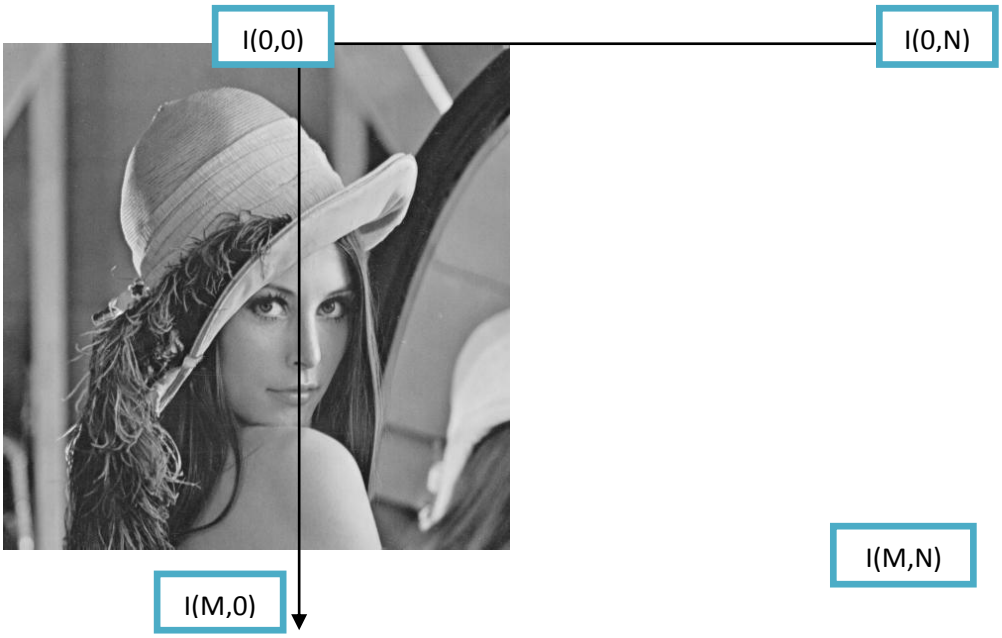
ΕΙΣΑΓΩΓΗ ΣΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Ο σκοπός αυτού του οδηγού είναι να αποκτήσετε εξοικείωση με την επεξεργασία εικόνας στο MATLAB . Αυτός ο οδηγός δεν περιέχει όλες τις διαθέσιμες συναρτήσεις που σχετίζονται με την επεξεργασία εικόνας στο MATLAB. Μπορείτε να χρησιμοποιήσετε το Help του MATLAB, αν έχετε απορίες.

Μία εικόνα μπορεί να οριστεί ως μία συνάρτηση δύο διαστάσεων f(x,y), όπου x και y οι συντεταγμένες, και η τιμή της f για κάθε ζευγάρι συντεταγμένων (x,y) καλείται ως φωτεινότητα της εικόνας στο σημείο αυτό. Στις μονοχρωματικές εικόνες, με τις οποίες και θα ασχοληθούμε, συνηθίζεται να χρησιμοποιείται ο όρος 'επίπεδο γκρι'.
Μία αναλογική εικόνα έχει συνεχείς πεδία τιμών x,y και συνεχείς τιμές φωτεινότητας. Για την ψηφιοποίηση μίας αναλογικής εικόνας, με άμεσο σκοπό την επεξεργασία της από τον Η/Υ, απαιτείται η μετατροπή των συντεταγμένων σε διακριτές τιμές (δειγματοληψία - sampling) και η μετατροπή των συνεχών τιμών φωτεινότητας σε διακριτές (κβάντιση - quantization). Με τον τρόπο αυτό προκύπτει μία ψηφιακή εικόνα.

Για παράδειγμα μία εικόνα MxN pixels θα απεικονίζεται στο Matlab με έναν δισδιάστατο πίνακα MxN στοιχείων. Λόγω της κβάντισης των επιπέδων του γκρι σε L επίπεδα, κάθε στοιχείο f(x,y) θα περιορίζεται στο διάστημα $0 \leq f(x,y) \leq L-1$. Η τιμή 0 αντιστοιχεί στο μαύρο χρώμα, ενώ η τιμή L-1 στο άσπρο χρώμα. Τυπικά , οπότε και απαιτούνται M*N*k bits για την αποθήκευση της δεδομένης εικόνας. Συνήθως k=8, οπότε $0 \leq f(x,y) \leq 255$.



Η ***επεξεργασία εικόνας*** ασχολείται με την αλλαγή των χαρακτηριστικών μιας εικόνας, προκειμένου είτε να
1. βελτιώσουμε την πληροφορία της εικόνας ώστε να φαίνεται καλύτερη στο ανθρώπινο μάτι (οι άνθρωποι θέλουν τις εικόνες τους ευκρινείς, καθαρές και λεπτομερείς),
2. καταστεί περισσότερο κατάλληλη μία εικόνα για την επεξεργασία της από ηλεκτρονικό

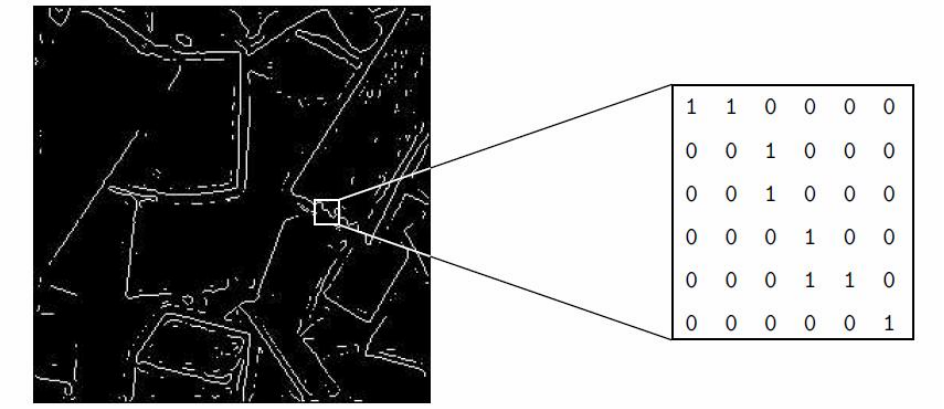
υπολογιστή (σε αντίθεση με τους ανθρώπους , οι υπολογιστές προτιμούν οι εικόνες τους να είναι απλές και απέριττες!!) .

