

Ζητούμενα:

1) Το πρόβλημα με τις εικόνες flowers και Merilin είναι πως δεν χρησιμοποιούν όλο το διαθέσιμο εύρος τιμών για την αναπαράστασή τους, αλλά ένα διάστημα αυτού. Θα προσπαθήσουμε να διορθώσουμε το πρόβλημα αυτό αξιοποιώντας πλήρως όλο το εύρος τιμών (0,255) εφαρμόζοντας τον γραμμικό μετασχηματισμό. Λαμβάνοντας υπόψη από τις διαλέξεις του μαθήματος τα εξής:

$[i_{min}, i_{max}]$: αρχικό εύρος τιμών της εικόνας

$[d_{min}, d_{max}]$: επιθυμητό εύρος τιμών της εικόνας

το οποίο επιθυμητό εύρος προκύπτει από τον κατάλληλο γραμμικό μετασχηματισμό.

$$w_1 * i_{min} + w_2 = d_{min}$$

$$w_1 * i_{max} + w_2 = d_{max}$$

Προκύπτει ένα σύστημα 2 αγνώστων, το οποίο το λύνω και υπολογίζω τα w_1 και w_2 . Έτσι έχω:

$$w_1 = \frac{d_{max} - d_{min}}{i_{max} - i_{min}}$$

$$w_2 = d_{min} - \left(\frac{d_{max} - d_{min}}{i_{max} - i_{min}} \right) * i_{min} = d_{min} - w_1 * i_{min}$$

και τέλος πολλαπλασιάζω κάθε pixel της εικόνας με το w_1 και του προσθέτω το w_2 .

Παρακάτω φαίνεται η συνάρτηση που υλοποιεί τον παραπάνω γραμμικό μετασχηματισμό:

```
function new_im = erwthma1(im, dmin, dmax)
    imin = min(im(:));
    imax = max(im(:));
    w1 = (dmax-dmin)/(imax-imin);
    w2 = dmin - w1*imin;
    new_im = round(w1*im+w2);
    figure(1);
    imshow(uint8(im));
    figure(2);
    imshow(uint8(new_im));
end
```

Η παραπάνω συνάρτηση παίρνει σαν είσοδο την εικόνα προς επεξεργασία, η οποία δεν αξιοποιεί όλο το δυνατό εύρος τιμών, την μικρότερη και την μεγαλύτερη τιμή του

επιθυμητού εύρους. Δίνει σαν έξοδο την εικόνα με το επιθυμητό εύρος τιμών, κάνοντας τα βήματα που περιγράψαμε παραπάνω.

Αφού φορτώσουμε τα αρχεία flowers.mat και Merilin.mat πληκτρολογώντας **load flowers;** και **load Merilin;** αντίστοιχα,

Στιγμιότυπο μεταβλητών:

Name ▲	Value	Min	Max
Mer	<256x256 double>	100	255
x_fl	<256x256 double>	23	137

καλούμε την συνάρτηση ως εξής:

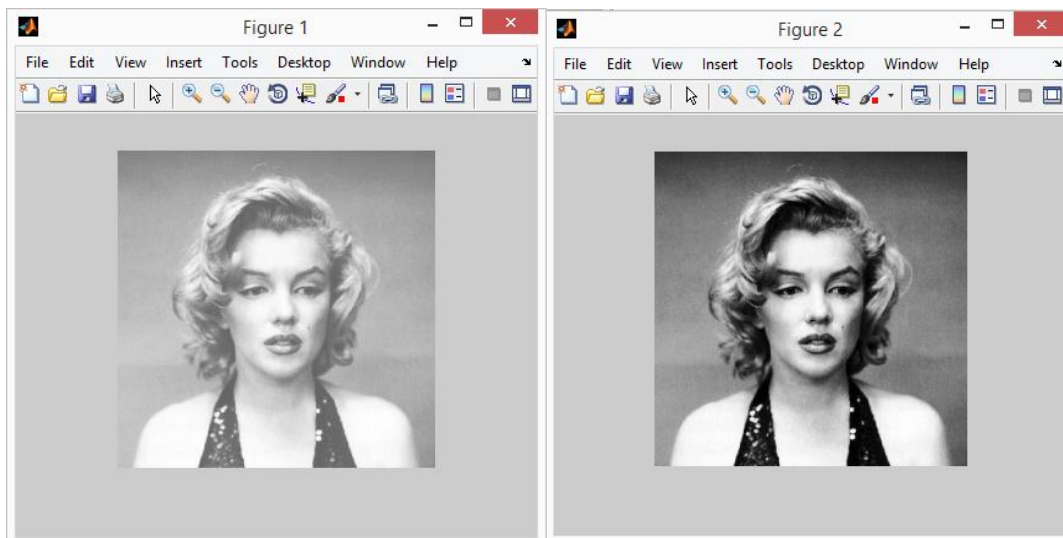
```
new_Mer = erwthma1(Mer,0,255);
```

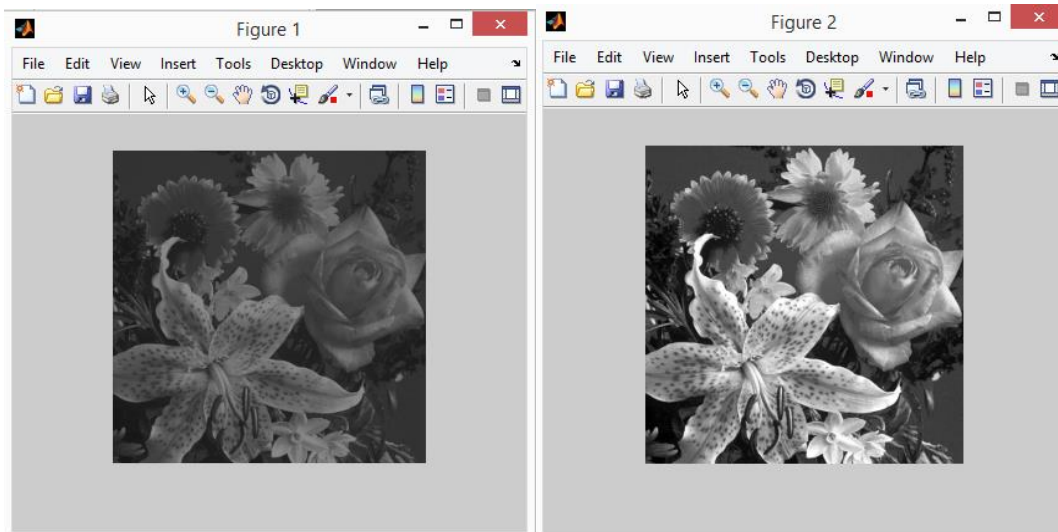
```
new_flow = erwthma1(x_fl,0,255);
```

Αποτελέσματα εικόνων πριν και μετά από την εκτέλεση της συνάρτησης:

Πριν

Μετά





Παρατηρούμε την μεγάλη διαφορά ανάμεσα στις εικόνες που είχαμε και στις εικόνες που πήραμε, αφού χρησιμοποιήσαμε όλο το δυνατό εύρος τιμών.

