Τ.Ε.Ι ΚΡΗΤΗΣ
Σχολή Τεχνολογικών Εφαρμογών
Τμήμα Εφαρμοσμένης Πληροφορικής και Πόλυμέσων

ΤΕΧΝΟΛΟΓΙΚΟ
ΕΚΠΑΙΔΕΥΤΙΚΟ
ΙΔΡΥΜΑ ΚΡΗΤΗΣ

Πτυχιακή Εργασία

Ανάλυση Εικόνας με χρήση του OpenCV

Εισηγητής: Παπαδουράκης Γεώργιος

Σπουδαστής: Κιόσης Αθανάσιος 901

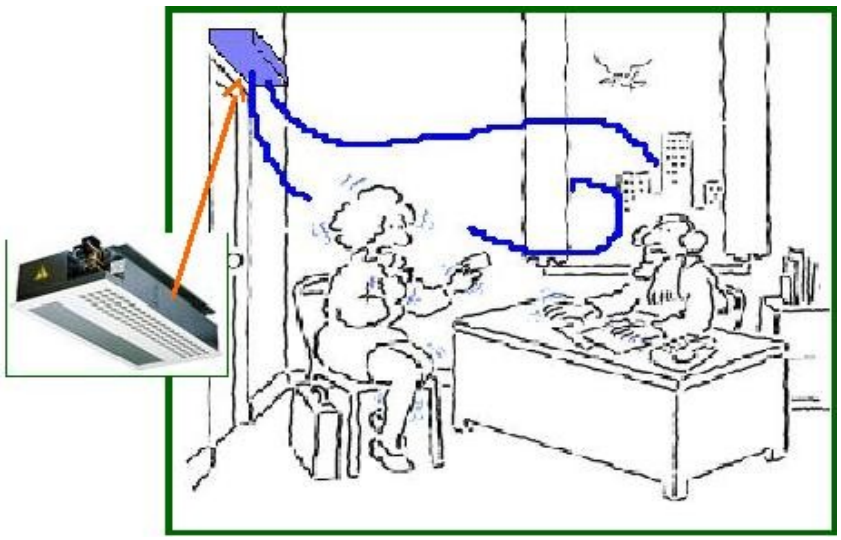
Ημερομηνία: 23/02/2009

Περιεχόμενα:

1. Εισαγωγή

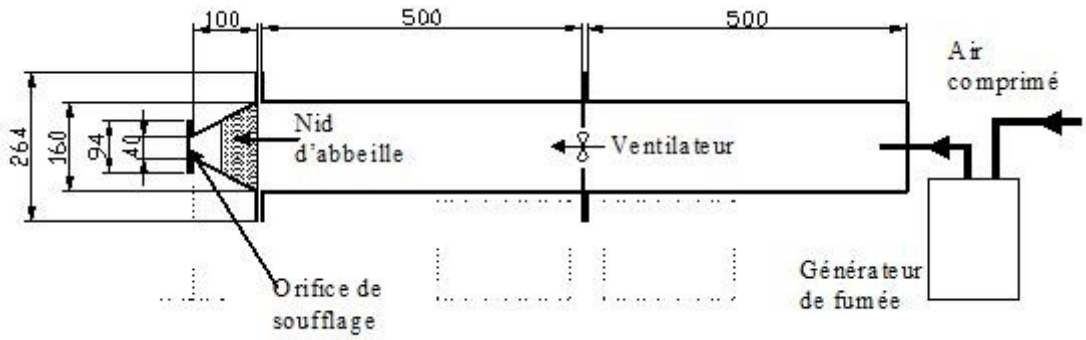
1.1 Σκοπός της Πτυχιακής

Ο σκοπός αυτής της πτυχιακής είναι η ανάλυση,η επεξεργασία και η εξαγωγή συμπερασμάτων σε εικόνες,με την βοήθεια της βιβλιοθήκης OpenCV που χρησιμοποιεί η γλώσσα προγραμματισμού C++. Με την χρήση της OpenCV θα δημιουργηθεί ένα πρόγραμμα,το οποίο θα επεξεργάζεται κατάλληλα μία ή περισσότερες εικόνες κάθε φορά,με σκοπό την εξαγωγή σημαντικών πληροφοριών.Συγκεκριμένα θα κατασκευαστεί ένα πρόγραμμα το οποίο θα μπορεί να ανοίγει μια ακολουθία από εικόνες και θα εφαρμόζει σε αυτές διάφορους αλγορίθμους,όπως είναι η κατωφλιοποίηση,η εύρεση κέντρου,περιγράμματος και ακτίνας σε μια επιφάνεια.Οι εικόνες τις οποίες θα επεξεργάζεται το πρόγραμμα είναι εικόνες στροβίλων και είναι προιόν πειραμάτων. Συγκεκριμένα,στο μαθηματικό εργαστήριο του πανεπιστήμιου του La Rochelle της Γαλλίας γίνανε ορισμένα πειράματα με σκοπό την επεξεργασία και ανάλυση των στροβίλων που παράγουν τα κλιματιστικά συστήματα. Ορισμένα πειράματα έδειξαν ότι με την σωστή μελέτη και επεξεργασία των στροβίλων που παράγουν τα κλιματιστικά μπορούμε να επιτύχουμε καλύτερη θέρμανση ή ψύξη του χώρου,όπως βλέπουμε και στην παρακάτω εικόνα 1.1.



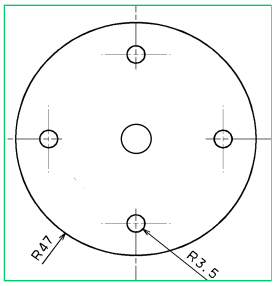
Εικόνα1.1

Γι'αυτό τον λόγο γίνανε φωτοραφίσεις (με την βοήθεια laser) των στροβίλων αυτών καθώς έβγαιναν από κλιματιστικά τα οποία είχαν πολλών διαφορετικών ειδών και μεγεθών στόμια.Οι φωτογραφίες αυτές απεικονίζουν τον στρόβιλο είτε σε οριζόντια είτε σε κάθετη μορφή.

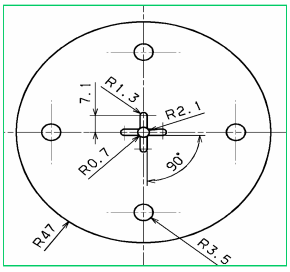


Εικόνα 1.2

Όπως βλέπουμε και στην παραπάνω εικόνα,ο αέρας βγαίνει από το στόμιο του κλιματιστικού,το οποίο μπορεί να έχει διαφορετικό μήκος,από 40-264 mm. Στο πρόγραμμα θα μελετηθούν οι στροβίλοι για μήκος 40 και 50mm.Τα στόμια είναι δύο ειδών,τα κυκλικά(εικόνα 1.3) και τα αυτά με στόμιο σε σχήμα σταυρού(εικόνα 1.4).

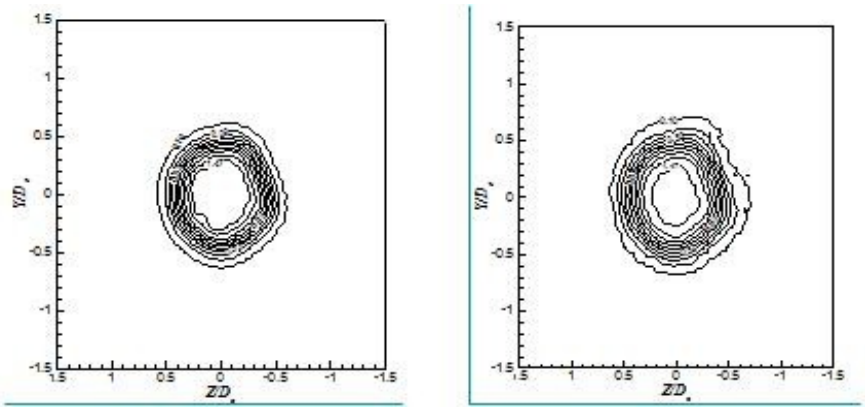


εικόνα 1.3

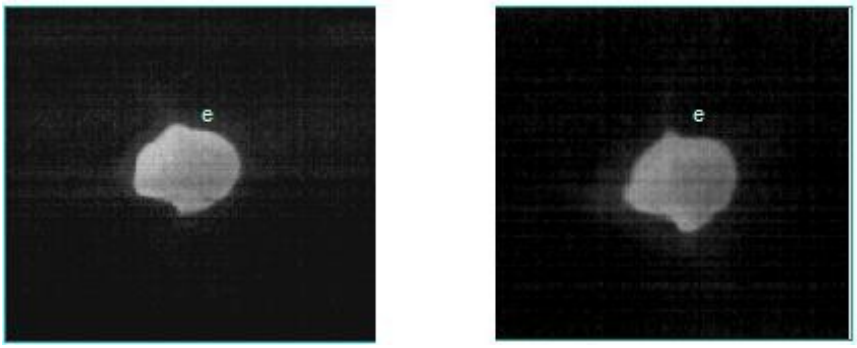


εικόνα 1.4

Στην περίπτωση που χρησιμοποιείται το κυκλικό στόμιο,ο αέρας που βγαίνει έχει την μορφή κυκλικού στροβίλου και φαίνεται στις παρακάτω εικόνες σε μορφή διαγράμματος αλλά και σε φωτογραφία (1.5 και 1.6).

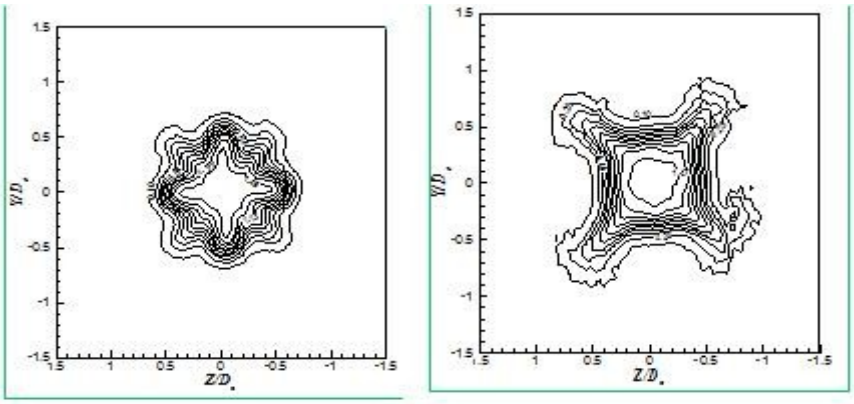


εικόνα 1.5

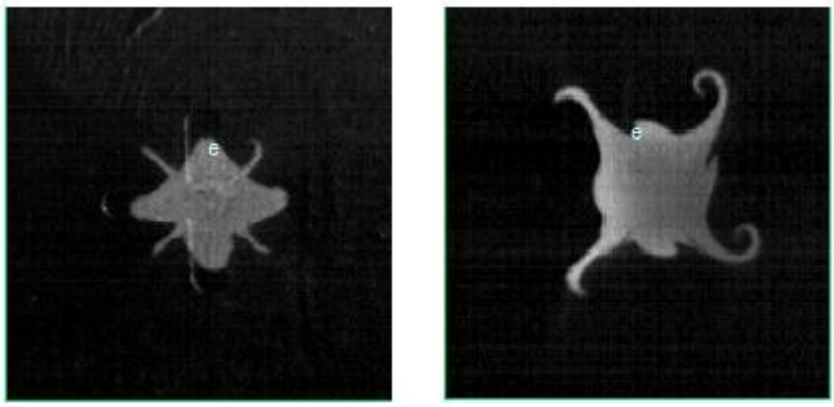


εικόνα 1.6

Στην περίπτωση που χρησιμοποιείται το στόμιο με σχήμα σταυρού,ο αέρας που βγαίνει έχει την μορφή μαργαριτοειδούς στροβίλου και φαίνεται στις παρακάτω εικόνες σε μορφή διαγράμματος αλλά και σε φωτογραφία (1.7 και 1.8).