1. Είδη ψηφιακών εικόνων

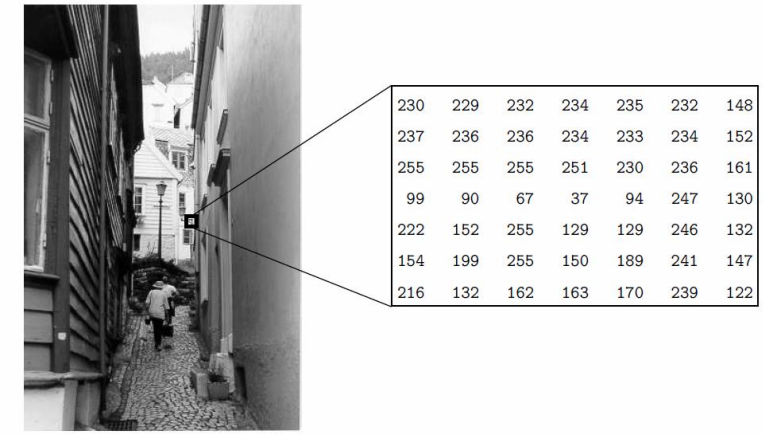
Το πρώτο βήμα στην επεξεργασία εικόνας MATLAB είναι να γίνει κατανοητό ότι μια ψηφιακή εικόνα αποτελείται από ένα δισδιάστατο ή τρισδιάστατο πίνακα εικονοστοιχείων (pixels). Κάθε pixel περιέχει έναν αριθμό ή περισσότερους αριθμούς που μας δείχνει μία τιμή στην κλίμακα του γκρι ή σε έγχρωμη κλίμακα που έχει εκχωρηθεί σε αυτό.

Binary. Κάθε pixel είναι μαύρο ή άσπρο. Εφόσον υπάρχουν δύο μόνο τιμές που μπορεί να πάρει ένα pixel χρειαζόμαστε μόνο ένα bit για κάθε pixel.

Ένα παράδειγμα δείχνουμε παρακάτω όπου έχουμε μία εικόνα που έχει τα δύο μόνο χρώματα , λευκό για τις ακμές και μαύρο για το background.



Greyscale: Κάθε pixel παίρνει μία τιμή γκρίζου , από το 0 (μαύρο) μέχρι το 255 (άσπρο) όπως φαίνεται στη παρακάτω εικόνα.



True colour, or RGB: Κάθε pixel έχει μία τιμή χρώματος , και το χρώμα περιγράφεται με βάση το ποσοστό που περιέχει σε κόκκινο(**R**ed), μπλε(**B**lue), πράσινο (**G**reen) . Κάθε ένα από αυτά τα RGB στοιχεία μπορεί να πάρει τιμή 0-255 (αυτό μας δίνει 255^3=16,777,216 διαφορετικά πιθανά χρώματα στην εικόνα)

49	55	56	57	52	53
58	60	60	58	55	57
58	58	54	53	55	56
83	78	72	69	68	69
88	91	91	84	83	82
69	76	83	78	76	75
61	69	73	78	76	76

Red

64	76	82	79	78	78
93	93	91	91	86	86
88	82	88	90	88	89
125	119	113	108	111	110
137	136	132	128	126	120
105	108	114	114	118	113
96	103	112	108	111	107

Green

66	80	77	80	87	77
81	93	96	99	86	85
83	83	91	94	92	88
135	128	126	112	107	106
141	129	129	117	115	101
95	99	109	108	112	109
84	93	107	101	105	102

Blue

2. Εισαγωγή (Loading) μιας εικόνας στο MATLAB

Πολλές φορές θα θέλετε να επεξεργαστεί μια συγκεκριμένη εικόνα. Αν επιλέξετε να το κάνετε αυτό σε MATLAB θα χρειαστεί να φορτώσετε την εικόνα ώστε να μπορείτε να αρχίσετε την επεξεργασία. Αν η εικόνα που έχετε είναι έγχρωμη, αλλά το χρώμα δεν είναι σημαντικό για την τρέχουσα εφαρμογή, τότε μπορείτε να αλλάξετε την εικόνα σε grayscale. Αυτό καθιστά την επεξεργασία πολύ απλούστερη από τότε υπάρχουν μόνο το ένα τρίτο των τιμών των pixels στη νέα εικόνα.

Για παράδειγμα το χρώμα δεν μπορεί να είναι σημαντικό σε μια εικόνα όταν προσπαθούμε να εντοπίσουμε ένα συγκεκριμένο αντικείμενο που έχει καλή αντίθεση με το περιβάλλον του.

Το παράδειγμα , παρακάτω, δείχνει πώς μπορείτε να φορτώσετε διαφορετικές εικόνες στο Matlab.