2) Όπως αναφέρεται και στην εκφώνηση το πρόβλημα με τις εικόνες image1 και church1, είναι ότι έχουν χαμηλή αντίθεση (contrast). Για βελτιώσουμε την αντίθεση τους θα χρησιμοποιήσουμε την ολική ισοστάθμιση. Για το ερώτημα 2 υλοποιήθηκαν 2 συναρτήσεις η μία παίρνει σαν είσοδο την εικόνα, δίνοντας μας το ιστόγραμμα της και η άλλη παίρνει σαν είσοδο την εικόνα εφαρμόζοντας την ολική ισοστάθμιση. Για να φτιάξουμε το ιστόγραμμα μίας εικόνας χρησιμοποιήσαμε τον εξής τύπο από τις διαλέξεις του μαθήματος:

$$p(r_k) = \frac{n_k}{n}, \quad k=0,1,2,\dots,L-1, \text{ όπου}$$

r_k : η k -οστή τιμή της έντασης

n_k : ο αριθμός των pixels με ένταση r_k

n : ο συνολικός αριθμός των pixels της εικόνας,

τέλος το άθροισμα όλων των τιμών του ιστογράμματος πρέπει να είναι ίσο με 1, εφόσον αναφερόμαστε σε πιθανότητες. Για κάθε τιμή έντασης 0,...,255, υπολογίζουμε την πιθανότητα εμφάνισης της στην εικόνα. Παρακάτω φαίνεται ο κώδικας της συνάρτησης που υπολογίζει το ιστόγραμμα της εικόνας.

```

function [] = erwthma2_1(image)
%pinakas gia kathe timh entashs apo min ews max
x(256)=0;
%pinakas p(r) me tis pathanothtes emfanishs
y(256)=0;
for i=0:255
    x(i+1) = i;
    y(i+1) = size(find(image==i),1)/(size(image,1)*
size(image,2));
end
figure;
plot(x,y);
xlabel('H k-OSTH TIMH THS ENTASHS: rk');
ylabel('H PITHANOTHTA EMFANISHS ENOS EPIPEDOY ENTASHS rk:
p(rk)');
title('ISTOGRAMMA THS EIKONAS');
end

```

Για να εφαρμόσουμε τώρα την ολική ισοστάθμιση, ώστε να αποδώσουμε μια καινούργια τιμή έντασης σε κάθε παλιά, θα χρειαστεί να εφαρμόσουμε τον εξής τύπο:

$$S_k = T(r_k) = (L-1) * \sum_{i=0}^k pr(ri), k=0,1,2,...,L-1$$

Παρακάτω φαίνεται ο κώδικας την συνάρτησης που παίρνει σαν είσοδο την εικόνα και δημιουργεί μία καινούργια στην οποία αποδίδει μία καινούργια τιμές έντασης για κάθε παλιά.

```

function [new] = erwthma2_2(image)
%pinakas gia kathe timh entashs apo min ews max
x(256)=0;
%pinakas p(r) me tis pithanothtes emfanishs
y(256)=0;
%pinakas t(r) isostathmishs istogrammatos
t(256)=0;
for i=0:255
    x(i+1) = i;
    y(i+1) = size(find(image==i),1)/(size(image,1)*
size(image,2));
end
for i=0:255
    sum = 0;
    for j=0:i
        sum = sum + y(j+1);
    end
    t(i+1)=max(x)*sum;
end

```

```


end
new = image;
for i=1:size(new,1)
    for j=1:size(new,2)
        new(i,j) = t(find(x==image(i,j)));
    end
end
figure;imshow(uint8(image));
figure;imshow(uint8(new));
end

```

Φορτώνουμε αυτές τις δύο εικόνες πληκτρολογώντας:

load image1;

load church1;

Name ▲	Value	Min	Max
 ch	<440x427 double>	0	255
 im	<544x811 double>	0	255

Για να εφαρμόσουμε την ολική ισοστάθμιση καλούμε την δεύτερη συνάρτηση και για τις δύο εικόνες πληκτρολογώντας:

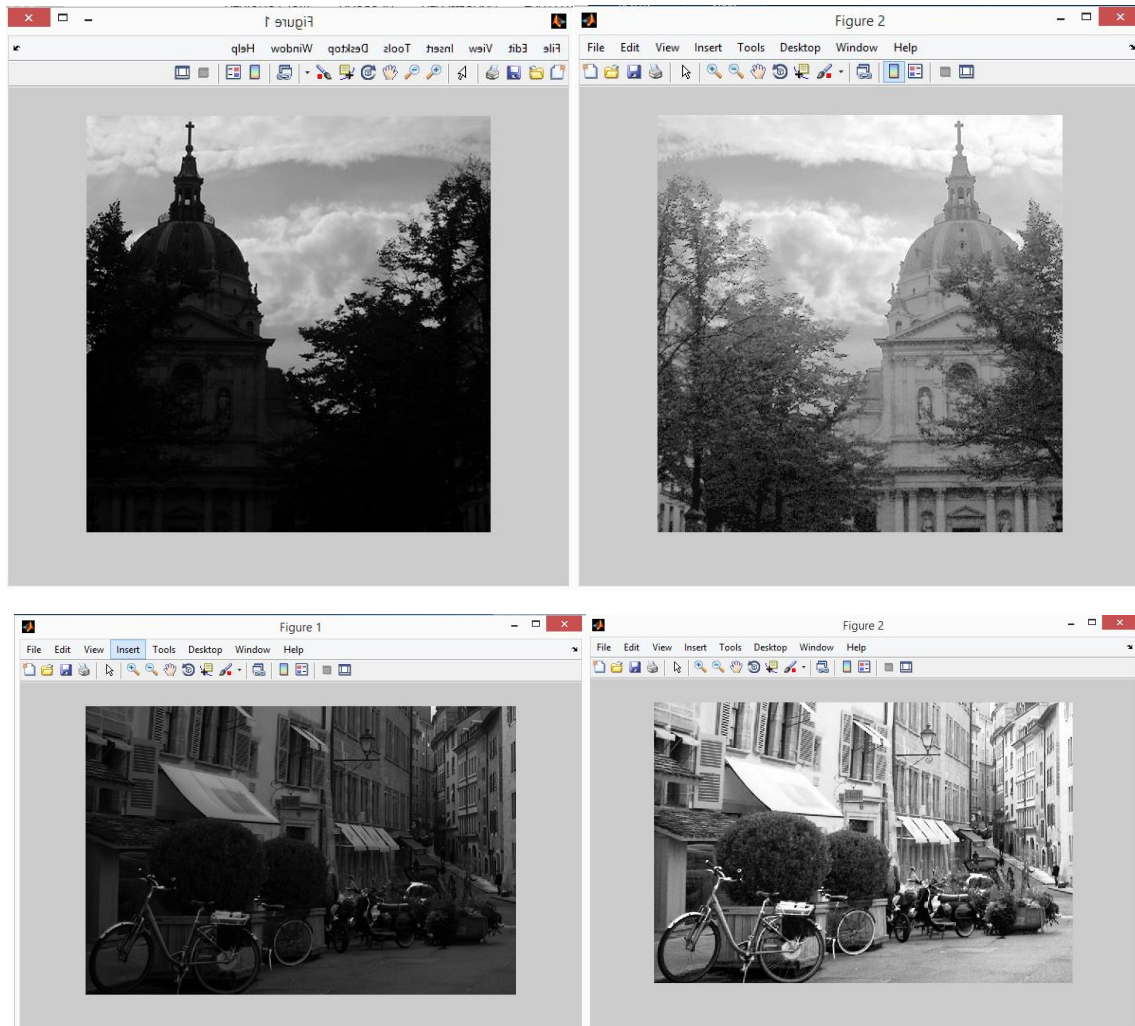
new_ch = erwthma2_2(ch);

new_im = erwthma2_2(im); .

