωθεί ότι είναι απαραίτητη η χρήση φίλτρου λείανσης για τη μείωση του θορύβου.

4.4 Συμπεράσματα

Στην ενότητα αυτή μελετήθηκε το πρόβλημα της εξαγωγής χαρακτηριστικών από μια εικόνα, το οποίο αποτελεί καίριο βήμα για την υλοποίηση του συστήματος μας. Έγινε μια παρουσίαση διάφορων αλγορίθμων ανίχνευσης σημείων καθώς και το θεωρητικό υπόβαθρο στο οποίο στηρίζονται αυτοί. Όλες οι μέθοδοι αυτοί κρίνονται κατάλληλοι για εφαρμογές σχετικές με κίνηση , όμως δείχνει να υπερισχύει ο αλγόριθμος του Harris, ο οποίος είναι και ο ποιο διαδεδομένος. Κατά την διάρκεια της υλοποίησης της εργασίας αυτής δοκιμάστηκαν δυο διαφορετικές μέθοδοι του Emgu Cv όπου χρησιμοποιούσαν και οι δυο των αλγόριθμο Harris. Αυτές είναι η μέθοδος GoodFeaturesToTrack( ) και η μέθοδος FindChessboardCorners( ). Η πρώτη βρίσκει χαρακτηριστικά σημεία-γωνίες μέσα σε μια εικόνα, βρίσκοντας μεγάλες ιδιοτιμές και μπορεί να χρησιμοποιηθεί σε συνδυασμό με τον αλγόριθμο Harris. Ουσιαστικά είναι μια μετεξέλιξη του. Η μέθοδος FindChessboardCorners( ) αρχικά χρησιμοποιείται για την βαθμονόμηση της κάμερας. Ανιχνεύει τις εσωτερικές γωνίες μιας σκακιέρας με εξαιρετική ακρίβεια και ταχύτητα. Λειτουργεί με μια ακόμη υλοποίηση του Harris. Αναμφισβήτητα είναι η καλύτερη μέθοδος για εύρεση σημείων που δοκιμάστηκε στην εργασία αυτή και αν δεν υπήρχε η αναγκαιότητα αναγνώρισης πολλαπλών markers , η σκακιέρα θα ήταν τι μοναδικό marker που θα είχε χρησιμοποιηθεί. Τα πλεονεκτήματα της FindChessboardCorners( ) σε συνδυασμό με την σκακιέρα δεν περιορίζονται μόνο στην γρήγορη ανίχνευση γωνιών, αλλά όπως θα δούμε σε επόμενη ενότητα, τα σημεία της είναι πολύ καλύτερα για track (χάνονται πολύ δύσκολα από μεθόδους OpticalFlow) και εκτός αυτού μέσω αυτόν γίνεται με μεγαλύτερη ακρίβεια η εκτίμηση θέσης της κάμερας. Για τον λόγο αυτό κρίνεται σκόπιμο στο σημείο αυτό, να επεξηγηθεί η μέθοδος αυτή του Emgu Cv.

Επεξήγηση μεθόδου *FindChessboardCorners( )* :

```
public static bool FindChessboardCorners(
        Image<Gray, byte> image,
        Size patternSize,
        CALIB_CB_TYPE flags,
        out PointF[] corners
)
```

Όπου : image

Εικόνα – στιγμιότυπο .

patternSize

Αριθμός εσωτερικών γωνιών σκακιέρας προς ανίχνευση.

flags

Τύπος επεξεργασίας εικόνας.

corners

Πίνακας επιστροφής σημείων που βρέθηκαν.

Return Value

True εάν βρεθεί σκακιέρα , false εάν όχι.

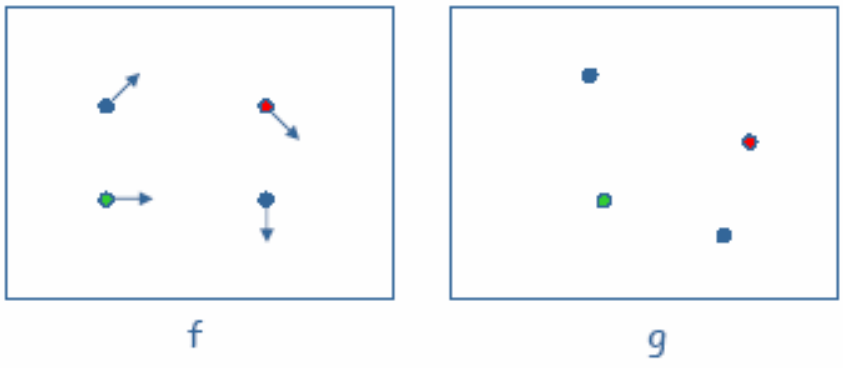
5. ΟΠΤΙΚΗ ΡΟΗ (OPTICAL FLOW)

5.1 Ορισμός της Οπτικής Ροής (Optical Flow)

Ότι παρατηρούμε γύρω μας είναι μια τρισδιάστατη απεικόνιση του περιβάλλοντος. Μπορούμε να θεωρήσουμε λοιπόν, ότι κάθε εικονοστοιχείο «αναπαριστά» ένα μικρό κομμάτι του (pinhole model). Η αναπαράσταση αυτή μετράται μέσω του προσπίπτοντος φωτός στην αντίστοιχη κατεύθυνση. Στην πράξη, βέβαια, η μέτρησή της εξαρτάται από τη ραδιομετρική και χρωματική ευαισθησία της κάμερας, καθώς και από τα χαρακτηριστικά της επιφάνειας του αντικειμένου σε σχέση με τη σκηνή στην οποία κινείται.

Θεωρούμε ότι δύο εικόνες *f* και *g* είναι δύο πλαίσια μια αληθινής τρισδιάστατης σκηνής, όπως τα έχει συλλάβει η κάμερα. Η κάμερα είναι ακίνητη και παρατηρούμε ότι κάθε ένα από τα τέσσερα κινούμενα αντικείμενα ακολουθεί τη δική του ανεξάρτητη τροχιά. Θέλουμε έναν τρόπο για να ανιχνευτεί και να εκτιμηθεί το πεδίο κινήσεων. Τη λύση μας δίνει η εκτίμηση της οπτικής ροής.

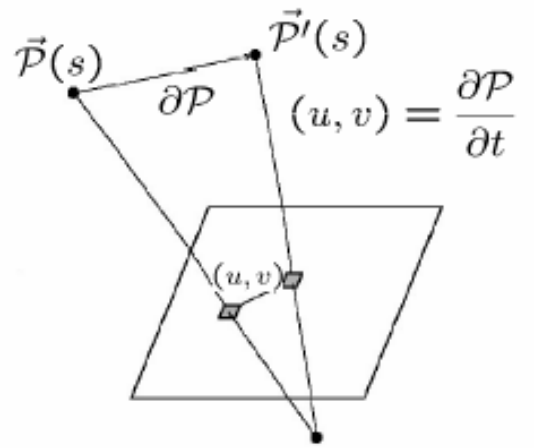
.



Εικόνα 5.1 : Παράδειγμα δύο διαδοχικών πλαισίων ακολουθίας βίντεο.