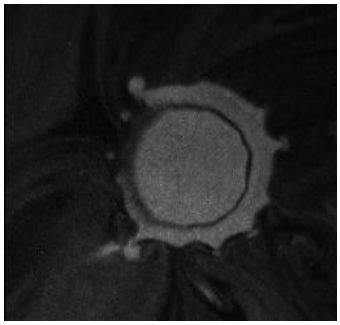


εικόνα 1.7

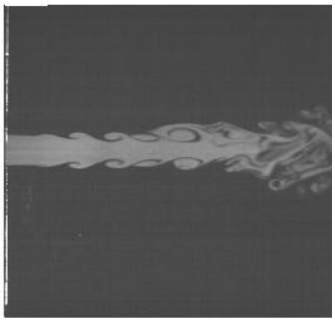


εικόνα 1.8

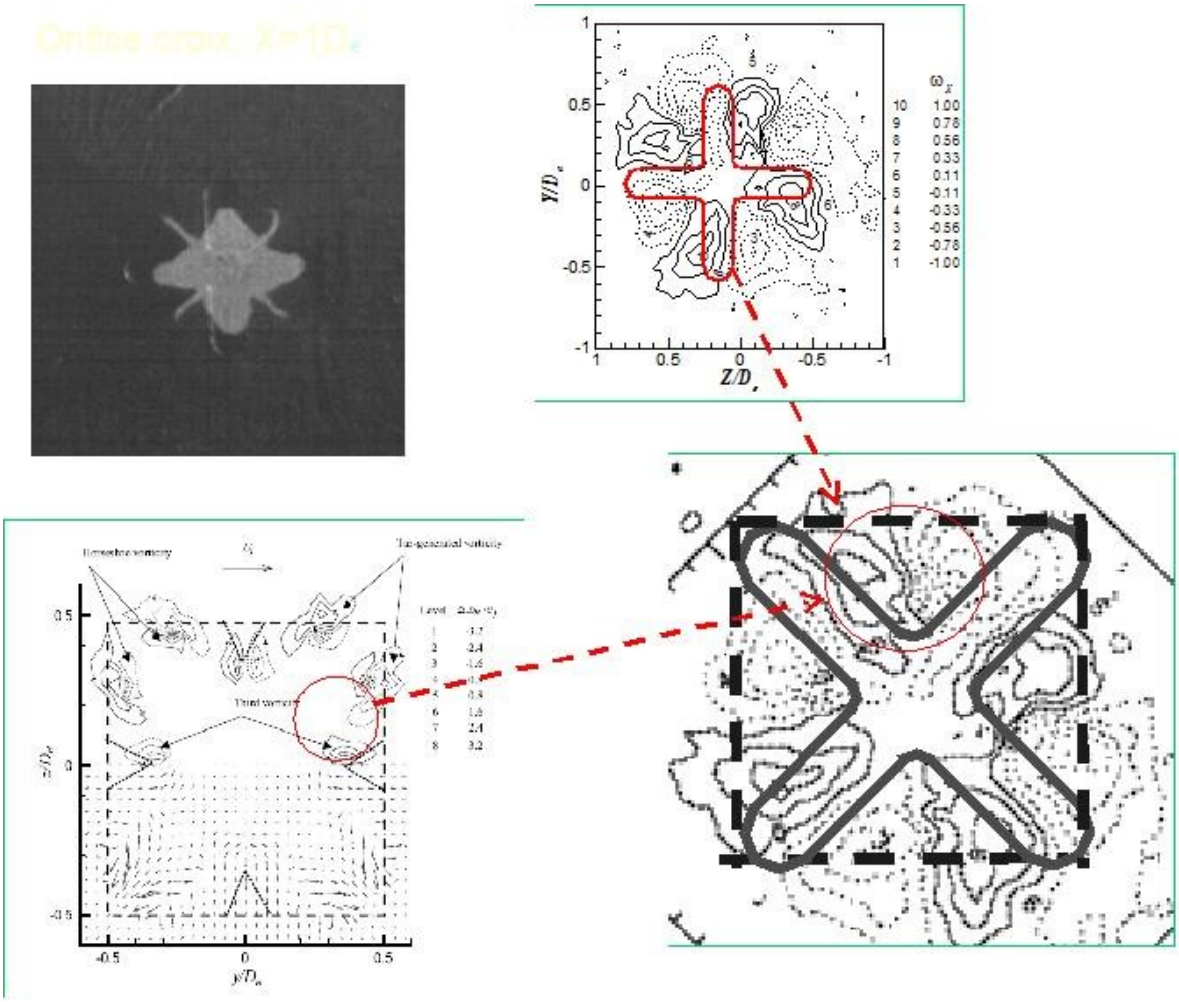
Επίσης υπάρχουν και οι φωτογραφίες στροβίλων σε κάθετη (εικόνα 1.9) και οριζόντια λήψη (εικόνα 1.10).



εικόνα 1.9



εικόνα 1.10



εικόνα 1.11

Στην παραπάνω εικόνα βλέπουμε τον τρόπο με τον οποίο δημιουργούντε οι στρόβιλοι γύρω απο το στόμιο του κλιματιστικού,στην συγκεκριμένη περίπτωση το στόμιο έχει σχήμα σταυρού.

2.OpenCV

Στο πρόγραμμα μας δημιουργήθηκε με την χρήση της C++ και της βιβλιοθήκης OpenCV, παρακάτω γίνεται αναφορά για την OpenCV.

2.1 Εισαγωγή στην OpenCV

Η OpenCV (Open Computer Vision) είναι μία υψηλού επιπέδου,βιβλιοθήκη τεχνητής όρασης,η απoία δημιουργήθηκε απο την Intel το 1999. Μετά απο κάποιες beta εκδόσεις που προυπήρξαν στο παρελθόν,η πρώτη έκδοση της OpenCV κυκλοφόρησε το 2006.Η βιβλιοθήκη αυτή είναι ανοιχτού κώδικα και είναι ελεύθερη για εκπαιδευτική αλλά και για εμπορική χρήση.Είναι γραμμένη κυρίως στην γλώσσα προγραμματισμού C και χρησιμοποιείται στις κυριότερες πλατφόρμες όπως Windows,Linux,Mac Os. Η OpenCV έχει ενσωματωμένους αλγορίθμους επεξεργασίας εικόνας και τεχνητής όρασης.

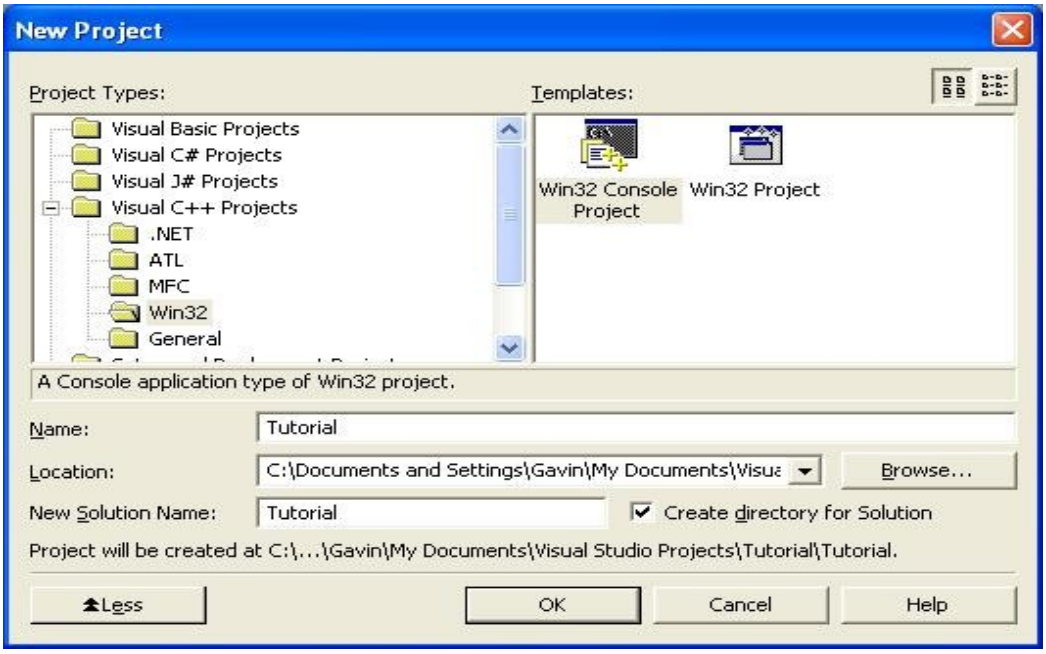
Μερικές απο τις εφαρμογές που καλύπτει η OpenCV είναι:

- Αναγνώρισης Προσώπου
- Αναγνώρισης Χειρονομιών
- Διεπαφής Χρήστη-Υπολογιστή
- Κινητά Ρομπότ
- Αναγνώρισης και Ταυτοποίησης αντικειμένων

Εγκατάσταση

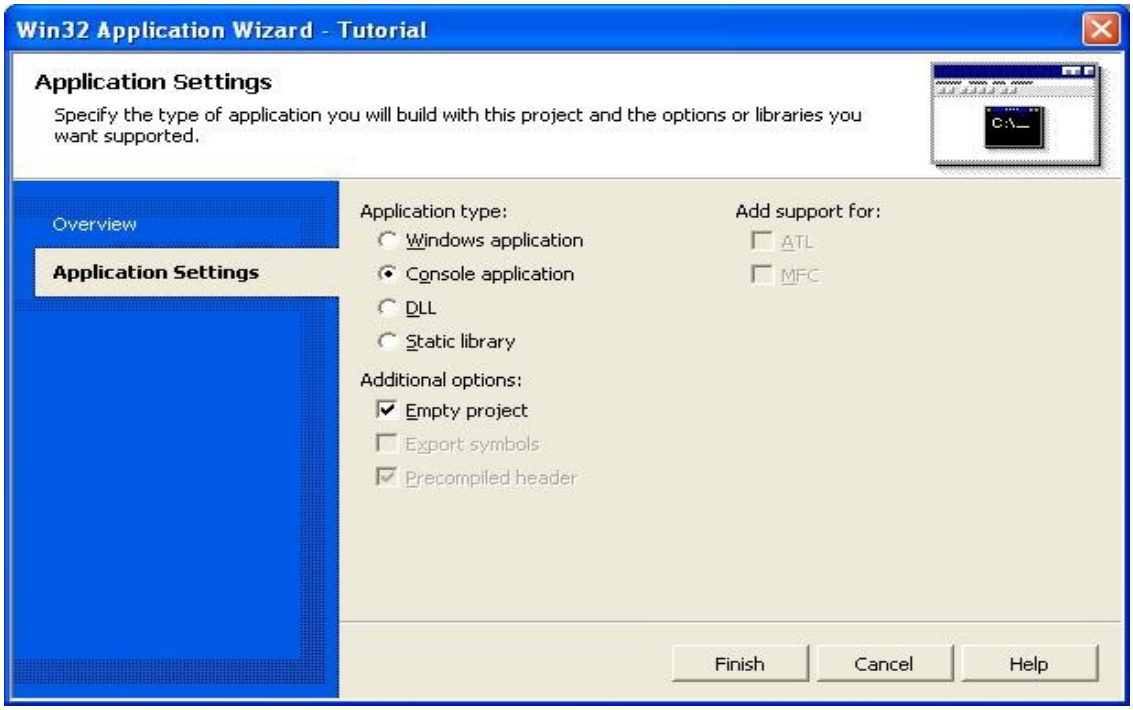
Η βιβλιοθήκη αυτή μπορεί να βρεθεί και να κατεβαστεί απο τον παρακάτω σύνδεσμο:
http://opencvlibrary.sourceforge.net/
και όταν κατεβαστέι απλά κάνουμε install το .exe αρχείο. Τα αρχεία αποθηκεύονται στα αρχεία εφαρμογών του σκληρού δίσκου και στην συνέχεια προσθέτουμε τις βιβλιοθήκες που χρειαζόμαστε στην C++. Παρακάτω θα δούμε βήμα βήμα τον τρόπο εγκατάστασης.

Αφού κάνουμε install το εκτελέσιμο αρχείο της OpenCV,ανοίγουμε το πρόγραμμα της C++. Δημιουργούμε ένα καινούργιο project **File->New->Project** και στην συνέχεια επιλέγουμε **Win23 Console Project,**η παρακάτω εικόνα(2.1) θα μας βοηθήσει να το καταλάβουμε καλύτερα.



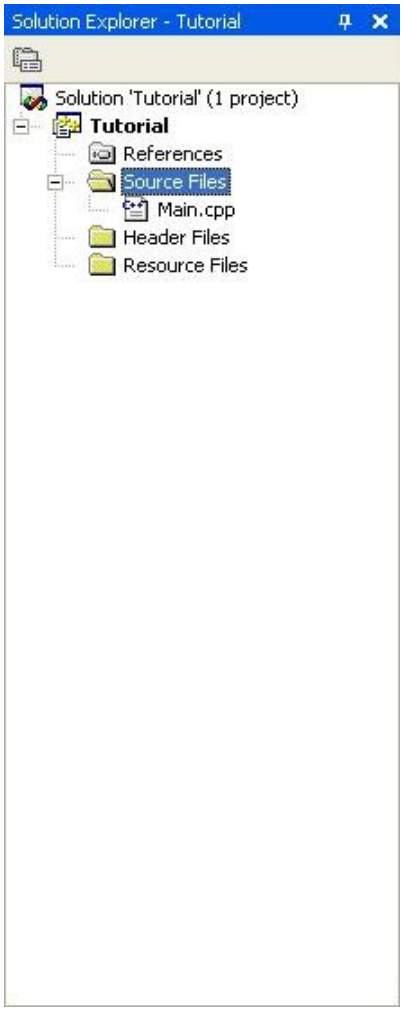
Εικόνα 2.1

Στην συνέχεια δημιουργούμε ένα άδειο project **Empty Project,** επιλέγοντας το κουτί κάτω απο το "application settings" όπως φαίνεται ξεκάθαρα στην παρακάτω εικόνα 2.2.

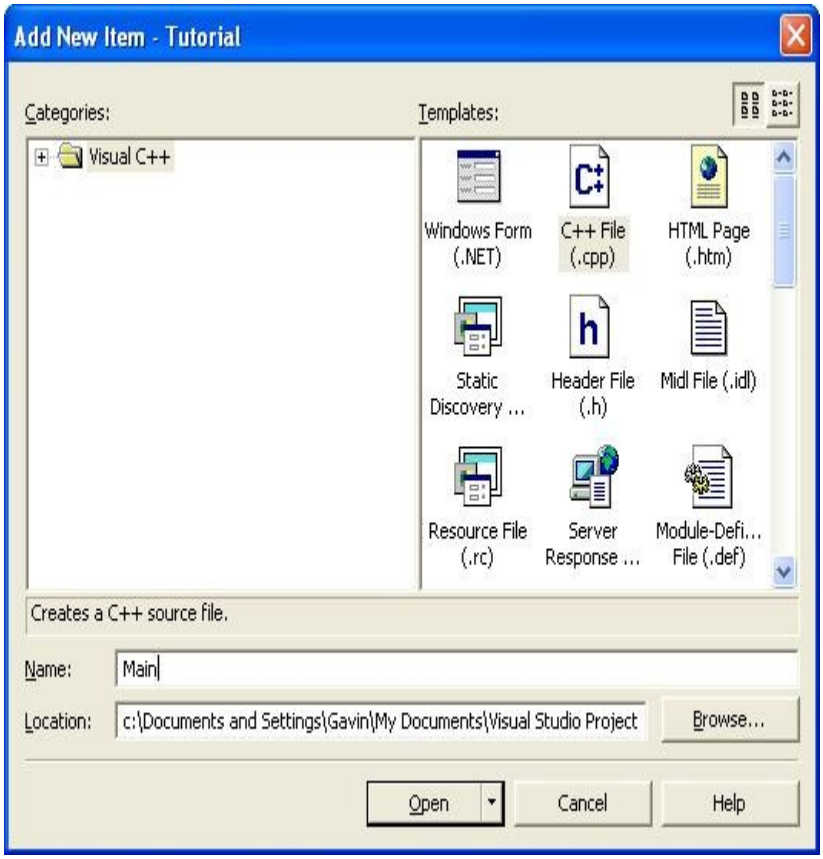


εικόνα 2.2

Στην συνέχεια κάνουμε δεξί click στο "**Source Files**" και στην συνέχεια κάνουμε **Add → Add new Item**(εικόνα 2.3). Μετά επιλέγουμε ένα αρχείο C++ και του δίνουμε ένα όνομα(εικόνα 2.4).

