ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF EPIRUS
SCHOOL OF APPLIED TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας :

ΤΕΧΝΙΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ ΜΕ ΤΗ ΧΡΗΣΗ MATLAB

Σπουδαστής: ΚΑΛΟΓΗΡΟΥ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων καθηγητής : ΣΤΕΡΓΙΟΥ ΕΛΕΥΘΕΡΙΟΣ

Άρτα Σεπτέμβριος 2014

ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΗΠΕΙΡΟΥ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

TECHNOLOGICAL EDUCATIONAL INSTITUTE OF EPIRUS
SCHOOL OF APPLIED TECHNOLOGY
DEPARTMENT OF COMPUTER ENGINEERING

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τίτλος Πτυχιακής Εργασίας :

ΤΕΧΝΙΚΕΣ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ ΜΕ ΤΗ ΧΡΗΣΗ MATLAB

Σπουδαστής: ΚΑΛΟΓΗΡΟΥ ΚΩΝΣΤΑΝΤΙΝΟΣ

Επιβλέπων καθηγητής : ΣΤΕΡΓΙΟΥ ΕΛΕΥΘΕΡΙΟΣ

Εξεταστική επιτροπή :
1……..
2……
3……

Άρτα Σεπτέμβριος 2014

ΠΕΡΙΛΗΨΗ

Στην εργασία αυτή θα δούμε διάφορους τρόπους επεξεργασίας εικόνας , καθώς και το πως μπορούμε να τους εφαρμόσουμε με τη χρήση Matlab. Θα γίνει μια επισκόπηση του τι γίνεται με τις εικόνες και πως λύνονται ορισμένα σημαντικά προβλήματα. Θα εξηγήσουμε επίσης πώς να προγραμματίσουμε κάποιες από τις λειτουργίες που χρειάζονται για να δημιουργήσουμε το επιθυμητό αποτέλεσμα σε μια εικόνα .

ABSTRACT

In this paper we will look at different ways of image processing, and how we can apply them using Matlab. It will be an overview of what is done with the pictures and that solved some important problems. It will also explain how to program some of the functions needed to create the desired effect on an image.

ΠΕΡΙΕΧΟΜΕΝΑ

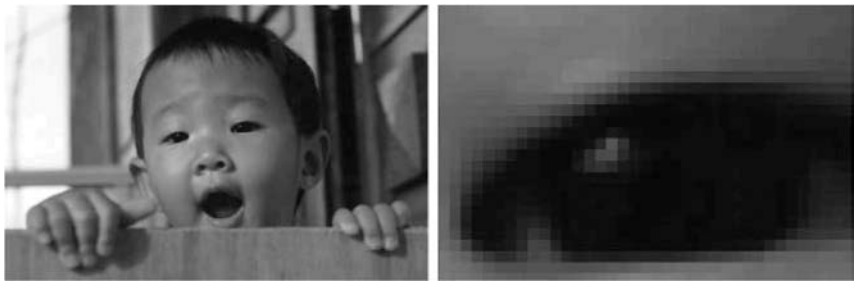
ΕΙΣΑΓΩΓΗ

Στην εργασία αυτή θα δούμε διάφορους τρόπους επεξεργασίας εικόνας, καθώςκαι το πως μπορούμε να τους εφαρμόσουμε με τη χρήση Matlab. Θα γίνει μια επισκόπηση του τι γίνεται με τις εικόνες και πως λύνονται ορισμένα σημαντικά προβλήματα. Θα εξηγήσουμε επίσης πώς να προγραμματίσουμε κάποιες από τις λειτουργίες που χρειάζονται για να δημιουργήσουμε το επιθυμητό αποτέλεσμα σε μια εικόνα .

Η προβολή εικόνας, παλέτα χρωμάτων

Θα θεωρήσουμε την εικόνα ως ένα σύνολο από εικονοστοιχείαπου συνδέονται με ένα ορθογωνικό πλέγμα (Εικόνα1).

Στο MATLAB, υπάρχουν διάφοροι τρόποι για να εμφανιστεί μια εικόνα, είτε άμεσα με ένα (N x M x 3) ή (N x M x 4) ανάλογα με το μοντέλο χρωμάτων: RGB (κόκκινο, πράσινο και μπλε), CMYK (κυανό, ματζέντα, κίτρινο και μαύρο), HSL (Hue , κορεσμός και φωτεινότητα), CIE Lab.



Εικόνα1

Εικόνα : Κάθε σημείο της αρχικής εικόνας έχει ένα 8-bit κωδικοποιημένο "γκρι επίπεδο". Κάθε pixel εμφανίζεται ως ένα γκρι τετράγωνο.

Στο ακόλουθο παράδειγμα, μια εικόνα σε μορφή JPEG εισάγεται με τη χρήση της λειτουργίας imreadμε διάσταση 800 χ 580 χ 3 πίνακα, το 3 υποδεικνύει ότι υπάρχουν τρία RGB επίπεδα . Παρατηρήστε ότι ο τύπος δεδομένων που χρησιμοποιείται είναι 8-bit ακέραιος χωρίς πρόσημο:

```
» xx=imread('10.jpg','jpeg')

Name                Size
ans             xx
Bytes Class
188 char array 1392000 uint8
array

1x94            800x580x3
```

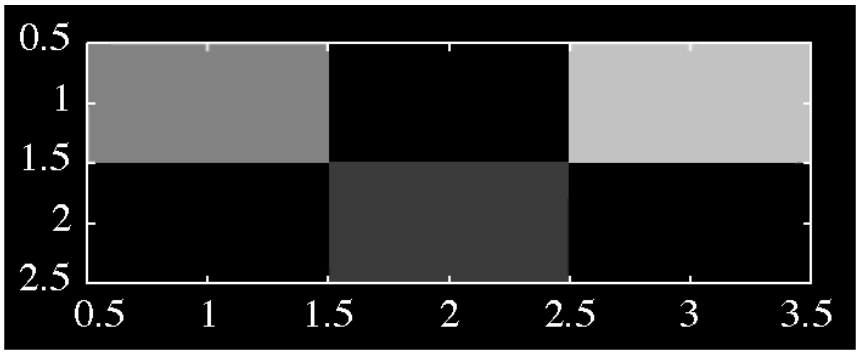
Με τη χρήση μιας 2D εικόνας και μια παλέτα χρωμάτων αυτός είναι ο τρόπος προβολήςτου indexworkspace.

Έστω A = [a (i, j)], με 1 <i <N και 1 <j <M, είναι ένας πίνακας N χ M. Ο αριθμός α (i, j), τοποθετημένο στη γραμμή I και στη στήλη j, υποδεικνύει το χρώμα του σημείο με συντεταγμένες (i, j) της εικόνας αφού έχει δειγματοληπτηθεί και υπολογιστεί. Ο δείκτης γραμμής i αντιπροσωπεύει την οριζόντια θέση, και ο δείκτης στήλης j αντιπροσωπεύει την κατακόρυφη θέση. Το σημείο με τις συντεταγμένες (1, 1) τοποθετείται στην πάνω αριστερή γωνία (Εικόνα 2).

Παράδειγμα 1.1 (Pixelizing μιας εικόνας)

```
imagel = [32 0 48; 0 16 0];
image (imagel);
colormap ('gray');
```

Η εικόνα που εμφανίζεται αποτελείται από 6 σημεία, ή λογικαpixels, και το ένα σημείο μας που σχετίζεται με την τοποθέτηση του σημείου μας (1,1) είναι αυτό στη πάνω αριστερή γωνία (Εικόνα 2).



Εικόνα 2

Εικόνα 2- Έξι εικονοστοιχεία: το x- ακέραιο και y-συντεταγμένη που αντιστοιχεί στο "κέντρο" του κάθε pixel.

Παρατηρούμε ότι ένα στοιχείο του πίνακα με το δείκτη (i, j) μπορεί να συνδέεται με διάφορα φυσικά pixels(εικονοστοιχεία) του παραθύρου οθόνης. Στην πραγματικότητα, δεν υπάρχει κανένας λόγοςο αριθμός των τιμών των στοιχείων α (i, j) του πινακα να είναι ίσος με τον αριθμό των φυσικών εικονοστοιχείων του παραθύρου οθόνης. Ως εκ τούτου, ένα σημείο με τις συντεταγμένες (i, j) μπορεί να αντιπροσωπεύεται από διάφορα φυσικαpixels, ακριβώς όπως ένα φυσικοεικονοστοιχείο μπορεί να χρησιμοποιηθεί για να αντιπροσωπεύσει πολλά σημεία με τις συντεταγμένες (i, j). Από τώρα και στο εξής, όταν χρησιμοποιούμε τη λέξη pixel, εννοούμε ένα λογικοpixel, δηλαδή τα στοιχεία που προσδιορίζονται από το ζεύγος (i, j).

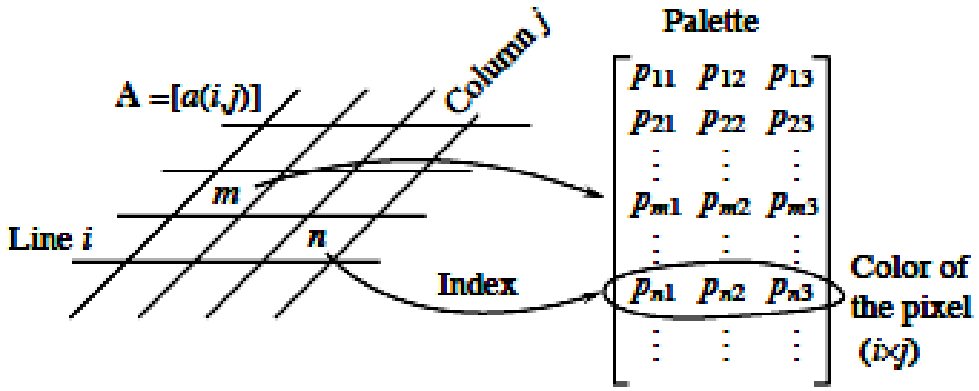
Εάν θέλουμε να εμφανίσουμε μια εικόνα και να διατηρηθεί το πραγματικό μέγεθος της (ένα pixel οθόνης που αντιστοιχεί σε ένα pixel της εικόνας), θα πρέπει να χρησιμοποιήσουμε τις ιδιότητες units, Position, AspectRatio... (αυτές οι παράμετροι μπορούν να αλλάξουν από τη μία έκδοση MATLAB στην επόμενη) . Στο παράδειγμα 1.1, μια οθόνη σε πραγματικό μέγεθος επιτυγχάνεται με την πληκτρολόγηση:

set (gca,'units','pixels','Position', [20 20 fliplr (size (imagel))])

Στην αναπαράσταση ενδείξεων, το (i, j) υποδεικνυει μια σειρά χρωμάτων που ονομάζεται παλέτα (Εικόνα 3). Η χρωματική παλέτα είναι ένας (P x 3) πίνακας όπου κάθε γραμμή χρησιμοποιείται για να κωδικοποιήσει ένα χρώμα σύμφωνα με τα κόκκινα, πράσινα και μπλε συστατικά της (RGB) χρησιμοποιώντας έναν πραγματικό αριθμό μεταξύ 0 και 1.

Αυτή η παράσταση είναι εύκολη δεδομένου ότι τα περισσότερα προγράμματα επεξεργασίας bitmap μπορεί να παρέχουν μια περιγραφή της εικόνας σε τρία επίπεδα, το καθένα από αυτά να αντιστοιχει σε ένα πρωτεύον χρώμα R, G ή B, και κωδικοποιημενο ως ακέραιος αριθμός μεταξύ 0 και 2n - 1 (n-bit κωδικοποίηση). Οι εικόνες που θα εξετάσει θα είναι "σε επίπεδα του γκρι". Η MATLAB έχει μια προεπιλεγμένη παλέτα που μπορεί να ενεργοποιηθεί με τη χρήση του χάρτη χρωμάτων («γκρίζων»).

Μπορούμε να εξάγουμε έναν (64 x 3) πίνακα με τρεις πανομοιότυπες στήλες τιμών μεταξύ 0 και 1:



Εικόνα 3

- Στο παράδειγμα 1.1 οι μηδενικές τιμές της εικόνας 1 θα επαναπροσδιοριστούν ως 1 και ως εκ τούτου, ο δείκτης χρώματος (0, 0, 0), το οποίο είναι μαύρο (Εικόνα 2).
- Οι εντολές εικόνας, imagesc και colormap πρέπει ιδιαίτερα να τις προσέχουμε ως προς το τι θέλουμε να εξάγουμε ως αποτέλεσμα.
- Η τυπική παλέτα κατασκευάζεται γραμμικά. Κάθε στήλη είναι του τύπου [0: 1/63: 1] »(1/63 ~ 0.0159).

Αυτό δεν αντιστοιχεί με την αντίληψη που έχουμε για τη φωτεινότητα.

- Άλλα παλέτες έρχονται πρότυπο στην βασική έκδοση του MATLAB να κάνει το έργο πιο εύκολο του χρήστη. Χρησιμοποιούμε την εντολή helpcolor για να μάθουμε περισσότερα γι 'αυτα.
Επίσης, τίποτα δεν μας σταματά από το να καθορίσουμε τις δικές μας παλέτες.

Για παράδειγμα, για να πάρουμε μια οθόνη με 256 επίπεδα του γκρι, το μόνο που πρέπει να κάνουμε είναι να δημιουργήσουμε μια συστοιχία CMAP ως εξής:

```
cmap=[0:255] '*ones (1,3)/255 ;
colormap(cmap);
```

Εισαγωγή εικόνων

ΣτηMATLAB μπορούμε να δημιουργήσουμε πάντα μια βάση εικόνων rawformatμε τη χρήση του λογισμικού επεξεργασίας εικόνας. Την ίδια στιγμή, μπορούμε να αποθηκεύσουμε την παλέτα, αν αυτό είναι δυνατόν. Οι παρακάτω λειτουργίες μας επιτρέπουν να διαβάσουμε και να δημιουργήσουμε εικόνες που μπορούν να χρησιμοποιηθούν άμεσα από την MATLAB.
Για την ανάγνωση εικόνων μέσα από το περιβάλλον της Matlabχρησιμοποιούμε την εντολή imread(‘filename’).

Παράδειγμα: f=imread('SUN.jpg');

Το αποτέλεσμα της παραπάνω εντολής είναι η δημιουργία του τρισδιάστατου πίνακαf ο οποίος αντιστοιχεί στην εικόνα SUN.jpg

Για να βρούμε τις διαστάσεις τις εικόνας χρησιμοποιούμε την εντολή size(f). Ειδική περίπτωση για τις εικόνες με παλέτα χρωμάτων:

[fmap]=imread('SUN.gif');

Για την προβολή της εικόναςστην οθόνη χρησιμοποιούμετην εντολήimshow(matrixname)matrixname είναι το όνομα τουπίνακα με τον οποίοαναπαρίσταται η εικόνα μετάτην ανάγνωση της.