Τα αποτελέσματα πριν και μετά από την εκτέλεση της συνάρτησης φαίνονται παρακάτω:

Πριν

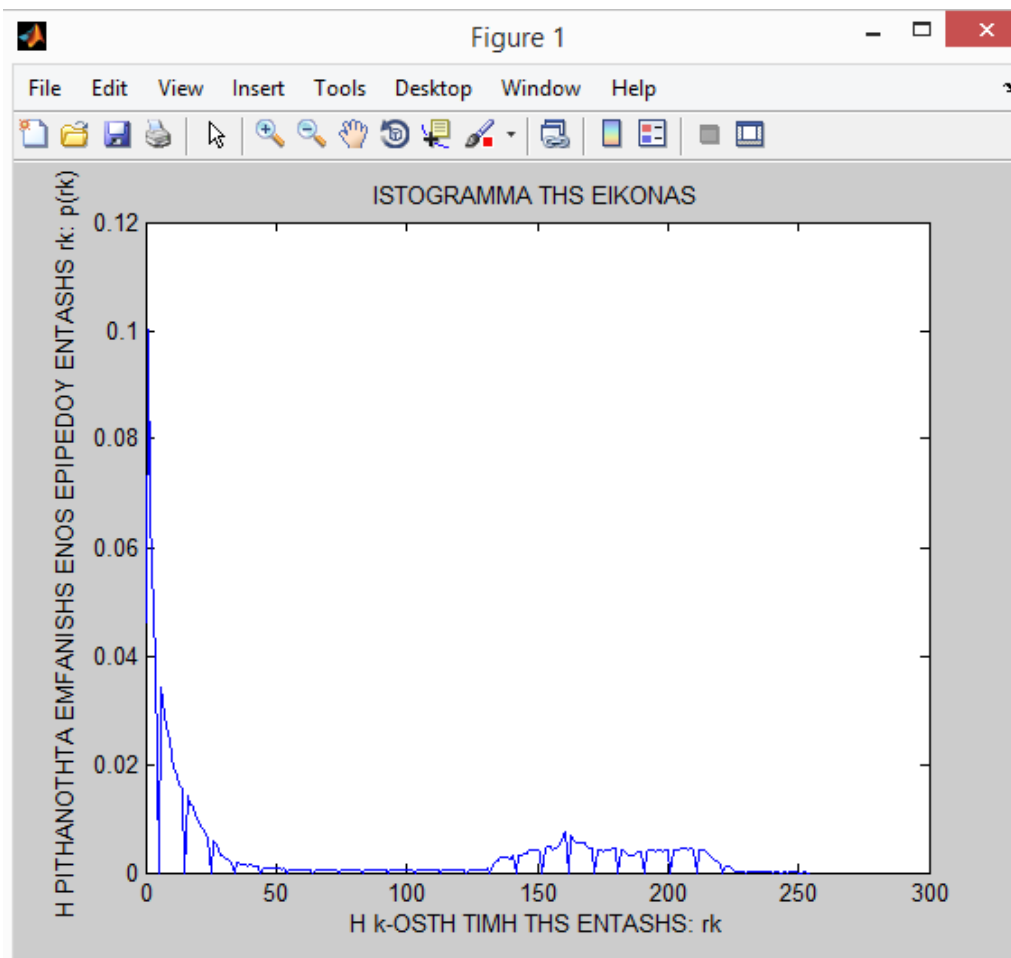
Μετά



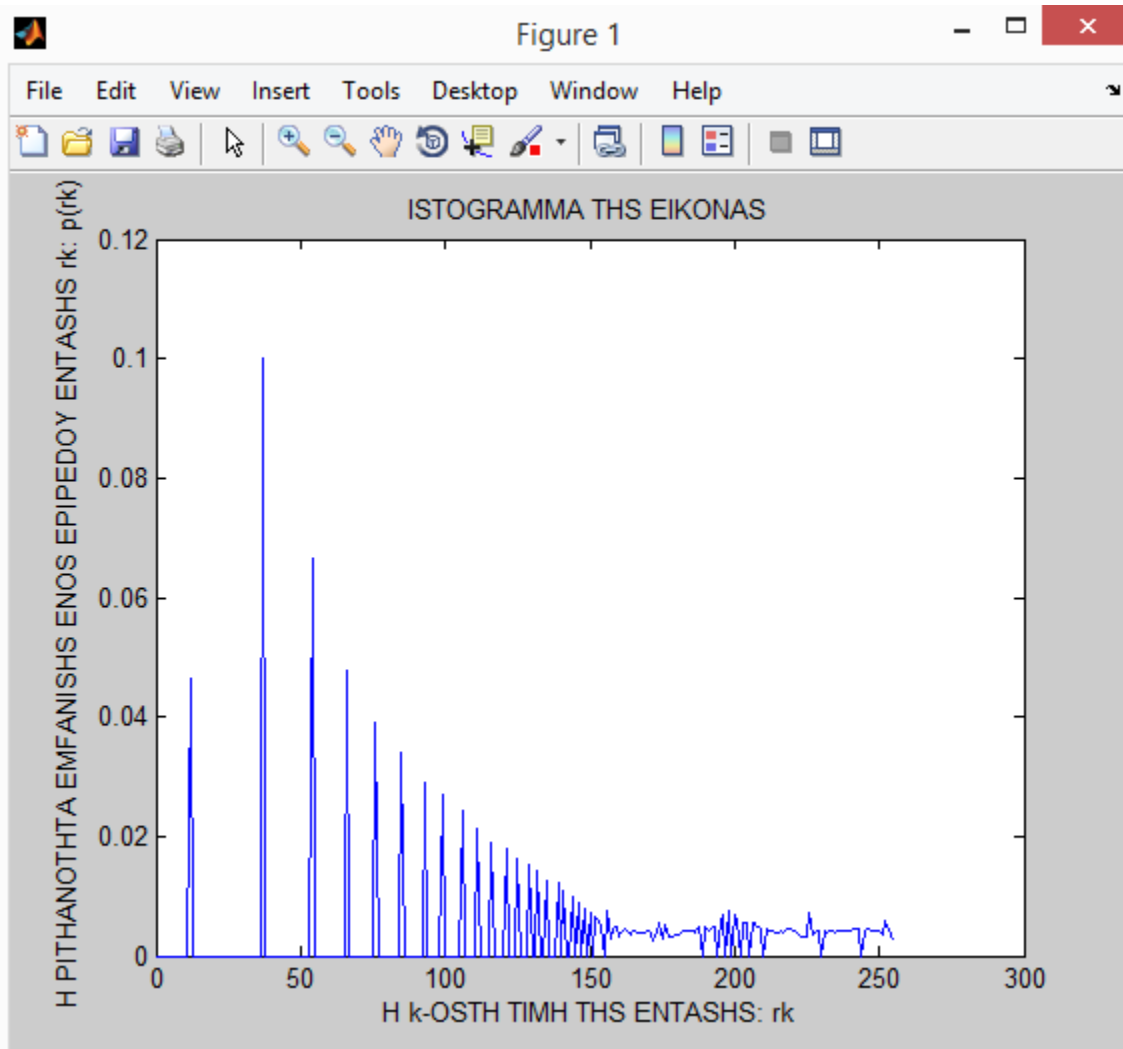
Για να πάρουμε τα ιστογράμματα κάθε εικόνας πριν και μετά την εφαρμογή της ολικής ισοστάθμισης εκτελούμε:

ι)για την εικόνα church1 θα πάρουμε το ιστόγραμμα της πριν την επεξεργασία πληκτρολογώντας **erwthma2_1(ch)**; και το ιστόγραμμα της μετά την επεξεργασία δίνοντας σαν είσοδο τον επεξεργασμένο πλέον πίνακα πληκτρολογώντας **erwthma2_1(new_ch)**; αφού πρώτα τον στρογγυλοποιήσουμε πληκτρολογώντας **new_ch = round(new_ch)**; τα αποτελέσματα πριν και μετά την ισοστάθμιση φαίνονται παρακάτω.

Πρω

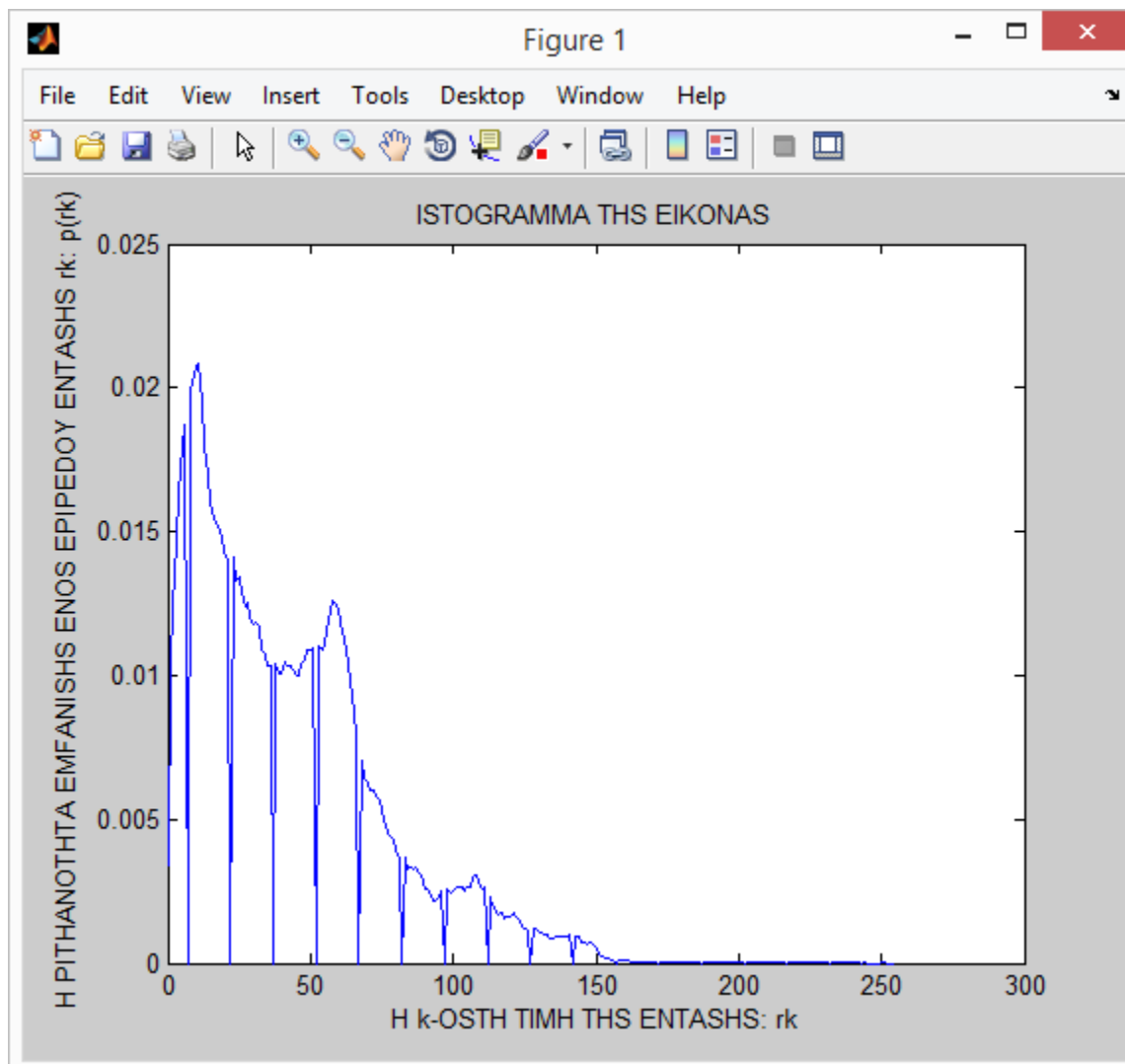


Μετά



ii) για την εικόνα `image1` θα πάρουμε το ιστόγραμμα της πριν την επεξεργασία πληκτρολογώντας `erwthma2_1(im)`; και το ιστόγραμμα της μετά την επεξεργασία δίνοντας σαν είσοδο τον επεξεργασμένο πλέον πίνακα πληκτρολογώντας `erwthma2_1(new_im)`; αφού πρώτα τον στρογγυλοποιήσουμε πληκτρολογώντας `new_im = round(new_im)`; τα αποτελέσματα πριν και μετά την ισοστάθμιση φαίνονται παρακάτω.