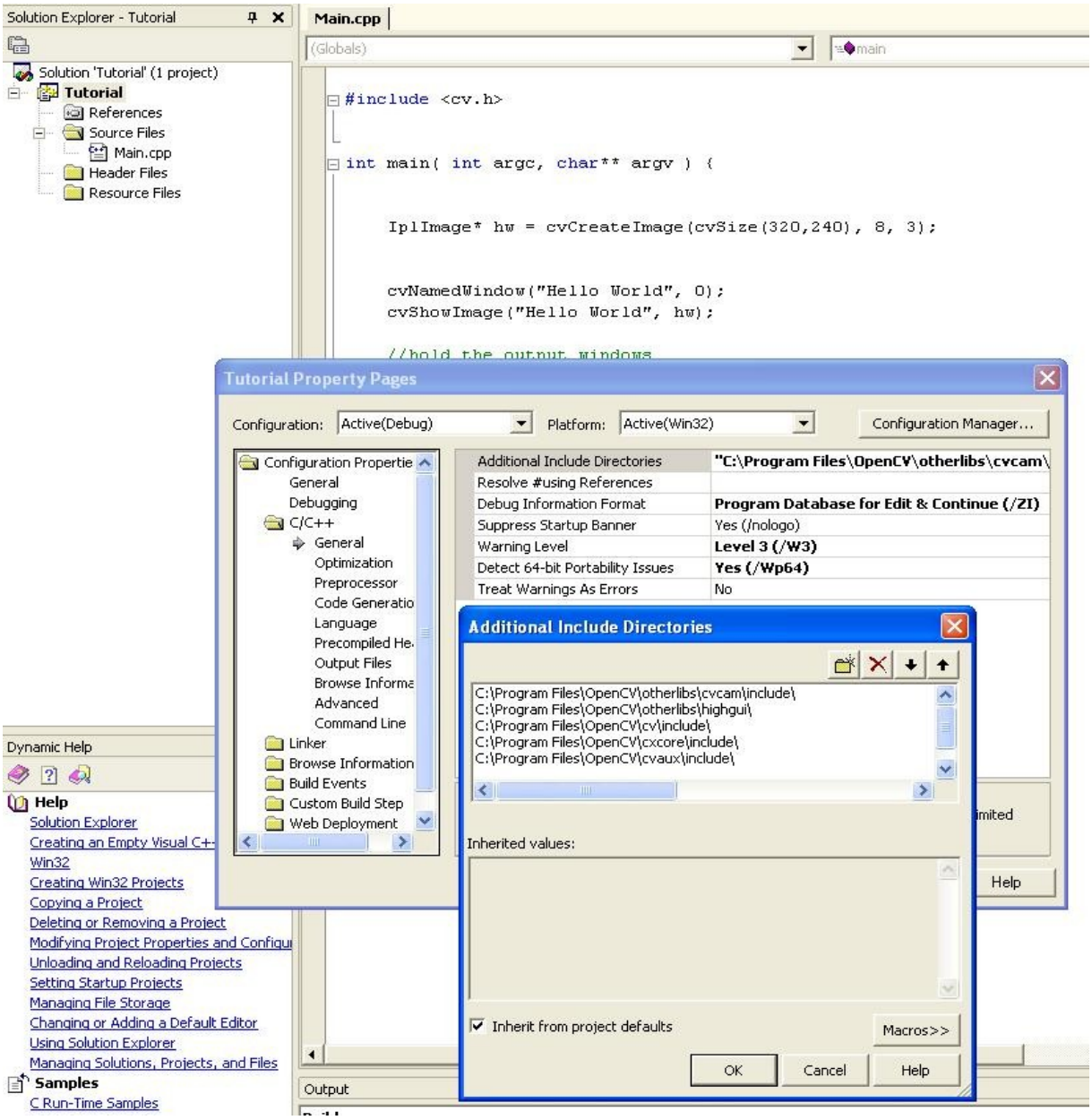


εικόνα 2.3



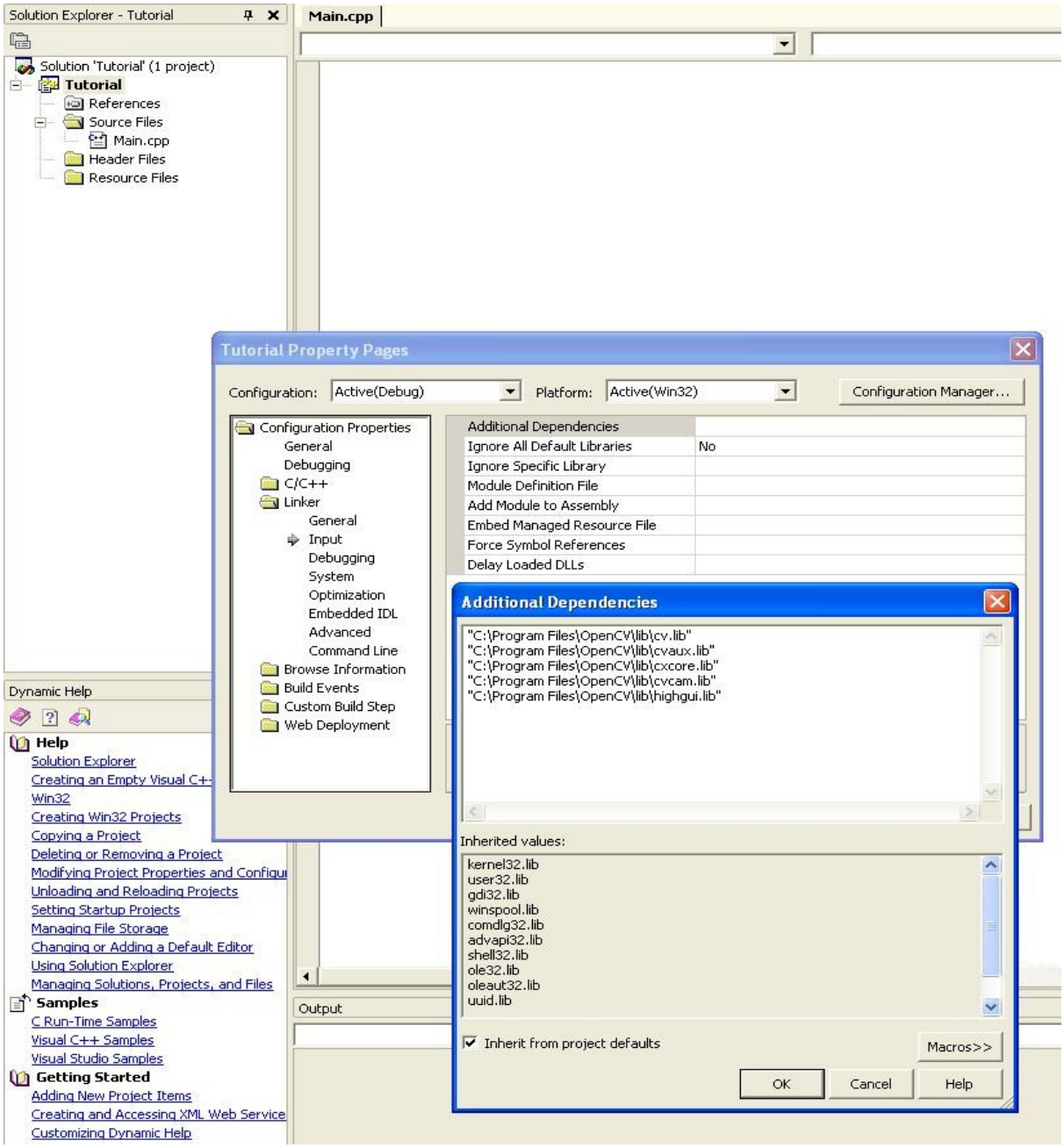
εικόνα 2.4

Μετά απο αυτό επιλέγουμε το **General** που βρίσκεται κάτω απο το tab C/C++ ,μετά επιλέγουμε το **Additional Include Directives** και στην συνέχεια προσθέτουμε ολόκληρο το path για κάθε φάκελο που περιέχει τα .h(headers) αρχεία του OpenCV.Στην παρακάτω εικόνα (2.5)φαίνεται αυτή η διαδικασία.


Solution Explorer - Tutorial
Solution 'Tutorial' (1 project)
Tutorial
References
Source Files
Main.cpp
Header Files
Resource Files
Main.cpp
(Globals)
main
#include <cv.h>
int main(int argc, char** argv) {
IplImage* hw = cvCreateImage(cvSize(320,240), 8, 3);
cvNamedWindow("Hello World", 0);
cvShowImage("Hello World", hw);
//hold the output windows
Tutorial Property Pages
Configuration: Active(Debug)
Platform: Active(Win32)
Configuration Manager...
Configuration Properties
General
Debugging
C/C++
General
Optimization
Preprocessor
Code Generation
Language
Precompiled Headers
Output Files
Browse Information
Advanced
Command Line
Linker
Browse Information
Build Events
Custom Build Step
Web Deployment
Additional Include Directories
"C:\Program Files\OpenCV\otherlibs\cvcam\
Resolve #using References
Debug Information Format
Program Database for Edit & Continue (/ZI)
Suppress Startup Banner
Yes (/nologo)
Warning Level
Level 3 (/W3)
Detect 64-bit Portability Issues
Yes (/Wp64)
Treat Warnings As Errors
No
Additional Include Directories
C:\Program Files\OpenCV\otherlibs\cvcam\include\
C:\Program Files\OpenCV\otherlibs\highgui\
C:\Program Files\OpenCV\cv\include\
C:\Program Files\OpenCV\cxcore\include\
C:\Program Files\OpenCV\cvaux\include\
Inherited values:
Inherit from project defaults
Macros>>
OK
Cancel
Help
Help
Dynamic Help
Help
Solution Explorer
Creating an Empty Visual C++ Win32
Creating Win32 Projects
Copying a Project
Deleting or Removing a Project
Modifying Project Properties and Configu
Unloading and Reloading Projects
Setting Startup Projects
Managing File Storage
Changing or Adding a Default Editor
Using Solution Explorer
Managing Solutions, Projects, and Files
Samples
C Run-Time Samples
Output

Εικόνα 2.5

Στην συνέχεια κάτω από το tab Linker επιλέγουμε το **Input,** μετά επιλέγουμετο **Additional Dependencies** και προσθέτουμε όλα τα .lib αρχεία που χρειάζονται για την OpenCV όπως δείχνει η παρακάτω εικόνα 2.6 και τελειώσαμε.



εικόνα 2.6

Προγραμματισμός με OpenCV

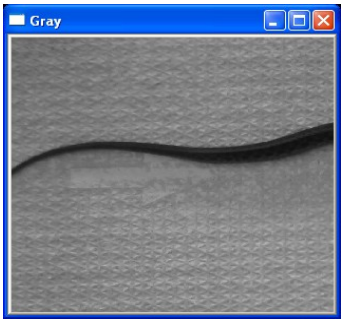
Η OpenCV όπως αναφέρθηκε και πιο πάνω,χρησιμοποιεί αλγορίθμους επεξεργασίας εικόνας και τεχνητής όρασης.Παρακάτω θα δούμε ένα απλό παράδειγμα με την χρήση της OpenCV,καθώς και μερικές βασικές εντολές της.

```
//Δημιουργία μιας εικόνας με ίδιες διαστάσεις με την κανονική
IplImage* grayImage = cvCreateImage(cvSize(im->width,im->height),
IPL_DEPTH_8U, 1);
//Μετέτρεψε την εικόνα στη κλιμακα του γκρι
cvCvtColor(im, grayImage, CV_BGR2GRAY);
//Δημιουργια μιας εικόνας που θα περέιχει το ιστόγραμμα
IplImage* histImage = cvCreateImage(cvSize(320,200), 8, 1);
//Δημιουργία ενός ιστογράμματος που θα περιέχει της πληροφορίες μιας εικόνας
CvHistogram* hist =
cvCreateHist(1, &hist_size, CV_HIST_ARRAY, ranges, 1);
//Υπολογισμός του ιστογράμματος
cvCalcHist( &grayImage, hist, 0, NULL );
//Κράτα τις μεγαλύτερες και μικρότερες τιμές
cvGetMinMaxHistValue( hist, &min_value, &max_value, &min_idx, &max_idx);
//Κάνε scale στις διαδικές τιμές ετσι ώστε να αντιπροσωποεύουν την εικόνα
cvScale( hist->bins, hist->bins, ((double)histImage->height)/max_value, 0 );
//Κάνε όλες τις τιμές του ιστογράμματος 255
cvSet( histImage, cvScalarAll(255), 0 );
//Δημιούργησε ένα παράγωντα για scaling κατά μήκος του βάθους
bin_w = cvRound((double)histImage->width/hist_size);
for( i = 0; i < hist_size; i++ ) {
//Σχεδίασε τα δεδομένα του ιστογράμματος στο ιστόγραμμα της εικόνας
cvRectangle( histImage, cvPoint(i*bin_w, histImage->height),
cvPoint((i+1)*bin_w,
histImage->height - cvRound(cvGetReal1D(hist->bins,i))),
cvScalarAll(0), -1, 8, 0 );
//Πάρε την τιμή από το προσορινό ιστόγραμμα
float* bins = cvGetHistValue_1D(hist,i);
//Αύξησε την ενδιάμεση τιμή
mean += bins[0];
}
//Τελείωσε τον υπολογισμο της ενδιάμεσης τιμής
mean /= hist_size;
//Τελείωσε τον υπολογισμό των μεταβλητών
variance /= hist_size;
```