Παράδειγμα 2.1

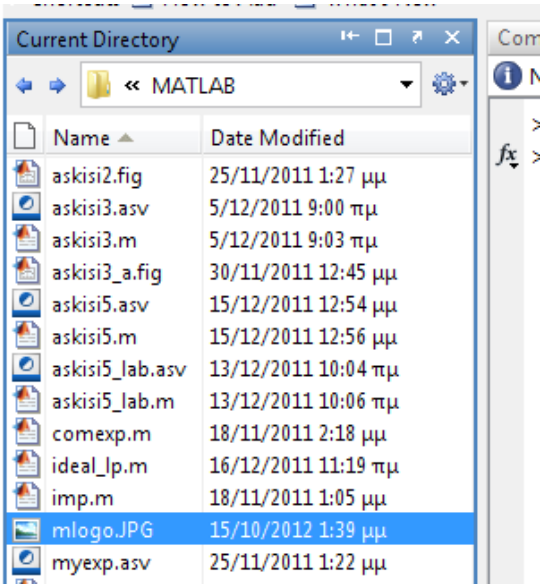
Οι εντολές που θα χρησιμοποιήσουμε είναι

- Η εντολή **imread** χρησιμοποιείται για να διαβάσει ένα αρχείο εικόνας με μια συγκεκριμένη μορφή. Συμβουλευτείτε τη βοήθεια για να βρείτε ποιες μορφές υποστηρίζονται. Γενικά η εντολή imread διαβάζει τις τιμές των pixels της εικόνας και επιστρέφει ένα πίνακα με όλες αυτές τις τιμές. Επομένως έχουμε διαβάσει την

εικόνα και την έχουμε πλέον σε ένα πίνακα στο Matlab , οπότε μπορούμε να εφαρμόσουμε οτιδήποτε λειτουργίες θέλουμε πάνω σε αυτόν τον πίνακα.

- Η εντολή **imshow** εμφανίζει έναν πίνακα σε ένα Matlab figure.
- Η εντολή **rgb2gray** μπορεί να χρησιμοποιηθεί για να αλλάξει μια έγχρωμη εικόνα σε ασπρόμαυρη.

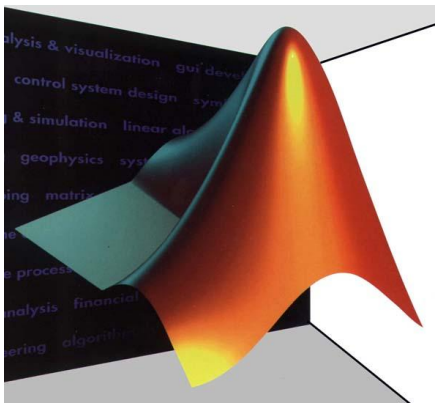
Έστω ότι στον τρέχον φάκελο του Matlab έχουμε ένα αρχείο εικόνας.

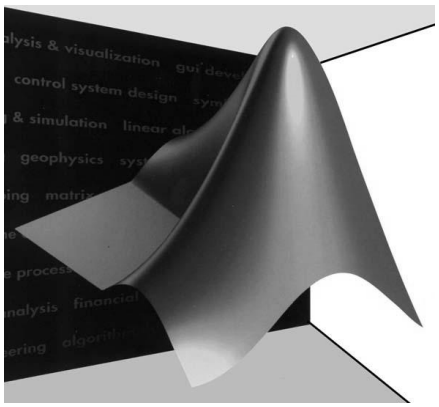


Στη συνέχεια τρέξτε το παρακάτω παράδειγμα

```
A=imread('mlogo.jpg');

figure(1);imshow(A)

B=rgb2gray(A);

figure(2);imshow(B)
```

Πρέπει να δείτε σε δύο διαφορετικά figures την αρχική εικόνα και τη ίδια εικόνα σε grayscale

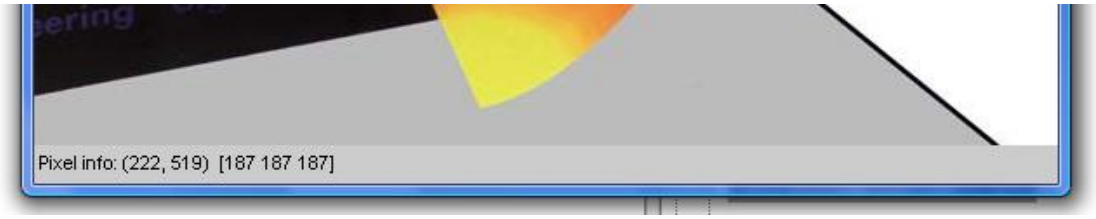




Δείτε στην συνέχεια πως μπορούμε να προσπελάσουμε για παράδειγμα συγκεκριμένα pixels σε μία εικόνα και να αλλάξουμε την τιμή τους.

```
A=imread('mlogo.jpg');

imtool(A)  %Examine RGB image in interactive viewer

A(265,493,:)  %print pixel value at location (265,493)
```

Με την εντολή ***imtool*** βλέπουμε την εικόνα σε ένα interactive viewer στο Matlab (είναι σαν ένα νέο figure) και πηγαίνοντας με το ποντίκι μας πάνω στην εικόνα βλέπουμε από κάτω τις τιμές των pixels για κάθε σημείο της εικόνας.