Έστω, ότι η φωτεινότητα στο σκηνικό παραμένει σταθερή παρά την κίνηση των αντικειμένων και συνεπώς το προσπίπτον φως από την αντίστοιχη κατεύθυνση παραμένει σταθερό. Θεωρούμε, επίσης, την προβολή μίας τρισδιάστατης στοιχειώδους περιοχής s τη χρονική στιγμή t στην εικόνα $\vec{p}'$, καθώς και τη χρονική στιγμή $t+1$ στη νέα εικόνα $\vec{P}'(s)$ (το επόμενο πλαίσιο της ακολουθίας βίντεο). Η κίνηση μπορεί να έχει προκύψει είτε από την κίνηση του ίδιου αντικειμένου είτε από την κίνηση της κάμερας και ισχύει ότι $\vec{P}(s) \neq \vec{P}'(s)$.

Σύμφωνα με τα παραπάνω, μπορούμε να ορίσουμε την οπτική ροή ως την ενέργεια ανάκτησης του δισδιάστατου διανύσματος της ταχύτητας (u, v) της τρισδιάστατης στοιχειώδους περιοχής s στο χρόνο. Δηλαδή $(\mathrm{u}, \mathrm{v}) = \vec{P}(s) - \vec{P}'(s)$. Στην ουσία η εκτίμηση της οπτικής ροής είναι η εκτίμηση της κίνησης σε μια ακολουθία βίντεο.

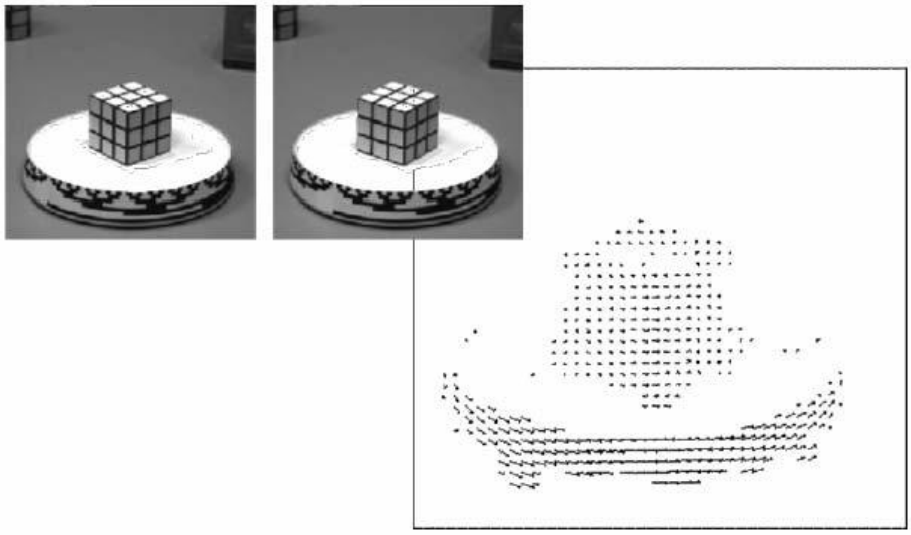


Εικόνα 5.2 : Προβολές της στοιχειώδους περιοχής πάνω στις εικόνες.

Αν η προβολή της s πάνω στην $\vec{P}(s)$ είναι $f\left(\vec{P}(s)\right)$, η προβολή της ίδιας περιοχής στο επόμενο πλαίσιο της ίδιας ακολουθίας βίντεο είναι $g\left(\vec{P}'(s)\right)$ και θεωρήσουμε ότι η φωτεινότητα παραμένει σταθερή, τότε ισχύει :

$$f\left(\vec{P}(s)\right) = g\left(\vec{P}'(s)\right) \text{ και } f\left(\vec{P}(s)\right) = g\left(\vec{P}(s) + (\mathrm{u},\mathrm{v})\right) \quad (5.1)$$

Στην Εικόνα 5.3 δίνεται ένα παράδειγμα εκτίμησης οπτικής ροής. Δίνονται τα δύο διαδοχικά πλαίσια της ακολουθίας βίντεο και το ροόγραμμά της. Στα επόμενα κεφάλαια θα αναλυθούν οι υπάρχουσες μέθοδοι για την εκτίμηση της οπτικής ροής και θα προταθεί μια νέα προσέγγιση στο πρόβλημα της σωστότερης εκτίμησής της.



Εικόνα 5.3 : Παράδειγμα εκτίμησης οπτικής ροής.

Η εκτίμηση της οπτικής ροής είναι ένα πρόβλημα που η λύση του βρίσκει εφαρμογή σε σημαντικούς τομείς, όπως στην:

i. **Κωδικοποίηση και Συμπίεση Βίντεο**. Το πρότυπο MPEG4 λειτουργεί βάσει της εκτίμησης της οπτικής ροής. Μεταξύ διαδοχικών πλαισίων δε χρειάζεται παρά η γνώση του αρχικού και η κωδικοποίηση μόνο των διανυσμάτων κίνησης για τα επόμενα πλαίσια.

ii. **Τρισδιάστατη Ανακατασκευή**. Μπορεί να θεωρηθεί ότι η μια κινούμενη κάμερα είναι στην ουσία δύο για το στατικό σκηνικό, οπότε η εκτίμηση της κίνησης ανάγεται στην εκτίμηση των δύο διαδοχικών θέσεων της κάμερας.

iii. **Ανάλυση κίνησης και τροχιάς με εφαρμογή σε θέματα ασφάλειας**. Για παράδειγμα, μπορεί να γίνει ανίχνευση μεθυσμένων οδηγών αυτοκινήτων, που κινούνται με ιλιγγιώδη ταχύτητα ή με ασταθή πορεία.

5.2 Χρήση της Οπτικής Ροής στην εργασία

Η ανάγκη εύρεσης οπτικής ροής στην εργασία αυτή έγκειται στο ότι δεν είναι δυνατόν να ανιχνεύεται σε κάθε εικόνα από την αρχή ένας marker και να εξάγονται τα σημεία από αυτόν. Αυτό γιατί εκτός του ότι δεν εξάγονται έγκυρα αποτελέσματα, καθυστερεί κατά πολύ την εκτέλεση του προγράμματος.

Για τον λόγο αυτό, στην παρούσα εργασία κρίθηκε πως καταλληλότερη μέθοδος οπτικής ροής (optical flow), για την παρακολούθηση των σημείων που βρέθηκαν (βλέπε εικόνα 4.1) στην ροή των εικόνων, είναι η πυραμιδωτή Lucas – Kanade. Η μέθοδος αυτή είναι αρκετά ακριβής στα αποτελέσματά της, καθώς και αρκετά γρήγορη. Παίρνει σαν είσοδο δύο εικόνες, εκ των οποίων η πρώτη είναι μία αρχική κατάσταση της εικόνας και η δεύτερη, η αμέσως επόμενη στιγμή. Επίσης λαμβάνει ως είσοδο και τα σημεία που θα παρακολουθηθούν από την πρώτη στην δεύτερη εικόνα. Ακόμη δέχεται ως είσοδο και κάποια κριτήρια για το πού περίπου θα είναι τα σημεία στην επόμενη κατάσταση . Τέλος επιστρέφονται οι συντεταγμένες των σημείων που βρέθηκαν μετατοπισμένα στην δεύτερη εικόνα. Ο αλγόριθμος PyrLK είναι ο αλγόριθμος του Emgu Cv και θα αναλυθεί εκτενέστερα σε επόμενη υποενότητα.