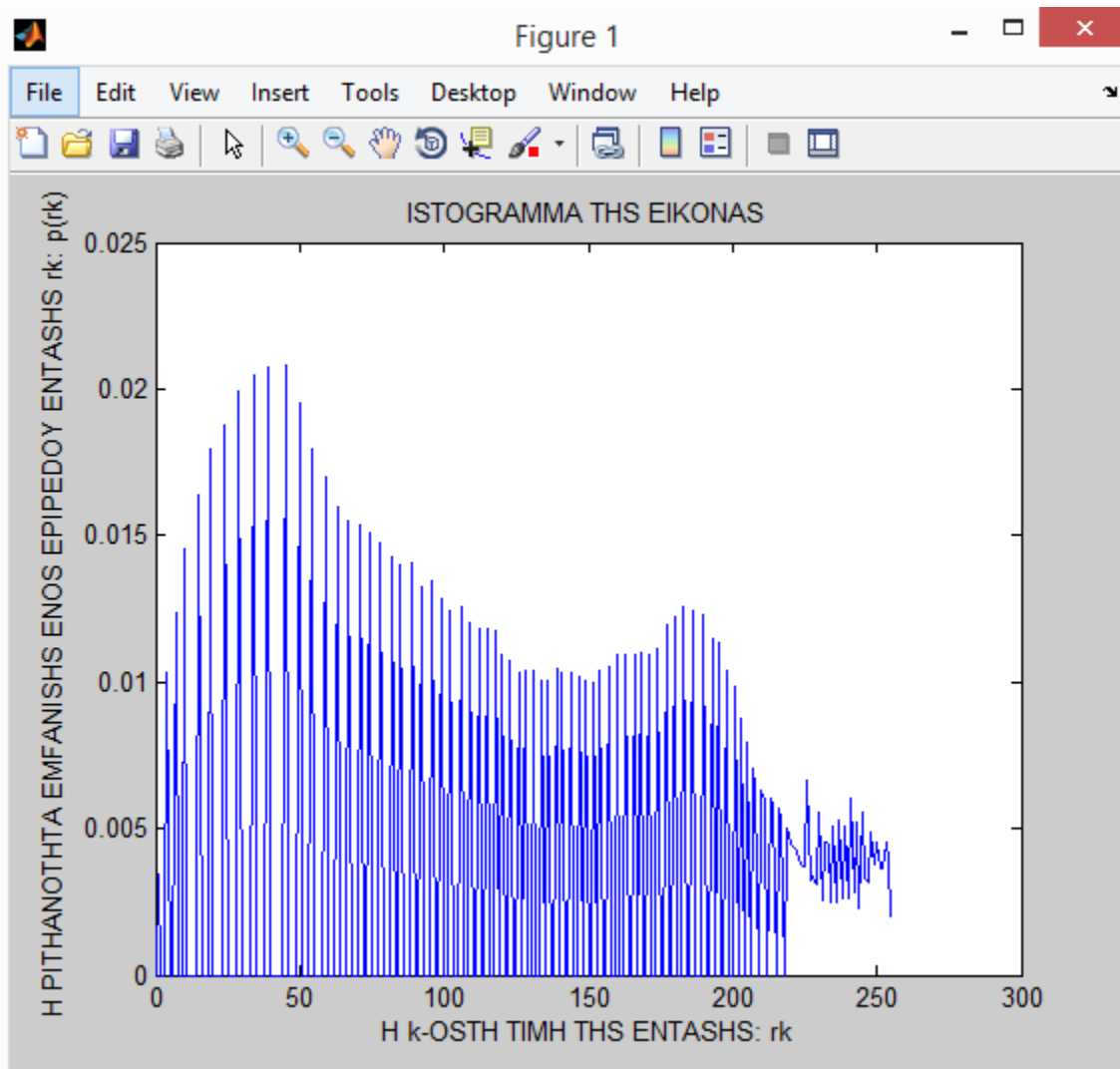
Πριν



Στιγμιότυπο μεταβλητών:

Name ▲	Value	Min	Max
ch	<440x427 double>	0	255
im	<544x811 double>	0	255
new_ch	<440x427 double>	12	255
new_im	<544x811 double>	1	255

Μετά



Παρατηρούμε και για τις δύο εικόνες πως πριν την επεξεργασία το ιστόγραμμα έχει συγκεντρωμένες τις μεγαλύτερες πιθανότητες κοντά στο μηδέν πράγμα το οποίο σημαίνει πως η εικόνα είναι πιο σκοτεινή, ενώ μετά την επεξεργασία το ιστόγραμμα έχει απλώσει, πράγμα που σημαίνει πως η αντίθεση έχει πλέον βελτιωθεί σε σχέση με πριν.

3)α) Θα προσπαθήσουμε να βελτιώσουμε τις έγχρωμες εικόνες museum και dscn1078, εφαρμόζοντας ολική ισοστάθμιση σε κάθε μια συνιστώσα χωριστά. Αυτό θα γίνει χρησιμοποιώντας την συνάρτηση `erwthma2_2()`, που υλοποιήσαμε στο προηγούμενο ερώτημα, αλλά δίνοντας της κάθε φορά σαν είσοδο μόνο την μία συνιστώσα. Φτιάχνουμε μια νέα συνάρτηση `erwthma3_a()`, η οποία έχει ακριβώς τον ίδιο κώδικα

της `erwthma2_2()`, εκτός από τις παρακάτω γραμμές:

```
figure;imshow(uint8(image));  
figure;imshow(uint8(new)); .
```

Επιλέγοντας New -> Script, γράφουμε τον εξής κώδικα με όνομα αρχείου `erwthma3main_a()`:

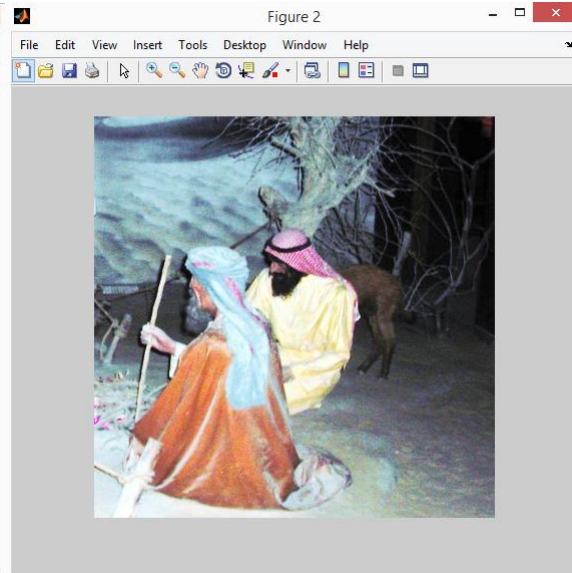
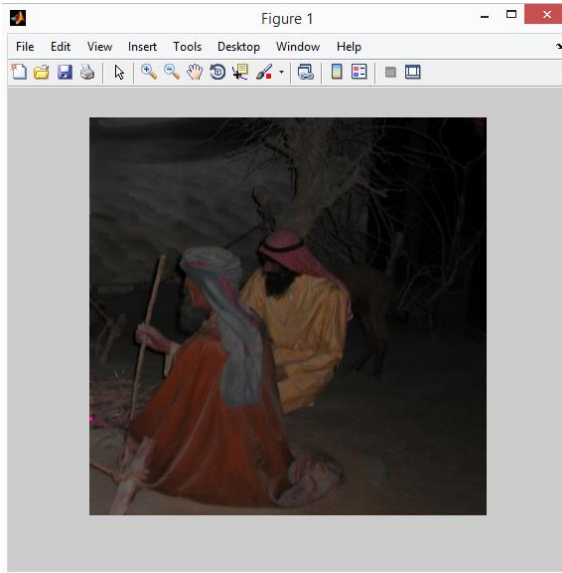
```
museum=imread('museum.jpg');  
museum=double(museum);  
new_museum=zeros(size(museum));  
new_museum(:,:,1)=erwthma3_a(museum(:,:,1));  
new_museum(:,:,2)=erwthma3_a(museum(:,:,2));  
new_museum(:,:,3)=erwthma3_a(museum(:,:,3));  
figure(1);  
imshow(uint8(museum));  
figure(2);  
imshow(uint8(new_museum));  
  
dscn1078=imread('dscn1078.jpg');  
dscn1078=double(dscn1078);  
new_dscn1078=zeros(size(dscn1078));  
new_dscn1078(:,:,1)=erwthma3_a(dscn1078(:,:,1));  
new_dscn1078(:,:,2)=erwthma3_a(dscn1078(:,:,2));  
new_dscn1078(:,:,3)=erwthma3_a(dscn1078(:,:,3));  
figure(3);  
imshow(uint8(dscn1078));  
figure(4);  
imshow(uint8(new_dscn1078));
```

Στον παραπάνω κώδικα για την κάθε εικόνα εκτελούμε τα ίδια ακριβώς βήματα. Αρχικά διαβάζουμε την εικόνα και την αποθηκεύουμε σε ένα πίνακα, κάνουμε αυτό τον πίνακα μετατροπή σε `double`, δημιουργούμε έναν άλλο πίνακα στον οποίο αποθηκεύεται η καινούργια εικόνα δίνοντας του διαστάσεις ίδιες με τις διαστάσεις την παλιάς εικόνας και περιεχόμενο όλα μηδενικά, θέτουμε κάθε συνιστώσα της καινούργιας εικόνας ίση με την παλιά συνιστώσα αφού πρώτα εφαρμόσουμε την ολική ισοστάθμιση, μέσω της συνάρτησης και τέλος εμφανίζουμε την παλιά και την καινούργια εικόνα. Τρέχουμε το παραπάνω script πληκτρολογώντας **erwthma3main_a** .