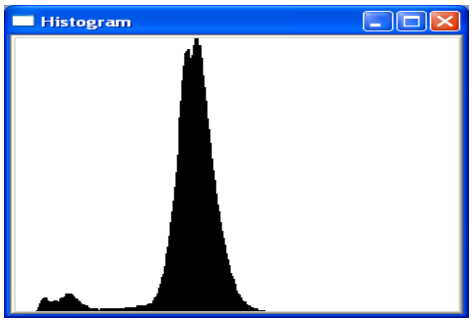
```
std::cout << "Histogram Mean: " << mean << std::endl;
std::cout << "Variance: " << variance << std::endl;
std::cout << "Standard Deviation: " << sqrt(variance) << std::endl;
//Εμφάνισε τις τρεις εικόνες
cvNamedWindow("Original", 0);
cvShowImage("Original", im );
cvNamedWindow("Gray", 0);
cvShowImage("Gray", grayImage );
cvNamedWindow("Histogram", 0);
cvShowImage("Histogram", histImage );

//Κράτα τις εικόνες μέχρι να πατηθεί ένα κουμπί
cvWaitKey(0);

//Αποδέσμευσε την μνήμη για της εικόνες
cvReleaseImage(&histImage);
cvReleaseImage(&grayImage);
cvReleaseImage(&im);

//Κλείσε τα παράθυρα στα οποία εμφανίζονται οι οθόνες
cvDestroyWindow("Original");
cvDestroyWindow("Gray");
cvDestroyWindow("Histogram");
```

Τρέχοντας αυτό τον κώδικα θα έχουμε το παρακάτω αποτέλεσμα,το οποίο περιλαμβάνει τρεις εικόνες,την κανονική(2.7),την κανονική σε κλίμακα του γκρι(2.8) και το ιστόγραμμα(2.9).

Εικόνα 2.7



Εικόνα 2.8



Εικόνα 2.9

Βασικές Εντολές OpenCV:

- cvLoadImage("όνομα εικόνας.jpg",1); -Φορτώνει μία εικόνα
- IplImage * -Δεσμεύει χώρο για μια εικόνα
- cvCreateImage(διαστάσεις,βάθος χρώματος); -Δημιουργεί μία εικόνα
- cvSize(διαστάσεις) -Ορίζει τις διαστάσεις μίας εικόνας
- cvGetSize(εικόνα) -Δέχεται τις διαστάσεις μιας εικόνας
- cvCvtColour (εικόνα1,εικόνα2,χρώμα); -Μετατρέπει το χρώμα τις μιας εικόνας στο χρώμα που επιλέγουμε εμείς
- CV_BGR2GRAY -Μετατρέπει από RGB κλίμακα σε grayscale κλίμακα.
- CvNamedWindow(όνομα); -Ονομάζει το πλαίσιο όπου βρίσκεται η εικόνα.
- cvShowImage("όνομα",όνομα εικόνας); -Εμφανίζει μία εικόνα
- cvReleaseImage(& όνομα εικόνας); -Αποδεσμεύει χώρο για μία εικόνα
- cvWaitKey(0); - Κρατάει την εικόνα μέχρι να πατήσουμε ένα κουμπί
- cvDestroyWindow("όνομα"); -Κλείνει το παράθυρο της εικόνας.
- CvThreshold(όνομα εικόνας,κατώφλι,) -Κατωφλιοποιεί μια εικόνα
- cvCreateMemStorage(); -Δεσμεύει χώρο στην μνήμη
- cvCreateHist(μέγεθος,πίνακας ιστ.); -Δημιουργεί ένα ιστόγραμμα
- cvDrawContours(εικόνα,χρώμα); -Σχεδιάζει μια ακμή
- cvCircle(εικόνα,κέντρο,χρώμα); -Σχεδιάζει ένα κύκλο
- cvLine(εικόνα,σημείο.χρώμα); -Σχεδιάζει μία γραμμή

Περισσότερες εντολές μπορούμε να βρούμε στον παρακάτω σύνδεσμο:
http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_BasicFuncs.htm

3.Δημιουργία προγράμματος

3.1 Εισαγωγή

Σκοπός μας είναι να κατασκευάσουμε ένα πρόγραμμα με την χρήση της γλώσσας προγραμματισμού C++ και της βιβλιοθήκης OpenCV το οποίο θα έχει τις εξής δυνατότητες:

1. Ανοιγμα μιας εικόνας ή μιας ακολουθιάς από εικόνες
2. Εφαρμογή της τεχνικής του κατωφλιού(thresholding)
3. Υπολογισμός περιμέτρου των στροβίλων
4. Υπολογισμός κέντρου του στροβίλου
5. Υπολογισμός ακτίνας του στροβίλου
6. Δημιουργια ενός txt αρχείου που θα περιέχει χρήσιμες πληροφορίες για τους στροβίλους

Για να το πετύχουμε αυτό θα χωρίσουμε το πρόγραμμα μας σε ξεχωριστά βήματα. Στο κάθε βήμα θα δημιουργούμε ένα ξεχωριστό βήμα του προγράμματος μας.

Στα παρακάτω βήματα βρίσκεται μέρος του κώδικα που χρησιμοποιήσαμε,όλος ο κώδικας καθώς και οι εξωτερικές συναρτήσεις που χρησιμοποιήσαμε μαζί με τα header file τους,βρίσκονται στο παράρτημα.

3.2 Άνοιγμα και δημιουγία μιας εικόνας

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
```

Οι βιβλιοθήκες που χρησιμοποιούμε είναι οι παραπάνω. Η <stdio.h> και <math.h> είναι βιβλιοθήκες της C++ και οι υπόλοιπες είναι της OpenCV.

```
int main ()

{
        IplImage* newImg;
        IplImage* grayImg;
        IplImage* cunnyImg;
```

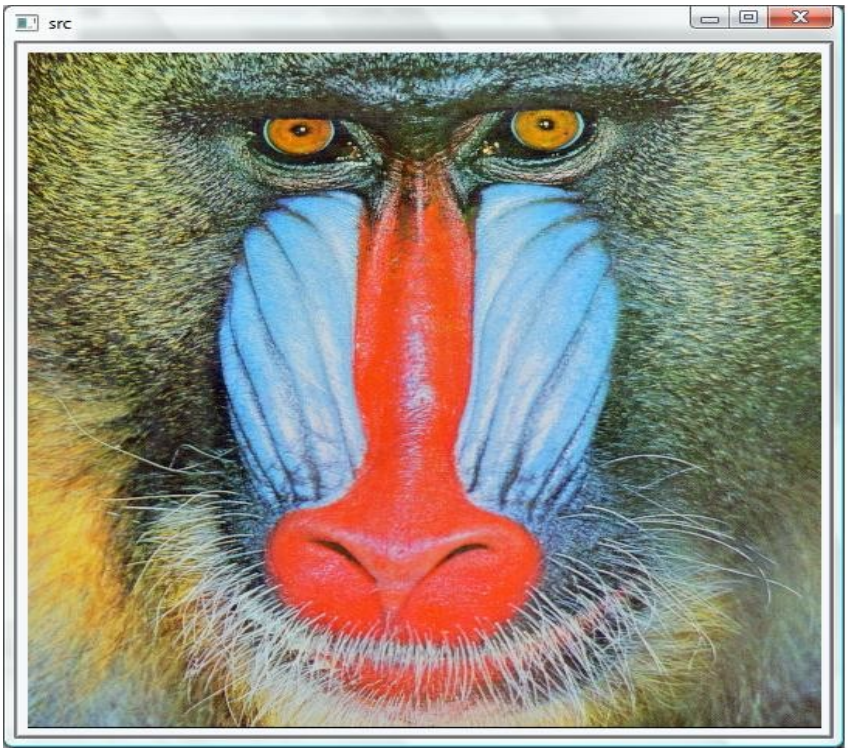
Εδώ δεσμεύουμε χώρο στην μνήμη για να μπορέσουμε να φορτώσουμε τις εικόνες που θέλουμε,στην συγκεκριμένη περίπτωση δεσμεύουμε χώρο για τρείς εικόνεις, την newImg,grayImg,cunnyImg.
Γενικά στην OpenCV όποτε θέλουμε να δεσμεύσουμε χώρο στην μνήμη για μία εικόνα χρησιμοποιούμε την εντολή IplImage.

```
newImg=cvLoadImage("baboon.jpg",1);
grayImg=cvCreateImage( cvSize(newImg->width,newImg->height),IPL_DEPTH_8U,1);
cunnyImg=cvCreateImage(cvGetSize(newImg), IPL_DEPTH_8U,1);
cvCvtColor (newImg,grayImg,CV_BGR2GRAY);
```

```
cvNamedWindow("src",1);
        cvNamedWindow ("cunny",1);
         cvShowImage( "src" , newImg);
        cvShowImage( "cunny", cunnyImg);

        cvReleaseImage( &newImg);
        cvReleaseImage( &grayImg);
        cvReleaseImage( &cunnyImg);

        cvWaitKey(0);

        cvDestroyWindow("src");
        cvDestroyWindow("cunny");


        return 0;

}
```

Στον παραπάνω κώδικα, δημιουργούμε τις εικόνες, ονομάζουμε τα πλαίσια στα οποία θα εμφανίζονται οι εικόνες,όπως επίσης αποδεσμεύουμε τον χώρο τον οποίο είχαμε χρησιμοποιήσει για να φορτώσουμε τις εικόνες και στο τέλος κλείνουμε τα παράθυρα.

Καθώς τρέχουμε το πρόγραμμα,μας ανοίγει την εικόνα που επιλέξαμε,στην συγκεκριμένη περίπτωση επιλέξαμε μια μη-σχετική με το πείραμα εικόνα(3.1).



Εικόνα 3.1

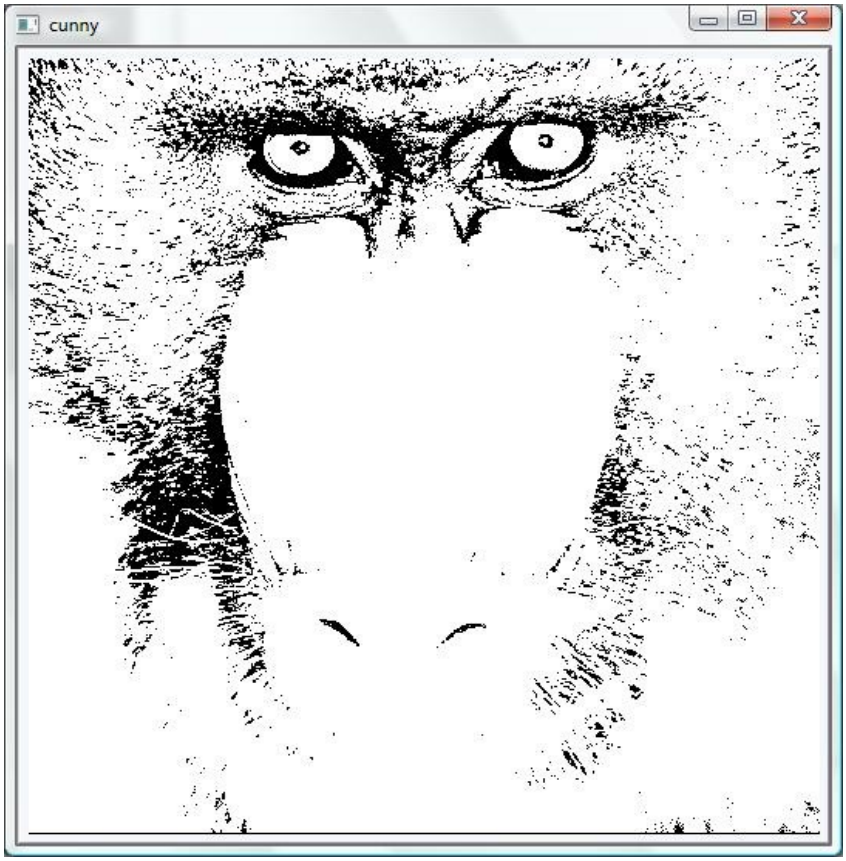
3.3 Κατωφλιοποίηση μιας εικόνας

Εάν στον προηγούμενο κώδικα γίνει πρόσθεση της παρακάτω γραμμής κώδικα τότε θα έχουμε πετύχει την κατωφλιοποίηση της εικόνας που επιλέξαμε.

```
CvThreshold(grayImg,cunnyImg,70,255,CV_THRESH_BINARY);
```

Αυτό γίνεται με την εντολή CvThreshold η οποία δέχεται ως ορίσματα τις εικόνες στις οποίες θέλουμε να εφαρμόσουμε την τεχνική του κατωφλίου, ορίζουμε το κατώφλι (π.χ 70 στην συγκεκριμένη περίπτωση).Με την εντολή CV_THRESH_BINARY δηλώνουμε οτι θα μας εμφανίσει την εικόνα σε χρώμα δύο καναλιών,μαύρο και άσπρο.

Τρέχοντας το πρόγραμμα τώρα μας εμφανίζει την κατωφλιοποιημένη εικόνα(3.2). Σε περίπτωση που δεν μας καλύπτει το αποτέλεσμα,μπορούμε να αλλάξουμε την τιμή του κατωφλιού μέχρι να επιτύχουμε το καλύτερο δυνατό αποτέλεσμα.



Εικόνα 3.2

3.4 Εισαγωγή μιας ακολουθίας από εικόνες

Μέχρι τώρα το πρόγραμμα άνοιγε μια μόνο εικόνα κάθε φορά που το τρέχαμε,αλλά αυτό δεν είναι βολικό για εμάς,επειδή θέλουμε το πρόγραμμα μας να επεξεργάζεται ακολουθίες από εικόνες.
Σε αυτό το βήμα σκοπός μας είναι να κάνουμε το πρόγραμμα να ανοίγει μια ακολουθία από εικόνες. Για να το πετύχουμε αυτό δημιουργούμε μία εξωτερική συνάρτηση την input.cpp η οποία θα κάνει αυτή ακριβώς την δουλεία.