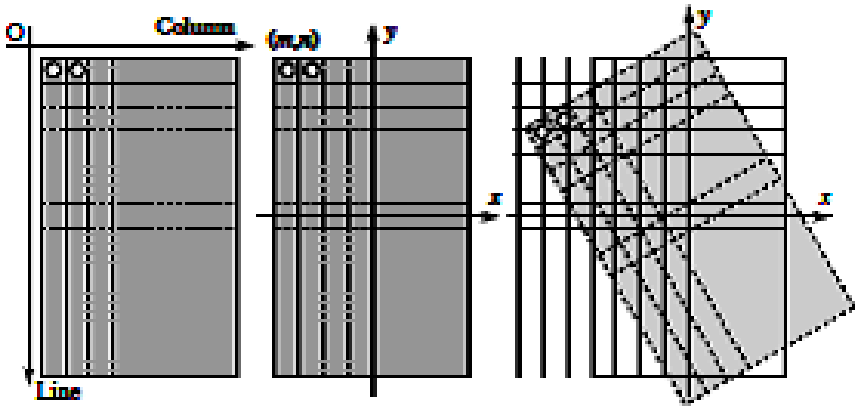
Παραδείγματα:
imshow(f)
imshow(f, map) (για εικόνες παλέτας χρωμάτων)
imshow(f, [lowhigh]) (για εικόνες αποχρώσεων του γκρι)

Προβολή περισσότερων απόμια εικόνες
Εντολή figure
Εντολή pixval

ΚΕΦΑΛΑΙΟ 1
ΓΕΩΜΕΤΡΙΚΟΙ ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΙΜΕ ΧΡΗΣΗ MATLAB

1.1 Γεωμετρικοί μετασχηματισμοί μιας εικόνας
Τυπικοί μετασχηματισμοί

Οι απλοί γεωμετρικοί μετασχηματισμοί, όπως οι μετατοπίσεις και οι παραμορφώσεις είναι προβληματικοί λόγω της ακεραίας φύσης των αριθμών και της θέσης των εικονοστοιχιων σε μια εικόνα .(Εικόνα 4: περιστροφή μιας εικόνας)



Εικόνα 4

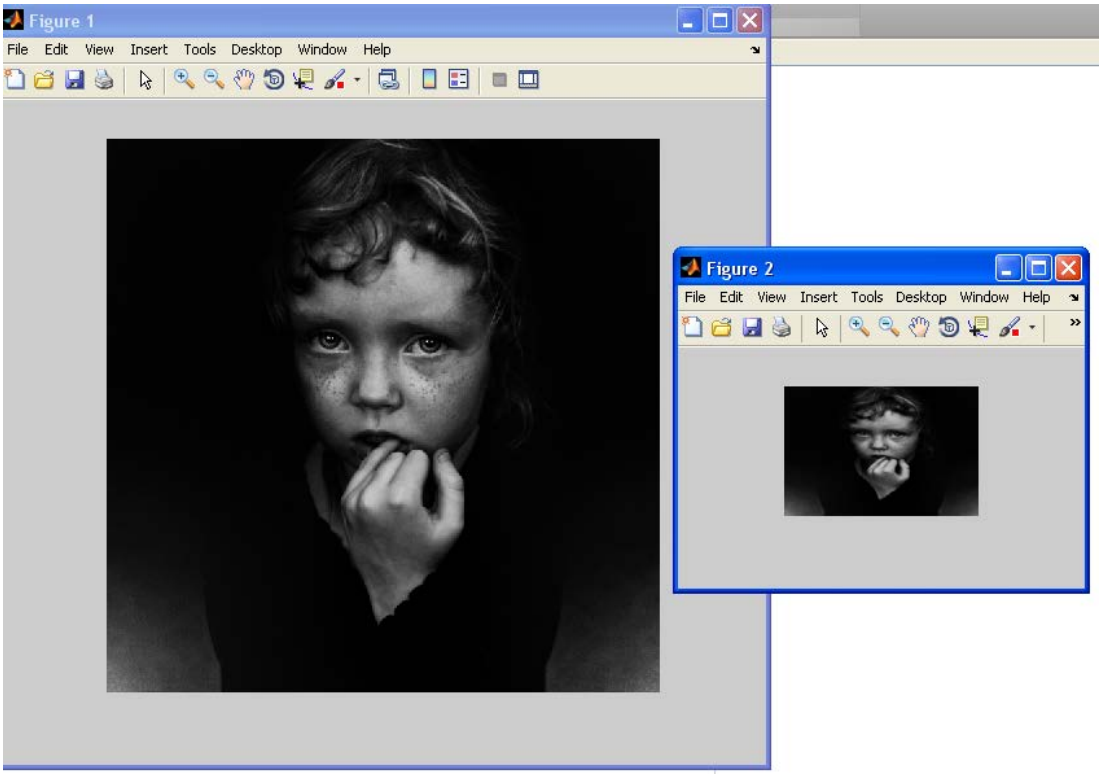
1.2 Μέγεθος εικόνας

Για να αλλάξουμε το μέγεθος μιας εικόνας αρκεί να χρησιμοποιήσουμε κατάλληλα στη Matlab την εντολή «imresize».

Η εντολή αυτή χρησιμοποιείται αλλάζοντας τον συντελεστή της όσο χρειάζεται για το επιθυμητό αποτέλεσμα μας όπως στην εικόνα 5.

1.2.1 Παράδειγμα

```
g = imread('10.png');
n = imresize(g,[90 140]);
imshow(g)
figure, imshow(n)
```



Εικόνα 5

1.3 Περιστροφή εικόνας

Παράδειγμα 1.3 Περιστροφή μιας εικόνας.
Επιθυμούμε να περιστρέψουμε μια εικόνα. Για να απλουστεύσουμε τα πράγματα θα χρησιμοποιήσουμε μια εικόνα στα επίπεδα του γκρι.

1.Ο πίνακας της περιστροφής έχει την μορφή :

$$\mathbf{M} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

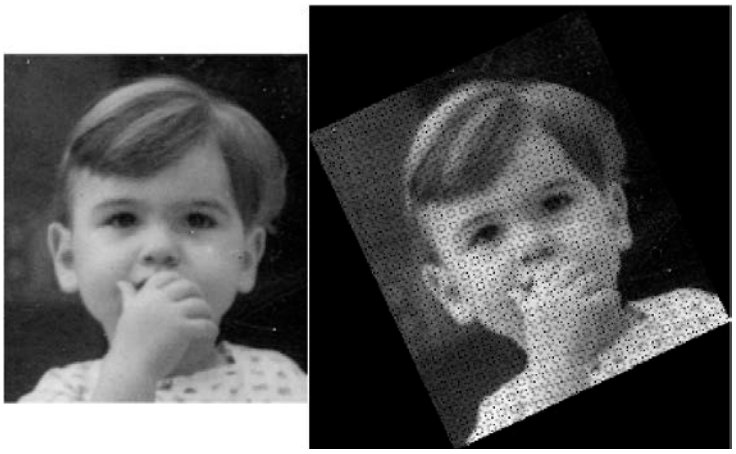
2. Δημιουργούμε έναν πίνακα για τις συντεταγμένες των pixels (διαφορετικές από τους αριθμούς γραμμών και στηλών στις συντεταγμένες X, Y). Η εφαρμογή της περιστροφής σε κάθε σημείο, αφού στρογγυλέψουμε την προκύπτουσα αξία, και αλλάξουμε τη γραμμή, την αναπαράσταση των στηλών, μας οδηγεί στην τελική εικόνα.

Περιστρέφοντας δυο γειτονικά pixel , μπορεί να οδηγήσει μετά την στρογγυλοποίηση σε πανομοιότυπες συντεταγμένες. Αυτό μας οδηγεί στο συμπέρασμα πως υπάρχουν ‘κενά’ στην εικόνα προορισμού. Αυτές είναι εμφανείς στην εικόνα που πρόεκυψε από την περιστροφή. (Εικόνα 6)

Γενικά , οι μετασχηματισμοί με συσχέτιση εκφράζονται ως εξής:

$$\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} = \begin{bmatrix} a & b & t_x \\ c & d & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

X , y είναι οι συντεταγμένες της S πηγής, X , Y είναι οι συντεταγμένες της επιλεγμένης εικόνας C , tx και ty καθορίζουν την μετατόπιση που εφαρμόστηκε στην εικόνα.



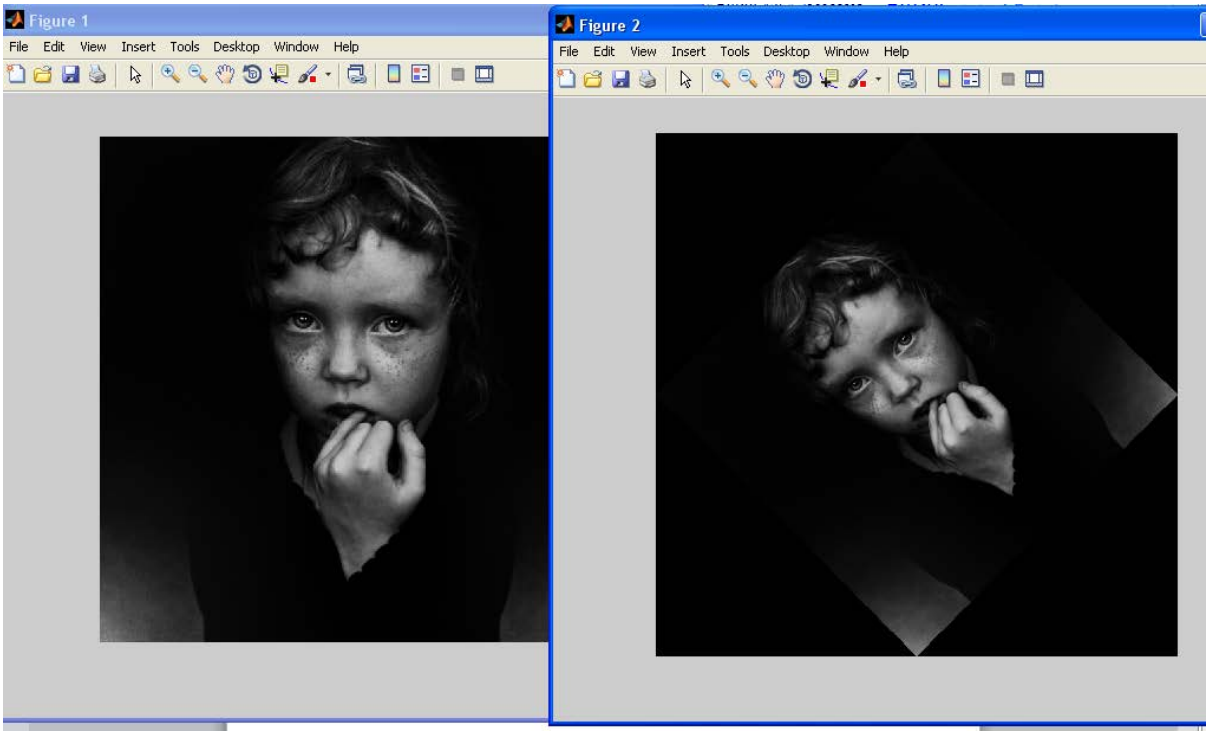
Εικόνα 6

Συνήθως για να μπορέσει να γίνει σε μια ψηφιακή εικόνα περιστροφή πρέπει να ακολουθηθεί συγκεκριμένος αλγόριθμος για να μπορέσει να εφαρμοστεί σωστά το rotate σε όλα τα pixels όπως: nearestneighbor, bilinear, bicubic, spline, sinc, lanczos κ.α.

Στο παρακάτω παραδειγμα για να κάνουμε περιστροφή θα πρέπει να επιλέξουμε την εντολή «imrotate» γράφοντας την γωνία που θέλουμε και τον αλγόριθμο που θέλουμε για να γίνει η περιστροφή όπως στην εικόνα 7.

1.3.1 Παράδειγμα

```
k = imread('10.png');
n = imrotate(k,45,'bilinear');
imshow(k)
figure, imshow(n)
```



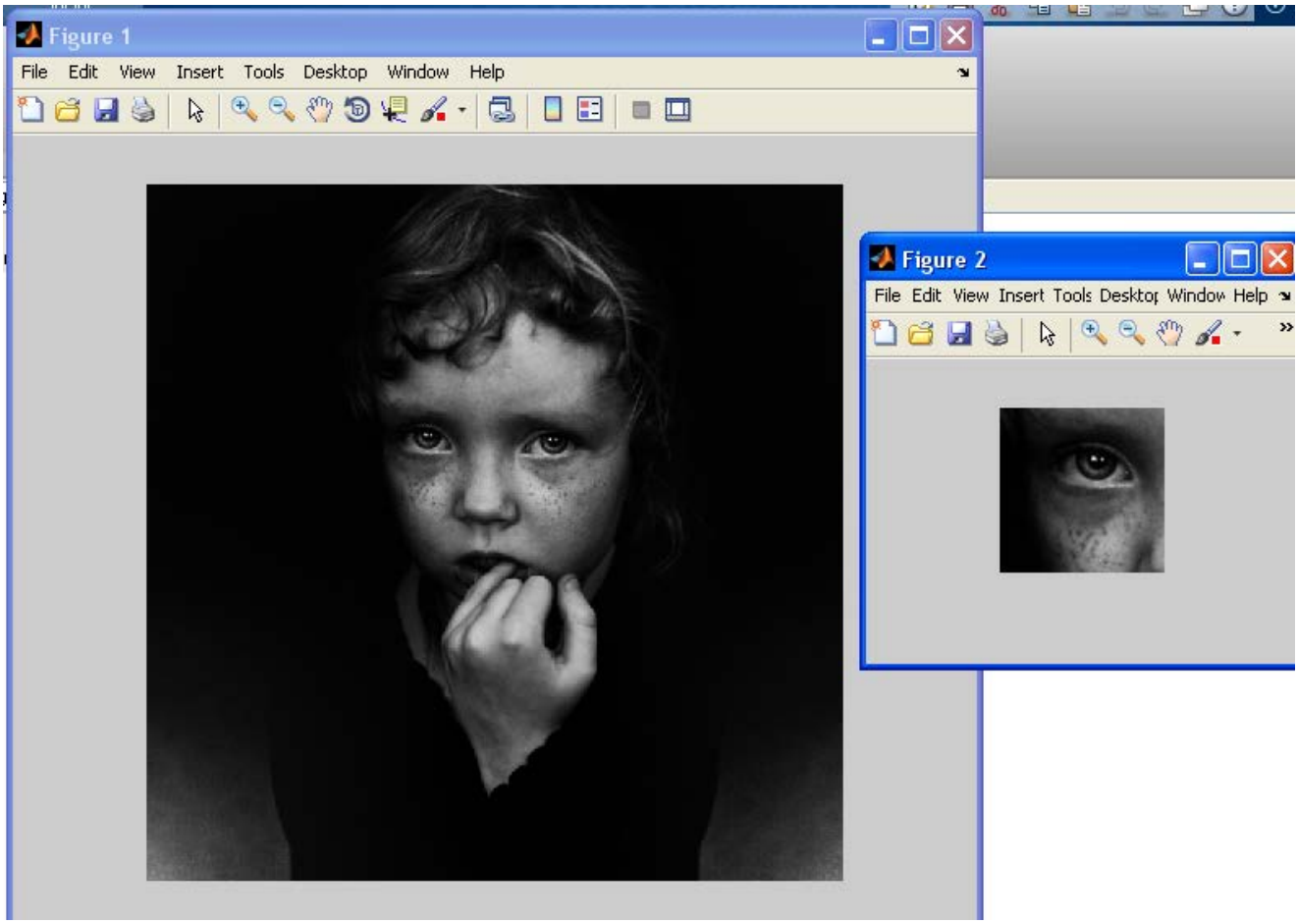
Εικόνα 7

1.4 Περικοπή μιας εικόνας

Για να πάρουμε ένα ορθογώνιο κομμάτι της εικόνας μας, χρησιμοποιούμε τη συνάρτηση imcrop. Η imcropέχει 2 μεταβλητές, την εικόνα που θα εφαρμόσουμε τη εντολή και τις συντεταγμένες του κομματιού που θέλουμε να κρατήσουμε όπως στην εικόνα 8.

1.4.1 Παράδειγμα

```
k= imread('10.png');
n = imcrop(k,[200 200 100 100]);
imshow(k)
figure, imshow(n)
```



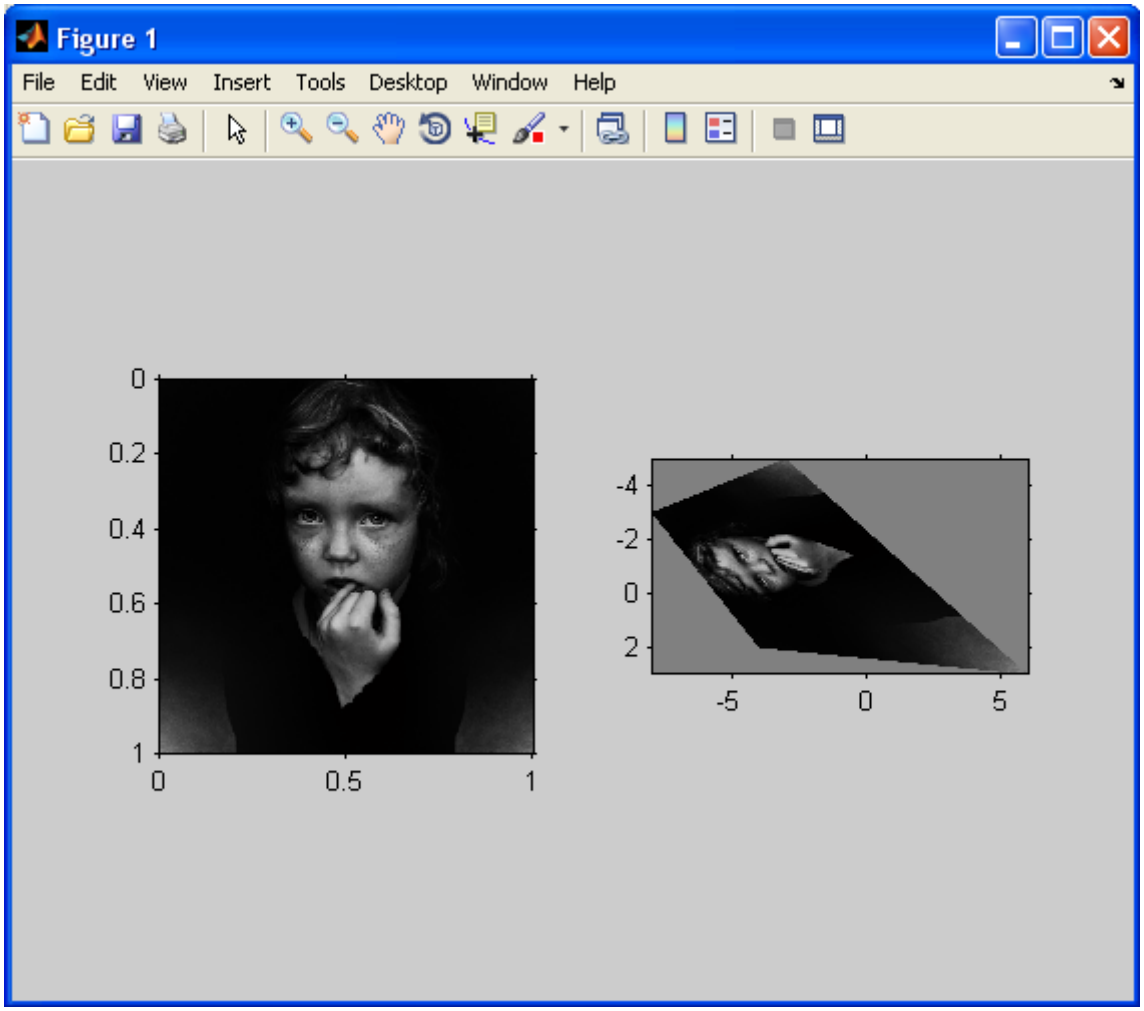
Εικόνα 8

1.5 Μετατόπιση , Κύρτωση

Παίρνουμε την εικόνα που θέλουμε να επεξεργαστούμε και τρέχουμε τον παρακάτω κώδικα πάνω σε αυτή. Θέτουμε τις παραμέτρους των διαστάσεων που θέλουμε και παίρνουμε το επιθυμητό μας αποτέλεσμα που φαίνεται στην εικόνα9.