Αποτελέσματα από την εκτέλεση του παραπάνω script πριν και μετά την ολική ισοστάθμιση κάθε συνιστώσας:

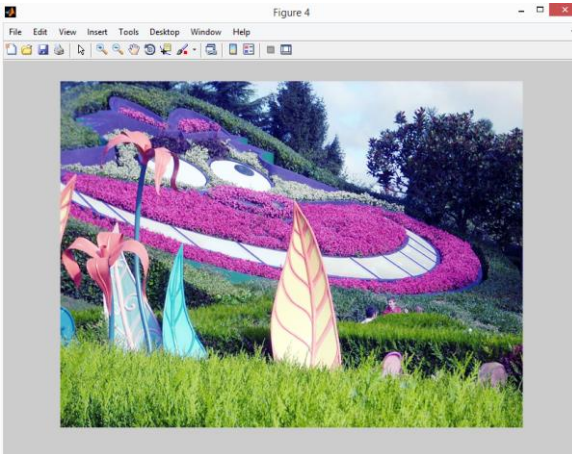
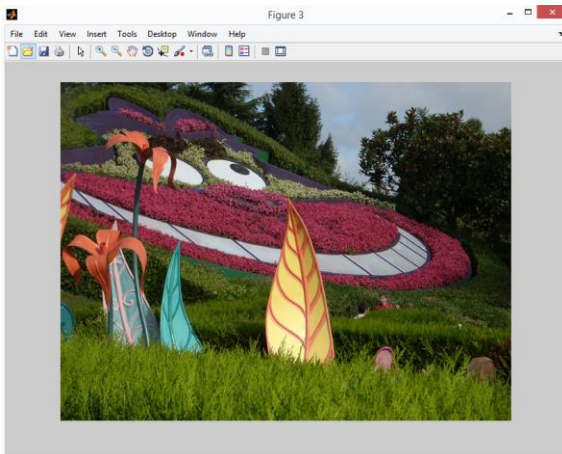
Πριν

Μετά



Πριν

Μετά



Στιγμιότυπο μεταβλητών:

Name ▲	Value	Min	Max
dscn1078	<500x667x3 double>	<Too many elements>	<Too many elements>
museum	<400x400x3 double>	0	244
new_dscn1078	<500x667x3 double>	<Too many elements>	<Too many elements>
new_museum	<400x400x3 double>	3.5063	255.0000

β) Το HSI μοντέλο είναι πιο κοντά στην περιγραφή του χρώματος με βάση το πώς το καταλαβαίνει ο άνθρωπος. Οι άνθρωποι περιγράφουν ένα χρώμα με βάση την απόχρωση του (Hue), την καθαρότητά του (τον κορεσμό - Saturation) και τη λαμπρότητά του (την έντασή του - Intensity). Στην περίπτωση μας θα εφαρμόσουμε την ολική ισοστάθμιση μόνο στην τρίτη συνιστώσα του HSI μοντέλου, η οποία αντιστοιχεί

στην ένταση – Intensity. Επίσης με το HSI μοντέλο μπορούμε να εφαρμόσουμε τεχνικές που εφαρμόζονται σε gray-scale εικόνες, λόγω του ότι αποσυνδέει τη χρωματική και τη gray-scale πληροφορία μιας εικόνας.

Πατώντας New -> Script δημιουργούμε ένα νέο script με όνομα **erwthma3main_b**, στο οποίο προσθέτουμε τον παρακάτω κώδικα:

```
mus=imread('museum.jpg');
mus=double(mus);
mus_hsv=rgb2hsv(mus);
new_mus=zeros(size(mus_hsv));
new_mus(:,:,1)=mus_hsv(:,:,1);
new_mus(:,:,2)=mus_hsv(:,:,2);
new_mus(:,:,3)=erwthma3_a(mus_hsv(:,:,3));
new_mus=hsv2rgb(new_mus);
figure(1);
imshow(uint8(mus));
figure(2);
imshow(uint8(new_mus));

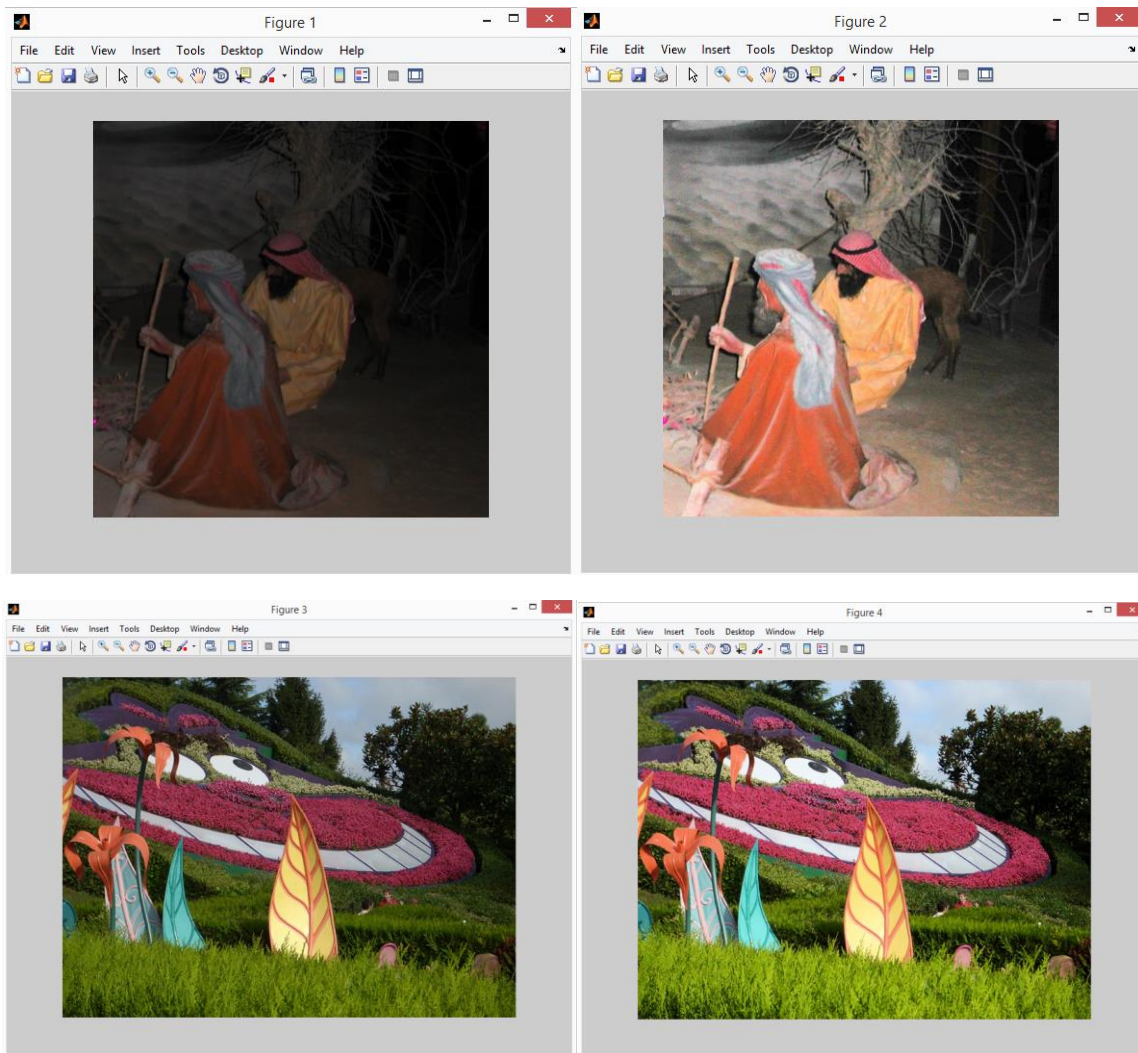
dscn=imread('dscn1078.jpg');
dscn=double(dscn);
dscn_hsv=rgb2hsv(dscn);
new_dscn=zeros(size(dscn_hsv));
new_dscn(:,:,1)=dscn_hsv(:,:,1);
new_dscn(:,:,2)=dscn_hsv(:,:,2);
new_dscn(:,:,3)=erwthma3_a(dscn_hsv(:,:,3));
new_dscn=hsv2rgb(new_dscn);
figure(3);
imshow(uint8(dscn));
figure(4);
imshow(uint8(new_dscn));
```