```
#include "input.h"
#include <stdio.h>
#include <stdlib.h>
```

```
string input::readName()
{
        char s[1000];
        scanf("%s",s);
        string strS = s;
        return strS;
}

string input::getImageFromSequence(string seqName, int seqNr)
{
        string s = seqName;
        std::stringstream strm;
        strm << seqNr;
        string nr = strm.str();//the number of the image as a string
        int PointPosition = s.find(".");
        int nr_len = nr.length();
        s.replace(PointPosition - nr_len, nr_len, nr);

        return s; }
```

Όπως βλέπουμε και στον κώδικα η συνάρτηση αυτή διαβάζει το όνομα της εικόνας που πρόκειται να φορτωθεί και στην συνέχεια φορτώνει την εικόνα αυτή αλλά και τον αριθμό όλων των υπολοίπων που βρίσκονται στον ίδιο φάκελο.

Μετά την δημιουργία αυτής της συνάρτησης,γίνεται και η κλήση της στο πρόγραμμα με σκοπό να φορτώσουμε μια ακολουθία από εικόνες.

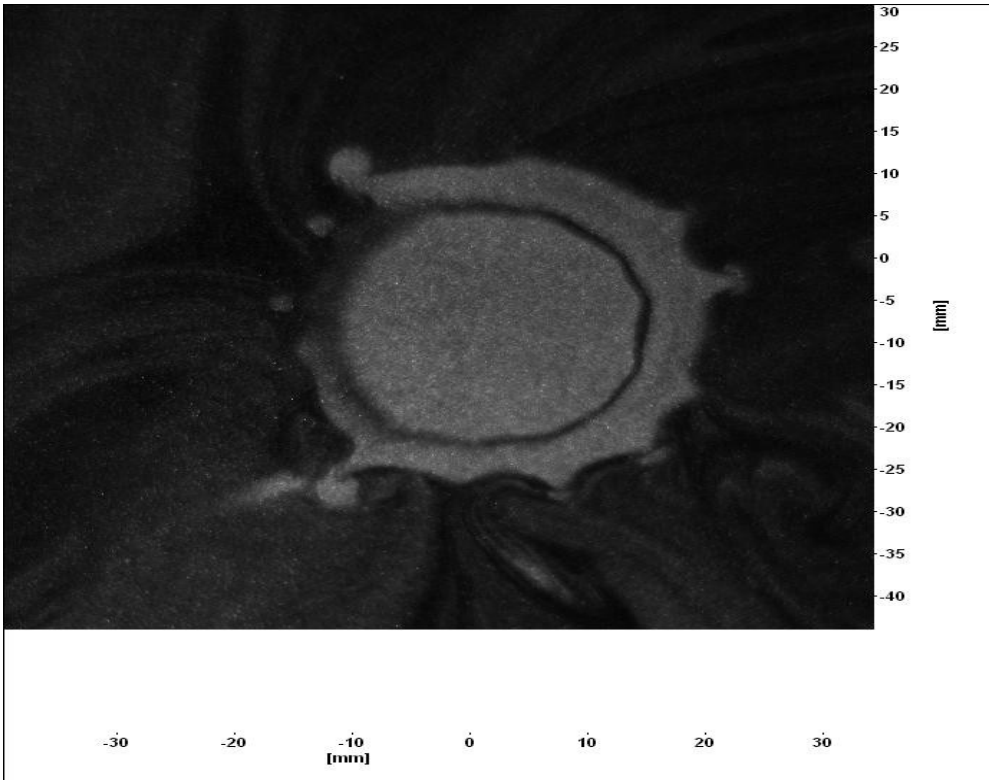
```
CvMemStorage *SpaceMemory = cvCreateMemStorage();
CvSeq *Sequence;
CvSeq *Debut;

for (i=1;i<400;i++)
   {
        input ob;
        string s;

        s=ob.getImageFromSequence("C:\\Users\\user\\Desktop\\Orifice circulaire\\JC
        X=50mm \\B00001.bmp",i);

          const char *image=s.c_str();
          sprintf(file,image);
}
```

Στην for δηλώνουμε πόσες εικόνες θέλουμε να φορτώσει η ακολουθία και μέσα στην getImageFromSequence ορίζουμε το path από το οποίο θα φορτώνει της εικόνες. Η cvCreateMemStorage() μας βοηθάει να δεσμεύσουμε μνήμη για την ακολουθία των εικόνων.

3.5 Υπολογισμός περιμέτρου

Μέχρι στιγμής έχουμε ένα πρόγραμμα το οποίο φορτώνει μια ακολουθία από εικόνες και εφαρμόζει σε αυτές την τεχνική του κατωφλιού. Σε αυτό το βήμα θα προσπαθήσουμε να υπολογίσουμε την περίμετρο ενός σημείου μιας εικόνας,στην συγκεκριμένη περίπτωση θα υπολογίσουμε την περίμετρο του στροβίλου.

Στον υπολογισμό των στροβίλων συναντήσαμε κάποιες δυσκολίες,λόγω της περιττής πληροφορίας,που πρώτον μας ήταν άχρηστη και δεύτερον δημιουργούσε προβλήματα στον υπολογισμό της περιμέτρου. Αυτό φαίνεται ξεκάθαρα στην επόμενη εικόνα(3.3).



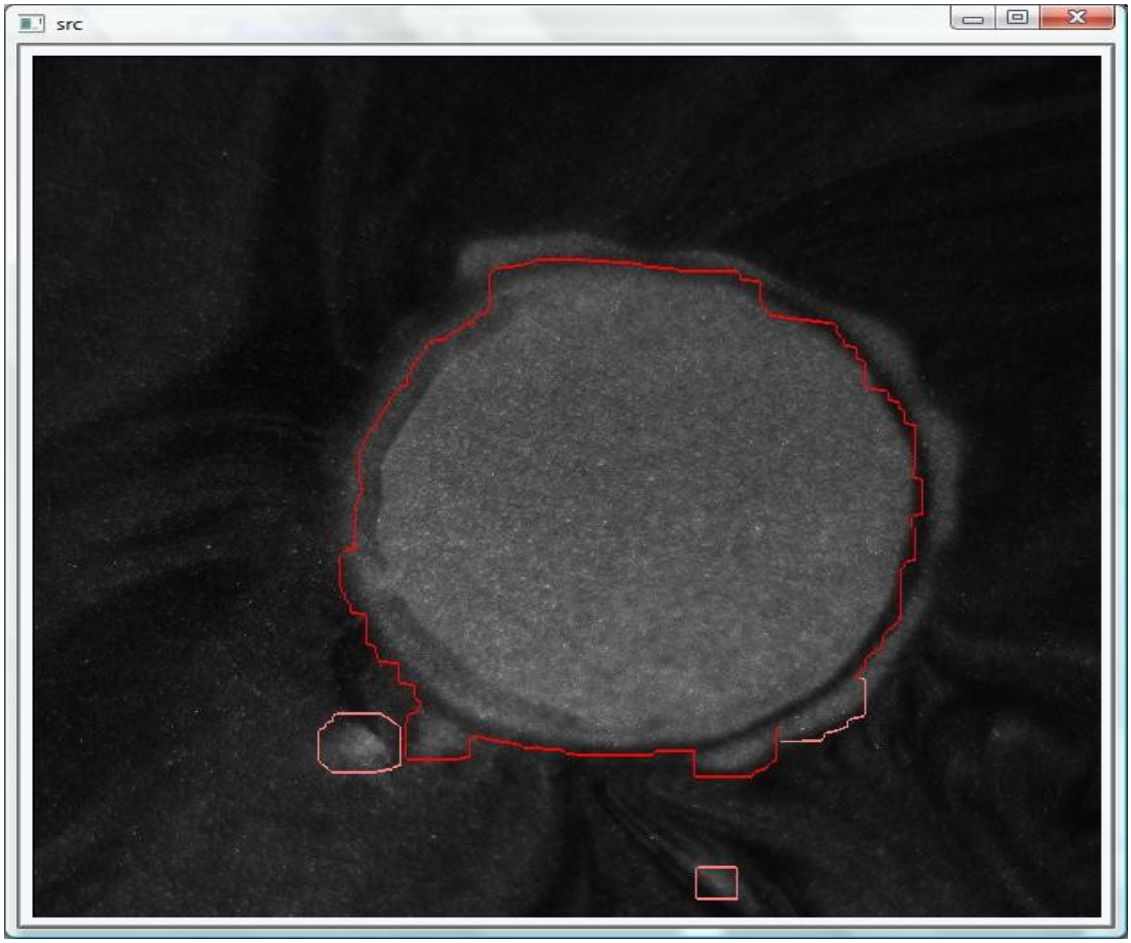
Εικόνα 3.3

```
cvErode( segmentedImg, segmentedImg, 0, 1 );
cvDilate( segmentedImg, segmentedImg, 0, 10);
cvCanny(segmentedImg,contourImg,50,150,3);
cvFindContours(contourImg, SpaceMemory, &Sequence, sizeof(CvContour),
CV_RETR_LIST , CV_LINK_RUNS);
```

Με τις παράπανω εντολές ξεπερνάμε τις δυσκολίες καθώς διαστέλουμε(cvDilate) και διαβρώνουμε(cvErode) τις εικόνες με σκοπό να απομακρύνουμε την περιττή πληροφορία και στην συνέχεια με την εντολή της OpenCV cvFindContours υπολογίζουμε την περίμετρο.

Στην συνέχεια το εφαρμόζουμε για όλη την ακολουθία όπως βλέπουμε στον πίνακα που ακολουθεί.

```
Debut=Sequence;
        for( ; Sequence != 0; Sequence = Sequence->h_next)
                        {
                                Long = cvArcLength(Sequence,CV_WHOLE_SEQ,1);
                                re++;
                                if(Long > LongMax)
                                {
                                        LongMax = Long;
                                }
                        }
        Sequence=Debut;

                        for( ; Sequence != 0; Sequence = Sequence->h_next)
                        {
                        CvScalar color;
                                if (LongMax == cvArcLength(Sequence,CV_WHOLE_SEQ,1))
                                {
                                        color = CV_RGB( 255, 0, 0 );
                                }
                                else
                                        color = CV_RGB(255,127,127);
                                cvDrawContours( contourImg, Sequence,
                                        color, color, 255, 1 , 8 );
                                cvDrawContours( newImg, Sequence, color,
                                        color, 255, 1 , 8 );
                                cvDrawContours( edgeImg, Sequence, color,
                                        color, 255, 1 , 8 );
                        }
        Sequence=Debut;
```



Εικόνα 3.4

Εδώ (εικόνα 3.4) βλέπουμε την περίμετρο του στροβίλου αλλά υπάρχουν ακόμη σφάλματα όπως μερικές άχρηστες πληροφορίες που δεν χρειαζόμαστε.

3.6 Εντοπισμός του κέντρου.

Επόμενο μας βήμα είναι ο υπολογισμός του κέντρου του στροβίλου.Σε αυτό το βήμα συναντήσαμε τα περισσότερα προβλήματα και δοκιμάσαμε πολλούς διαφορετικούς τρόπους.
Έπρεπε ο αλγόριθμος που θα βρούμε να δουλεύει για όλους τους τύπους στροβίλων,όχι μόνο των κυκλικών.Έτσι καταλήξαμε ότι ο καλύτερος τρόπος είναι να υπολογίσουμε τον αριθμό των pixel από το ένα ακρότατο μέχρι το άλλο,δηλαδή των αριθμό των pixels από το άκρο αριστερό pixel μέχρι το άκρο δεξιό pixel και απο το ανώτατο pixel μέχρι το κατώτατο pixel. Αφού το κάνουμε αυτό παίρνουμε την μέση τιμή των pixel η οποία είναι και το σημείο τομής των pixel και έτσι βρίσκουμε το κέντρο του στροβίλου.
Η παρακάτω συνάρτηση TrouverCentreG υπολογίζει το κέντρο του στροβίλου,για οριζόντιους στροβίλους.