Το αποτέλεσμα που βλέπουμε στην οθόνη για τη συγκεκριμένη εικόνα είναι

```
ans(:,:,1) =

  174

ans(:,:,2) =

   46

ans(:,:,3) =

   33
```

Τιμές για το κόκκινο , πράσινο , μπλε

Δοκιμάστε στη συνέχεια το παρακάτω, όπου όπως φαίνεται και από το αποτέλεσμα αλλάξαμε την τιμή αυτού του pixel σε λευκό

```
A(265,493,:)=255

imshow(A)
```

```
ans(:,:,1) =

  255

ans(:,:,2) =
```

255

ans(:,:,3) =

255

Πολλές πληροφορίες για την εικόνα μπορούμε να πάρουμε με την συνάρτηση **imfinfo**

```
>> A=imfinfo('mlogo.jpg')

A =

           Filename: 'mlogo.jpg'
        FileModDate: '15-Οκτ-2012 13:39:23'
           FileSize: 68678
             Format: 'jpg'
      FormatVersion: ''
              Width: 589
             Height: 543
           BitDepth: 24
          ColorType: 'truecolor'
    FormatSignature: ''
    NumberOfSamples: 3
       CodingMethod: 'Huffman'
      CodingProcess: 'Sequential'
            Comment: {}
```

Τα στοιχεία σε έναν πίνακα του Matlab ανήκουν σε ένα τύπο δεδομένων. Οι πιο κοινοί τύποι δεδομένων φαίνονται στον παρακάτω πίνακα (υπάρχουν και άλλοι τύποι δεδομένων αλλά δεν μας χρειάζονται στην επεξεργασία εικόνας). Οι τύποι δεδομένων που δίνουμε στον πίνακα αυτό αποτελούν επίσης συναρτήσεις για να μετατρέπουμε από τον ένα τύπο στον άλλο .

Data Type	Περιγραφή	Range
int8	8-bit integer	-128 … 127
uint8	8-bit unsigned integer	0 … 255
int16	16-bit integer	-32768 … 32767
uint16	16-bit unsigned integer	0 … 65535
double	Double precision real number	Machine specific

Για παράδειγμα

```
>> a=23;
>> b=uint8(a);
>> b

b =

   23

>> whos a b
  Name        Size              Bytes  Class     Attributes

  a           1x1                   8  double
  b           1x1                   1  uint8
```

Αν και οι μεταβλητές α,b έχουν την ίδια αριθμητική τιμή δεν έχουν το ίδιο τύπο(data type) . Αυτό που πρέπει να σημειώσουμε εδώ είναι ότι δεν επιτρέπονται αριθμητικοί υπολογισμοί με του τύπους int8, int16, uint8 και uint16 .

Μία grayscale εικόνα μπορεί να αποτελείται από pixels τα οποία είναι τύπου uint8. Αυτού του είδους οι εικόνες μπορεί να είναι αποδοτικές όσο αφορά το χώρο που καταλαμβάνουν (κάθε pixel χρειάζεται ακριβώς ένα byte) αλλά δεν μπορούμε να εκτελέσουμε καμία αριθμητική λειτουργία σε αυτές . Οπότε χρειάζεται να μετατρέψουμε μία τέτοια εικόνα σε double πριν κάνουμε αριθμητικές πράξεις .

Παράδειγμα 2.2

Συνάρτηση που μετατρέπει μία εικόνα από grayscale σε RGB

Φτιάξτε την παρακάτω συνάρτηση σε ένα ξεχωριστό m file.

```
function Image=gray2rgb(Image)
%Gives a grayscale image an extra dimension
%in order to use color within it
[m n]=size(Image);
rgb=zeros(m,n,3);
rgb(:,:,1)=Image;
rgb(:,:,2)=rgb(:,:,1);
rgb(:,:,3)=rgb(:,:,1);
Image=rgb/255;
end
```

Στη συνέχεια πηγαίνετε στο Command window και τρέξτε τις παρακάτω εντολές

```
A=imread('mlogo.jpg');
 B=rgb2gray(A);
```

Και με την εντολή whos βλέπουμε τι ακριβώς μέγεθος και τύπο έχουν οι μεταβλητές μας.

```
>> whos A
  Name         Size                Bytes  Class    Attributes

  A          543x589x3           959481  uint8

>> whos B
  Name         Size                Bytes  Class    Attributes

  B          543x589             319827  uint8
```

Έπειτα εφαρμόστε την συνάρτηση gray2rgb που μόλις φτιάξαμε , στην εικόνα B. Τι παρατηρείτε ;

```
new_A=gray2rgb(B);
imshow(new_A)
```

Η εικόνα που προέκυψε δεν είναι μία έγχρωμη εικόνα , αλλά μία εικόνα στην οποία οι τιμές των pixels είναι ίδιες με την εικόνα B (που ήταν στην κλίμακα του γκρι).

Δείτε με την εντολή whos , τι ακριβώς περιέχει η νέα εικόνα .

```
>> whos new_A
  Name         Size                Bytes  Class     Attributes

  new_A      543x589x3            7675848  double
```

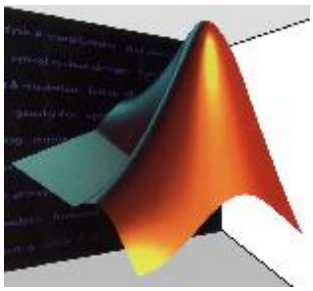
Δηλαδή, στην συνάρτηση gray2rgb φτιάξαμε ένα νέο πίνακα με 3 στήλες (μία για το reb, μία για το green, και μία για το blue) και απλά επαναλάβαμε τις τιμές του γκρι για κάθε κάθε pixel σε κάθε στήλη. Η μετατροπή από RGB σε grayscale είναι μία μη αντιστρέψιμη διαδικασία , καθώς η πληροφορία για το χρώμα στην αρχική εικόνα χάνεται στη μετατροπή και δεν μπορεί να ανακτηθεί.