Αρχικά διαβάζουμε την εικόνα που θα επεξεργαστούμε, την αποθηκεύουμε σε έναν πίνακα και την μετατρέπουμε σε double. Προχωράμε σε μετάβαση από το rgb στο hsv μοντέλο και το αποτέλεσμα το αποθηκεύουμε σε νέο πίνακα. Θέτουμε τις διαστάσεις του νέου πίνακα της εικόνας και τον γεμίζουμε με μηδενικά. Αντιγράφουμε τις δύο πρώτες συνιστώσες στον νέο πίνακα και εφαρμόζουμε ολική ισοστάθμιση μόνο στην τρίτη συνιστώσα. Τέλος κάνουμε μετάβαση στο rgb μοντέλο και απεικονίζουμε τις εικόνες μας. Εκτελούμε το script πληκτρολογώντας **erwthma3main_b**.

Αποτελέσματα από την εκτέλεση του παραπάνω script πριν και μετά την ολική ισοστάθμιση της τρίτης συνιστώσας:

Πριν

Μετά



Παρατηρούμε πως η ολική ισοστάθμιση είχε καλύτερα αποτελέσματα στο HSI μοντέλο από ότι στο RGB μοντέλο. Οι εικόνες που υπέστησαν επεξεργασία και στις τρεις συνιστώσες του RGB μοντέλου φαίνονται σαν να έχουν ξεθωριάσει σε σχέση με τις εικόνες που υπέστησαν επεξεργασία στην μία μόνο συνιστώσα της έντασης του HSI μοντέλου. Αυτό συμβαίνει, γιατί μεταβάλλοντας και τις τρεις συνιστώσες του RGB μοντέλου είναι σαν να μεταβάλλουμε και το χρώμα του κάθε pixel, εφόσον και οι τρεις συνιστώσες χρησιμοποιούνται για την αναπαράσταση ενός pixel.

4) Με την color slicing τεχνική επιτυγχάνεται ο τονισμός συγκεκριμένου εύρους χρωμάτων με σκοπό να το απομονώσουμε από το background. Η τεχνική που θα χρησιμοποιήσουμε στο συγκεκριμένο ερώτημα για την επεξεργασία των εικόνων wall και Saint-Nikolaos, είναι αυτή που χρησιμοποιούμε την σφαίρα για τον καθορισμό των χρωμάτων που μας ενδιαφέρουν και όχι τον κύβο, λόγω του ότι η σφαίρα έχει

καλύτερα αποτελέσματα. Για την υλοποίηση της παραπάνω τεχνικής δημιουργήθηκε μια συνάρτηση με όνομα `erwthma4` η οποία παίρνει σαν είσοδο το όνομα της εικόνας προς επεξεργασία και απομονώνει το χρώμα που εμείς θα υποδείξουμε. Ο κώδικας φαίνεται παρακάτω:

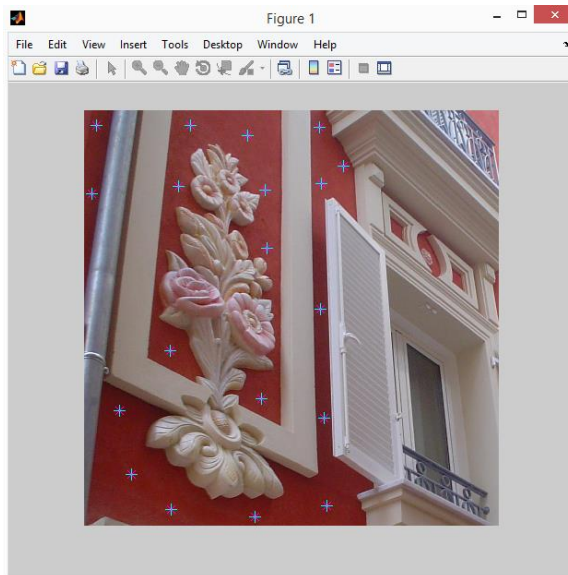
```
function [] = erwthma4(image)
wall=imread(strcat(image, '.jpg'));
wall=double(wall);
pixels=imixel(uint8(wall));
main_point = [mean(pixels(:,1)) mean(pixels(:,2))
mean(pixels(:,3))];

for i=1:size(wall,1)
    for j=1:size(wall,2)
        sum=0;
        for k=1:size(wall,3)
            temp=power(wall(i,j,k)-main_point(k),2);
            sum=sum+temp;
        end
        if (sum<power(50,2))
            wall(i,j,1)=255;
            wall(i,j,2)=255;
            wall(i,j,3)=0;
        end
    end
end
figure(2);
imshow(uint8(wall));
end
```