```
void TrouverCentreG(CvSeq *seq,int *x, int *y)
{
	CvPoint *point;
	long sommex = 0;
	long sommey = 0;
	for(int i = 0; i < seq->total; i++)
	{
		point = (CvPoint *)(cvGetSeqElem(seq, i));
		sommex += point->x;
		sommey += point->y;
	}
	*x = sommex / seq->total;
	*y = sommey / seq->total;
}
```

Επίσης χρησιμοποιούμε την συνάρτηση resizeImage με σκοπό να απομακρύνουμε το πλαίσιο της εικόνας το οποίο δεν μας χρειάζεται για την επεξεργασία.

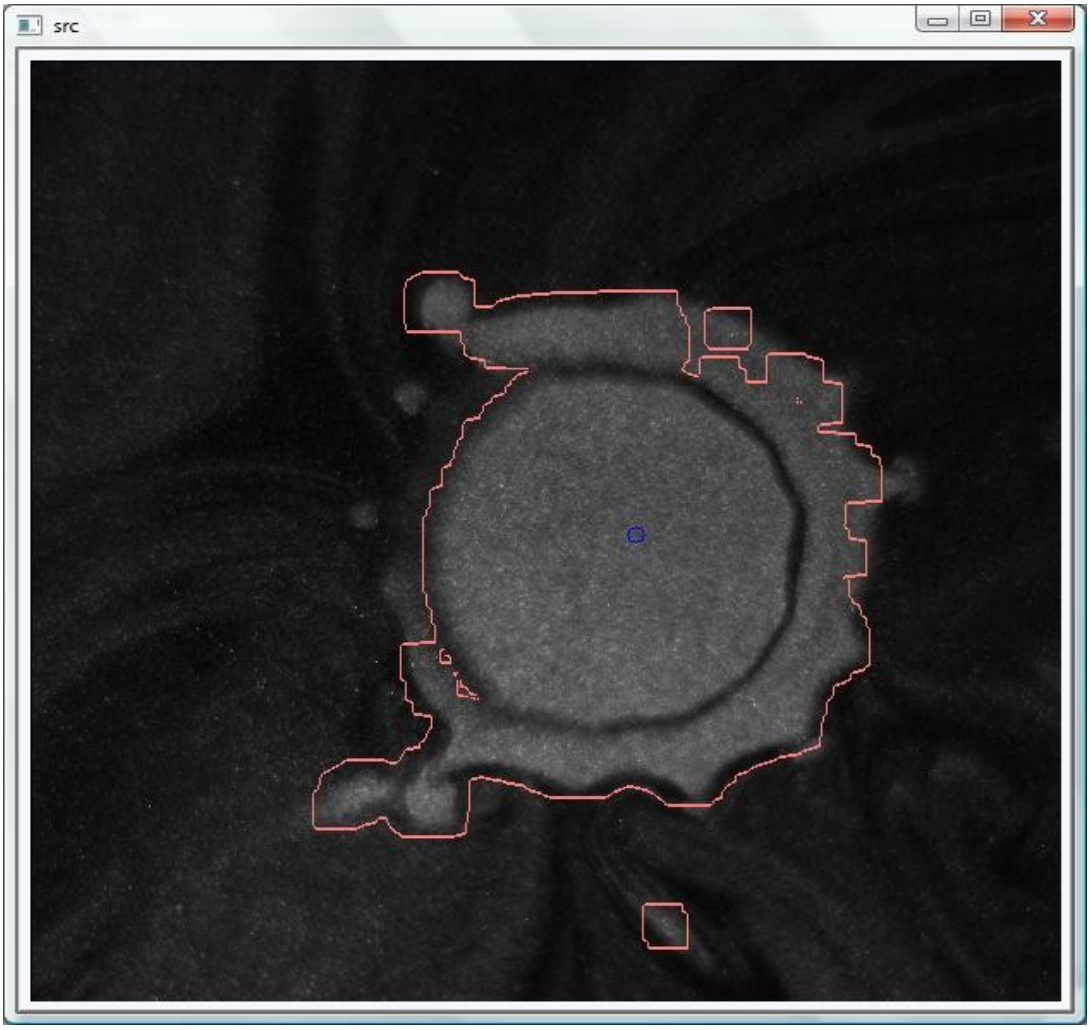
```
void resizeImage(IplImage * newImg,IplImage * newnewImg)
{
int i,j;
int shift=60;
unsigned char *pixelcourant;
	cpix p;

	pixelcourant= (unsigned char *)malloc(1*sizeof(unsigned char ) );
```

```
        for (i=shift;i<newImg->width;i++)
                for (j=shift; j<newImg->height; j++ )
                {

p.GetGrayPixel(newImg,i,j,pixelcourant);
p.PutGrayPixel(newnewImg,i-shift,j-shift,*pixelcourant);

                }
                free (pixelcourant);
}
```

Μετά εισάγουμε στο πρόγραμμα τις συναρτήσεις αυτές, εάν ο στρόβιλος είναι οριζόντιος τότε θα χρησιμοποιεί την συνάρτηση TrouverCentreG ενώ αν είναι κάθετος τότε θα χρησιμοποιεί την συνάρτηση TrouverCentre1. Με την χρήση της εντολής CvPoint center; δηλώνουμε ότι τα αποτελέσματα που θα βρεί είναι σημεία ενος κέντρου.

```
CvPoint center;

 if(choix == IMG_TRAN)
                                        {
                                                TrouverCentreG(Sequence,&CgX,&CgY);
                                        }


TrouverCentre1(newImg,&cX,&cY,CgX,CgY);

center.x=cX;
center.y=cY;

cvCircle( newImg,center,5, CV_RGB(0,0,255),1 );
```

Στην παρακάτω εικόνα (3.6) βλέπουμε τον εντοπισμό του κέντρου.



Εικόνα 3.6

3.7 Υπολογισμός ακτίνας

Αφού υπολογίσαμε το κέντρο του στροβίλου,επόμενος μας στόχος είναι να υπολογίσουμε και την ακτίνα του, για να το πετύχουμε αυτό υπολογίζουμε την απόσταση του κέντρου με το άκρο δεξιό pixel,δηλαδή αφαιρούμε τη τιμή που έχει το pixel στο κέντρο με αυτή που έχει το πιο δεξιό pixel. Αυτό γίνεται κυρίως με την βοήθεια των συναρτήσεων που θα δούμε παρακάτω.

```
void SearchCX(int *xw,int *yw,int CX,int *xmin,int *xmax)
 {
          int i,j;
          int *debutx;
          int *debuty;
                  debutx=xw;
                  debuty=yw;
              *xmin=999;
              *xmax=0;

         while(*yw!=-1)
                  {
                  if (*yw==CX)
                  {
                  if(*xw<*xmin)
                           {
                           *xmin=*xw;
                           }
                  if(*xw>*xmax)
                           {
                           *xmax=*xw;
                           }
                  }
                  xw++;
                  yw++;
         }
         xw=debutx;
         yw=debuty;
}
```

```
void initilization(IplImage* newImg, int *xw, int *yw)
         {
                  int i;

                  for(i=0; i< newImg->width*newImg->height;i++)
```

```
                {
                        *(xw+i)=-1;
                        *(yw+i)=-1;

                }
        }
```

```
void Searchmin(int *xw, int *yw,int *k)
{

        int min=9999;
        int kmin=0;
        int *debutx;
        debutx=xw;
   *k=0;
        while(*xw!=-1)
                {
                if(*xw<min)
                        {
                        min=*xw;
                        kmin=*k;
                        }
                (*k)++;
                xw++;
        }
        *k=kmin;
        xw=debutx;
}
```

```
void searchWhitePixel(IplImage *newImg,int *xw,int *yw)
{
        int i,j;
        int *debutx;
        int *debuty;
        int k=0;

unsigned char *pixelcourant;
        cpix p;
        pixelcourant= (unsigned char *)malloc(1*sizeof(unsigned char ) );
```

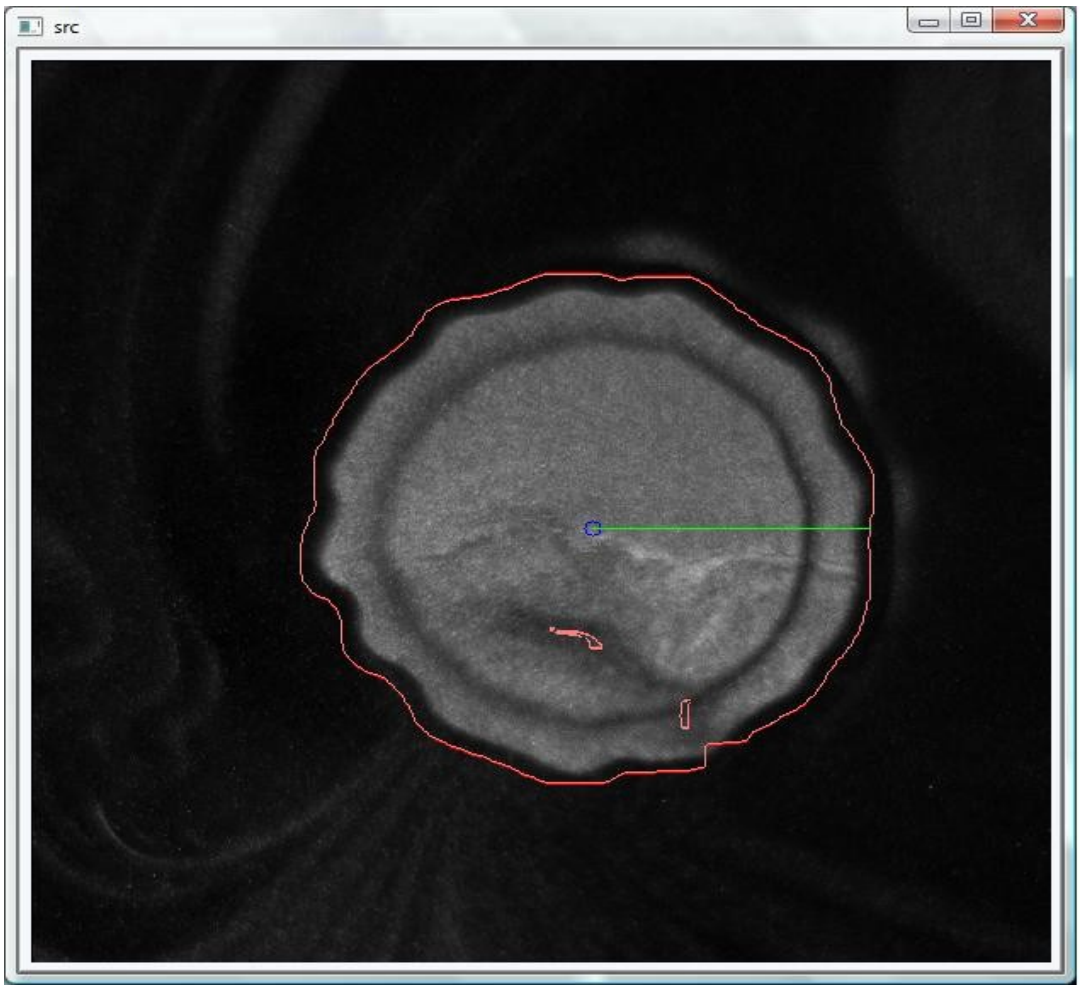
```
        for (i=0;i<newImg->width;i++)
                for (j=0; j<newImg->height; j++ )
                {

p.GetGrayPixel(newImg,i,j,pixelcourant);

                        if ((int)(*pixelcourant) !=0 ){
                                *(xw+k)=i;
                                *(yw+k)=j;
                                k++;
                        }
        }

}
```

```
void Searchmax(int *xw, int *yw,int *k)
{

        int max=0;
        int kmax=0;
        int *debutx;
        debutx=xw;
   *k=0;
        while(*xw!=-1)
                {

                if(*xw>max)
                        {
                        max=*xw;
                        kmax=*k;
                        }
                (*k)++;
                xw++;
        }

        *k=kmax;
        xw=debutx;
}
```

Μετά εισάγουμε στον κώδικα μας την συνάρτηση αυτή και έτσι υπολογίζουμε την ακτίνα του

στροβίλου.Στην εικόνα 3.7 βλέπουμε την ακτίνα.

```
searchWhitePixel(edgeImg,x1w,y1w);
        SearchCX(xw,yw,cY,ymin1,ymax1) ;
        pt1.x=*ymax1;
        pt1.y=cY;
        pt2.x=cX;
        pt2.y=cY;
        printf("%d,%d " ,pt1.x,pt1.y);
        cvLine (edgeImg,  pt1, pt2, CV_RGB (0, 255,255),1);
```

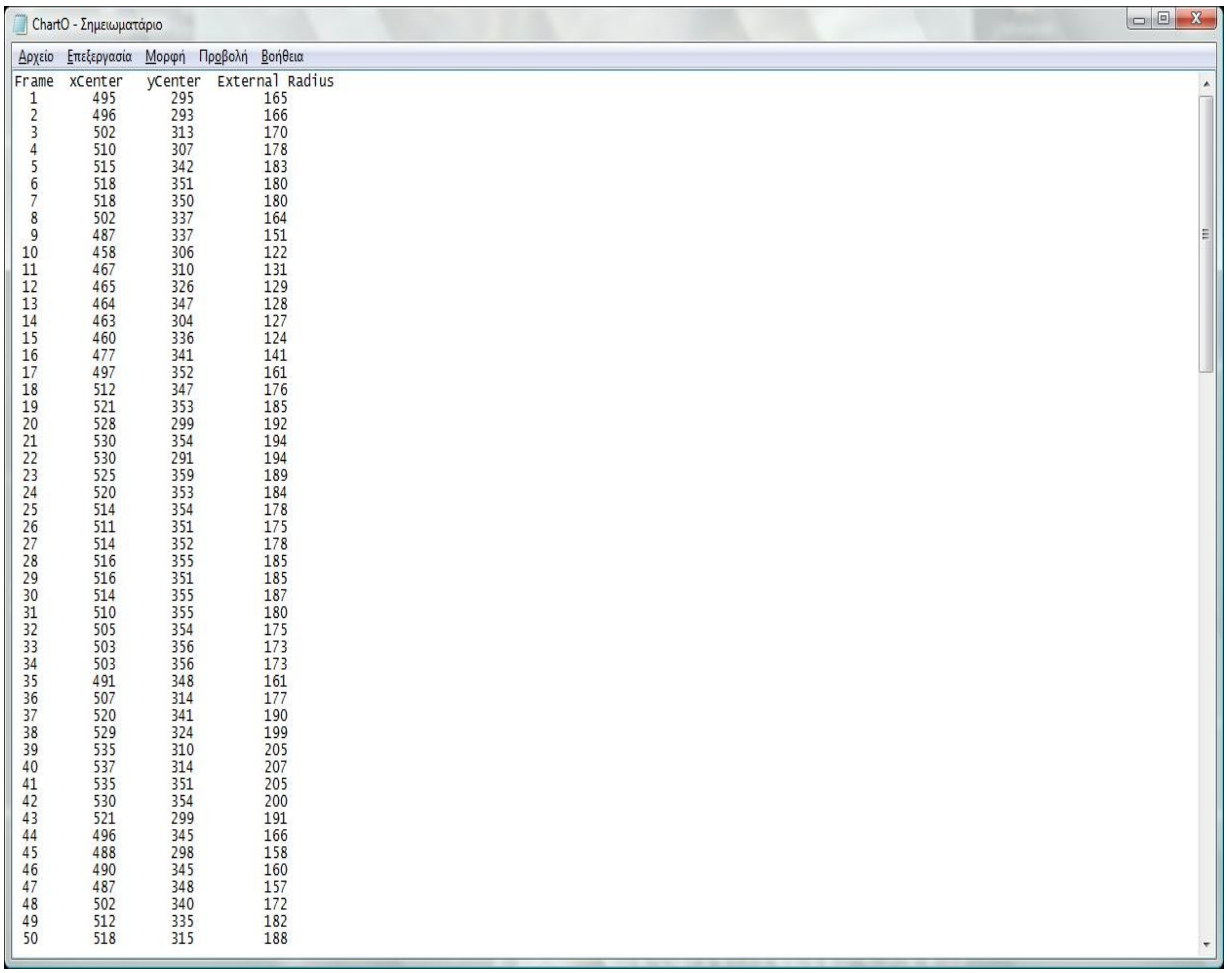


Εικόνα 3.7

3.8 Εκτύπωση των αποτελεσμάτων

Το τελικό μας βήμα είναι η εκτύπωση των αποτελεσμάτων,τα αποτελέσματα θα περιέχουν τον αριθμό των πλαισίων,τις τιμές x,y του κέντρου και την ακτίνα,όπως ακριβώς δείχνει η εικόνα 3.8.