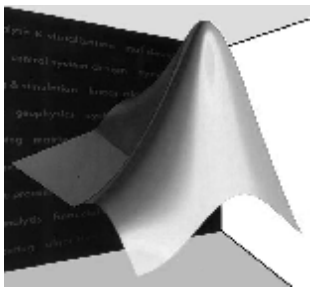
Παράδειγμα 2.3

Δείτε στη συνέχεια το παρακάτω παράδειγμα

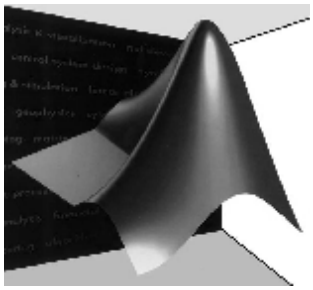
```
A=imread('mlogo.jpg');

Ared=A(:,:,1); %extract red channel

Agreen=A(:,:,2); %extract green channel

Ablue=A(:,:,3); %extract blue channel

subplot(2,2,1); imshow(A); axis image;

subplot(2,2,2); imshow(Ared); title('red');

subplot(2,2,3); imshow(Agreen); title('green');

subplot(2,2,4); imshow(Ablue); title('blue');
```

red

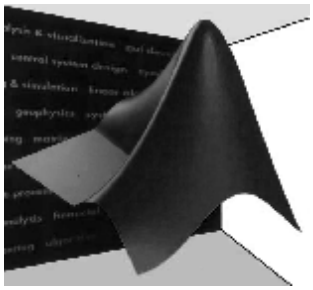




green



blue



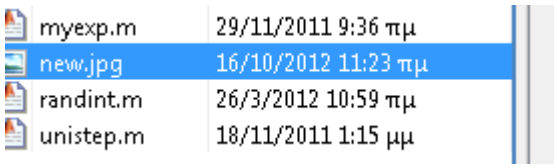
3. Αποθήκευση των νέων εικόνων

Τις περισσότερες φορές θα θέλαμε αφού ολοκληρώσουμε την επεξεργασία μίας εικόνας να να την μεταφέρουμε κάπου αλλού, να την σώσουμε δηλαδή σαν αρχείο. Αυτό γίνεται με χρησιμοποιώντας τη συνάρτηση **imwrite** , η οποία επιτρέπει να αποθηκεύσετε μια εικόνα ως οποιοδήποτε τύπο αρχείου που υποστηρίζεται από το MATLAB, (τα οποία είναι τα ίδια από με αυτά που υποστηρίζει η imread , δηλαδή bmp,jpg κτλ)

Για παράδειγμα δοκιμάστε

```
imwrite(new_A , 'new.jpg');
```

Και στο Current directory θα εμφανιστεί το νέο αρχείο της εικόνας.



4. Ιστόγραμμα

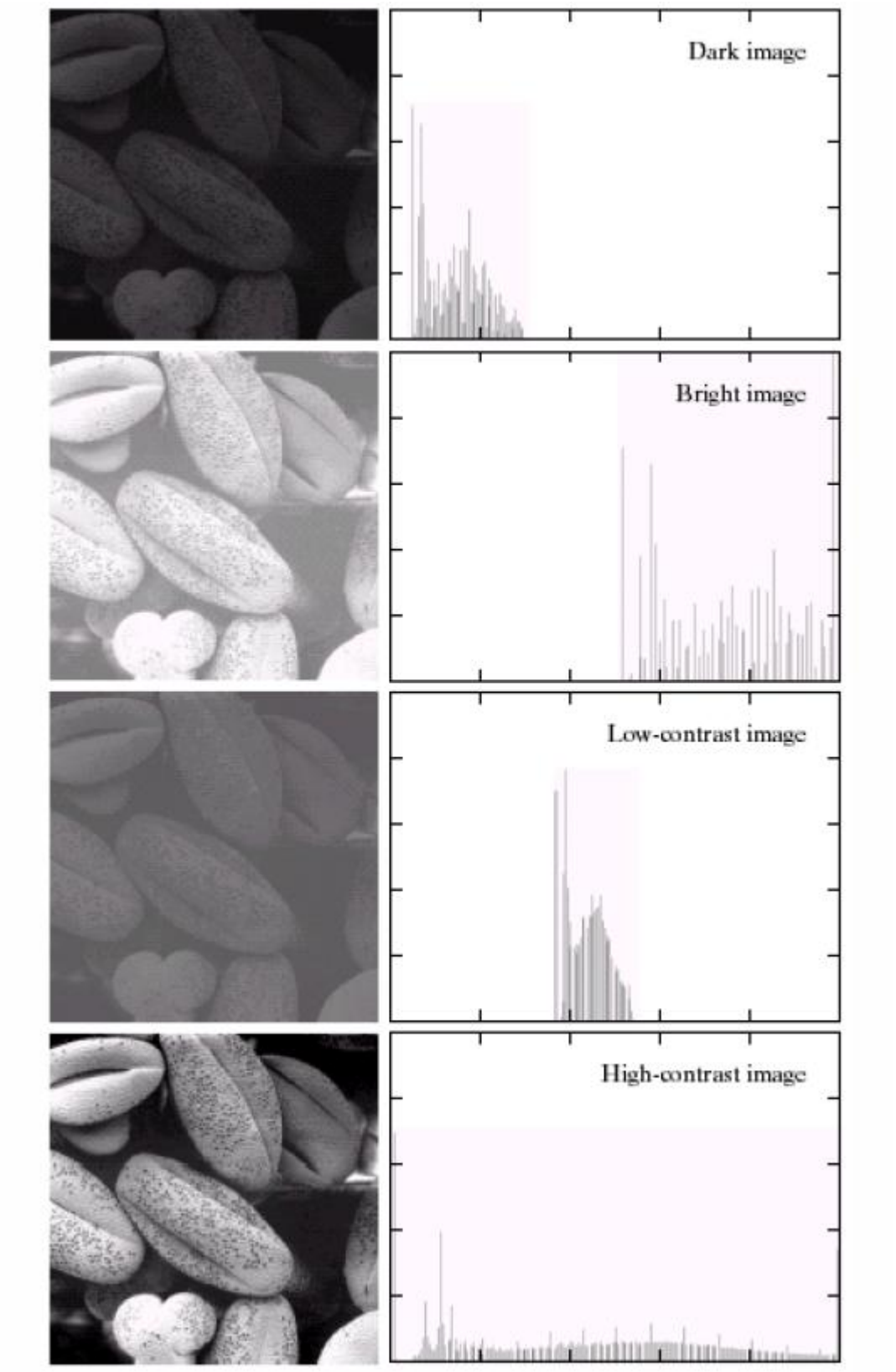
Ένα ιστόγραμμα είναι γράφημα που δείχνει την κατανομή των δεδομένων. Τα ιστογράμματα στην επεξεργασίας εικόνας χρησιμοποιούνται για να δείξουν την κατανομή των τιμών των pixles σε μια εικόνα.