1.5.1 Παράδειγμα

```
I = imread('10.png');
udata = [0 1];  vdata = [0 1];
tform = maketform('projective',[ 0 0;  1  0;  1  1; 0 1], [-4 2; -8 -3; -
3 -5; 6 3]);
[B,xdata,ydata] = imtransform(I, tform, 'bicubic','udata',
udata,'vdata', vdata, 'size', size(I),'fill', 128);
subplot(1,2,1), imshow(I,'XData',udata,'YData',vdata),axis on
subplot(1,2,2), imshow(B,'XData',xdata,'YData',ydata),axis on
```



Εικόνα 9

ΚΕΦΑΛΑΙΟ 2
ΙΣΤΟΓΡΑΜΜΑ

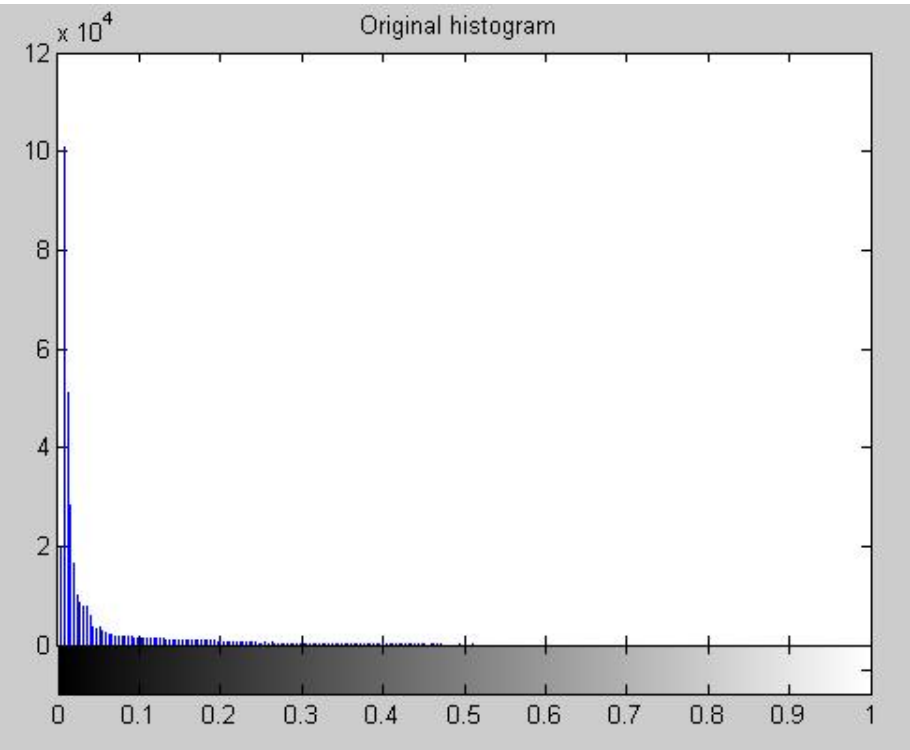
2.1 ΙΣΤΟΓΡΑΜΜΑ

Η αρχική εικόνα παρουσιάζεται στηνεικόνα 7. Το αντίστοιχο ιστόγραμμα είναι στην εικόνα 8. Εάν εξετάσουμε το ιστόγραμμα, βλέπουμε ότι η αντίθεση της εικόνας είναι πολύ χαμηλή επειδή οι περισσότερες greyscale τιμές συγκεντρώνονται γύρω από [0 ,0.5]. οι Greyscale τιμές εκτός αυτού του εύρους εμφανίζονται σπάνια , το όποιο δείχνει πως η διάφορα μεταξύ των χαμηλών και υψηλών τιμών έντασης είναι μικρή σε σχέση με το σύνολο των τιμών που θα ήταν διαθέσιμες για παρουσίαση 8-bit.

Εικόνα 7

Εξίσωση ιστογράμματος

Τελικός στόχος είναι ο υπολογισμός δυο διαφορετικών τιμών αντίθεσης μεταξύ αρχικής εικόνας και εικόνας μετά την εξίσωση. Αυτές είναι η αντίθεση Michelson και η RMS αντίθεση.



Εικόνα 8

Η Michelson αντίθεση ορίζεται ως

$$\frac{I_{\max} - I_{\min}}{I_{\max} + I_{\min}}$$

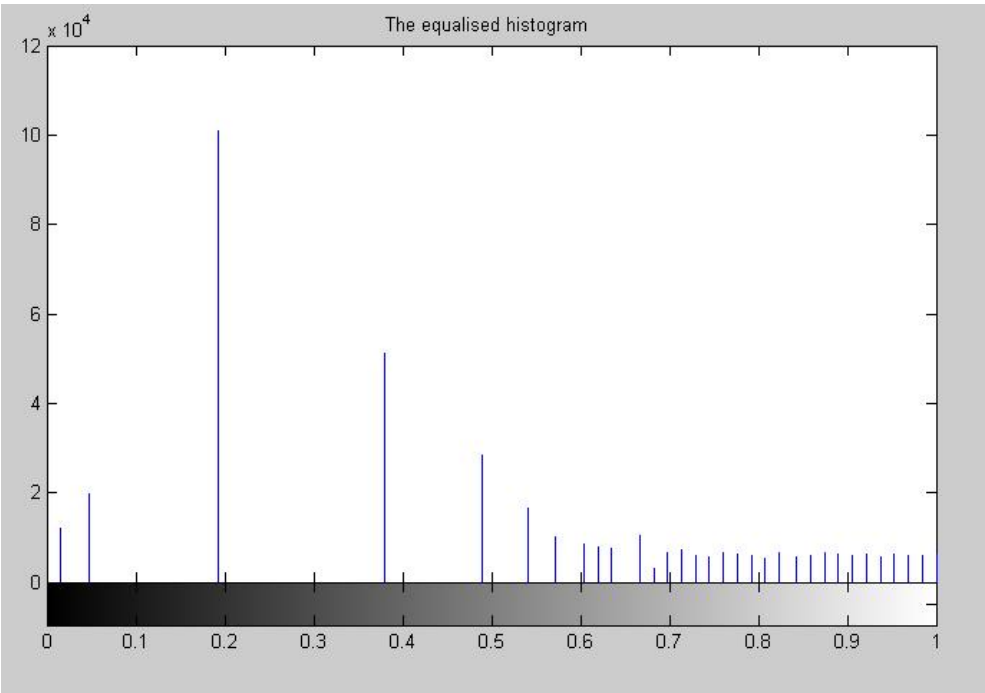
Εξαρτάται μόνο από το ολικό μέγιστο και ελάχιστο.

Η RMS αντίθεση εξαρτάται από όλο το χωρικό περιεχόμενο και ορίζεται ως

$$\sqrt{\frac{1}{MN}\sum_{i=1}^{N}\sum_{j=1}^{N}(I_{ij} - \bar{I})^2}$$

Όπου M και N είναι το πλάτος και ύψος της εικόνας αντίστοιχα και Ī είναι το ολικό μέσο των grayscale τιμών. Οι τιμές δείχνουν ποσοτικά ότι η εξισωμένη εικόνα έχει μεγαλύτερη αντίθεση. Παρόλα αυτά , πρέπει να σημειωθεί πως η αντίθεση Michelson μπορεί να είναι παραπλανητική αν η αρχική εικόνα περιείχε πολύ υψηλό η πολύ χαμηλά ελάχιστα και μέγιστα.

Ακόμα και αν οι περισσότερες grayscale τιμές είναι κεντροθετημένες γύρω από μια στενή περιοχή , η τιμή δεν θα επηρεαστεί.



Εικόνα 9

Η τελική εικόνα παρουσιάζεται στηνεικόνα 10. Το αντίστοιχο ιστόγραμμα είναι στην εικόνα 9. Με την εκτέλεση του κώδικα Matlab που παραθέτουμε στο τέλος του κεφαλαίου, δουλεύουμε πάνω στην αρχική εικόνα 7 καλώντας συγκεκριμένες συναρτήσεις της βιβλιοθήκης Matlabγια να ισοσκελίσουμε το χαμηλό σε αντίθεση αρχικό ιστόγραμμα που είναι στην εικόνας 8. Εάνεξετάσουμε τα αποτελέσματα της εκτέλεσης θα δούμε το τελικό ιστόγραμμα στην εικόνα 9. Βλέπουμε ότι η αντίθεση της εικόνας 10είναι ισοσκελισμένη επειδή οι greyscale τιμές δεν συγκεντρώνονται γύρω από κάποιο συγκεκριμένο σημείο αλλά διασπείρονται σε όλο το φάσμα τιμών [0,1] στο αντίστοιχο ιστόγραμμα της.

Εικόνα 10

ΚΕΦΑΛΑΙΟ 3

3.1 Φίλτρα

Πρώτα όμως πρέπει να δούμε την έννοια τις εφαρμογής φίλτρων σε εικόνες. Ο λόγος για ναφίλτραρουμετις εικόνες στη πιο γενική απάντηση που μπορούμε να πούμε είναι ότι θέλουμε οι επεξεργασμένες εικόνες να είναι καλύτερες από τις πρώτες . Τα φίλτρα που εφαρμόζονται πάνω στα pixels της εικόνας χωρίζονται σε γραμμικά και μη γραμμικά. Ένα σημαντικό χαρακτηριστικό που χρησιμοποιούμε στις εικόνες είναι η έννοια της συνδεσιμότητας . Δηλαδή το να βλέπουμε ποια pixels «γειτονεύουν» με άλλα pixelsόπως παρακάτω στα σχήματα .

Η εφαρμογή των φίλτρων σε μία εικόνα βασίζεται σε αυτή την έννοια της «γετονιάς» .

Δηλαδή η τιμή για κάθε pixel στην εικόνα (target) αντικαθίσταται με μία νέα τιμή η οποία εξαρτάται μόνο από την τιμή των pixels σε μία προκαθορισμένη γειτονιά γύρω από το targetpixel.

```
A=imread('10.jpg');
subplot(1,3,1), imshow(A);
Isp=imnoise(I,‘salt& pepper’,0.03);
subplot(1,3,2), imshow(Isp);
Ig=imnoise(A,’gaussian’,0.02);
subplot(1,3,3), imshow(Ig);
```

Πέρνουμε μία μάσκα όπως την αποκαλούμε, πάνω από την αρχική εικόνα και σαν αποτέλεσμα παίρνουμε μία νέα εικόνα στην οποία οι τιμές των pixels έχουν τιμές που έχουν υπολογιστεί από την εφαρμογή της μάσκας. Τα γραμμικά φίλτρα χρησιμοποιούν μία μάσκα μεγέθους NxN. Στον παρακάτω πίνακα για παράδειγμα απεικονίζεται μία μάσκα μεγέθους 3x3, δηλαδή .

$$\omega = \begin{bmatrix} \omega(-1,-1) & \omega(-1,0) & \omega(-1,1) \\ \omega(0,-1) & \omega(0,0) & \omega(0,1) \\ \omega(1,-1) & \omega(1,0) & \omega(1,1) \end{bmatrix}$$

Η βασική προσέγγιση είναι να αθροιστούν τα γινόμενα μεταξύ των συντελεστών της μάσκας και των εντάσεων των pixels που βρίσκονται κάτω από τη μάσκα, σε μία συγκεκριμένη θέση στην εικόνα.

Η μάσκας είναι στην θέση (x,y) ως κεντρική θεση, η τιμή που έχει η εικόνα εκεί αντικαθίσταται από το προηγούμενο άθροισμα. Η μάσκα έπειτα μετακινείται στο επόμενο pixel και η διαδικασία επαναλαμβάνεταικατατέτοι τρόπο μέχρι να περάσει από όλα τα εικονοστοιχεία. Δηλαδή, αν θεωρήσουμε ως f την αρχική εικόνα και g την εικόνα που προκύπτει μετά την εφαρμογή της μάσκας ω, θα έχουμε για κάθε pixel (x,y).

$$g(x,y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} f(x-i, y-j) * \omega(i,j)$$

Στην περίπτωση του γραμμικού φίλτρου μέσου όρου NxN, ισχύει $\omega_{i,j} = \frac{1}{N*N}$

, δηλαδή στο παράδειγμα, $\omega = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

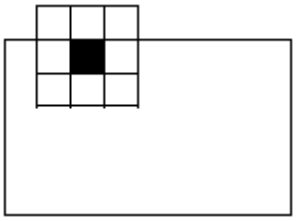
Άρα, αντικαθιστούμε κάθε pixel της εικόνας με τον μέσο όρο των γειτονικών του pixel.

	a	b	c	
	d	e	f	
	g	h	i	

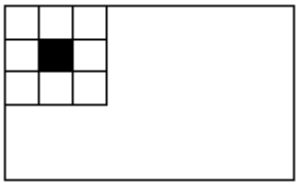
$$\frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

Τα μη γραμμικά φίλτρα έχουν την εξής διαφορά. Η τιμή για κάθε pixel είναι ένα μη γραμμικός συνδυασμός των γειτονικών pixels.

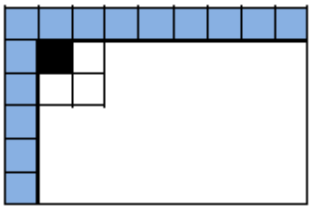
Στα άκρα της εικόνας Εκεί δηλαδή που ένα μέρος της μάσκας έξω από τα όρια της εικόνας όπως φαίνεται στο παρακάτω σχήμα .



Υπάρχουν δύο διαφορετικές τεχνικές στην επίλυση αυτού του προβλήματος. Η μία τεχνική λέει : αγνόησε τα άκρα και εφάρμοσε τη μάσκα εσωτερικά στην εικόνα. Οπότε η μάσκα δεν εφαρμόζεται στα άκρα της εικόνας , και παίρνουμε έτσι μία εικόνα μικρότερη σε μέγεθος από την αρχική. Το μειονέκτημα αυτής της μεθόδου είναι ότι χάνεται πληροφορία, ειδικά όταν η μάσκα έχει μεγάλο μέγεθος.



Η δεύτερη τεχνική γεμίζει την περιοχή που η μάσκα και η εικόνα δεν συμπίπτουν, με μηδενικά, και έτσι επιστρέφεται μία εικόνα που έχει ίδιο μέγεθος με την αρχική, αλλά μπορεί να μην έχει επιθυμητό αποτέλεσμα στα άκρα της εικόνας.



3.2 Θόρυβος

Το βασικότερο εμπόδιο στην ψηφιακή επεξεργασία σήματος αλλά και της εικόνας είναι ο θόρυβος. Με τον όρο θόρυβο εννοούμε μία μεταβολή στην τιμή του σήματος από την πραγματική του τιμή .

Παράδειγμα 3.2

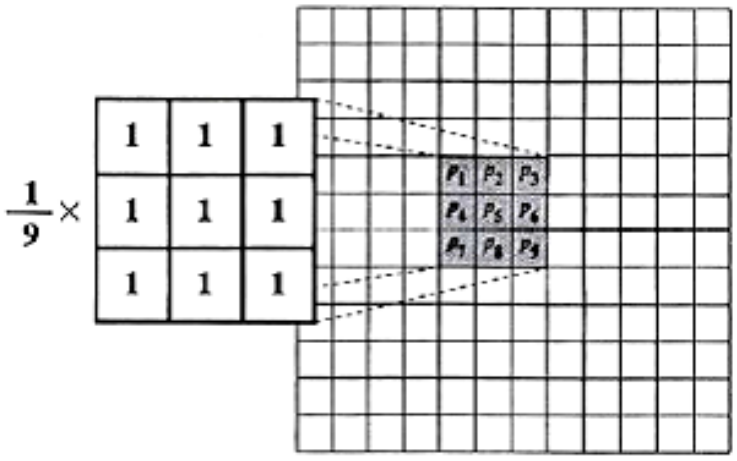
```
A=imread('mlogo.jpg');
subplot(1,3,1), imshow(A);
Isp=imnoise(I,‘salt& pepper’,0.03); subplot(1,3,2),
imshow(Isp);
Ig=imnoise(A,’gaussian’,0.02);
subplot(1,3,3), imshow(Ig);
```

Θα χρησιμοποιήσουμε τώρα φίλτρο του μέσου για να φιλτράρουμε την εικόνα με τον θόρυβο και θα προβάλουμε τα παρακάτω αποτελέσματα.
Όπως θα παρατηρήσουμε το medfilt2 κάνει καλύτερη δουλεία στην αφαίρεση θορύβου.