Αξίζει να σημειωθεί πως ο καλύτερος δυνατός συνδυασμός μεθόδων εύρεσης σημείων και οπτικής ροής πειραματικά, είναι ο συνδυασμός “FindChessboardCorners() και PyrLK() (Pyramid Lucas – Kanade) “. Βρίσκονται εξαιρετικά γρήγορα τα σημεία πάνω στην σκακιέρα και ακολουθούνται με θεαματική ταχύτητα και ακρίβεια. Ο συνδυασμός των μεθόδων εξαγωγής σημείων από το τετράγωνο του marker και PyrLK() έχει εξίσου μεγάλη ταχύτητα παρακολούθησης των σημείων ως προς την μετατόπιση αλλά δεν έχει τόσο μεγάλη ακρίβεια και είναι ευάλωτος στην περιστροφή κατά άξονα Z.

Επεξήγηση μεθόδου *PyrLK ( )* :

```
public static void PyrLK(
        Image<Gray, byte> prev,
        Image<Gray, byte> curr,
        PointF[] prevFeatures,
        Size winSize,
        int level,
        MCvTermCriteria criteria,
        out PointF[] currFeatures,
        out byte[] status,
        out float[] trackError
)
```

Όπου : prev
Εικόνα πρώτη.

curr
Εικόνα την επόμενη στιγμή.

prevFeatures
Πίνακας με σημεία της πρώτης εικόνας που πρέπει να ακολουθηθούν.

winSize
Μέγεθος της περιοχής που αναζητούνται τα σημεία σε κάθε επίπεδο της πυραμίδας.

level
Αριθμός επιπέδων που χρησιμοποιούνται στην πυραμίδα.

criteria
Προσδιορίζει πότε πρέπει να σταματήσει η διαδικασία εύρεσης ροής για κάθε σημείο, σε κάθε επίπεδο της πυραμίδας.

currFeatures
Πίνακας με σημεία της δεύτερης εικόνας που ακολουθήθηκαν από την πρώτη.

status
Πίνακας με 0 και 1 ανάλογα αν βρέθηκε κάθε σημείο ή όχι.

trackError
Πίνακας που περιέχει τις των σφαλμάτων μεταξύ των αρχικών και τελικών σημείων.

5.3 Μέθοδοι εύρεσης της Οπτικής Ροής

Οι μέθοδοι εύρεσης της οπτικής ροής ακολουθούν τα παρακάτω στάδια :

I. Φιλτράρισμα και ομαλοποίηση της εικόνας με κάποιο βαθυπερατό ή ζωνοπερατό φίλτρο, για την εξάλειψη του υψίσυχνου θορύβου και αύξηση του σηματοθορυβικού λόγου.

II. Διεξαγωγή μετρήσεων στην εικόνα όπως χρονική και χωρική παραγώγιση, ή εύρεση συσχετίσεων μεταξύ περιοχών στην εικόνα, ανάλογα με τη χρησιμοποιούμενη μέθοδο.

III. Χρήση των μετρήσεων και κάποιων παραδοχών για την εύρεση της οπτικής ροής.

Γενικά οι μέθοδοι οπτικής ροής μπορούν να ομαδοποιηθούν σε κατηγορίες όπως διαφορικές τεχνικές (Horn-Schunck, Lucas-Kanade), τεχνικές σύμπτωσης περιοχών (block-matching), τεχνικές που βασίζονται στον υπολογισμό της "ενέργειας" της εικόνας και στην ελαχιστοποίηση του αντίστοιχου συναρτησιακού (Heeger) και τεχνικές φάσης (Fleet - Jepson). Γενικά πάντως πραγματοποιείται η υπόθεση της διατήρησης της φωτεινότητας, που θεωρεί ότι η φωτεινότητα ή το χρώμα των αντικειμένων δεν αλλάζει σημαντικά μεταξύ δύο διαδοχικών καρέ μιας ακολουθίας εικόνων. Εδώ θα ασχοληθούμε μόνο με τις διαφορικές τεχνικές και τις μεθόδους τους.

5.3.1 Μέθοδος Lucas-Kanade

Η μέθοδος των Lucas και Kanade εξακολουθεί να είναι μια από τις πιο δημοφιλείς μεθόδους για την εκτίμηση της κίνησης μεταξύ δύο καρέ (frames) ενός βίντεο. Ανήκει στην κατηγορία των διαφορικών μεθόδων διότι βασίζεται σε προσέγγιση Taylor του φωτεινού σήματος της εικόνας, και συνεπώς χρησιμοποιούν μερικές παραγώγους στο χώρο και στο χρόνο.

Έστω μια ακολουθία εικόνων I, και ένα pixel στη θέση (x,y,z,t) με φωτεινότητα $I(x,y,z,t)$ στην εικόνα αναφοράς. Στο επόμενο καρέ (εικόνα ελέγχου) το pixel αυτό θα έχει μετακινηθεί κατά $\delta x, \delta y, \delta z, \delta t,$ οπότε η διατήρηση της φωτεινότητας δίνει:

$$I(x,y,z,t) = I(x+\delta x, y+\delta y, z+\delta z, t+\delta t) \qquad (5.2)$$

Υποθέτοντας ότι η κίνηση είναι μικρή, μπορούμε να αναπτύξουμε την εξίσωση (5.2) κατά Taylor και να πάρουμε:

$$I(x+\delta x, y+\delta y, z+\delta z, t+\delta t) = I(x,y,z,t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t + H.O.T. \qquad (5.3)$$

Όπου H.O.T. σημαίνει όροι υψηλότερης τάξεως, που είναι δυνατόν να αγνοηθούν. Από τις εξισώσεις αυτές προκύπτει ότι :

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial z}\delta z + \frac{\partial I}{\partial t}\delta t = 0 \qquad (5.4)$$

ή

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial z}\frac{\delta z}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \qquad (5.5)$$

που συνεπάγεται ότι :

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial z}V_z + \frac{\partial I}{\partial t}\delta = 0 \qquad (5.6)$$

όπου V_x, V_y, V_z είναι οι x, y, z συνιστώσες της ταχύτητας, ή αλλιώς η οπτική ροή για την εικόνα $I(x,y,z,t)$ και $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$, $\frac{\partial I}{\partial z}$, και $\frac{\partial I}{\partial t}$ είναι οι παράγωγοι της εικόνας στο σημείο (x,y,z,t) στις αντίστοιχες διευθύνσεις. Οι παραπάνω παράγωγοι μπορούν επίσης να γραφούν ως I_x , I_y , I_z και I_t ,οπότε η παραπάνω σχέση γράφεται :

$$I_x V_x + I_y V_y + I_z V_z = -It \implies \nabla I \cdot \vec{V} = -I_t \qquad (5.7)$$

Η παραπάνω εξίσωση έχει τρείς αγνώστους και ως εκ τούτου δε μπορεί να λυθεί άμεσα. Το γεγονός αυτό είναι γνωστό ως το πρόβλημα του διαφράγματος (aperture problem). Για να βρεθεί το πεδίο οπτικής ροής χρειάζεται ακόμα ένα σύνολο εξισώσεων. Το οποίο θα προκύψει από κάποιον επιπρόσθετο περιορισμό. Η λύση που έδωσαν οι Lucas και Kanade είναι μια μη επαναληπτική μέθοδος που υποθέτει ότι η οπτική ροή είναι τοπικά σταθερή.