Το ιστόγραμμα μίας ψηφιακής εικόνας με επίπεδα του γκρι στο διάστημα [0, L-1] είναι μία διακριτή συνάρτηση , όπου είναι το k επίπεδο γκρι και είναι το πλήθος των pixels της εικόνας, που έχουν τιμή $h(r_k) = n_k$επιπέδου γκρι . Συνήθως, κανονικοποιούμε το ιστόγραμμα, διαιρώντας κάθε τιμή με τον συνολικό αριθμό των pixels της εικόνας, έστω n. Τότε, το κανονικοποιημένο ιστόγραμμα δίνεται από την συνάρτηση $p(r_k) = \frac{n_k}{n}$, για k=0,1,…,L-1. Θα μπορούσαμε να πούμε, ότι η δίνει μία προσέγγιση της πιθανότητας της εμφάνισης ενός γκρι επιπέδου r_k.

Τα ιστογράμματα μπορεί να φανούν πολύ χρήσιμα γιατί μας βοηθούν να εξάγουμε συμπεράσματα για τη μορφή μιας εικόνας. Για παράδειγμα

- Μία σκούρα εικόνα (dark image) οι τιμές του γκρίζου θα είναι συγκεντρωμένες στα χαμηλότερα επίπεδα.
- Σε μία φωτεινή εικόνα αντίθετα οι τιμές του γκρίζου θα είναι συγκεντρωμένες σε υψηλότερα επίπεδα.
- Μία εικόνα με χαμηλό contrast θα έχει για παράδειγμα τις τιμές συγκεντρωμένες στο κέντρο.

Για να τονιστεί η χρησιμότητα του ιστογράμματος στην ψηφιακή επεξεργασία εικόνας παρατηρήστε το επόμενο σχήμα, όπου και απεικονίζεται η ίδια εικόνα σε 4 διαφορετικές μορφές, με τα αντίστοιχα ιστογράμματά τους. Ο οριζόντιος άξονας κάθε ιστογράμματος αντιστοιχίζεται στα r_k επίπεδα του γκρι , ενώ ο κάθετος άξονας στην $p(r_k)$. Στο πρώτο, π.χ. ιστόγραμμα βλέπουμε ότι τα περισσότερα pixels συγκεντρώνονται στις μικρές τιμές των γκρι επιπέδων, και άρα η εικόνα είναι αρκετά ‘σκούρα’, σε αντίθεση με το δεύτερο ιστόγραμμα. Στην τρίτη περίπτωση, παρατηρούμε ότι οι τιμές των pixels περιορίζονται σε ένα μικρό πεδίο των επιπέδων γκρι, και η εικόνα είναι χαμηλής αντίθεσης.



Θα μπορούσαμε να πούμε ότι από τις παραπάνω 4 περιπτώσεις, η ιδανική είναι η τελευταία, όπου το ιστόγραμμα «απλώνεται» σε όλο το εύρος των διαθέσιμων γκρι επιπέδων και η εικόνα είναι κατανοητή από το ανθρώπινο μάτι. Την τεχνική αυτή υλοποιεί

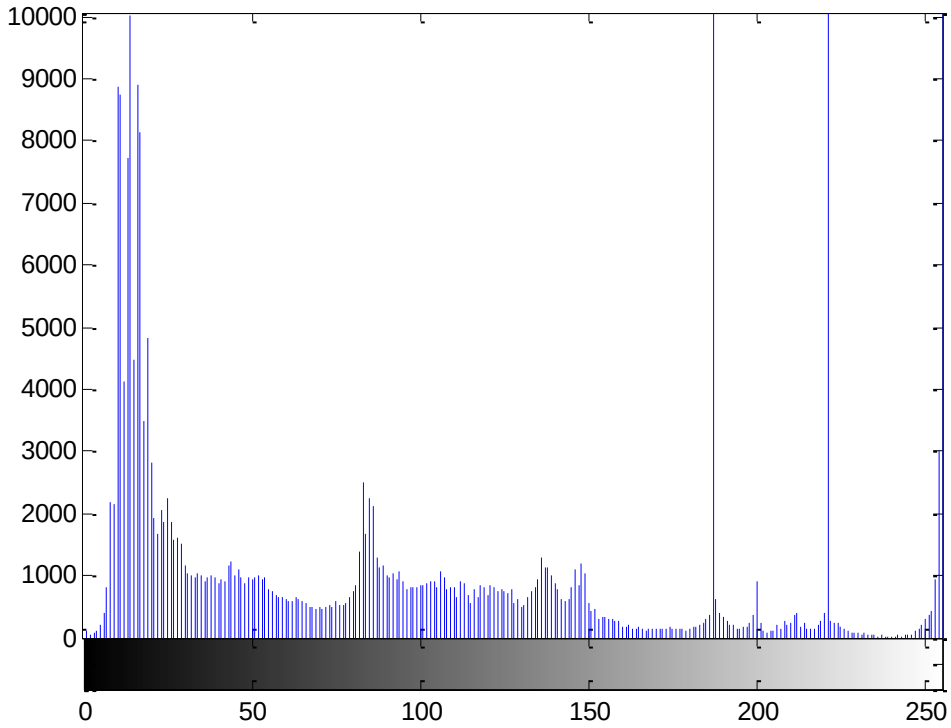
η εξισορρόπηση ιστογράμματος (histogram equalization), στοχεύει δηλαδή στο να «μετατρέψει» την κατανομή που ακολουθούν τα επίπεδα του γκρι μιας εικόνας σε ομοιόμορφη κατανομή. Την τεχνική αυτή θα δούμε στο επόμενο παράδειγμα.