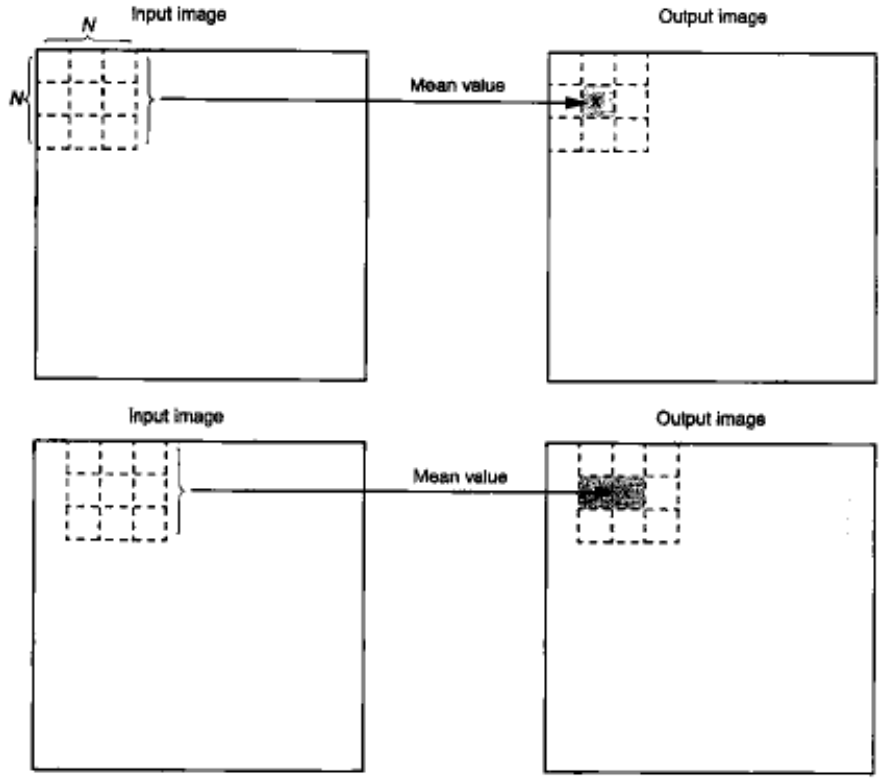
```
L=medfilt2(J,[3 3]);
subplot(1,2,1),imshow(K)
subplot(1,2,2),imshow(L)
```

3.3 ΦίλτροΜέσηςτιμής (Meanfilter)

Το φίλτρο της μέσης τιμής (meanfilter) είναι μια απλή τεχνική εξάλειψης καιεξομάλυνσης του θορύβου από ψηφιακές εικόνες αποχρώσεων του γκρι. Σύμφωνα με την τεχνική αυτή από την αρχική εικόνα παράγεται μία νέα εικόνα ιδίων διαστάσεων όπου κάθε εικονοστοιχείο της έχει φωτεινότητα τη μέση τιμή των τιμών φωτεινότητας μιας γειτονιάς του αντίστοιχου εικονοστοιχείου της αρχικής εικόνας.



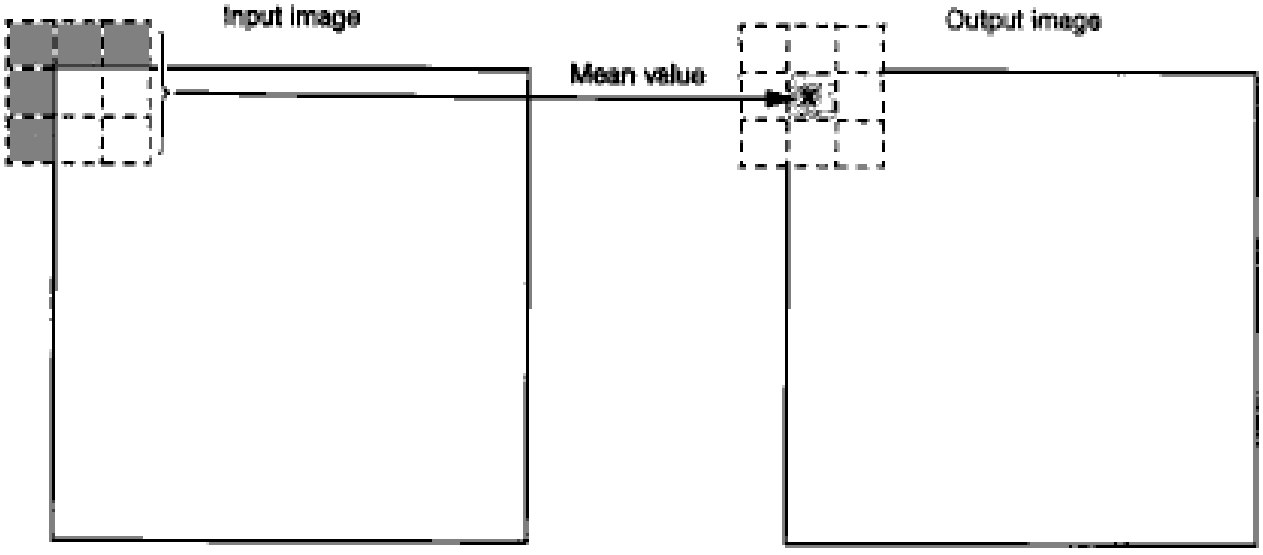
Το φίλτρο μέσης τιμής είναι ένα Lowpass φίλτρο. Το αποτέλεσμα είναι πολύικανοποιητικό όσον αφορά την απαλλαγή από τον θόρυβο (ειδικά αν πρόκειται για Gaussian ή Uniform) αλλά ταυτόχρονα δημιουργείται μια θαμπάδα στην εικόνα.



Η νέα εικόνα είναι το αποτέλεσμα του φιλτραρίσματος της αρχικήςεικόνας με τη διπλανή μάσκα. Είναι δυνατόν να χρησιμοποιηθούν και άλλες μάσκες για την εξομάλυνση μιας εικόνας. Μάσκες 5×5 ή μεγαλύτερες εξομαλύνουν ακόμα περισσότερο την εικόνα.

$$\frac{1}{9}\begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

Σημεία στα άκρα της εικόνας



Για να υπολογιστούν οι τιμές των pixel που βρίσκονται στα άκρα της νέας εικόνας η μάσκα τοποθετείται έτσι ώστε κάποια στοιχεία της βρίσκονται έξω

από τα όρια της αρχικής εικόνας. Σ'αυτή την περίπτωση θεωρείται αυθαίρετα ότι τα στοιχεία της αρχικής εικόνας που δεν υπάρχουν έχουν κάποια συγκεκριμένη τιμή (padding). Συνήθως η τιμή αυτή είναι το μηδέν (zeropadding) ή η κοντινότερη τιμή της αρχικής εικόνας.

Υλοποίηση του φίλτρου μέσης τιμής στο Matlab

Στο Matlab το φίλτρο μέσης τιμής καθώς και άλλα φίλτρα μπορούν να υλοποιηθούν με τη χρήση των παρακάτω συναρτήσεων:

Συνάρτησηfilter2

B = filter2(f,A)

Φιλτράρει τα δεδομένα στον πίνακα A με τη 2-διάστατη μάσκα του πίνακα f. Μετασχηματίζει πάντα τα δεδομένα εισόδου σε double, και η έξοδος είναι πάντα double.

Όταν η μάσκα εφαρμόζεται πάνω σε οριακά pixel του πίνακα εισόδου τα δεδομένα εισόδου που αντιστοιχούν στα στοιχεία που βρίσκονται εκτός ορίων θεωρείται ότι είναι 0 (zeropadding).

Η συνάρτηση filter2 δεν υποστηρίζει άλλες εναλλακτικές δυνατότητες padding.

Συνάρτησηimfilter

B = imfilter(A, f)

Είναι πιο γενική από την filter2. Εδώ ο πίνακας A μπορεί να είναι πολλών διαστάσεων. Μπορείναείναιοποιασδήποτεκλάσης (double, single, logical, char, cell, struct, int8, uint8, int16, uint16, int32, uint32, int64, uint64) και οποιουδήποτε μεγέθους. Το φίλτρο f μπορεί επίσης να είναι πολλών διαστάσεων. Το αποτέλεσμα B έχει τις ίδιες διαστάσεις και την ίδια κλάση με τον A.

Η συνάρτηση imfilter δε μετατρέπει τα δεδομένα εισόδου σε double.

B = imfilter(A, f, X)

Εκτελεί φιλτράρισμα θεωρώντας ότι οι τιμές του πίνακα εισόδου που βρίσκονται εκτός των ορίων, έχουν την τιμή X. Όταν δεν καθορίζεται καμία τιμή, η imfilter χρησιμοποιεί την τιμή X = 0, δηλαδή έχουμε zeropadding όπως και στη συνάρτηση filter2.

B = imfilter(A, f)

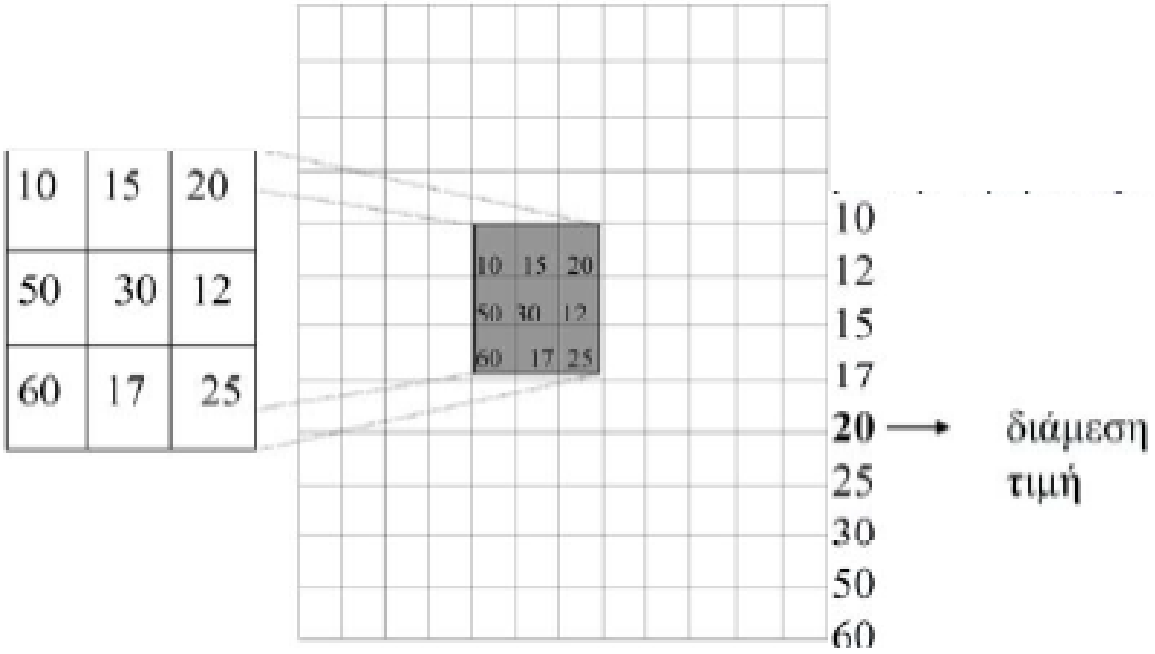
Εκτελεί φιλτράρισμα θεωρώντας ότι οι τιμές του πίνακα εισόδου που βρίσκονται εκτός των ορίων, έχουν την τιμή 0. Όταν ο A είναι double δίνει το ίδιο αποτέλεσμα με την filter2(f,A).

B = imfilter(A, f, 'replicate')

Εκτελεί φιλτράρισμα θεωρώντας ότι οι τιμές του πίνακα εισόδου που βρίσκονται εκτός των ορίων του, έχουν τιμή ίση με την κοντινότερη τιμή του πίνακα.

3.4 Φίλτρο Ενδιάμεσης τιμής (Medianfilter)

Μια άλλη μέθοδος εξομάλυνσης και μείωσης του θορύβου είναι το φίλτρο της ενδιάμεσης ή μεσαίας τιμής. Σύμφωνα με την τεχνική αυτή οι τιμές τωνεικονοστοιχείων μιας γειτονιάς ταξινομούνται και η τιμή κάθε pixel της νέας εικόνας είναι η μεσαία (διάμεση ή ενδιάμεση) από τις τιμές των εικονοστοιχείων της γειτονιάς του αντίστοιχου pixel της αρχικής.



Συνήθως οι εξωτερικές γραμμές και στήλες δεν αντικαθίστανται.
Το φίλτρο αυτό είναι ιδανικό για περιπτώσεις κρουστικού θορύβου (saltandpepper).

Όσο το μέγεθος της μάσκας μεγαλώνει τόσο μεγαλύτερη απώλεια πληροφορίας έχουμε στο αποτέλεσμα που τείνει να μοιάσει τεχνητά ζωγραφισμένο.

Παράδειγμα
Στο παρακάτω παράδειγμα έχουμε κατασκευάσει τη συνάρτηση median_filter με τηνοποία διαβάζουμε τις τιμές των pixel μιας εικόνας, τις αποθηκεύουμε σε ένα πίνακα I και στη συνέχεια εφαρμόζουμε στον πίνακα I με ένα φίλτρο ενδιάμεσης τιμής 3 x 3.

```
functionmedian_filter(name, format)
I = imread(name, format);
figure(1)
imshow(I)
I = double(I);
[r c] = size(I);
G = I;
```

```
for m = 2:r-1;
for n = 2:c-1;
v = sort(reshape(I(m-1:m+1,n-1:n+1),9,1));
G(m,n) = v(5);
end
end
G = uint8(G);
figure(2)
imshow(G)
filename = [name,'_median','.',format];
imwrite(G, filename, format)
```

Η γειτονιά 3 x 3 κάθε pixel με τη συνάρτηση reshape γίνεται ένα διάνυσμα 1 x 9 του οποίου τα στοιχεία με χρήση της συνάρτησης sort μπαίνουν σε διατεταγμένη σειρά και τέλος από αυτά διαλέγουμε το μεσαίο (εδώ είναι το πέμπτο).

Τα εικονοστοιχεία της πρώτης και τελευταίας σειράς και πρώτης και τελευταίας στήλης δεν επηρεάζονται. Αν το φίλτρο ήταν μεγέθους 5 x 5 ο διπλός βρόχος στο παραπάνω παράδειγμα θα γινόταν:

```
for m = 3:r-2;
for n = 3:c-2;
 v = sort(reshape(I(m-2:m+2,n-2:n+2),25,1));
G(m,n) = v(13);
end
end
```

Θα μπορούσαν να απλοποιηθούν κάπως τα πράγματα με τη χρήση της συνάρτησηςmedian(X) του Matlab η οποία αν X είναι διάνυσμα, επιστρέφει την ενδιάμεση τιμή των στοιχείων του X ενώ αν X είναι 2-διάστατος πίνακας η median(X) επιστρέφει ένα διάνυσμα που περιέχει τις ενδιάμεσες τιμές κάθε στήλης του X. Για να πάρουμε το ενδιάμεσο στοιχείο ενός 2-διάστατου πίνακα θα πρέπει να την εφαρμόσουμε δυο φορές: median(median(X)).

Σ' αυτή την περίπτωση και για φίλτρο 3 x 3 ο διπλός βρόχος του παραδείγματος θα γινόταν:

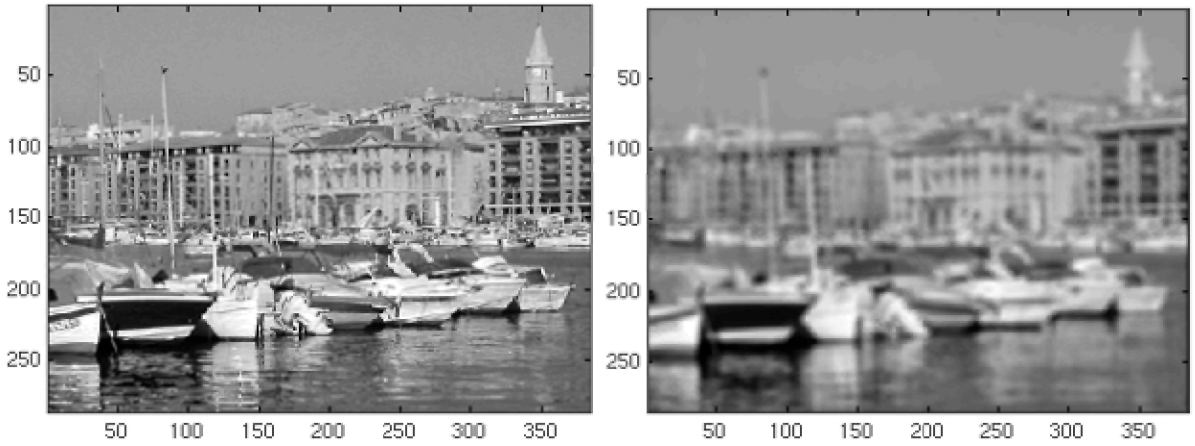
```
for m = 2:r-1;
for n = 2:c-1;
G(m,n) = median(median(I(m-1:m+1,n-1:n+1)));
end
end
```

3.5 Γραμμικόφιλτράρισμα

Η συνάρτησηfilter2 που χρησιμοποιείται στα 2Dφιλτραρίσματα, χρησιμοποιεί την 2D συνέλιξη της οποίας η εντολή στην MATLABαναφέρεται ως c=conv2(a,b).