Πράγματι, υποθέτοντας ότι η ροή (V_x, V_y, V_z) είναι σταθερή σε ένα μικρό παράθυρο γύρω από το εξεταζόμενο pixel, διαστάσεων $m \times m \times m$ με $m > 1$ και αριθμώντας τα pixels εντός του παραθύρου από $1 \ldots n$, μπορούμε να διατυπώσουμε το εξής σύνολο εξισώσεων :

$$\begin{aligned} I_{x1}V_x + I_{y1}V_y + I_{z1}V_z &= -I_{t1} \\ I_{x2}V_x + I_{y2}V_y + I_{z2}V_z &= -I_{t2} \\ &\vdots \\ I_{xn}V_x + I_{yn}V_y + I_{zn}V_z &= -I_{tn} \end{aligned} \qquad (5.8)$$

Με τις εξισώσεις αυτές έχουμε παραπάνω από 3 εξισώσεις για τους 3 αγνώστους και συνεπώς το σύστημα εξισώσεων :

$$\begin{bmatrix} I_{x1} & I_{y1} & I_{z1} \\ I_{x2} & I_{y2} & I_{z2} \\ \vdots & \vdots & \vdots \\ I_{xn} & I_{yn} & I_{zn} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} -I_{t1} \\ -I_{t2} \\ \vdots \\ -I_{tn} \end{bmatrix} \qquad (5.9)$$

ή

$$A\vec{v} = -b \qquad (5.10)$$

είναι υπερορισμένο. Έτσι, εφαρμόζουμε για την επίλυση του, τη μέθοδο των ελαχίστων τετραγώνων:

$$A^T A\vec{v} = A^T(-b) \qquad (5.11)$$

ή

$$\vec{v} = \left(A^T A\right)^{-1} A^T(-b) \qquad (5.12)$$

ή

$$\begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix} = \begin{bmatrix} \sum I_{x_i}^2 & \sum I_{x_i} I_{y_i} & \sum I_{x_i} I_{z_i} \\ \sum I_{x_i} I_{y_i} & \sum I_{y_i}^2 & \sum I_{y_i} I_{z_i} \\ \sum I_{x_i} I_{z_i} & \sum I_{y_i} I_{z_i} & \sum I_{z_i}^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum I_{x_i} I_{t_i} \\ -\sum I_{y_i} I_{t_i} \\ -\sum I_{z_i} I_{t_i} \end{bmatrix} \qquad (5.13)$$

με τα αθροίσματα να εκτείνονται από $i = 1$ ως n.

Η παραπάνω εξίσωση υποδηλώνει ότι η οπτική ροή μπορεί ν βρεθεί υπολογίζοντας τις παραγώγους τις εικόνας και στις 4 διαστάσεις (3 αν πρόκειται για δισδιάστατη εικόνα). Συνήθως προστίθεται κάποια συνάρτηση βάρους $W(i,j,k), i,j,k \in [1,m]$ που ενισχύει τη συνεισφορά του κεντρικού pixel εντός του παραθύρου. Οι γκαουσιανές συναρτήσεις προτιμώνται για το σκοπό αυτό, χωρίς βέβαια να αποκλείονται και άλλου τύπου συναρτήσεις.

Εκτός από τον υπολογισμό τοπικών μετατοπίσεων, το μοντέλο ροής μπορεί να επεκταθεί και σε αφινικές μεταβολές. Χαρακτηριστικό του αλγορίθμου LK είναι ότι δεν είναι ικανός να παράξει ένα ιδιαίτερα πυκνό πεδίο από διανύσματα οπτικής ροής, ιδιότητα κοινή για όλες τις μεθόδους εκτίμησης του πεδίου ροής που βασίζονται σε τοπική πληροφορία γύρω από κάθε pixel. Με άλλα λόγια, η χρήσιμη πληροφορία του πεδίου ροής μειώνεται γρήγορα στις περιοχές των συνόρων μεταξύ αντικειμένων και στο εσωτερικό μεγάλων ομογενών περιοχών χωρίς ιδιαίτερη υφή, όπου δε μπορεί να ανιχνευθεί ισχυρή κίνηση. Το πλεονέκτημα της εν λόγω μεθόδου είναι ότι είναι αρκετά εύρωστη απέναντι στο θόρυβο.

5.3.2 Μέθοδος Horn-Schunck

Η μέθοδος των Horn-schunck είναι μια μέθοδος καθολικής εκτίμησης του πεδίου ροής που εισάγει ένα καθολικό κριτήριο ομαλότητας προκειμένου να επιλύσει το πρόβλημα του διαφράγματος. Εδώ ζητείται η ελαχιστοποίηση του εξής συναρτησιακού ενέργειας :

$$f = \iiint \left(\left(\nabla I \cdot \vec{V} + I_t \right)^2 + \alpha (|\nabla V_x|^2 + |\nabla Vy|^2 + |\nabla Vz|^2) \right) dxdydz \qquad (5.14)$$

όπου $\nabla I = \begin{bmatrix} I_x \\ I_y \\ I_z \end{bmatrix}$ είναι οι παράγωγοι της εικόνας κατά μήκος των x, y και z διευθύνσεων, και I_t είναι η παράγωγος στο χρόνο, $\vec{V}$ είναι το διάνυσμα οπτικής ροής με συνιστώσες V_x, Vy, Vz. Η παράμετρος α είναι μια σταθερά κανονικοποίησης, με τις μεγαλύτερες τιμές του α να οδηγούν σε ομαλότερο πεδίο ροής. Η ελαχιστοποίηση της συνάρτησης αυτής μπορεί να γίνει υπολογίζοντας τις αντίστοιχες εξισώσεις Euler-Lagrange :

$$\Delta V_x - \frac{1}{\alpha} I_x \left(I_x V_x + I_y V_y + I_z V_z + I_t \right) = 0 \qquad (5.15)$$

$$\Delta V_y - \frac{1}{\alpha} I_y \left(I_x V_x + I_y V_y + I_z V_z + I_t \right) = 0 \qquad (5.16)$$

$$\Delta V_z - \frac{1}{\alpha} I_z \left(I_x V_x + I_y V_y + I_z V_z + I_t \right) = 0 \qquad (5.17)$$