```
pFile = fopen ("Chart1.txt","a");
                fprintf (pFile, "%3d      %d      %d        %d\n",i,pt1.x,pt1.y,
                (int)fabs(pt1.x-pt2.x));
        fclose (pFile);
```



Frame	xCenter	yCenter	External Radius
1	495	295	165
2	496	293	166
3	502	313	170
4	510	307	178
5	515	342	183
6	518	351	180
7	518	350	180
8	502	337	164
9	487	337	151
10	458	306	122
11	467	310	131
12	465	326	129
13	464	347	128
14	463	304	127
15	460	336	124
16	477	341	141
17	497	352	161
18	512	347	176
19	521	353	185
20	528	299	192
21	530	354	194
22	530	291	194
23	525	359	189
24	520	353	184
25	514	354	178
26	511	351	175
27	514	352	178
28	516	355	185
29	516	351	185
30	514	355	187
31	510	355	180
32	505	354	175
33	503	356	173
34	503	356	173
35	491	348	161
36	507	314	177
37	520	341	190
38	529	324	199
39	535	310	205
40	537	314	207
41	535	351	205
42	530	354	200
43	521	299	191
44	496	345	166
45	488	298	158
46	490	345	160
47	487	348	157
48	502	340	172
49	512	335	182
50	518	315	188

Εικόνα 3.8

3.9 Τελικά βήματα

Τελειώνοντας το πρόγραμμα αποδεσμεύουμε την μνήμη που είχαμε δεσμεύσει για κάποιες εικόνες και δομές δεδομένων,με σκοπό να μην κολάει το πρόγραμμα,όπως επίσης κλείνουμε και τα παράθυρα.

```
free(xw);
 free(yw);
free(x1w);
 free(y1w);
free(ymin1);
free(ymax1);

cvReleaseImage( &newImg);
	cvReleaseImage( &grayImg);
	cvReleaseImage( &contourImg);
	cvReleaseImage( &edgeImg);
	cvReleaseImage( &contour1Img);
	cvReleaseImage( &segmentedImg);

cvWaitKey(0);

	}
			cvReleaseMemStorage(&SpaceMemory);
	cvDestroyWindow("cunny");
	cvDestroyWindow("src");
	cvDestroyWindow("edge");

}
```

4.Βιβλιογραφία

Η βιβλιογραφεία που χρησιμοποιήσα για να κατανοήσω την τεχνητή όραση,την OpenCV αλλά και να μάθω για τους στροβίλους ,ήταν κυρίως πηγές του διαδικτίου,σημειώσεις απο μαθήματα καθώς και papers σχετικά με τους στρόβιλους. Αναλυτικά η βιβλιογραφία που χρησιμοποίησα είναι η παρακάτω:

- Vortex Dymanics Analysis in Jet Flows Using 2D-Planar PIV and High-Speed Laser Tomography Image Processing (Ilina Nastase,Amina Meslem,Thierry Bouwmans University of La Rochelle)
- Tutorial of OpenCV
- http://en.wikipedia.org/wiki/OpenCV
- http://sourceforge.net/projects/opencvlibrary/
- http://www710.univ-lyon1.fr/~bouakaz/OpenCV-0.9.5/docs/ref/OpenCVRef_BasicFuncs.htm
- Σημειώσεις τεχνητής όρασης (Γεώργιος Παπαδουράκης)
- Καθώς και διάφορες άλλες πηγές που βρήκα μέσω των μηχανών αναζήτησης

5. ΠΑΡΑΡΤΗΜΑ

Ο τελικός κώδικας sras.cpp

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "minMax.h"
#include "TrouverCentre.h"
#include <sstream>

#define PI 3.14159265

void main (void)
{
	IplImage* newImg;
	IplImage* grayImg;
	IplImage* contourImg;
	IplImage* contour1Img;
	IplImage* segmentedImg;
	IplImage* edgeImg;

	char *file=(char *)malloc(150);
	int i;
	int *ymax1,*ymin1,*xs,*ys,*xw,*yw,*x1w,*y1w,*k;
	int  xdep=300;
	int ydep=300;
	int shift=100;

	FILE * pFile;
	int *debutx,*debuty;
	int cY,cX;
	int CgY,CgX;
	float theta,theta1,difference,*v1x,*v1y,*v2x,*v2y;
```

```
        double LongMax = 0;
        double Long;
        int re = 0;

        CvPoint center,pt1,pt2,*point;
        CvMemStorage *SpaceMemory = cvCreateMemStorage();
        CvSeq *Sequence;
        CvSeq *Debut;

        pFile = fopen ("Chart1.txt","w");
        fprintf (pFile, "Frame  xCenter   yCenter  External Radius \n");
         fclose (pFile);


        newImg=cvLoadImage("C:\\Users\\user\\Desktop\\Orifice circulaire\\JC
X=50mm\\B00001.bmp",1);
 //newImg=cvLoadImage("C:\\Users\\user\\Desktop\\Orifice
Marguerite\\X=20mm\\B00001.bmp",1);

         xs=(int *)malloc(1*sizeof(int ) );
         ys=(int *)malloc(1*sizeof(int ) );
         v1x=(float *)malloc(1*sizeof(float ) );
         v2x=(float *)malloc(1*sizeof(float ) );
         v1y=(float *)malloc(1*sizeof(float ) );
         v2y=(float *)malloc(1*sizeof(float ) );
         pFile=(FILE *)malloc(1*sizeof(FILE ) );

   cvNamedWindow("src",1);
        cvNamedWindow ("segmented",1);
        cvNamedWindow("contour",1);
        cvNamedWindow("edge",1);

        for (i=1;i<400;i++)   // Begin of the Sequence
   {
                input ob;
           string s;

     s=ob.getImageFromSequence("C:\\Users\\user\\Desktop\\Orifice circulaire\\JC
X=50mm\\B00001.bmp",i);
    //  s=ob.getImageFromSequence("C:\\Users\\user\\Desktop\\Orifice
Marguerite\\X=20mm\\B00001.bmp",i);
      const char *image=s.c_str();
       sprintf(file,image);
```

```
	newImg=cvvLoadImage(file);
	 newImg->width=newImg->width-87;
	newImg->height=newImg->height-153;

	cvShowImage( "src" , newImg);

	grayImg=cvCreateImage( cvSize(newImg->width,newImg->height),IPL_DEPTH_8U,1);
	edgeImg=cvCreateImage( cvSize(newImg->width,newImg->height),IPL_DEPTH_8U,1);
	contourImg=cvCreateImage( cvSize(newImg->width,newImg-
>height),IPL_DEPTH_8U,1);
	segmentedImg=cvCreateImage( cvSize(newImg->width,newImg-
>height),IPL_DEPTH_8U,1);
	contour1Img=cvCreateImage( cvSize(newImg->width,newImg-
>height),IPL_DEPTH_8U,1);
	cvCvtColor (newImg,grayImg,CV_BGR2GRAY);

	cvSet(contourImg, CV_RGB (0,0,0));
	cvSet(contour1Img, CV_RGB (0,0,0));
	cvSet(edgeImg, CV_RGB (0,0,0));


	cvThreshold(grayImg,segmentedImg,70,255,CV_THRESH_BINARY);
	cvErode( segmentedImg, segmentedImg, 0, 1 );
	cvDilate( segmentedImg, segmentedImg, 0, 10);
	cvCanny(segmentedImg,contourImg,50,150,3);

	cvFindContours(contourImg, SpaceMemory, &Sequence, sizeof(CvContour),
CV_RETR_LIST , CV_LINK_RUNS);


	Debut=Sequence;

	for( ; Sequence != 0; Sequence = Sequence->h_next)
			{
				Long = cvArcLength(Sequence, CV_WHOLE_SEQ,1);
				re++;
				if(Long > LongMax)
				{
					LongMax = Long;
				}
			}
Sequence=Debut;
```

```
                        for( ; Sequence != 0; Sequence = Sequence->h_next)
                        {
                        CvScalar color;
                                if (LongMax == cvArcLength(Sequence, CV_WHOLE_SEQ,1))
                                {
                                        // if(choix == IMG_TRAN)
                                        {
                                                TrouverCentreG(Sequence,&CgX,&CgY);
                                        }
                                        color = CV_RGB( 255, 0, 0 );
                                }
                                else
                                        color = CV_RGB(255,127,127);
                                cvDrawContours( contourImg, Sequence, color, color, 255, 1 , 8 );
                                cvDrawContours( newImg, Sequence, color, color, 255, 1 , 8 );
                                cvDrawContours( edgeImg, Sequence, color, color, 255, 1 , 8 );
                        }
Sequence=Debut;
                        printf("%d,%d\n",CgX,CgY);


//-----------------------------------------------------

                        //find the centre

                        TrouverCentre1(newImg,&cX,&cY,CgX,CgY);

                        center.x=cX;
                        center.y=cY;

                        cvCircle( newImg,center,5, CV_RGB(0,0,255),1 );

// ---------------------------------------
                        //find radious

        xw=(int *)malloc (newImg->width*newImg->height*sizeof(int ) );
        yw=(int *)malloc (newImg->width*newImg->height*sizeof(int ) );
        x1w=(int *)malloc (newImg->width*newImg->height*sizeof(int ) );
        y1w=(int *)malloc (newImg->width*newImg->height*sizeof(int ) );
        ymin1=(int *)malloc (1*sizeof(int ) );
        ymax1=(int *)malloc (1*sizeof(int ) );
        k=(int *)malloc(1*sizeof(int ) );
```

```
        debutx=(int *)malloc (1*sizeof(int ) );
        debuty=(int *)malloc (1*sizeof(int ) );

         initilization(newImg, xw, yw);
         searchWhitePixel(segmentedImg,xw,yw);


        SearchCX(xw,yw,cY,ymin1,ymax1) ;   //deksia grammi - right line
         pt1.x=*ymax1;
         pt1.y=cY;
         pt2.x=cX;
         pt2.y=cY;
         printf("%d,%d " ,pt1.x,pt1.y);
         cvLine (newImg,  pt1, pt2, CV_RGB (0, 255,0),1);


searchWhitePixel(edgeImg,x1w,y1w);
        SearchCX(xw,yw,cY,ymin1,ymax1) ;   //deksia grammi - right line
        pt1.x=*ymax1;
        pt1.y=cY;
        pt2.x=cX;
        pt2.y=cY;
       printf("%d,%d " ,pt1.x,pt1.y);
        cvLine (edgeImg,  pt1, pt2, CV_RGB (0, 255,255),1);
                cvCircle( edgeImg,center,5, CV_RGB(0,0,255),1 );


                        xw++;
                        yw++;
                }

                xw=debutx;
                yw=debuty;

                        if (LongMax == cvArcLength(Sequence, CV_WHOLE_SEQ,1))
                                {
                                for(int i = 0; i <500; i++) //Sequence->total;i++ )
                                        {
                                        point = (CvPoint *)(cvGetSeqElem(Sequence, i));
                                         center.x= point->x;
                                         center.y= point->y;

                                        cvCircle( edgeImg,center,5, CV_RGB(0,255,255),1 );
```

```
                                }
                        Sequence = Sequence->h_next;
                        }

debutx=xw;
        debuty=yw;
        while (*xw!=-1){
                pt1.x=*xw;
                pt1.y=*yw;
                cvLine (edgeImg,  pt1, pt2, CV_RGB (0, 255,255),1);
                for(int j=0;j<10;j++){
                xw++;
                yw++;
                }
        }

        xw=debutx;
        yw=debuty;

        pFile = fopen ("Chart1.txt","a");
        fprintf (pFile, "%3d      %d      %d        %d\n",i,pt1.x,pt1.y,(int)fabs(pt1.x-pt2.x));
        fclose (pFile);



cvShowImage( "src" , newImg);
cvShowImage( "segmented", segmentedImg);
cvShowImage( "contour", contour1Img);
cvShowImage( "edge", edgeImg);
   printf("Frame %d\n",i);


 free(xw);
 free(yw);
 free(x1w);
 free(y1w);

 cvReleaseImage( &newImg);
cvReleaseImage( &grayImg);
cvReleaseImage( &contourImg);
cvReleaseImage( &edgeImg);
cvReleaseImage( &contour1Img);
cvReleaseImage( &segmentedImg);
```

```
cvWaitKey(0);

	}
			cvReleaseMemStorage(&SpaceMemory);
	cvDestroyWindow("cunny");
	cvDestroyWindow("src");
	cvDestroyWindow("edge");

}
```

Οι συναρτήσεις που χρησιμοποιήσαμε καθώς και τα headers τους.