Παράδειγμα 3.1

Η συνάρτηση που χρησιμοποιούμε για το ιστόγραμμα είναι η **imhist.** Δοκιμάστε για παράδειγμα να δείτε το ιστόγραμμα της grayscale εικόνας που έχουμε από τα παράδειγμα 2.1, δηλαδή του πίνακα B. Γράφουμε στο Command Window

```
imhist(B)
```

… και ανοίγει ένα νέο figure με το ιστόγραμμα της εικόνας



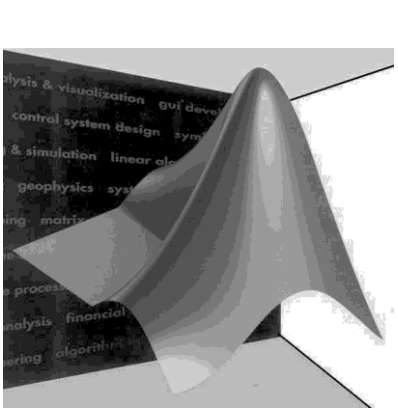
Αν το ιστόγραμμα της εικόνας δεν πληροί τις προδιαγραφές μας τότε μπορούμε να χρησιμοποιήσουμε την εντολή **histeq** (*histogram equalization*) που είναι μία αυτόματη διαδικασία η οποία αλλάζει το ιστόγραμμα σε ένα που είναι πιο ομοιογενές (uniform). Συγκεκριμένα η histeq δέχεται σαν είσοδο ένα πίνακα (που αναπαριστά μία εικόνα στο Matlab) και παράγει ένα νέο πίνακα , που είναι η νέα εικόνα στην οποία έχουν αλλάξει τα επίπεδα του γκρίζου, έτσι ώστε να έχουν πιο ομοιογενή κατανομή

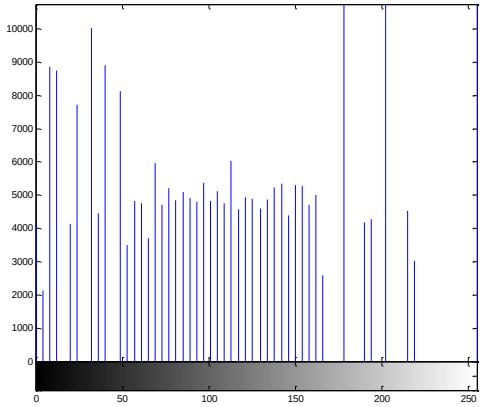
Δοκιμάστε το παρακάτω για να δείτε και μόνοι σας πως δουλεύει η histeq

```
new_B=histeq(B);
imshow(new_B)
```

Και στη συνέχεια τρέξτε την *imhist* για να δείτε το ιστόγραμμα της νέας εικόνας

```
imhist(new_B)
```





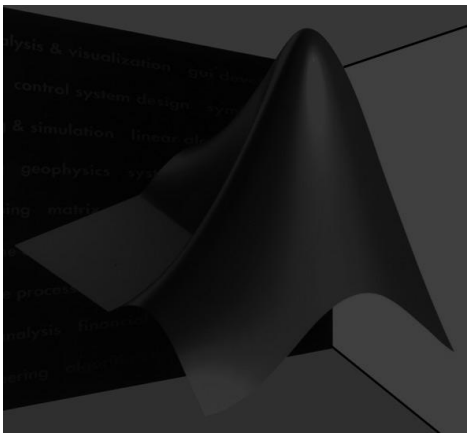
Παρατηρήστε ότι έγινε μία διαφορετική κατανομή των τιμών στο ιστόγραμμα

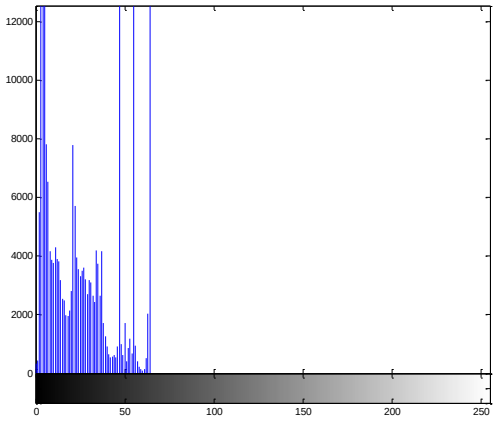
Παράδειγμα 3.2

Ας δούμε τώρα καλύτερα το πώς δουλεύει η histeq σε μία σκούρα εικόνα. Μπορούμε να δημιουργήσουμε μία πιο σκούρα εικόνα χρησιμοποιώντας την εντολή **imdivide** (divide image by constant) . Θα διαιρέσουμε δηλαδή όλες τις τιμές του γκρί με μία σταθερά για να πάρουμε μικρότερες τιμές , δηλαδή πιο κοντά στο μαύρο (που είναι το 0).