όπου Δ συμβολίζει τον τελεστή Laplace, ώστε $\Delta V_x = \frac{\partial}{\partial x}\frac{\partial V_x}{\partial x}, \Delta V_y = \frac{\partial}{\partial y}\frac{\partial y}{\partial y}, \Delta V_z = \frac{\partial}{\partial z}\frac{\partial V_z}{\partial z}$. Επιλύοντας τις εξισώσεις αυτές με τη μέθοδο Gauss-Seidel ως προς τις συνιστώσες V_x, V_y, V_z της οπτικής ροής παίρνουμε το εξής σχήμα :

$$V_x^{k+1} = \frac{\Delta V_x^k - \frac{1}{\alpha} I_x \left(I_y V_y^k + I_z V_z^k + I_t \right)}{\frac{1}{\alpha} I_x^2} \qquad (5.18)$$

$$V_y^{k+1} = \frac{\Delta V_y^k - \frac{1}{\alpha} I_y \left(I_x V_x^k + I_z V_z^k + I_t \right)}{\frac{1}{\alpha} I_y^2} \qquad (5.19)$$

$$V_z^{k+1} = \frac{\Delta V_z^k - \frac{1}{\alpha} I_z \left(I_x V_x^k + I_y V_y^k + I_t \right)}{\frac{1}{\alpha} I_z^2} \qquad (5.20)$$

όπου $k+1$ ο εκθέτης υποδηλώνει την επόμενη επανάληψη και ο k δείχνει στο πιο πρόσφατα υπολογισμένο αποτέλεσμα. Οι τιμές ΔV_i μπορούν να ληφθούν ως εξής :

$$\Delta V_i = \sum_{N(p)} V_i\big(N(p)\big) - V_i(p) \qquad (5.21)$$

Όπου $N(p)$ είναι η γειτονιά με τους 6 πλησιέστερους γείτονες του pixel p.

Μια εναλλακτική υλοποίηση του αλγορίθμου HS με τη μέθοδο Jacobi δίνει τα εξής :

$$V_x^{k+1} = \overline{V_x^k} - \frac{I_x\left(I_x\overline{V_x^k}+I_y\overline{V_y^k}+I_z\overline{V_z^k}+I_t\right)}{a^2+I_x^2+I_y^2+I_z^2} \qquad (5.22)$$

$$V_y^{k+1} = \overline{V_y^k} - \frac{I_y\left(I_x\overline{V_x^k}+I_y\overline{V_y^k}+I_z\overline{V_z^k}+I_t\right)}{a^2+I_x^2+I_y^2+I_z^2} \qquad (5.23)$$

$$V_z^{k+1} = \overline{V_z^k} - \frac{I_z\left(I_x\overline{V_x^k}+I_y\overline{V_y^k}+I_z\overline{V_z^k}+I_t\right)}{a^2+I_x^2+I_y^2+I_z^2} \qquad (5.24)$$

όπου $\overline{V_i^k}$ αντιστοιχεί στο μέσο όρο V_i^k στην περιοχή του τρέχοντος pixel.

Το πλεονέκτημα του αλγορίθμου HS είναι ότι μπορεί να δώσει μεγάλη πυκνότητα διανυσμάτων ροής, δηλαδή η πληροφορία που λείπει από το εσωτερικό ομογενών περιοχών συμπληρώνεται από τα όρια των περιοχών αυτών. Ωστόσο είναι πιο ευαίσθητος στο θόρυβο απ' ότι οι τοπικές μέθοδοι όπως ο LK.

Πυραμιδωτή Εκτέλεση των Αλγορίθμων Οπτικής Ροής

Η πυραμιδωτή εκτέλεση των αλγορίθμων οπτικής ροής είναι μια ιεραρχική υλοποίηση των αλγορίθμων Οπτικής Ροής, που εφαρμόζεται και στους δύο παραπάνω αλγορίθμους. Συγκεκριμένα εξετάζεται η κατασκευή μιας Λαπλασιανής πυραμίδας, όπου στο χαμηλότερο επίπεδο βρίσκονται οι εικόνες στην αρχική τους μορφή, ενώ σε ανώτερα επίπεδα κάποιες ομαλοποιημένες και υποδειγματοληπτημένες εκδοχές αυτών. Ξεκινώντας από τα χαμηλότερα επίπεδα (στην κορυφή της πυραμίδας) εκτελούνται οι αλγόριθμοι οπτικής ροής, και τα αποτελέσματά τους χρησιμοποιούνται ως μια αρχική εκτίμηση για την οπτική ροή του επόμενου επιπέδου. Το σύστημα που προτείνεται αποτελείται από 4 στάδια:

i. Κατασκευή της Λαπλασιανής πυραμίδας.

ii. Υπολογισμός του πεδίου ροής στο τρέχον επίπεδο, με οποιονδήποτε αλγόριθμο.

iii. Χρησιμοποίηση του υπολογισμένου πεδίου ροής για την εκτίμηση της εικόνας ελέγχου, και σύγκριση για εύρεση του σφάλματος εκτίμησης.

iv. Μεταφορά των παραμέτρων του μοντέλου εκτίμησης της οπτικής ροής (εάν υπάρχουν) και μεταφορά στο επόμενο επίπεδο. Το αποτέλεσμα του τρέχοντος επιπέδου τροποποιείται κατάλληλα ώστε να αντιστοιχεί στις διαστάσεις του επόμενου επιπέδου.

Η εκτέλεση των αλγορίθμων σε ιεραρχική πυραμίδα εξυπηρετεί τόσο από άποψη χρόνου εκτέλεσης, καθώς το πεδίο ροής υπολογίζεται ταχύτατα στα υψηλότερα επίπεδα και η αρχική εκτίμηση βοηθά στην επιτάχυνση της διαδικασίας στα χαμηλότερα, όσο και από την άποψη της ποιότητας των αποτελεσμάτων. Για παράδειγμα, η χρήση της πυραμίδας επιτρέπει σε τοπικές μεθόδους όπως ο αλγόριθμος LK να ανιχνεύσουν με επιτυχία και μεγαλύτερες μετατοπίσεις απ' ότι ο απλός αλγόριθμος (που μπορεί να ανιχνεύσει πολύ μικρές μόνο κινήσεις), καθώς οι μετατοπίσεις αυτές ανιχνεύονται με επιτυχία στα υψηλότερα επίπεδα όπου το μέγεθός τους έχει μειωθεί σημαντικά, και μεταφέρονται κατόπιν στα χαμηλότερα επίπεδα.

6. ΕΚΤΙΜΗΣΗ ΘΕΣΗΣ ΚΑΜΕΡΑΣ (POSE ESTIMATION)

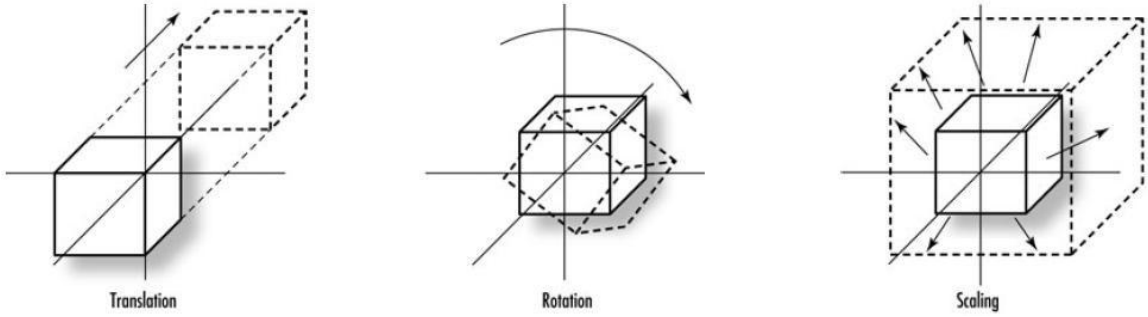
6.1 Γενικά

Η εύρεση των εξωτερικών παραμέτρων (με δεδομένη την εκ των προτέρων γνώση των εσωτερικών παραμέτρων) που δίνουν τη θέση και τον προσανατολισμό της κάμερας σε σχέση με τον πραγματικό τρισδιάστατο κόσμο, είναι γνωστή σαν pose estimation.