Κυκλικόφίλτρο

Θεωρούμε το κυκλικόφίλτροh(k,l) το οποίο δίνεταιπαρακάτω το παραπάνω πρόγραμμαομαλοποιεί την εικόνα



Figureομαλοποίηση μιας εικόνας χρησιμοποιώντας το κυκλικόφίλτρο

Το φίλτρο αυτό τείνει να εξαφανίζει τις υψηλέςσυχνότητες ειδικά όσεςπεριέχονται στα περιγράμματα και έτσιπροκύπτει μια εικόνα θολή σε σύγκριση με την αρχική.

3.5 Το Gaussianφίλτροομαλοποίησης

Το Gaussianχαμηλοπερατό φίλτρο ορίζεται στις δύο διαστάσεις ως h(u,v) = (1/(2*π*σ^2)) * e^(- (u^2 + v^2) / σ^2), όπου σ το standarddeviation. Δηλαδή το μεγαλύτερο βάρος έγκειται στο κέντρο και υπάρχει ομαλή εξασθένηση όσο απομακρυνόμαστε από το κέντρο. Συνήθως οι υψηλές συχνότητες στο φάσμα μιας εικόνας συνεισφέρουν στο διαχωρισμό των αντικειμένων δηλαδή στις λεπτομέρειες ενώ οι χαμηλές αντιπροσωπεύουν μεγάλες περιοχές αντικειμένων. Εφαρμόζοντας το χαμηλοπερατό φίλτρο οι υψηλές συχνότητες χάνονται και ο διαχωρισμός των αντικειμένων, οι ακμές, γίνονται περισσότερο δυσδιάκριτες. Με την εφαρμογή του lowpass φίλτρου οι περιοχές της εικόνας που διαχωρίζουν τα αντικείμενα, δηλαδή οι ακμές της θα έρθουν σε πιο κοντινές τιμές μετά την εφαρμογή του φίλτρου, το οποίο τείνει αν εξαλείψει τις σημαντικές διαφορές μεταξύ γειτονικών pixels. Έτσι οι ακμές των αντικειμένων γίνονται δυσδιάκριτες και θολές.

ΠαραδειγμαGaussianΦΙΛΤΡΟΥ

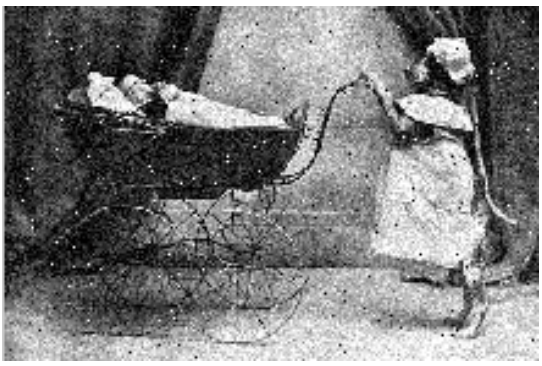
```
A=imread('10.png');
subplot(1,3,1), imshow(A);
Isp=imnoise(I,'salt& pepper',0.03);
subplot(1,3,2), imshow(Isp);
Ig=imnoise(A,'gaussian',0.02);
subplot(1,3,3), imshow(Ig);
```

Στο παραπάνω παράδειγμα εισάγουμε θόρυβο γνωστό στη Matlabως τύπο salt&pepper.

Έτσι αφού βάλουμε αυτό τον θόρυβο τότε θα προσπαθήσουμε, με την εφαρμογή του Gaussianφίλτρου και εισάγοντας την ανάλογη στον θόρυβο παράμετρο, να επεξεργαστούμε την εικόνα μας εφαρμόζοντας τον κώδικα στην παρακάτω εικόνα και παίρνοντας τα παρακάτω αποτελεσματα.

Αρχική

Προσθέσαμε θόρυβο



Εφαρμογή φίλτρου Gaussian

3.6Ο 2D-DFTμετασχηματισμός συχνοτήτων

Ξεκινώντας με την κυκλική συνέλιξη είναι πιθανό να υλοποιήσουμε ένα φιλτράρισμαπολλαπλασιάζοντας τον **2D-DFT μιας εικόνας** με τον **2D-DFT του** PSF του φίλτρου.

Θεωρούμε τον PSF*h(k,l)* ενός *K*x*L φίλτρου και μια M*x*N εικόναx(k,l).*

Θα υποθέσουμε*M>K*and*N>L.*

Τα H(m,n) andX(m, n) αναφέρονται στο 2D-DFT του PSF και της εικόνας αντίστοιχα.

Και τα δυο υπολογίζονται για *M*x*Nσημεία.*

Συμφώνα με το (κυκλικόφιλτράρισμα) το αντίστροφο2D-DFT του παραγωγού*H(m,n)X(m,n)μπορεί να γραφτεί , για k*E{O,...,M-1} και για LE*{O, . , . , N* - 1}:

$$y(k,\ell) = \sum_{u=0}^{K-1} \sum_{v=0}^{L-1} h(u,v)x(k-u \bmod M, \ell - v \bmod N)$$

για *k>K* — 1, *(k — u*mod*M) = k — u: δεν έχει περιεχόμενο*aliasing όταν καταλήγουμε στο *u*από το 0 στο *(K* — 1).

Αυτό επίσης ισχύει και για l>= L - 1, (l - v mod n) = l – v.
Στην περίπτωση αυτή, τα υπολογισμένα σημεία αντιστοιχούν με εκείνα της συνέλιξης που συνδέονται με το φιλτράρισμα. Ωστόσο, για k <K - 1 και / ή l <N-1. Υπάρχει ένας δείκτης "aliasing", το οποίο οδηγεί σε εσφαλμένο αποτέλεσμα. Ένας τρόπος για να αποφευχθεί αυτό το φαινόμενο είναι η ολοκλήρωση της εικόνας με μηδενικά K κατά μήκος του οριζόντιου άξονα, και L μηδενικά κατά μήκος του κάθετου άξονα.

2-D διακριτοςμετασχηματισμοςFourier2D-DFT

Ο 2D διακριτός μετασχηματισμός Fourier, ή 2D-DFT, από την πεπερασμένη ακολουθία {x (k,l)}, με k E{0, ..., M - 1} και lE{0, ..., N - 1 }, είναι η αλληλουχία που ορίζεται, για m E{0, ..., M - 1} και nE{0, ..., n - 1}, από:

$$X(m,n) = \sum_{k=0}^{M-1}\sum_{\ell=0}^{N-1} x(k,\ell)\exp\left\{-2\pi j\left(\frac{km}{M}+\frac{\ell n}{N}\right)\right\}$$

Εάν αλλάξουμε την έκφραση της X (m, n) για:

$$X(m,n) = \sum_{\ell=0}^{N-1}\left(\exp\left\{-2\pi j\frac{\ell n}{N}\right\}\sum_{k=0}^{M-1} x(k,\ell)\exp\left\{-2\pi j\frac{km}{M}\right\}\right)$$

για κάθε τιμή του l, η 1D-DFT της ακολουθίας x (k, l) της μεταβλητής k εμφανίζεται στην παρένθεση. Με MATLAB®, οι FFTs N αντιστοιχεί στην δεύτερη έκφραση παραπάνωκαι υπολογίζεται εφαρμόζοντας τη λειτουργία fft στο (M x N) πίνακα x. Η 2D-DFT στη συνέχεια επιτυγχάνεται απλά με την εκτέλεση μιας άλλης FFT για την προκύπτουσα μεταφορά πίνακα.

Για να συνοψίσω, η 2D-DFT λαμβάνεται με τον τρόπο:

fft(fft(x).')·'

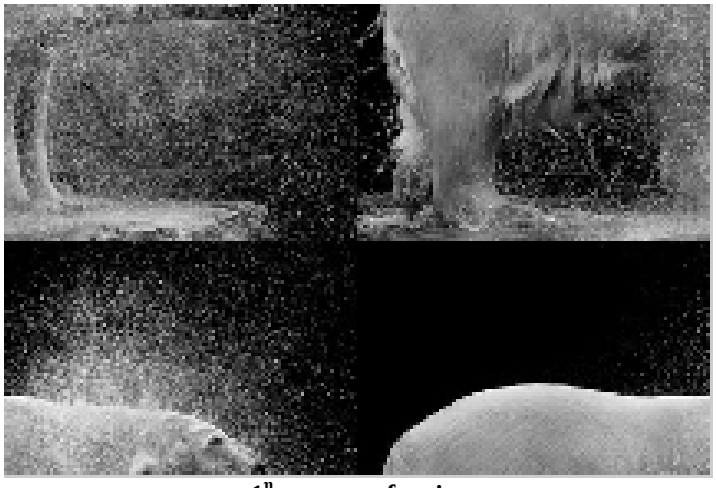
Η λειτουργία fft2, είναι διαθέσιμη στη βασική έκδοση του MATLAB, και εκτελεί την ίδια λειτουργία. Πληκτρολογώντας fft2 (x, M, N) συμπληρώνει, εάν είναι αναγκαίο, η συστοιχία χ με μηδενικά έτσι ώστε να έχει μια σειρά M χ N. Είναι συχνά προτιμότερο να εμφανιστούν οι χωρικές συχνότητες με τιμές μεταξύ 0 και 1, ή μεταξύ 0 και 1/2.

Παράδειγμα 2D DFT

```
I = imread('11.png');
subplot(3,1,1)
imshow(I)
F= fftshift(I)
subplot(3,1,2)
imshow(F)
k= fftshift(F)
subplot(3,1,3)
imshow(k)
```

έτσι έχουμε τα παρακάτω αποτελέσματα

Αρχική εικόνα



1η εφαρμογη fourier

2η εφαρμογή fourier

ΚΕΦΑΛΑΙΟ 4

Ανίχνευση Ακμών

4.1 Ανίχνευση ακμών

Χρησιμοποιώνταςτοορο ακμές για μια ασπρόμαυρη εικόνα, εννοουμε τις αλλαγές της φωτεινότητας μεταξύ τωνγειτονικών περιοχών της.

Αλλαγές τηςφωτεινότητας αντιστοιχούν συνηθωσε διαφοροποίηση ιδιοτήτων της απεικόνισης τρισδιάστατων αντικειμένων όπως αλλαγές στηνυφή, στοβάθος, στα όρια των αντικειμένων καισε διαφορετικό φωτισμό και αντανάκλαση.

Ωσεκ τουτου όταν ανιχνευουμεακμες μπορούμε να εξαγουμε πληροφορίες για τις φυσικές ιδιότητες για τα εικονιζόμενα πραγματικά αντικείμενα.

Η ανίχνευση ακμών μιας εικόνας εμφανιζειπολλές δυσκολίες. Οι ακμές ενδεχεται να χαρακτηρίζονται από προοδευτικές ή ακόμα και πολύ μικρές αλλαγές στην φωτεινότητα της εικόνας. Όταν ο θορύβος είναι παρων σε μια εικόνα ,μπορει να οδηγήσει στην ανίχνευση εσφαλμένων ακμών αλλοιώνοντας τα όρια των αντικειμένων. Το διαφορετικό φώς στα χρώματαπολλές φορες δημιουργεί ψεύτικες ακμές ενώ δεν αντιστοιχούν σε κανονικες ακμές.Η ανίχνευση ακμών περιεχεται στις εφαρμογές για την ανάλυση εικόνων με τη βοήθεια αλγεβρικών τελεστών (όπως το άθροισμα, η διαφορά, το γινόμενο και το πηλίκο). Ειδικοτερα,οσων αφορά στην ανίχνευση ακμών, το να αφαιρεθουνκαποιες εικόνες μπορεί να χρησιμοποιηθεί για τον υπολογισμό μιας σημαντικής παραγώγου της εικόνας, της βαθμίδας. Βασισμενοιστην ανίχνευση ακμών, κάθε ένα από τα εικονοστοιχεία της εικόνας εξετάζεται ωστε να προσδιοριστεί εάν και ποσο ανήκει στο σύνορο ενός αντικειμένου. Τέτοιου είδους pixelκαθοριζονται ως στοιχεία ακμών. Η ανίχνευση ακμών συντελειται με εξέταση της περιοχής καθενοςpixel και με τον υπολογισμό του ποσου της κλίσης και της κατεύθυνσης της αλλαγής επιπέδων γκρίζου(graylevels). Για τον σκοπό αυτό χρησιμοποιούνται καποιοισυνελικτικοί τελεστές.Για παράδειγμα ανίχνευση ακμών κατά Roberts, Sobel, Prewitt, Kirsch.

4.2 ΑνίχνευσηακμώνκατάRoberts, Sobel, Prewitt

Ο κώδικας που χρησιμοποιείται για την υλοποίηση των διάφορων αλγορίθμων ανίχνευσης ακμών έχει ως εξής:

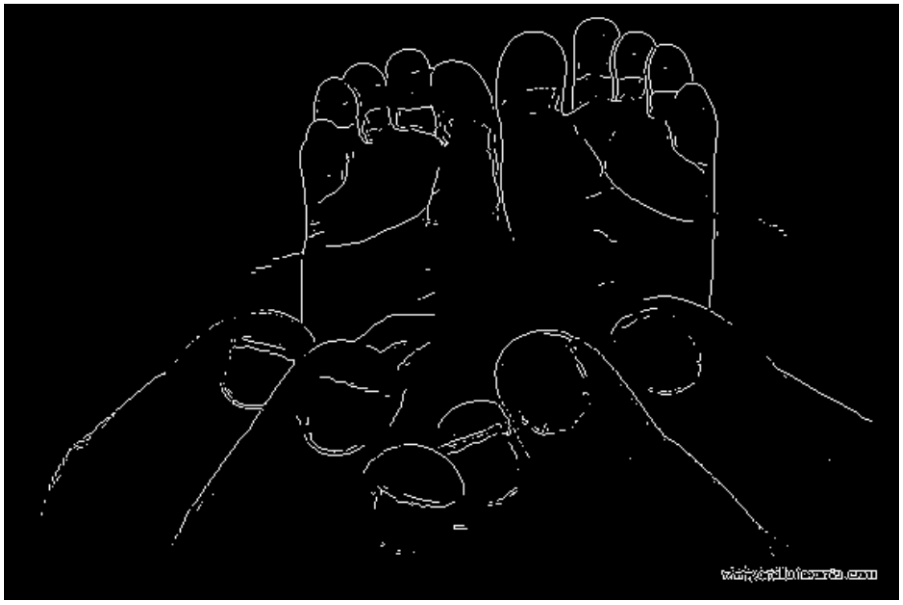
```
I = imread('10.png')
prewitt = edge(I,'prewitt');
```

```
roberts = edge(I,'roberts');
sobel = edge(I,'sobel');
figure, imshow(prewitt)
figure, imshow(roberts)
figure, imshow(sobel)
```

Στις οθόνες που ακολουθούν, επιστρέφεται το αποτέλεσμα της εικόνας που προκύπτει από την εφαρμογή του κάθε αλγορίθμου πάνω στο αρχικό αρχείο.