Συνεχίζοντας με την grascale εικόνας μας που είναι αποθηκευμένη στον πίνακα B γράφουμε

```
E=imdivide(B,4);
 imshow(E)
imhist(E)
```

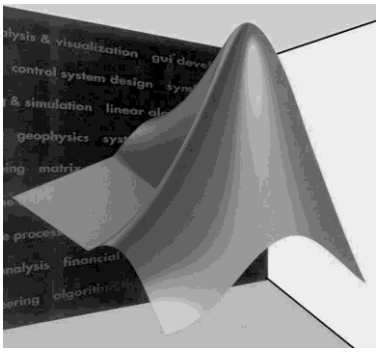


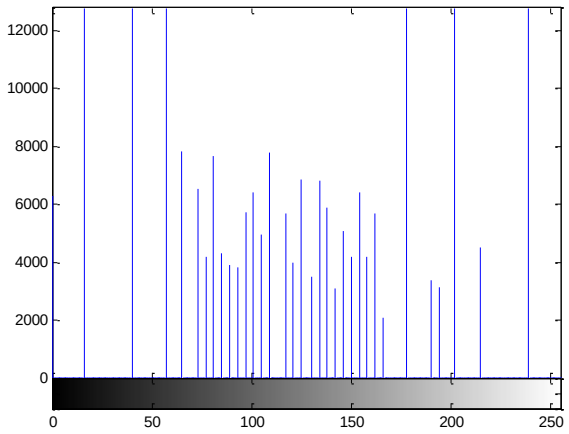


Και βλέπουμε τη διαφορά στην φωτεινότητα της εικόνας ενώ στο ιστόγραμμα της παρατηρούμε ότι οι τιμές είναι συγκεντρωμένες στα χαμηλότερα επίπεδα.

Ας χρησιμοποιήσουμε τώρα την histeq όπως κάναμε και πριν

```
Eh=histeq(E);
 imshow(Eh) ;
figure;
imhist(Eh)
```





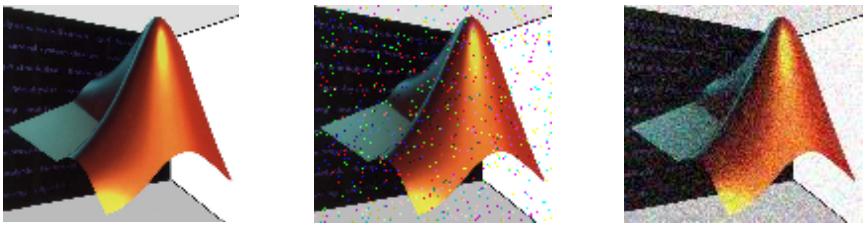
5. Θόρυβος

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 5.1

Τρέξτε τον παρακάτω κώδικα

```
A=imread('mlogo.jpg');
subplot(1,3,1), imshow(A);
Isp=imnoise(I,'salt & pepper',0.03); %Add 3% (0.03) salt and pepper noise
subplot(1,3,2), imshow(Isp); %Display result image Isp
Ig=imnoise(A,'gaussian',0.02); %Add Gaussian noise (with 0.02 variance)
subplot(1,3,3), imshow(Ig); %Display result image Ig
```



Η αποκατάσταση του θορύβου θα σας δοθεί σαν άσκηση …

Πρώτα όμως πρέπει να δούμε την έννοια τις εφαρμογής φίλτρων σε εικόνες. Γιατί όμως να εφαρμόσω φίλτρα σε εικόνες ; Η πιο γενική απάντηση που μπορούμε να δώσουμε σε ένα τέτοιο ερώτημα είναι ότι θέλουμε οι επεξεργασμένες εικόνες να είναι πιο «καλές» από τις αρχικές . 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 pixels (δείτε σχήμα παρακάτω) .
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 target pixel.

Η ιδέα είναι ότι μετακινούμε μία «μάσκα» όπως λέγεται, πάνω από τη δοσμένη εικόνα και σαν αποτέλεσμα παίρνουμε μία νέα εικόνα στην οποία οι τιμές των pixels έχουν τιμές που έχουν υπολογιστεί από την εφαρμογή της μάσκας. Γενικά, 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. Αν το κέντρο της μάσκας είναι στην θέση (x,y), η τιμή που έχει η εικόνα εκεί αντικαθίσταται από το προηγούμενο άθροισμα. Η μάσκα έπειτα μετακινείται στο επόμενο pixel και η διαδικασία επαναλαμβάνεται. 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, $\omega = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$

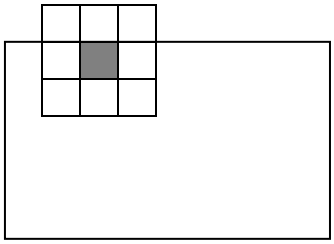
Άρα, αντικαθιστούμε κάθε pixel της εικόνας με τον μέσο όρο των γειτονικών του pixel.

	a	b	c	
	d	e	f	
	g	h	i	

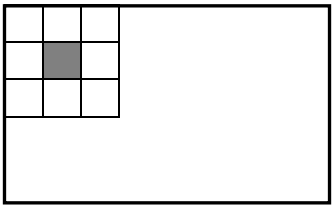
$$\frac{1}{9}(a+b+c+d+e+f+g+h+i)$$

Τα μη γραμμικά φίλτρα εφαρμόζονται στην εικόνα με τον ίδιο τρόπο με τα γραμμικά με τη μόνη διαφορά ότι η τιμή για κάθε pixels είναι ένα μη γραμμικός συνδυασμός των γειτονικών pixels.

Τι γίνεται όμως 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.