Η σημασία της στην παρούσα εργασία είναι και η πιο σημαντική, αφού στην ουσία όλα τα παραπάνω γίνονται για να βρεθεί η πόζα της κάμερας. Βρίσκοντας τη θέση και τον προσανατολισμό της πραγματικής κάμερας, μπορούμε να “ευθυγραμμίσουμε” με αυτή την εικονική κάμερα του αντίστοιχου τρισδιάστατου μοντέλου. Σαν αποτέλεσμα έχουμε την εμφάνιση της σωστής όψης του τρισδιάστατου μοντέλου, στο κατάλληλο μέγεθος και στο κατάλληλο μέρος(βλέπε κεφάλαιο 7). Οι εξωτερικές παράμετροι της κάμερας που αναζητούνται κατά την διαδικασία της εκτίμησης της θέσης της (pose estimation) είναι δύο πίνακες (βλέπε εικόνα 6.1) :

- Ο πίνακας (rotation), μεγέθους 3x3, που αποτελείται από 3 διανύσματα στοιχειωδών περιστροφών $R = [R_1\ R_2\ R_3]$. Τα διανύσματα αυτά αντιπροσωπεύουν, όπως προαναφέρθηκε, την περιστροφή της κάμερας γύρω από τους 3 άξονες ενός συστήματος συντεταγμένων (3 βαθμοί ελευθερίας για την rotation).

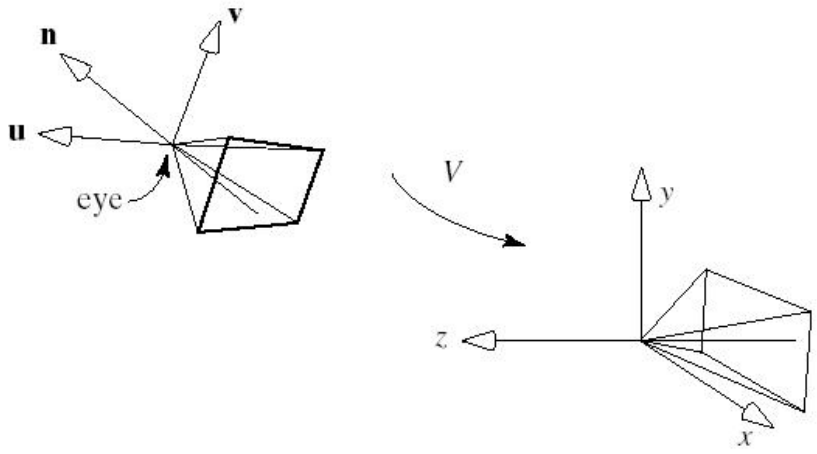
- Ο πίνακας t (translation), μεγέθους 3x1, που αντιπροσωπεύει τη μετακίνηση της κάμερας. Κάθε στοιχείο του διανύσματος εκφράζει τη μετακίνηση στον κατάλληλο άξονα του συστήματος συντεταγμένων (3 βαθμοί ελευθερίας για το translation).



Εικόνα 6.1 : Rotation, Translation και Scaling ενός αντικειμένου.

Συνοψίζοντας, κατά την εκτίμηση της θέσης της κάμερας (Pose Estimation) γίνεται προσπάθεια για να βρεθούν 6 παράμετροι.

Όπως μπορεί να γίνει εύκολα αντιληπτό, ο συνδυασμός αυτός του πίνακα περιστροφής (Rotation) και του πίνακα μετατόπισης (Translation) μπορεί να μοντελοποιήσει οποιαδήποτε κίνηση της κάμερας (βλέπε εικόνα 6.2).



Εικόνα 6.2 : Αναπαράσταση Rotation και Translation μιας κάμερας.

6.2 Εύρεση εξωτερικών παραμέτρων(Extrinsic Parameters)

Όλες οι διαδικασίες που ακολουθήθηκαν μέχρι στιγμής, έγιναν με απώτερο στόχο την εύρεση της θέσης της κάμερας. Τα ως τώρα δεδομένα είναι οι εσωτερικές παράμετροι της κάμερας, οι οποίες πάρθηκαν από την διαδικασία βαθμονόμησης της κάμερας (camera calibration) και τα σημεία που ανιχνεύτηκαν πάνω στον marker , όπου ως προς αυτά θα γίνει η εκτίμηση της θέσης της κάμερας. Έχοντας λοιπόν αυτά σαν δεδομένα, το μόνο που απομένει είναι η εύρεση του πίνακα εξωτερικών παραμέτρων (extrinsic parameters). Για τον λόγο αυτό δοκιμάστηκαν διάφορες μέθοδοι του Emgu Cv. Αρχικά ο πίνακας αυτός θεωρήθηκε πως μπορούσε να βρεθεί μέσω της μεθόδου CalibrateCamera(), όμως στην συνέχεια διαπιστώθηκε ότι δεν μπορεί να συμβεί αυτό στην παρούσα εργασία, γιατί χρειάζονται σταθερές εσωτερικές παράμετροι, όπου βάσει αυτών να υπολογίζονται οι εξωτερικές παράμετροι και όχι να μεταβάλλονται και οι δύο. Στην συνέχεια δοκιμάστηκε η μέθοδος Posit() όπου φαινομενικά έδειχνε να λειτουργεί, αλλά πρακτικά δεν λειτούργησε ποτέ ολοκληρωτικά σωστά. Αυτό μπορεί να οφείλεται σε λάθος χρήση της μεθόδου. Τέλος, μετά από πολύ έρευνα αποδείχτηκε πως η απάντηση βρισκόταν στην πιο προφανή μέθοδο. Η μέθοδος αυτή ονομάζεται FindExtrinsicCameraParams2(). Δέχεται σαν είσοδο κάποια “εικονικά” σημεία (Πρέπει να αντιπροσωπεύουν το σχήμα που σχηματίζουν τα σημεία πάνω στην εικόνα, όπως ακριβώς συνέβαινε και με την μέθοδο CalibrateCamera(), βλέπε εικόνα 2.6) όπου θα αντιστοιχηθούν με τα σημεία που ανιχνεύθηκαν στον marker, τα οποία επίσης εισάγονται στην μέθοδο. Ακόμα εισάγονται και οι εσωτερικοί παράμετροι της κάμερας και βάσει όλων αυτών υπολογίζεται ο πίνακας των εξωτερικών παραμέτρων της κάμερας. Ο πίνακας αυτός περιέχει όλα τα στοιχεία που χρειάζονται για την απεικόνιση της θέσης (περιστροφή – μετατόπιση) μιας κάμερας σε σχέση με ένα marker σε ένα χώρο. Αξίζει να σημειωθεί πως όσο περισσότερα σημεία χρησιμοποιούνται, τόσο πιο ακριβές είναι και το αποτέλεσμα.

```
public static ExtrinsicCameraParameters FindExtrinsicCameraParams2(
        MCvPoint3D32f[] objectPoints,
        PointF[] imagePoints,
        IntrinsicCameraParameters intrin
)
```

Όπου : objectPoints

Πίνακας “εικονικών” σημείων.

imagePoints

Πίνακας σημείων πάνω στην εικόνα.

intrin

Πίνακας εσωτερικών παραμέτρων της κάμερας.