Αρχική εικόνα

Ανίχνευση ακμών Prewitt

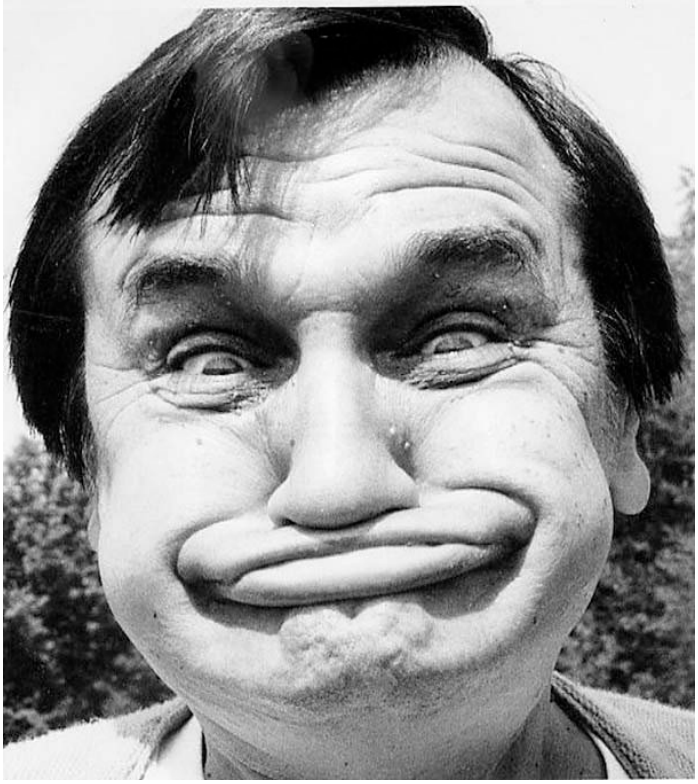


Ανίχνευση ακμών Roberts

Ανίχνευση ακμών Sobel

4.3 Αλγόριθμος kirch

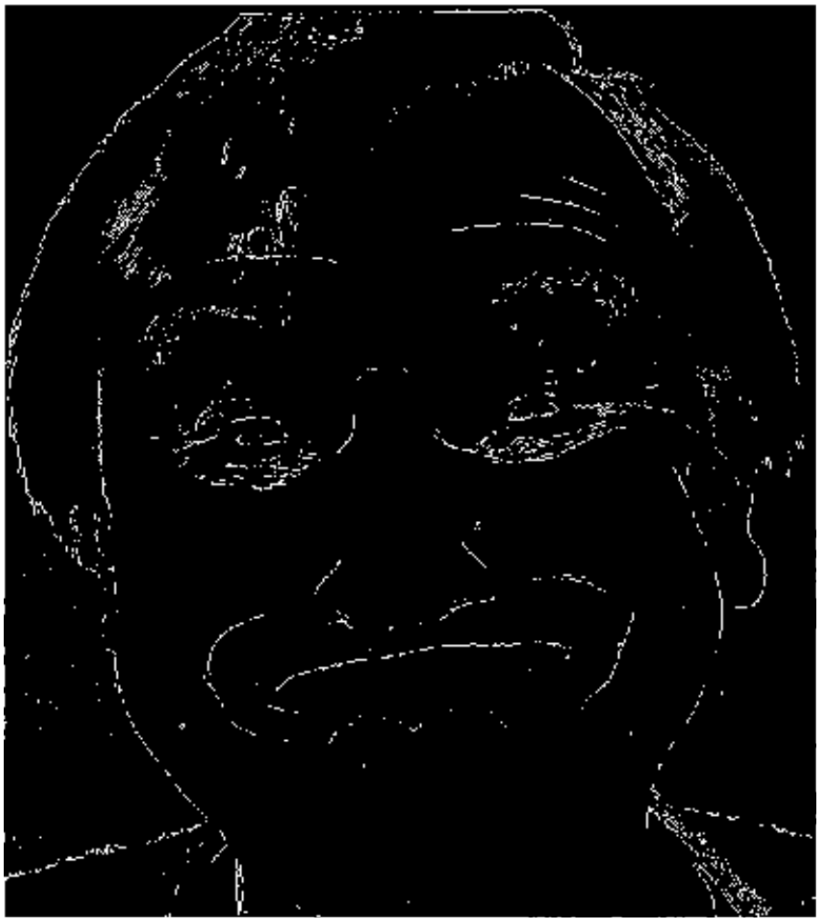
όσων αφορά στην ανίχνευση ακμών σύμφωνα με τον αλγόριθμο «kirch», ο κώδικας και τα αρχεία που χρησιμοποιούνται απεικονίζονται παρακατω:



```
function [imout, thresh] = edge_ed( im, method, thresh, param2 )
[n,m]= size(im);
xx= 2:m-1;
yy= 2:n-1;
ifstrcmp(method,'kirsch')
filt1= [1 2 1;0 0 0;-1 -2 -1]; fim1= conv2(im,filt1,'same');
filt2= [2 1 0;1 0 -1;0 -1 -2]; fim2= conv2(im,filt2,'same');
filt3= [1 0 -1;2 0 -2;1 0 -1]; fim3= conv2(im,filt3,'same');
filt4= [0 1 2;-1 0 1;-2 -1 0]; fim4= conv2(im,filt4,'same');
imo= reshape(max([abs(fim1(:)) abs(fim2(:)) abs(fim3(:)) abs(fim4(:))]'),n,m);
ifnargin<3
thresh= 2* mean(mean( imo(yy,xx) )) ;
end
imout= imo>= thresh ;
xpeak= imo(yy,xx-1) <= imo(yy,xx) &imo(yy,xx) >imo(yy,xx+1) ;
ypeak= imo(yy-1,xx) <= imo(yy,xx) &imo(yy,xx) >imo(yy+1,xx) ;
imout(yy,xx)= imout(yy,xx) & ( xpeak | ypeak );
end

im = imread('6.png');
[edgeim, thresh] = kirsch (im,'kirsch',0);
figure(1), imshow(edgeim);
```

Το αποτέλεσμα που επιστρέφεται από την εκτέλεση του αλγόριθμου, σε ότι αφορά την ανίχνευση ακμών πάνω στην αρχική εικόνα έχει ως εξής:



Όλοι οι τελεστές δεν έχουν την επιθυμητή συμπεριφορά παρουσία θορύβου και ειδικά για τον λευκό θόρυβο. Οι τελεστές Robinson,Kirch n δεν πλησιάζουν καν τις πραγματικές ακμές, ακόμα και για μια τόσο απλή εικόνα. Ο λόγος που συμβαίνει αυτό είναι ότι οι μητρώα συνέλιξης που χρησιμοποιήσαμε, στην ουσία αποτελούν ψηφιακά υψιπερατά φίλτρα. Έτσι, ενισχύουν τον υψίσυχνο θόρυβο οδηγώντας την έξοδο μακριά από τα επιθυμητά αποτελέσματα. Μια λύση είναι να χρησιμοποιήσουμε μη γραμμικά φίλτρα πριν την συνέλιξη της εικόνας με τα μητρώα ανίχνευσης ακμών. Για παράδειγμα ο saltandpepper θόρυβος μπορεί να εξαλειφθεί με ένα φίλτρο μέσης τιμής, όμως δεν θα έχει την ίδια επίδραση και για τον λευκό θόρυβο. Καταλαβαίνουμε πως δεν είναι μια λύση που θα δίνει πάντα αξιόπιστα αποτελέσματα.

ΚΕΦΑΛΑΙΟ 5
ΚΑΤΩΦΛΙΩΣΗ

5.1 Κατωφλίωση

Τα γκρι επίπεδατων pixels που ανήκουν στο αντικείμενο διαφέρουν από τα γκρι επιπεδατων pixels που ανήκουν στο φόντο.

Η κατωφλίωσηείναιένας τρόπος να ξεχορίσουμε ολόκληρα σχήματαμιας εικόνας ενα φόντοαυτο πουσε μερικέςεικόνες είναι απλόάλλα και αποτελεσματικό. Η κατωφλίωση μιας εικόνας δίνεισυνήθως ως αποτέλεσμα μια δυαδική (binary) εικόνα.Στο αποτέλεσμα, το προσκήνιο μαρκάρει τα αντικείμενα που θέλουμε στην εικόνα.

Οι τιμές των pixels μπορεί να είναι 0 (μαύρο)και 255 (ασπρο).

Προφανώς σε τέτοια διαδικασία υπάρχουν πολλοιπαραγοντες που επιρεάζουν το αποτέλεσμα μας. Όπως ένας απλός θόρυβος, φωτεινότητα, το εύρος τωνgraylevelsστις εικόνες μας.

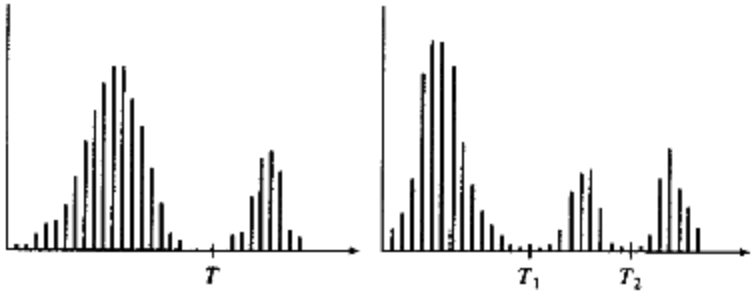
5.2Τι εννοούμε κατωφλίωση

Βλέπουμε το ιστογραμμα της εικόνας20 αντιστοιχεί σε μια εικόνα.

Τα εικονοστοιχεια του αντικειμένου και του background έχουν ομαδοποιημένα γκρι επιπεδα σε δύο βασικες περιοχές. Τότε ένας τρόπος για να εξάγουμε το αντικείμενο που μας ενδιαφερειαπευθειαςαπό το background, είναι να επιλεξουμε ένα κατώφλι *T* που να διαχωρίζει αυτές τις δύο περιοχές.

T λεγεταιτο σημείο του αντικειμένου που μας ενδιαφερει. Σε αντιθετηπεριπτωσηλεγεται σημείο του background.

Η εικόνα20 βλέπουμε την γενική έννοια της κατωφλίωσης, στην οποια 3 περιοχές χαρακτηρίζουν το ιστόγραμμα της εικόνας. Φωτεινά αντικείμενα σε ένα σκοτεινό background.



Εικόνα 20

Εικόνα 20: Gray-level ιστογράμματα

5.3 Αυτόματη επιλογή κατωφλίου των Gonzales και Woods

Μια πρωτηπροσεγγιση για την αυτοματηεπιλογηκατωφλιουπραγματοποιηθηκε από τους Gonzales και Woods [1]η όποια περιγράφει την ακολουθη επαναληπτική διαδικασία :

Στην αρχηεπιλέγεται μια αρχική εκτίμηση για το κατώφλι Τ (μέσο μεταξύ ελάχιστης και μέγιστης τιμής τωνlevels)

Διαχωρίζουμε την εικόνα χρησιμοποιώντας το Τ. Θα προκύψουν δυο ομάδεςpixels.

Υπολογίζουμε μέση τιμή έντασης για κάθε περιοχής.

Και βρίσκουμε το νέο κατώφλι

Επαναλαμβάνουμεωσπου η διαφορά μεταξύ των τιμών του Τα να γίνει μικρότερη από την αρχική ή σε όποιο όριο εμεις επιλέξουμε

5.4 Μη παραμετρική μέθοδος Otsu

Η μέθοδος κατωφλιωσηςτου Ostu (Otsu 1979) βασίζεται στην επεξεργασία του ιστογράμματος της εικόναςκαι στον προσδιορισμό του κατωφλιού με βάση το κριτηριο της μεγιστοποίησης της διαχωρισιμότητας μεταξύ των περιοχών κειμένου και υποβάθρου.

Πρωτο βήμα είναι να υπολογίζουμε το ολικό τετράγωνο της τυπικής απόκλισης (globalvariance) των επιπέδων του γκρι της εικόνας.

Υποθέτουμε ότι τα pixels της εικόνας διακρίνονται σε L διακριτά επίπεδα graylevel [1, 2, …, L] και niέστω το πλήθος των εικονοστοιχειωνπου έχουν επίπεδο γκρίζου τόνου i και N=n1+n2+…+nL το συνολικό πλήθος των εικονοστοιχειωντης εικόνας.

$$pi = \frac{ni}{N} \text{ και } pi \geq 0, \sum_i^{L-1} pi = 1 \quad (4.3.1)$$

Υποθέτουμε ότι για μια τιμή κατωφλίου T δημιουργούνται δύο κλάσεις C0 και C1 (παρασκήνιο και αντικείμενα ενδιαφέροντος, αντίστοιχα). Η C0 αναφέρεται στα pixels με τιμές επιπέδων γκρίζου [1,2,...,T], και η C1 αναφέρεται στα pixels με τιμές επιπέδων γκρίζου [T+1,T+2,...,L].

$$\omega_0 = \Pr(C_0) = \sum_{i=0}^{T} pi = \omega(T) \quad (4.3.2)$$

$$\omega_1 = \Pr(C_1) = \sum_{i=T+1}^{L} pi = 1 - \omega(T) \quad (4.3.3)$$

$$\mu_0 = \sum_{i=0}^{T} iPr(i|C_0) = \sum_{i=0}^{T} ipi/\omega_0 = \mu(T)/\omega(T) \quad (4.3.4)$$

$$\mu_1 = \sum_{i=T+1}^{L} iPr(i|C_1) = \sum_{i=T+1}^{L} ipi/\omega_1 = \frac{\mu_T - \mu(T)}{1-\omega(T)} \quad (4.3.5)$$

Όπου $\omega(T) = \sum_{i=0}^{T} pi$ και $\mu(T) = \sum_{i=0}^{T} ipi$ είναι οι μηδενικής και πρώτης τάξης, αντίστοιχα, αθροιστικές ροπές του ιστογράμματος μέχρι το κατώφλι T, και

$$\mu_T = \mu(L) = \sum_{i=1}^{L} ipi \quad (4.3.6)$$

είναι η ολική μέση τιμή για την εικόνα. Για οποιοδήποτε T εύκολα βλέπουμε πως ισχύει :

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T \text{ και } \omega_0 + \omega_1 = 1 \quad (4.3.7\ \alpha,\beta)$$

Οι διακυμάνσεις των κλάσεων είναι :

$$\sigma_0^2 = \sum_{i=1}^{T}(i-\mu_0)^2 \Pr(i|C_0) = \sum_{i=1}^{T}(i-\mu_0)^2 \mathrm{pi}/\omega_0 \qquad (4.3.8)$$

$$\sigma_1^2 = \sum_{i=T+1}^{L}(i-\mu_1)^2 \Pr(i|C_1) = \sum_{i=1}^{T}(i-\mu_1)^2 \mathrm{pi}/\omega_1 \qquad (4.3.9)$$

Για την αξιολόγηση της απόδοσης του διαχωρισμού σε κλάσεις ανάλογα με την επιλογή της τιμής του κατωφλίου, εισάγουμε τους εξής όρους:

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2}, \kappa = \frac{\sigma_T^2}{\sigma_W^2} \text{ και } n = \frac{\sigma_B^2}{\sigma_T^2} \quad (4.3.10)$$

Όπου $\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2$ και

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 = \omega_0\omega_1(\mu_1 - \mu_0) \quad (4.3.11)$$

Και $\sigma_T^2 = \sum_{i=0}^{L}(i-\mu_T)^2 p_i$ (4.3.12)

Όπου σ_W^2 είναι η διακύμανση μεταξύ των κλάσεων, σ_E^2 η διακύμανση σε κάθε κλάση και σ_T^2 η ολική διακύμανση. Επομένως, το πρόβλημα μπορεί να αντιμετωπιστεί σαν ένα πρόβλημα βελτιστοποίησης για την επιλογή της τιμής του T που μεγιστοποιεί κάποιον από τα όρους τα σχέσης (9).

Όμως, επειδή ισχύει : $\sigma_W^2 + \sigma_B^2 = \sigma_T^2$ (4.3.13)

οι τρεις όροι της (9) είναι ισοδύναμοι *[κ=1+λ, n=λ/(λ+1)],* άρα αρκεί να βρεθεί το T που μεγιστοποιεί οποιονδήποτε όρο από τα λ, κ, n. Επειδή ο υπολογισμός του n είναι ευκολότερος, επιλέγουμε να λύσουμε ως προς αυτό. Είναι (συναρτήσει του T) :

$$n(T) = \frac{\sigma_B^2(T)}{\sigma_T^2} \quad (4.3.14)$$

$$\Rightarrow \sigma_B^2(T) = \frac{[\mu_T\omega(T) - \mu(T)]^2}{\omega(T)[1-\omega(T)]} \quad (4.3.15)$$

Τελικά, η βέλτιστη τιμή κατωφλίου Topt, που μεγιστοποιεί το n, είναι το T που μεγιστοποιεί την ποσότητα:

$$\sigma_B^2(Topt) = \max_{1\le T\le L} \sigma_B^2(T) \quad (4.3.16)$$

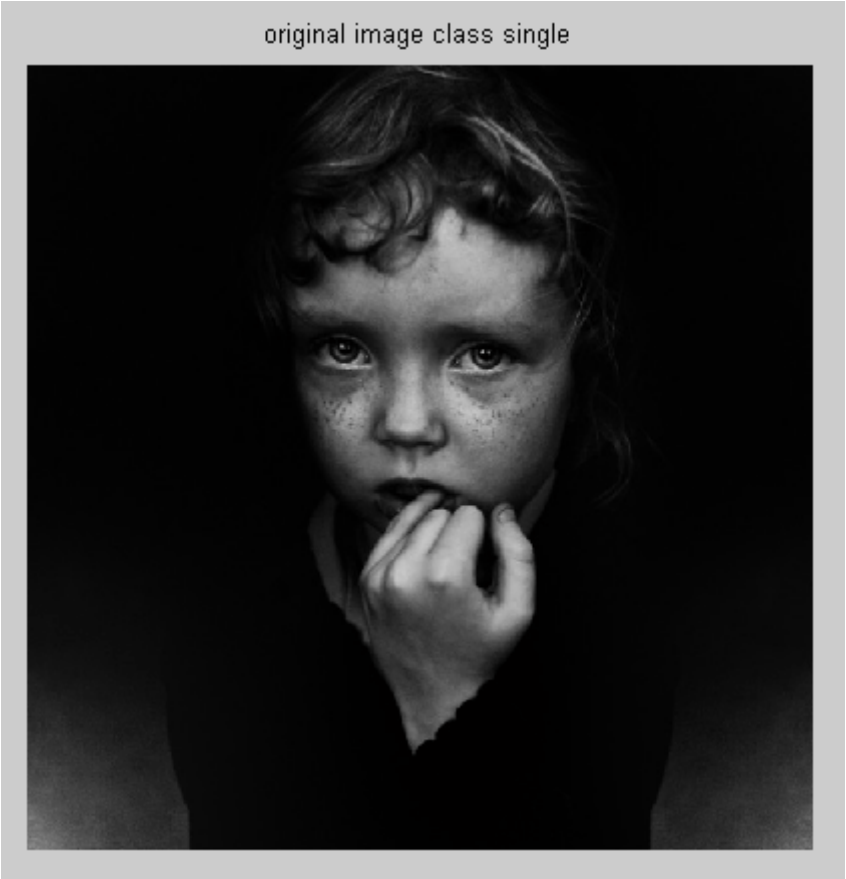
Στο matlab, η συνάρτηση graythresh υλοποιεί τον αλγόριθμο του Otsu. Η graythresh δέχεται σαν όρισμα μια εικόνα, υπολογίζει το ιστόγραμμά της και έπειτα βρίσκει την τιμή του κατωφλίου που μεγιστοποιεί την σ_B^2. Η σύνταξη της συνάρτησης είναι:

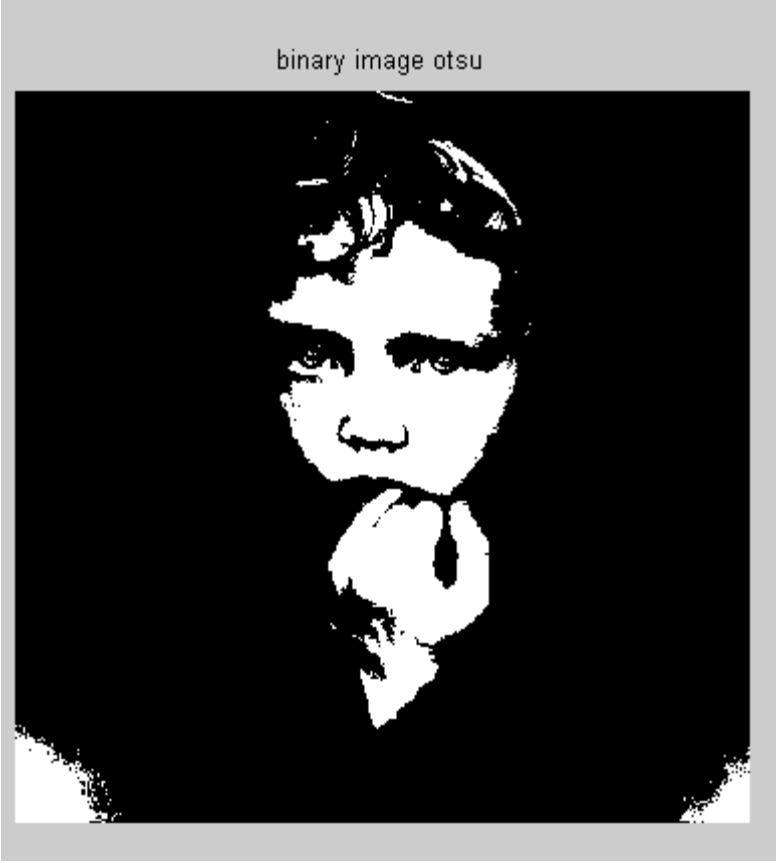
T=graythresh(M)

όπου M η εικόνα εισόδου και T το τελικό κατώφλι. Για να εμφανιστει ηκατωφλιωμένη εικόναχρησιμοποιείται την τιμή του T που υπολογίζεται από τη συνάρτηση αυτή, στη συνάρτηση im2bw. Επειδήομως το κατώφλι είναιστις τιμές [0,1], θα πρέπει να σχετίσουμε την τιμή του στην τάξη της φωτεινότητας της εικόνας πριν χρησιμοποιηθεί, π.χ. αν η f είναι της τάξης uint8, πολλαπλασιάζουμε την τιμή του T με το 255.

Παρακάτω φαίνεται η εφαρμογή της παραπάνω συνάρτησης:

```
close all
clear all
filename='1.png';
I=imread(filename);
figure,imshow(I);title('original image');
Io = im2single(I);
figure
imshow(Io);title('original image class single')
T=graythresh(Io);
BW=im2bw(Io,T);
figure,imshow(BW);title('binary image otsu')
```


original image class single


binary image otsu

5.5 Προσαρμόσιμηκατωφλίωση

Η προσαρμόσιμη η αλλιωστοπικηκατωφλίωσηδέχεται συνήθως μια γκριζουτονου ή έγχρωμη εικόνα σαν είσοδο και, στην πιο απλη εφαρμογή της, εξάγει μια δυαδική εικόνα που ισοδυναμει με το διαχωρισμό του αντικειμένου ενδιαφέροντος από το φοντο (background). Για καθεναεικονοστοιχειο στην εικόνα πρέπει να υπολογιστεί ένα κατώφλι. Αν η τιμή του εικονοστοιχειουείναι μικρότερη από την τιμή του κατωφλίου τότε αυτό ανήκει στο background. Αν όχι παιρνειτην τιμή 1, και ετσι ανήκει στο αντικείμενο ενδιαφέροντος.

Δυο είναι οι βασικές προσεγγίσεις για την εύρεση του κατωφλίου: η προσέγγιση Chow και Kaneko και η τοπική κατωφλίωση. Οι εικόνες σε αυτές τις περιπτώσεις προς επεξεργασία είναι αυτές με εξισοροπημένο ιστόγραμμα.

Chow και Kaneko

Οι Chow και Kanekoπεριέκοψαν την εικόνα σε κομμάτια και μετα από αυτο βρήκαν το βέλτιστο κατώφλι για κάθε κομμάτι. Για να βρουμε το κατώφλι κάθε pixelθα αναγαγουμεταποτελεσματα των κομματιών στην αρχική εικόνα. Με αυτό τον τρόπο έχουμε πρόβλημα στο κώστος σε περίπτωση κατασκευής αλγορίθμου και ετσι δεν την χρησιμοποιούμε σε πραγματικό χρόνο.

Το μέγεθος της γειτονιάς πρέπει να είναι αρκετά μεγάλοωστε να καλυφθουν επαρκώς τα pixels στο background και στο foreground. Σε αντιθετηπεριπτωση θα επιλεγεί ένα ασθενές κατώφλι. Από την άλλη, επιλέγοντας περιοχές που είναι πολύ μεγάλες ενδεχεται να παραβιαστεί η υπόθεση περί ομοιόμορφου φωτισμού. Αυτή η μεθοδοςομως είναι μικρότερης υπολογιστικής διάρκειας συγκριτικα με την προσέγγιση Chow και Kaneko και παράγει επαρκη αποτελέσματα σε ορισμένες εφαρμογές.

Το προσαρμόσιμο κατώφλι όπως και το γενικο κατωφλι,χρησιμοποιουνται για το διαχωρισμό της επιθυμητής εικόνας σε αντικείμενα ενδιαφέροντος και σε background, ο οποίος διαχωρισμός στηρίζεται στη διαφορά έντασης των εικονοστοιχειων της κάθε περιοχής.

Η γενική κατωφλίωση χρησιμοποιεί ένα σταθερό κατώφλι για όλα ταεικονοστοιχεια της εικόνας και ως εκ τουτου λειτουργεί μόνο στην περίπτωση που το ιστόγραμμα έντασης της εικόνας εισόδου περιέχει αρκετά διαχωρισμένες κορυφές που αντιστοιχούν στο επιθυμητό αντικείμενο(-α) ενδιαφέροντος και στο φοντο(background) .Ετσι , η μέθοδος αυτή δεν μπορεί να εφαρμοστεί σε εικόνες που παρουσιάζουν, παράδειγματοςχαριν, ισχυρή κλίση φωτεινότητας. Η τοπική προσαρμοστική κατωφλίωση, σε αντιθεση, επιλέγει ένα μεμονωμένο κατώφλι για κάθε pixel με βάση το εύρος των τιμών έντασης στηνπροκαθορισμένη γειτονιά του. Ως αποτελεσμαέχουμε την κατωφλίωση μιας εικόνας, της οποίας το γενικό ιστόγραμμα έντασης δεν περιέχει διακριτές κορυφές.

ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΕΦΑΡΜΟΓΗ ΣΤΗ ΚΑΘΗΜΕΡΙΝΟΤΗΤΑ

Επεξεργαζόμενος αυτό το θέμα μου γέννησε πολλές σκέψεις για το πώς μπορούμε να εκμεταλλευτούμε αυτό το απίστευτο πολυεργαλείο που λέγεται MATLAB. Με απλές σκέψεις και εύκολες εφαρμογές που έχουμε στην άκρη του μυαλού μας μπορούμε να κάνουμε τις καθημερινές μας δυσκολίες παιχνιδάκι πράγμα που θα μας ανέβαζε την απόδοση σε χρήσεις του σπιτιού του ή της εργασίας μας.

Φέρνοντας σε πέρας την πρακτική μου άσκηση ταυτόχρονα με αυτή την εργασία συνάντησα κάποια απλά προβληματάκια που μου <<έκαναν τη ζωή δύσκολη>>.

Στο εργασιακό μου περιβάλλον η καθημερινή μου ενασχόληση ήταν η διαχείριση λογισμικού και η βοήθεια σε αγρότες να πραγματοποιήσουν διάφορες ετήσιες υποχρεώσεις όπως Ενιαία Δήλωση Αγροτών, Υποβολή τιμολογίων για επιστροφή Φπα κ.α.

Οι διάφορες λοιπόν αυτές εργασίες εκτελούνται σε όλη την Ελλάδα από διαπιστευμένους φορείς όπως και η Ένωση Αγροτικών Συνεταιρισμών Κορίνθου στην οποία και εκτέλεσα την πρακτική μου.

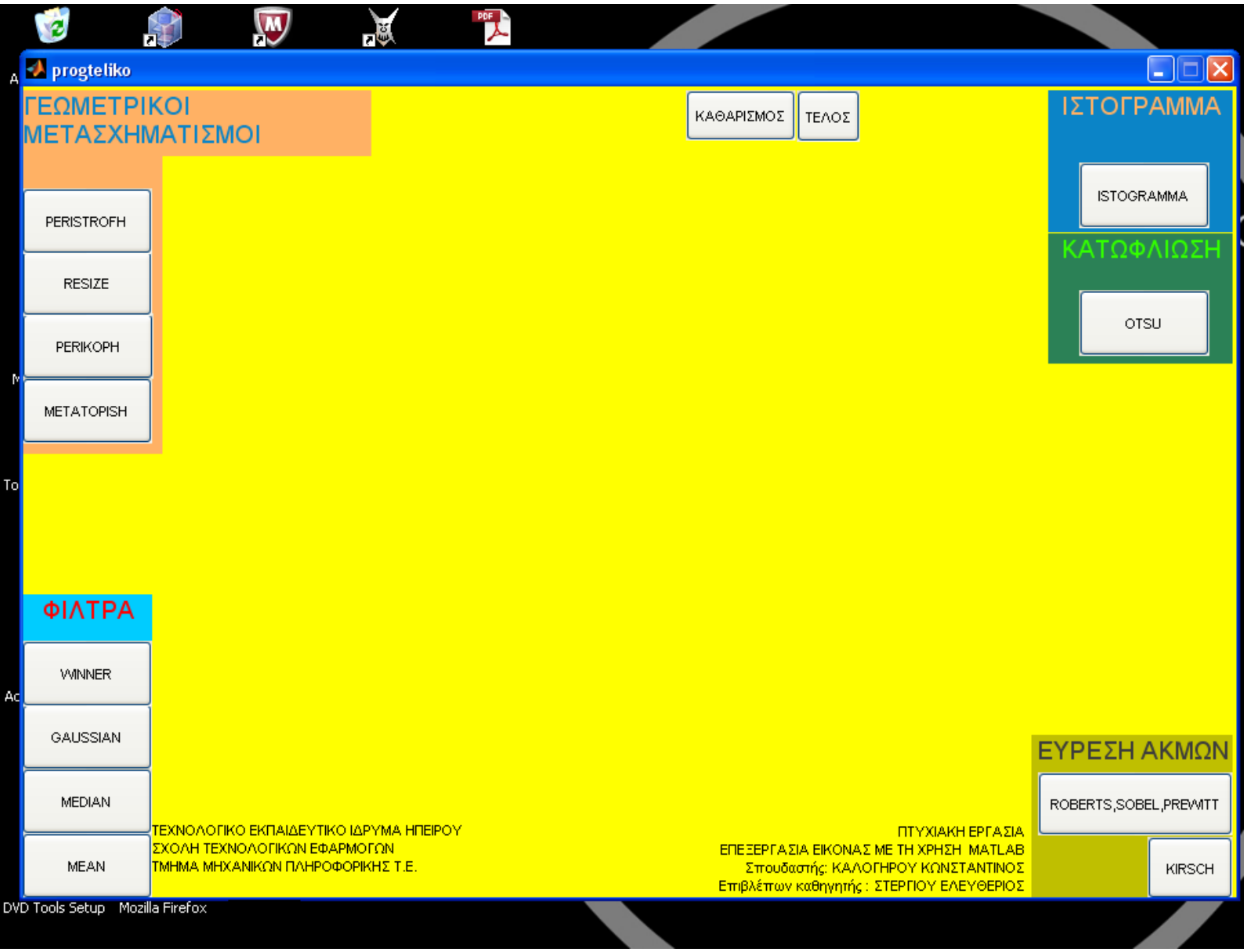
Οι εργασίες λοιπόν αυτές απαιτούν καθημερινά σκαναρίσματα εγγράφων τα οποία αποθηκεύονται σε κεντρικούς serversστην Αθήνα και στην συνέχεια επεξεργάζονται από τις ενδιαφερόμενες υπηρεσίες.

Τα προβλήματα που παρουσιάζονταν καθημερινά ήταν το στραβό σκανάρισμα, τα θολά έγραφα, δυσδιάκριτες υπογραφές, δυσκολία ανάγνωσης των barcodes που αντιστοιχούν αποκλείστηκα σε συγκεκριμένο παραγωγό και άλλες παρόμοιες καταστάσεις.

Το αποτέλεσμα η καθυστερημένη αντίληψη των προβλημάτων αυτών και η χρονοβόρα διαδικασία εύρεσης και διόρθωσης των λαθών ανάμεσα σε δεκάδες χιλιάδες σκαναρίσματα.

Αφού όλοι οι διαπιστευμένοι φορείς χρησιμοποιούν τα ίδια σκανερ τις ίδιες ρυθμίσεις σκαναρίσματος μέσα από το ίδιο λογισμικό και επεξεργάζονται ασπρόμαυρα έγραφα με κοινά καθημερινά προβλήματα θα ήταν σκόπημο να μπορέσει ο υπάλληλος που θα βρεί αυτά τα προβλήματα να δημιουργήσει μία αυτόματη διόρθωση του εγράφου με ένα απλό κλίκ.

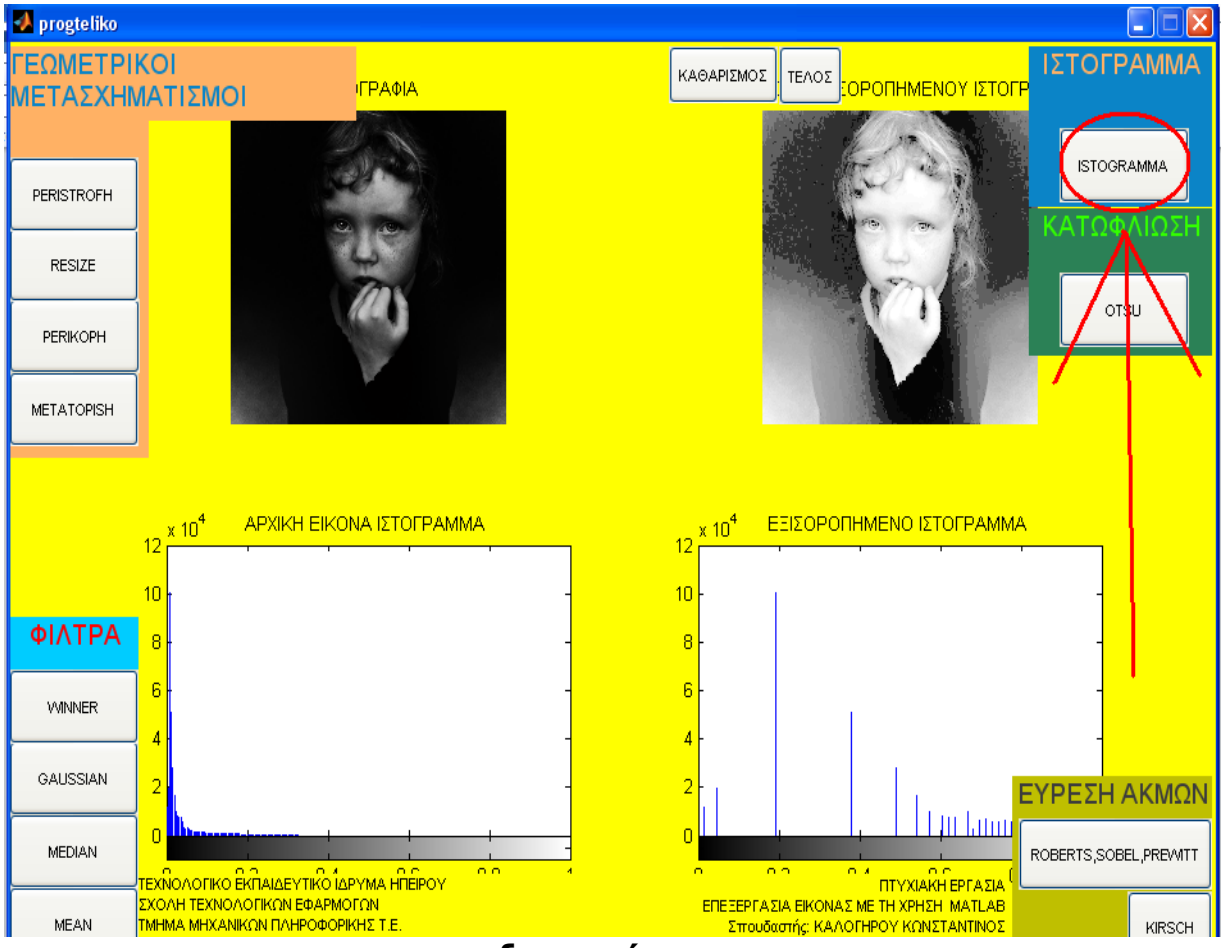
Παρακάτω δίνω μια υλοποίηση αυτής της με την εφαρμογή αυτών τον αλγορίθμων που αναφέρω σε αυτή την εργασία και δημιουργώντας το απαραίτητο γραφικό περιβάλλον για τον χρήστη.