cpix.cpp και cpix.h

```
#include ".\cpix.h"
#include <stdio.h>

cpix::cpix(void)
{
}

IplImage* cpix::FindingMask(IplImage *InputImage,int Threshold)
{

	//InputImage = the image we have to extract the mask from
	IplImage *InputImage1=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
	IplImage *InputImage2=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
	IplImage *InputImage3=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
	IplImage *ResultImage=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,3);
	IplImage *ResultImage1=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
	IplImage *ResultImage2=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
	IplImage *ResultImage3=cvCreateImage(cvSize(InputImage->width,InputImage-
```

```
>height),IPL_DEPTH_32F,1);
        IplImage *OrImage12=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
        IplImage *OrImage123=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,1);
        IplImage *aux=cvCreateImage(cvSize(InputImage->width,InputImage-
>height),IPL_DEPTH_32F,3);

        //Initialising OR images
        cvFillImage(OrImage12,0.0);
        cvFillImage(OrImage123,0.0);

        //Separate the image in 3 different planes
        cvCvtPixToPlane(InputImage,InputImage1,InputImage2,InputImage3,0);

        //InputImage - Colored images but not the Foreground
        //InputImage1,2,3 - Gray images
        cvThreshold(InputImage1,ResultImage1,Threshold/(float)255,1,CV_THRESH_BINARY);
        cvThreshold(InputImage2,ResultImage2,Threshold/(float)255,1,CV_THRESH_BINARY);
        cvThreshold(InputImage3,ResultImage3,Threshold/(float)255,1,CV_THRESH_BINARY);

        //ResultImage1,2,3 - Binary images
        cvOr(ResultImage1,ResultImage2,OrImage12,NULL);
        cvOr(OrImage12,ResultImage3,OrImage123,NULL);
        cvCvtPlaneToPix(OrImage123,OrImage123,OrImage123,0,ResultImage);

        cvReleaseImage(&InputImage1);
        cvReleaseImage(&InputImage2);
        cvReleaseImage(&InputImage3);
        cvReleaseImage(&ResultImage1);
        cvReleaseImage(&ResultImage2);
        cvReleaseImage(&ResultImage3);
        cvReleaseImage(&OrImage12);
        cvReleaseImage(&OrImage123);
        cvReleaseImage(&aux);

        return (ResultImage);
}
void cpix::GetPixel(IplImage *image,int m,int n,unsigned char *pixelcourant)
{
        int k;

        for (k=0;k<3;k++)
```

```
                {
                pixelcourant[k]=((unsigned char*)(image->imageData + image->widthStep*n))
[m*3 + k];
                }
}
void cpix::GetGrayPixel(IplImage  *image,int m,int n,unsigned char *pixelcourant)
{
        *pixelcourant =((unsigned char*)(image->imageData + image->widthStep*n))[m];
}

void cpix::PutPixel(IplImage *image,int p,int q,unsigned char *pixelcourant)
{
        int r;
        for (r=0;r<3;r++)
        {
                ((unsigned char*)(image->imageData + image->widthStep*q))[p*3 +
r]=pixelcourant[r];
        }
}
void cpix::PutGrayPixel(IplImage *image,int p,int q,unsigned char pixelcourant)
{
        ((unsigned char*)(image->imageData + image->widthStep*q))[p]=pixelcourant;
}

void cpix::GetPixel(IplImage *image,int m,int n,float *pixelcourant)
{
        int k;

        for (k=0;k<3;k++)
                {
                pixelcourant[k]=((float*)(image->imageData + image->widthStep*n))[m*3 + k];
                }
}
void cpix::GetGrayPixel(IplImage  *image,int m,int n,float *pixelcourant)
{
        *pixelcourant =((float*)(image->imageData + image->widthStep*n))[m];
}

void cpix::PutPixel(IplImage *image,int p,int q,float *pixelcourant)
{
        int r;
        for (r=0;r<3;r++)
        {
```

```
                ((float*)(image->imageData + image->widthStep*q))[p*3 + r]=pixelcourant[r];
        }
}
void cpix::PutGrayPixel(IplImage *image,int p,int q,float pixelcourant)
{
        ((float*)(image->imageData + image->widthStep*q))[p]=pixelcourant;
}
cpix::~cpix(void)
{
}



#pragma once
#include <cv/cv.h>
#include <cv/highgui.h>

class cpix
{
public:
        cpix(void);

        IplImage* FindingMask(IplImage *InputImage,int Threshold);

        void GetPixel(IplImage *image,int m,int n,unsigned char *pixelcourant);
        void GetGrayPixel(IplImage *image,int m,int n,unsigned char *pixelcourant);

        void PutPixel(IplImage *image,int p,int q,unsigned char *pixelcourant);
        void PutGrayPixel(IplImage *image,int p,int q,unsigned char pixelcourant);

        void GetPixel(IplImage *image,int m,int n,float *pixelcourant);
        void GetGrayPixel(IplImage *image,int m,int n,float *pixelcourant);

        void PutPixel(IplImage *image,int p,int q,float *pixelcourant);
        void PutGrayPixel(IplImage *image,int p,int q,float pixelcourant);

        ~cpix(void);
};
```

flux.cpp και flux.h

```
#include <stdio.h>
```

```
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "minMax.h"
#include "TrouverCentre.h"

#include <sstream>



void searchWhitePixel(IplImage *newImg,int *xw,int *yw)
{
	int i,j;
	int *debutx;
	int *debuty;
	int k=0;

unsigned char *pixelcourant;
	cpix p;

	pixelcourant= (unsigned char *)malloc(1*sizeof(unsigned char ) );

	for (i=0;i<newImg->width;i++)
		for (j=0; j<newImg->height; j++ )
		{

p.GetGrayPixel(newImg,i,j,pixelcourant);



				if ((int)(*pixelcourant) !=0 ){
					*(xw+k)=i;
					*(yw+k)=j;
					k++;
```

```
                        }
        }
        }



void initilization(IplImage* newImg, int *xw, int *yw)
        {

                int i;


                for(i=0; i< newImg->width*newImg->height;i++)
                {
                        *(xw+i)=-1;
                        *(yw+i)=-1;
                                        }
        }




#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "minMax.h"
#include "TrouverCentre.h"


#include <sstream>



void searchWhitePixel(IplImage *newImg,int *xw,int *yw);
```

```
void initilization(IplImage* newImg, int *xw, int *yw);
```

input.cpp και input.h

```
#include "input.h"
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <iostream>
#include <sstream>
using namespace std;

input::input(void)
{
}

string input::readName()
{
	char s[1000];
	scanf("%s",s);
	string strS = s;
	return strS;
}

string input::getImageFromSequence(string seqName, int seqNr)
{
	string s = seqName;
	std::stringstream strm;
	strm << seqNr;
	string nr = strm.str();//the number of the image as a string
	int PointPosition = s.find(".");
	int nr_len = nr.length();
	s.replace(PointPosition - nr_len, nr_len, nr);

	return s; }
input::~input(void)
{
}



#pragma once
#include <iostream>
using namespace std;
```

```
class input
{
public:
	input(void);
	string readName(void);
	string getImageFromSequence(string seqName, int seqNr);
	~input(void);
};
```

minmax.cpp και minmax.h

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "minMax.h"
#include "TrouverCentre.h"

#include <sstream>


void Searchmax(int *xw, int *yw,int *k)
{

	int max=0;
	int kmax=0;
	int *debutx;
	debutx=xw;
   *k=0;
	while(*xw!=-1)
		{

		if(*xw>max)
```

```
                        {
                        max=*xw;
                        kmax=*k;

                        }
                (*k)++;
                xw++;

        }

        *k=kmax;

        xw=debutx;

}



void Searchmin(int *xw, int *yw,int *k)
{

        int min=9999;
        int kmin=0;
        int *debutx;
        debutx=xw;
   *k=0;

        while(*xw!=-1)
                {
                if(*xw<min)
                        {
                        min=*xw;
                        kmin=*k;
                        }
                (*k)++;
                xw++;
        }

        *k=kmin;
        xw=debutx;
}
```

```
void resizeImage(IplImage * newImg,IplImage * newnewImg)
{
int i,j;
int shift=60;
unsigned char *pixelcourant;
        cpix p;

        pixelcourant= (unsigned char *)malloc(1*sizeof(unsigned char ) );

        for (i=shift;i<newImg->width;i++)
                for (j=shift; j<newImg->height; j++ )
                {

p.GetGrayPixel(newImg,i,j,pixelcourant);
p.PutGrayPixel(newnewImg,i-shift,j-shift,*pixelcourant);

                }
                free (pixelcourant);
}



#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "TrouverCentre.h"

#include <sstream>

void Searchmax(int *xw, int *yw,int *k);
void Searchmin(int *xw, int *yw,int *k);
void resizeImage(IplImage * newImg,IplImage * newnewImg);
```

```
void TrouverCentre(IplImage * newImg, int *xw, int *yw) ;
void Searchymax(int *xw, int *yw,int * xcentre,int *ymin);
void Searchymin(int *xw, int *yw,int * xcentre,int *ymin);
```

TrouverCentre.cpp και TrouverCentre.h

```
#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "minMax.h"
#include "TrouverCentre.h"
#include <sstream>



void TrouverCentre1(IplImage * Img, int *xs, int *ys, int xdep, int ydep) //vres to kentro
{
	int holdx = 0, holdy = 0, holdholdx = 0, holdholdy = 0, holdholdholdx = 0, holdholdholdy
= 0;
	int holdholdholdholdx = 0, holdholdholdholdy = 0;
	bool sortie = false;
	cpix cp;
	int x = xdep, y = ydep;
	int xg = x, xd = x;
	int yh = y, yb = y;
	unsigned char pixel[3];

	do
	{
		sortie=true;

		holdholdholdholdx = holdholdholdx;
		holdholdholdholdy = holdholdholdy;
```

```
holdholdholdx = holdholdx;
holdholdholdy = holdholdy;
holdholdx = holdx;
holdholdy = holdy;
holdx = x;
holdy = y;
printf("%d\n",holdx);
for( x = holdx; x > 0; x--)
{
	cp.GetPixel(Img,x,y,pixel);
	if(pixel[0] == 0 && pixel[1] == 0 && pixel[2] == 255)
	{
		xg = x;
		x = 0;
	}
}

for( x = holdx; x < cvGetSize(Img).width; x++)
{
	cp.GetPixel(Img,x,y,pixel);
	if(pixel[0] == 0 && pixel[1] == 0 && pixel[2] == 255)
	{
		xd = x;
		x = cvGetSize(Img).width;
	}
}
x = (xd + xg) / 2;

for( y = holdy; y > 0; y--)
{
	cp.GetPixel(Img,x,y,pixel);
	if(pixel[0] == 0 && pixel[1] == 0 && pixel[2] == 255)
	{
		yh = y;
		y = 0;
	}
}
for( y = holdy; y < cvGetSize(Img).width; y++)
{
	cp.GetPixel(Img,x,y,pixel);
	if(pixel[0] == 0 && pixel[1] == 0 && pixel[2] == 255)
	{
		yb = y;
```

```
                                y = cvGetSize(Img).width;
                        }
                }
                y = (yh + yb) /2;

                if(holdx <= x + 5 && holdy <= y + 5 && holdx >= x - 5 && holdy >= y - 5)
                {
                        sortie = true;
                        *xs = x;
                        *ys = y;
                }
                else if(holdholdx <= x + 5 && holdholdy <= y + 5 && holdholdx >= x - 5 &&
holdholdy >= y - 5)
                {
                        sortie = true;
                        *xs = x;
                        *ys = y;
                }
                else if(holdholdholdx <= x + 5 && holdholdholdy <= y + 5 && holdholdholdx >=
x - 5 && holdholdholdy >= y - 5)
                {
                        sortie = true;
                        *xs = x;
                        *ys = y;
                }
                else if(holdholdholdholdx <= x + 5 && holdholdholdholdy <= y + 5 &&
holdholdholdholdx >= x - 5 && holdholdholdholdy >= y - 5)
                {
                        sortie = true;
                        *xs = x;
                        *ys = y;
                }

        }while(!sortie);

}


void TrouverCentreG(CvSeq *seq,int *x, int *y)
{
        CvPoint *point;
        long sommex = 0;
        long sommey = 0;
```

```
        for(int i = 0; i < seq->total; i++)
        {
                point = (CvPoint *)(cvGetSeqElem(seq, i));
                sommex += point->x;
                sommey += point->y;
        }
        *x = sommex / seq->total;
        *y = sommey / seq->total;
}


void SearchCX(int *xw,int *yw,int CX,int *xmin,int *xmax)
{

         int i,j;

         int *debutx;
         int *debuty;

                debutx=xw;
                debuty=yw;
  *xmin=999;
        *xmax=0;



        while(*yw!=-1)
                {
                if (*yw==CX)
                {
                if(*xw<*xmin)
                        {
                        *xmin=*xw;

                        }
                if(*xw>*xmax)
                        {
                        *xmax=*xw;

                        }
                }
                xw++;
                yw++;
```

```
        }


        xw=debutx;
        yw=debuty;
}


void vectorN(float *vx,float *vy )
{
         *vx=(*vx/sqrt((pow(*vx,2))+(pow(*vy,2))));
         *vy=(*vy/sqrt((pow(*vx,2))+(pow(*vy,2))));
}



#include <stdio.h>
#include "cv.h"
#include "highgui.h"
#include <cvaux.h>
#include <math.h>
#include "cpix.h"
#include <stdlib.h>
#include <string.h>
#pragma once
#include <iostream>
using namespace std;
#include "input.h"
#include "flux.h"
#include "TrouverCentre.h"

#include <sstream>

void TrouverCentre1(IplImage * Img, int *xs, int *ys, int xdep, int ydep);
void TrouverCentreG(CvSeq *seq,int *x, int *y);
void SearchCX(int *xw,int *yw,int CX,int *ymin,int *ymax);
```