Return Value

Πίνακας εξωτερικών παραμέτρων.

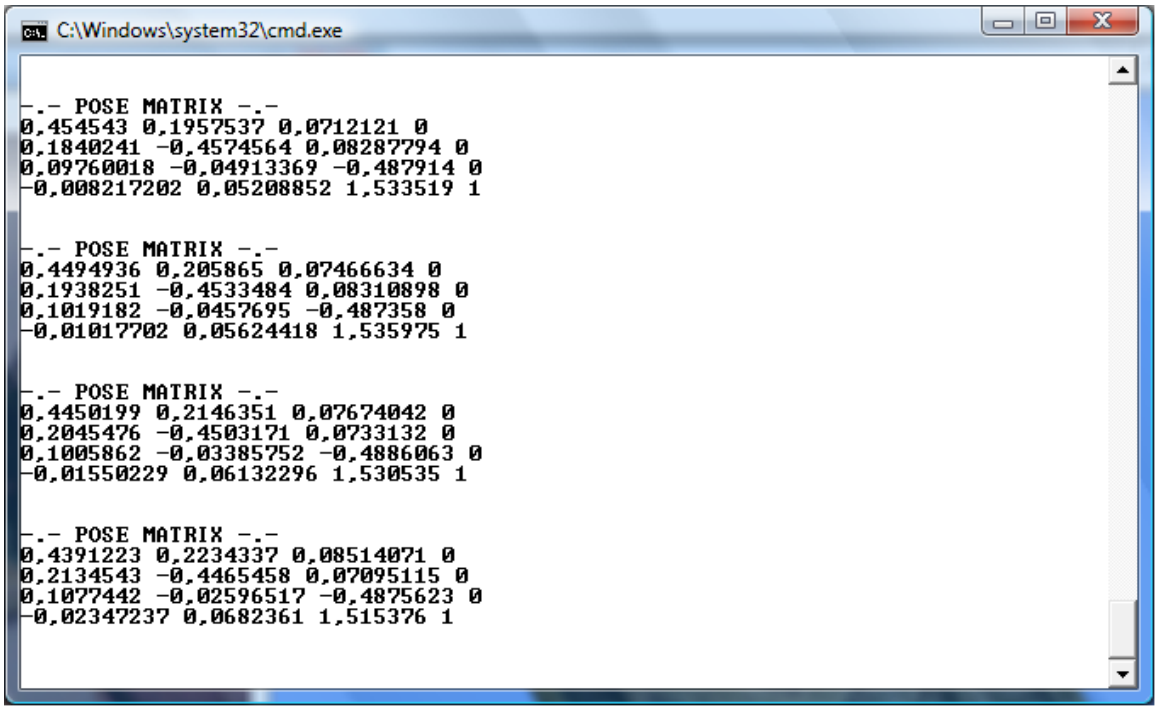
7. XNA

7.1 Γενικά

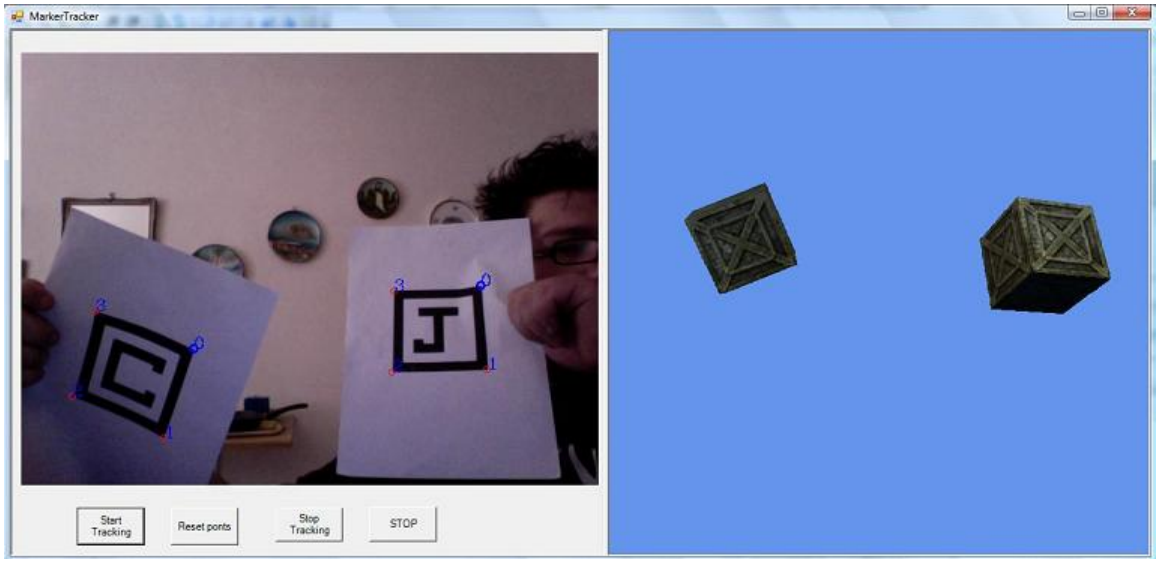
XNA γενικά (η Microsoft ισχυρίζεται ότι δεν είναι αρχικά λέξεων) ονομάζεται όλο το σετ των εργαλείων που παρέχει η Microsoft για την ανάπτυξη παιχνιδιών σε Windows, Xbox360 και Zune. Περιλαμβάνει εργαλεία όπως το PIX για γραφικά, το XACT για ήχο, το Xbox Development Kit, το Visual Studio κλπ. Το XNA Game Studio είναι ένα υποσύνολο των εργαλείων αυτών που στοχεύει αποκλειστικά στην ανάπτυξη παιχνιδιών σε Windows, Xbox360 και Zune χρησιμοποιώντας τη τεχνολογία .NET της ίδιας εταιρίας. Βασίζεται πάνω στο περιβάλλον ανάπτυξης Visual Studio, την γλώσσα προγραμματισμού C#, σε μια ειδική έκδοση του DirectX και φυσικά στο .NET. Το XNA Game Studio έχει φτάσει στην έκδοση 3.0 και είναι διαθέσιμο δωρεάν σε όλους.