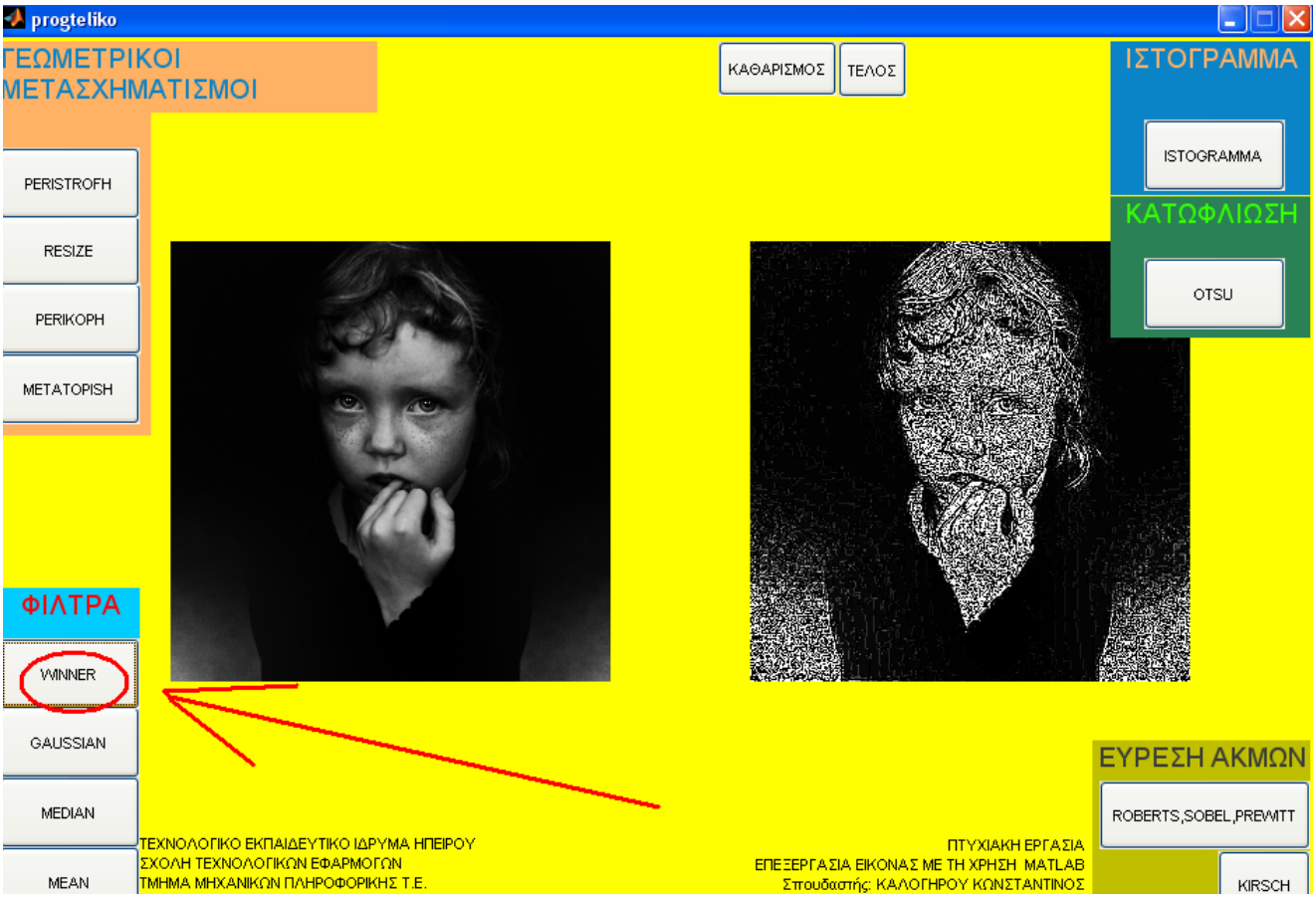
Με την εφαρμογή απλών παραμέτρων, που διορθώνουν μαζικά προβλήματα , σε κάθε έναν από τους αλγορίθμους κάποιας από τις λειτουργίες που έβαλα δημιουργούν το επιθυμητό αποτέλεσμα χωρίς να χρειαστεί το επαναλαμβανόμενο σβανάρισμα χωρίς λόγο. Προφανώς είναι μια λύση που μπορεί να επιτευχτεί και με άλλους τρόπους, αλλά κάνοντας την παρούσα εργασία βρήκα σκόπιμο να καταγράψω το πρόβλημα με την λύση αυτή για μελλοντική μελέτη και τελειοποίηση της.

Παρακάτω προβάλω μερικές μεταποιήσεις εικόνας:

περιστροφή



εξισορρόπηση



ΦΙΛΤΡΑΡΙΣΜΑ

ΚΑΤΩΦΛΙΩΣΗ

ΒΙΒΛΙΟΓΡΑΦΙΑ

ΕΛΛΗΝΟΓΛΩΣΣΗ

[1] Πήτας Ι. (2010) Ψηφιακή επεξεργασία εικόνας θεωρεία & ασκήσεις

[2] Παπαμάρκος(2010)Ν. Ψηφιακή επεξεργασία & ανάλυση εικόνας.

[3] Κόλλιας Σ. Δ. (2001) *Επεξεργασία, ανάλυση και τεχνολογία εικόνων Βίντεο*,

[4] Κουτσούρης Δ. *Επεξεργασία εικόνων DICOM με τη χρήση Matlab*

ΞΕΝΟΓΛΩΣΣΗ

[5] Anoraganingrum D.(1999) "Cell segmentation with median filter and Mathematical Morphology Operation", Proceedings, International Conference on *Image Analysis and Processing*, pp. 1043-1046

[6] Gonzalez R. C., Woods R. E., Eddins S. L.(2004), *Digital Image Processing Using MATLAB*, Prentice Hall, Upper Sandle River, NJ

ΔΙΑΔΙΚΤΥΟ

[7] MATLABwww.mathworks.com

ΠΑΡΑΘΕΣΗ ΣΥΝΟΛΙΚΟΥ ΚΩΔΙΚΑ

```
functionvarargout = progteliko(varargin)
% PROGTELIKO MATLAB code for progteliko.fig
%      PROGTELIKO, by itself, creates a new PROGTELIKO or raises the
existing
%      singleton*.
%
%      H = PROGTELIKO returns the handle to a new PROGTELIKO or the handle
to
%      the existing singleton*.
%
%      PROGTELIKO('CALLBACK',hObject,eventData,handles,...) calls the
local
%      function named CALLBACK in PROGTELIKO.M with the given input
arguments.
%
%      PROGTELIKO('Property','Value',...) creates a new PROGTELIKO or
raises the
%      existing singleton*.  Starting from the left, property value pairs
are
%      applied to the GUI before progteliko_OpeningFcn gets called.  An
%      unrecognized property name or invalid value makes property
application
%      stop.  All inputs are passed to progteliko_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only
one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help progteliko

% Last Modified by GUIDE v2.5 05-Sep-2014 22:14:35

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
'gui_Singleton',  gui_Singleton, ...
'gui_OpeningFcn', @progteliko_OpeningFcn, ...
'gui_OutputFcn',  @progteliko_OutputFcn, ...
'gui_LayoutFcn',  [] , ...
'gui_Callback',   []);
ifnargin&&ischar(varargin{1})
gui_State.gui_Callback = str2func(varargin{1});
end

ifnargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT


% --- Executes just before progteliko is made visible.
functionprogteliko_OpeningFcn(hObject, ~, handles, varargin)
```

```
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to progteliko (see VARARGIN)

% Choose default command line output for progteliko
handles.output = hObject;

% Update handles structure
guidata(hObject, handles);

% UIWAIT makes progteliko wait for user response (see UIRESUME)
% uiwait(handles.figure1);


% --- Outputs from this function are returned to the command line.
functionvarargout = progteliko_OutputFcn(~, ~, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;




% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(~, ~, ~)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = imread('10.png');
J = imrotate(I,45,'bicubic');
subplot(1,2,1);imshow(I)
subplot(1,2,2); imshow(J)




% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(~, ~, ~)
% hObject    handle to pushbutton4 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = imread('10.png');
J = imcrop(I,[200 200 100 100]);
subplot(1,2,1);imshow(I)
subplot(1,2,2); imshow(J)

% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton5 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = imread('10.png');
J = imresize(I,[100 150]);
```

```
subplot(1,2,1);imshow(I)
subplot(1,2,2); imshow(J)

% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(~, ~, ~)
% hObject    handle to pushbutton6 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = imread('10.png');
udata = [0 1];  vdata = [0 1];
tform = maketform('projective',[ 0 0;  1  0;  1  1; 0 1],...
                               [-4 2; -8 -3; -3 -5; 6 3]);
[B,xdata,ydata] = imtransform(I, tform, 'bicubic', ...
'udata', udata,...
'vdata', vdata,...
'size', size(I),...
'fill', 128);
subplot(1,2,1), imshow(I,'XData',udata,'YData',vdata), ...
axison
subplot(1,2,2), imshow(B,'XData',xdata,'YData',ydata), ...
axison


% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(~, ~, ~)
% hObject    handle to pushbutton7 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
img = double ( imread ('10.png'))/255.;
subplot(2,2,1)
imshow ( img )
title'ΦΩΤΟΓΡΑΦΙΑ '
print-depsc' 10.png'

% ΠΡΟΒΟΛΗ ΑΡΧΙΚΗΣ ΕΙΚΟΝΑΣ ΚΑΙ ΙΣΤΟΓΡΑΜΜΑΤΟΣ
subplot(2,2,3)

imhist (img)
axisauto% rescale the image so that the entire histogram fits in
title'ΑΡΧΙΚΗΕΙΚΟΝΑΙΣΤΟΓΡΑΜΜΑ '
print-depsc' original_histogram .eps '

print-depsc' equalised_image .eps '
%%% ΔΕΥΤΕΡΟ ΚΟΜΜΑΤΙ % ΕΞΙΣΟΡΟΠΗΣΗ ΚΑΙ ΠΡΟΒΟΛΗ
equalised = histeq (img );
subplot(2,2,2)
imshow ( equalised )
title'ΤΕΛΙΚΗ ΕΙΚΟΝΑ ΕΞΙΣΟΡΟΠΗΜΕΝΟΥ ΙΣΤΟΓΡΑΜΜΑΤΟΣ '

print-depsc' equalised_image .eps '
subplot(2,2,4)
imhist ( equalised )

title'ΕΞΙΣΟΡΟΠΗΜΕΝΟΙΣΤΟΓΡΑΜΜΑ '
axisauto
print-depsc' equalised_histogram .eps '
%%% ΤΡΙΤΟΚΟΜΜΑΤΙ
fprintf ('Michelson contrast ΑΡΧΙΚΗΣΕΙΚΟΝΑΣ : %f\n', Michelson (img ))
fprintf ('Michelson contrast ΕΞΙΣΟΡΟΠΗΜΕΝΗΣΕΙΚΟΝΑΣ : %f\n', ...
```

```
Michelson ( equalised ))
fprintf ('RMS contrast ΑΡΧΙΚΗΣΕΙΚΟΝΑΣ : %f\n', RMSContrast (img ))
fprintf ('RMS contrast ΕΞΙΣΟΡΟΠΗΜΕΝΗΣΕΙΚΟΝΑΣ : %f\n', ...
RMSContrast( equalised ))


% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(~, ~, handles)
% hObject    handle to pushbutton8 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I=imread('10.png');
subplot(2,2,1);imshow(I);title('original image');
Io = im2single(I);
subplot(2,2,2);imshow(Io);title('original image class single')
T=graythresh(Io);
BW=im2bw(Io,T);
subplot(2,2,3);imshow(BW);title('binary image otsu')


% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton10 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=imread('10.png');
subplot(2,2,1), imshow(A);
I=imnoise(A,'salt& pepper',0.01);
subplot(2,2,2), imshow(I);
Ig=imnoise(I,'gaussian',0.05);
subplot(2,2,3), imshow(Ig);

% --- Executes on button press in pushbutton11.
function pushbutton11_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton11 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=imread('10.png');
J = wiener2(A);
subplot(1,2,1),imshow(A)
subplot(1,2,2),imshow(J)

% --- Executes on button press in pushbutton12.
function pushbutton12_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton12 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=imread('10.png');
L=conv2(A,[0 0.01]);
subplot(1,2,1),imshow(A)
subplot(1,2,2),imshow(L)

% --- Executes on button press in pushbutton13.
function pushbutton13_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton13 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
A=imread('10.png');
L=medfilt2(A,[3 3]);
```

```
subplot(1,2,1),imshow(A)
subplot(1,2,2),imshow(L)


% --- Executes on button press in pushbutton14.
function pushbutton14_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton14 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
I = imread('3.png')
prewitt = edge(I,'prewitt');
roberts = edge(I,'roberts');
sobel = edge(I,'sobel');
subplot(2,2,1); imshow(I)
subplot(2,2,2); imshow(prewitt)
subplot(2,2,3);imshow(roberts)
subplot(2,2,4); imshow(sobel)


% --- Executes on button press in pushbutton15.
function pushbutton15_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton15 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
im = imread('6.png');
[edgeim, thresh] = kirsch(im,'kirsch',200);
subplot(1,2,1); imshow(im);
subplot(1,2,2); imshow(edgeim);


% --- Executes on button press in pushbutton16.
function pushbutton16_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton16 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
X = imread('10.png');
udata = [0 1];  vdata = [0 1];
tform = maketform('projective',[ 0 0;  1  0;  1  1; 0 1],...
[-4 2; -8 -3; -3 -5; 6 3]);
[B,xdata,ydata] = imtransform(X, tform, 'bicubic', ...
'udata', udata,...
'vdata', vdata,...
'size', size(X),...
'fill', 128);
subplot(1,2,1), imshow(X,'XData',udata,'YData',vdata), ...
axison
subplot(1,2,2), imshow(B,'XData',xdata,'YData',ydata), ...
axison


% --- Executes on button press in pushbutton17.
function pushbutton17_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton17 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
clc;
closeall;
clear;
```

```
% --- Executes on button press in pushbutton18.
function pushbutton18_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton18 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cloce(gcf);

function [imout, thresh] = edge_ed( im, method, thresh, param2 )
[n,m]= size(im);
xx= 2:m-1;
yy= 2:n-1;
ifstrcmp(method,'kirsch')
filt1= [1 2 1;0 0 0;-1 -2 -1]; fim1= conv2(im,filt1,'same');
filt2= [2 1 0;1 0 -1;0 -1 -2]; fim2= conv2(im,filt2,'same');
filt3= [1 0 -1;2 0 -2;1 0 -1]; fim3= conv2(im,filt3,'same');
filt4= [0 1 2;-1 0 1;-2 -1 0]; fim4= conv2(im,filt4,'same');
imo= reshape(max([abs(fim1(:)) abs(fim2(:)) abs(fim3(:))
abs(fim4(:))]'),n,m);
ifnargin<3
thresh= 2* mean(mean( imo(yy,xx) )) ;
end
imout= imo>= thresh ;
xpeak= imo(yy,xx-1) <= imo(yy,xx) &imo(yy,xx) >imo(yy,xx+1) ;
ypeak= imo(yy-1,xx) <= imo(yy,xx) &imo(yy,xx) >imo(yy+1,xx) ;
imout(yy,xx)= imout(yy,xx) & ( xpeak | ypeak );
end
```