7.2 Εξομοίωση κίνησης κάμερας με XNA

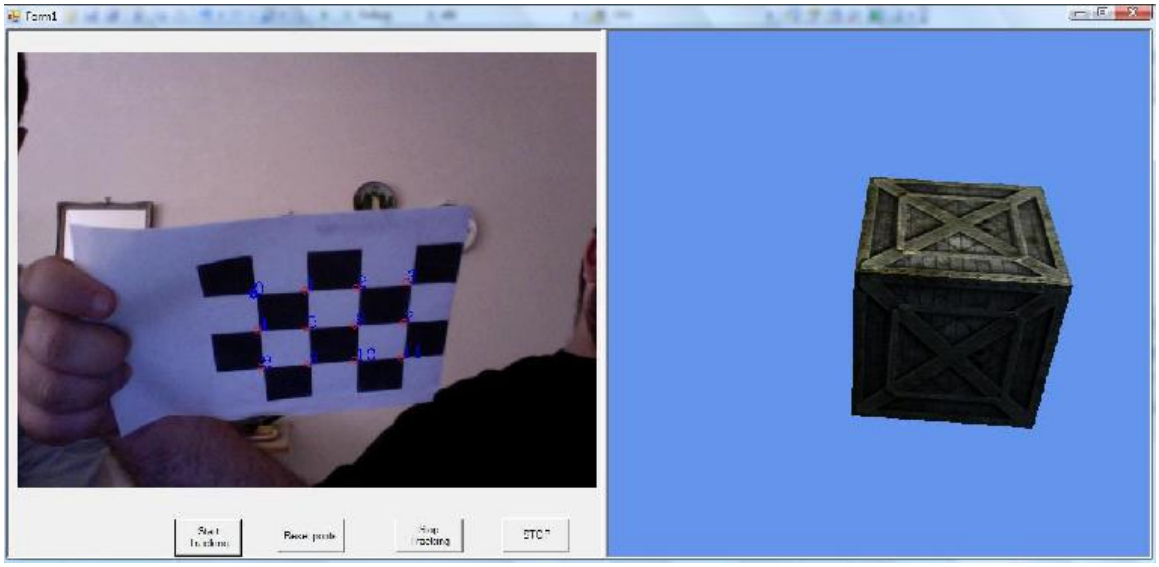
Για την απεικόνιση της κίνησης μιας κάμερας σε σχέση με ένα marker , χρησιμοποιήθηκε το XNA. Προβάλλεται ένα τρισδιάστατο (3D) μοντέλο (στην εργασία αυτή ένας κύβος) και εξομοιώνει την κίνηση της κάμερας στον χώρο σε σχέση με τον marker. Για την κίνηση του αντικειμένου αυτού χρησιμοποιείται ο πίνακας εξωτερικών παραμέτρων που εξηγήθηκε πως βρέθηκε στην προηγούμενη ενότητα, τροποποιημένος κατάλληλα ούτως ώστε να μπορεί να χρησιμοποιηθεί από το XNA. Ειδικότερα η μέθοδος FindExtrinsicCameraParams2() επιστρέφει τον πίνακα με τις περιστροφές και μετατοπίσεις ως γραμμές ενώ το XNA δέχεται πίνακες για την απεικόνιση της κίνησης των αντικειμένων ως στήλες. Για τον λόγο αυτό δημιουργείται ένας πίνακας με τα στοιχεία της προαναφερθείσας μεθόδου τοποθετημένα σε στήλες. Ακόμη πρέπει να γίνουν κάποιες μετατροπές στις τιμές του πίνακα αυτού έτσι ώστε να προσαρμοστούν σε κατάλληλη κλίμακα για την ομαλή και σωστή απεικόνιση της κίνησης του 3D αντικειμένου.



Εικόνα 7.1 : Στιγμιότυπα Πίνακα Περιστροφής και Μετατόπισης κατάλληλα διαμορφωμένου για τη κίνηση ενός αντικειμένου στο ΧΝΑ.



Εικόνα 7.2 : Στιγμιότυπο χρήσης της εφαρμογής με ταυτόχρονη αναγνώριση δύο markers , η εκτίμηση θέσης τους και η ανάλογη απεικόνιση σε ΧΝΑ.



Εικόνα 7.3 : Στιγμιότυπο χρήσης της εφαρμογής με αναγνώριση μιας σκακιέρας 3χ4 , η εκτίμηση θέσης της και η ανάλογη απεικόνιση σε XNA.

ΒΙΒΛΙΟΓΡΑΦΙΑ

1. J. Shi and C. Tomasi : “Good Features to Track”, Proc. IEEE Conf. Computer Vision and Pattern Recognition, 1994.

2. D. G. Lowe : “Distinctive Image Features from Scale-Invariant Keypoints”, International Journal of Computer Vision, 2004.

3. T. Kanade and D. D. Morris : “Factorization Methods for Structure From Motion”, 1997.

4. D. F. Dementhon, and L. S. Davis D. Oberkampf : “Iterative pose estimation using coplanar points," in IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '93), 1993. pp. 626-627.

5. Martin Hirzer : “Marker Detection for Augmented Reality Applications," TU-Graz, 2008.

6. C. Harris and M. Stephens : "A combined corner and edge detector : “ in Proceedings of the 4th Alvey Vision Conference, 1988.

7. Lewis, J. P. : “Fast Template Matching. s.l. : Vision Interface", 1995.

8. Gerhand X. Ritter, Joseph N. Wilson. : "Handbook of Computer Vision Algorithms in Image Algebra.

9. T.Kanade, C.Tomasi. : “Detetion and tracking of point features". Carnegie Mellon University Technical Report CMU CS. 1991.

10. J. L. Barron, D. J. Fleet, and S.S. Beauchemin: “Performance of Optical Flow Techniques,” , International Journal of Computer Vision, February 1994.

11. David J. Fleet, Yair Weiss : “Optical Flow Estimation”.

12. Aaron Reed : “Learning XNA 3.0" - O’Reilly, November 2008: First Edition.

13. Jesse Liberty : “ Programming C# " -O'Reilly Second Edition February 2002.

14. Gary Bradski , Adrian Kaehler : “ Learning OpenCV " - O'Reilly, September 2008: First Edition

15. Κάτσενου Αγγελική : ” Εκτίμηση Οπτικής Ροής χρησιμοποιώντας Υπερδειγματοληπτημένες Ακολουθίες Βίντεο” - Μεταπτυχιακή Διπλωματική Εργασία , Πάτρα 2007

16. Γεωργακάκης Νικόλαος : “ Ταυτόχρονη κατάρτιση χάρτη και εκτίμηση θέσης κάμερας με παράλληλη αναγνώριση αντικειμένων και εισαγωγής τους στον χάρτη σε πραγματικό χρόνο “ – Διπλωματική Εργασία , Θεσσαλονίκη 2008

17. Τζιώνας Δημήτριος : “ Ευέλικτοι αλγόριθμοι ανίχνευσης markers σε εικόνα ” – Διπλωματική Εργασία , Θεσσαλονίκη 2009

18. Wikipedia - Pinhole camera model. [Online]. http://en.wikipedia.org/wiki/Pinhole_camera_model.

19. Emgu Cv forum. [Online]. http://www.emgu.com/forum/viewforum.php?f=1

20. Strang Gilbert : “ ΓΡΑΜΜΙΚΗ ΑΛΓΕΒΡΑ ΚΑΙ ΕΦΑΡΜΟΓΕΣ", 2005

21. Emgu Cv Documentation.