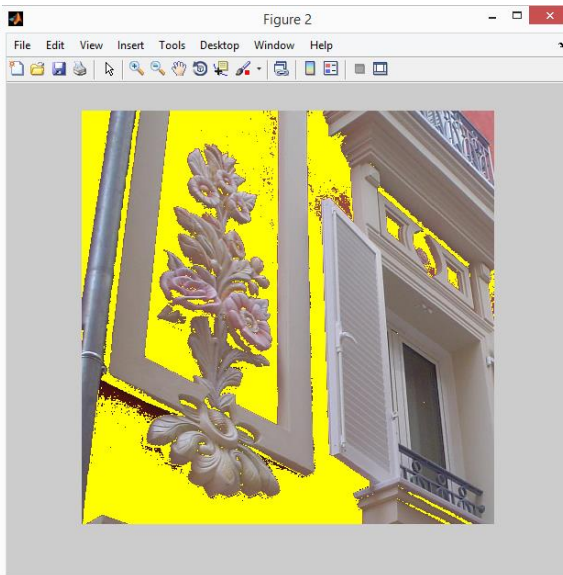
Αφού διαβάσουμε την .jpg εικόνα με το όνομα της σαν είσοδο στην συνάρτηση, την μετατρέψουμε σε `double`, την προβάλλουμε στον χρήστη με σκοπό να κάνει κλικ στα σημεία που θέλει να αλλάξει το χρώμα της, χρησιμοποιώντας την `imixel` η οποία επιστρέφει 3 τιμές για κάθε κλικ που κάναμε πάνω στην εικόνα, μια για κάθε συνιστώσα. Στην συνέχεια υπολογίζουμε το διάνυσμα `main_point` το οποίο έχει τρεις τιμές μια για κάθε συνιστώσα χρώματος. Την κάθε συνιστώσα την υπολογίσαμε ως τον μέσο όρο των αντίστοιχων επιμέρους συνιστωσών, του κάθε σημείου που μας υπέδειξε ο χρήστης. Στην συνέχεια προσπελάσαμε κάθε `pixel` της εικόνας, ώστε να δούμε σε πιο θα αλλάξουμε χρώμα σε κίτρινο και πιο θα αφήσουμε ίδιο, εφαρμόζοντας τον τύπο της σφαίρας που βρίσκεται στις διαλέξεις. Η εξίσωση της σφαίρας λειτουργεί καλύτερα, λόγω του σχήματος και της καλύτερης ομοιομορφίας που έχει σε σχέση με τον κύβο. Σαν ακτίνα τις σφαίρας ορίσαμε τον αριθμό 50 δοκιμαστικά, έτσι ώστε να έχουμε ικανοποιητικό αποτέλεσμα. Εκτελούμε την συνάρτηση για κάθε μια από τις δύο εικόνες πληκτρολογώντας `erwthma4('wall');` και `erwthma4('Saint-Nikolaos');`. Για

παράδειγμα τις λείει και στην εκφώνηση θα προσπαθήσουμε στην εικόνα wall να αλλάξουμε το χρώμα του τοίχου σε κίτρινο και στην εικόνα Saint-Nikolaos το χρώμα των φυτών σε κίτρινο. Αποτελέσματα πριν και μετά την εκτέλεσης της συνάρτησης για κάθε μια από τις δύο εικόνες:

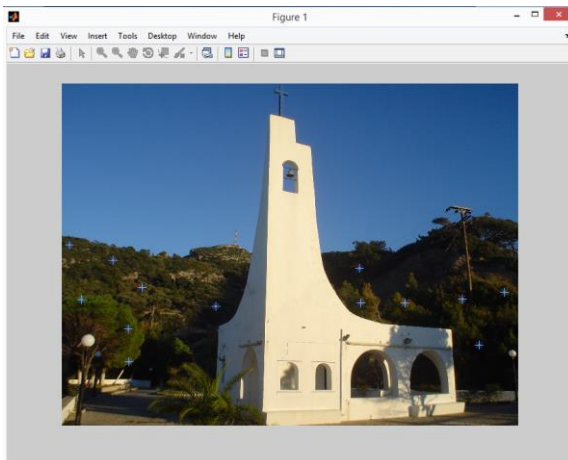
Πριν



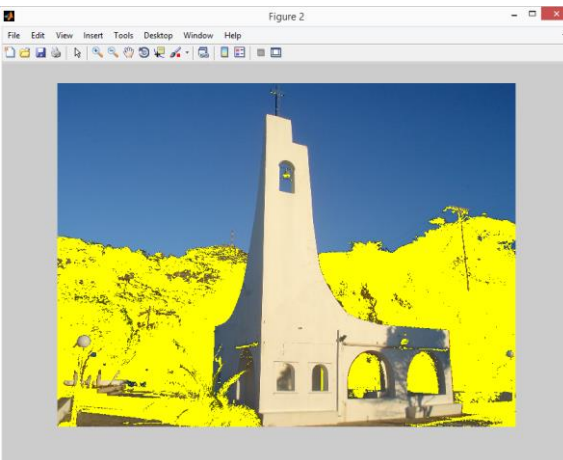
Μετά



Πριν



Μετά



5) Για την ανίχνευση των ακμών στις εικόνες church2 και San-Francisco θα χρησιμοποιήσουμε την μέθοδο του διαφορικού τελεστή Sobel. Η κλίση μιας

δισδιάστατης συνάρτησης $f(x,y)$ σε κάθε σημείο (x,y) είναι ένα διάνυσμα δύο στοιχείων.

$$\nabla f(x, y) = \begin{bmatrix} G_x(x, y) \\ G_y(x, y) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(x, y)}{\partial x} \\ \frac{\partial f(x, y)}{\partial y} \end{bmatrix}$$

Έτσι η ανίχνευση σημείων ασυνέχειας στη φωτεινότητα της εικόνας πραγματοποιείται με τη βοήθεια του υπολογισμού της κλίσης (gradient) της εικόνας. Στην περίπτωση που η συνάρτηση $f(x,y)$ δεν είναι συνεχής, όπως συμβαίνει με τις ψηφιακές εικόνες, η κλίση της εικόνας υπολογίζεται με τη βοήθεια των τελεστών Sobel:

$$s_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \cdot \quad s_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

οι οποίοι εφαρμόζονται επαναληπτικά σε κάθε pixel της εικόνας για τον υπολογισμό της μεταβολής της φωτεινότητας στην κάθετη και οριζόντια κατεύθυνση αντίστοιχα. Σχετικά τώρα με την ολική κατωφλίωση πρέπει να δώσουμε έμφαση στα παρακάτω:

- i)εφαρμόζεται δύσκολα σε εικόνες με περισσότερα του ενός αντικείμενα,
- ii)για να έχει αποτέλεσμα πρέπει να γνωρίζουμε εκ των προτέρων αν το αντικείμενο είναι πιο φωτεινό από το φόντο ή το αντίστροφο,
- iii)το κατώφλι T που θα επιλέξουμε θα παραμένει ίδιο για όλη την εικόνα, σε αντίθεση με την τοπική κατωφλίωση.

Ένας καλός τρόπος να επιλέξουμε κατώφλι είναι από το ιστόγραμμα της εικόνας. Εφόσον επιλέξουμε το κατώφλι T , διακρίνουμε δύο περιπτώσεις:

- i)αν το αντικείμενο είναι φωτεινότερο από το φόντο τότε αν $f(x,y) > T$, τότε το pixel με συντεταγμένες (x,y) ανήκει στο αντικείμενο, αλλιώς ανήκει στο φόντο, και
- ii)αν το αντικείμενο είναι σκοτεινότερο από το φόντο τότε αν $f(x,y) < T$, τότε το pixel με συντεταγμένες (x,y) ανήκει στο αντικείμενο, αλλιώς ανήκει στο φόντο.

Για την υλοποίηση των παραπάνω δημιουργήθηκε μια συνάρτηση με όνομα `erwthma5_1()`, η οποία εκτελεί τον εξής κώδικα:

```
function [image_new] = erwthma5_1(name)
image=imread(strcat(name, '.jpg'));
image=rgb2gray(image);
```

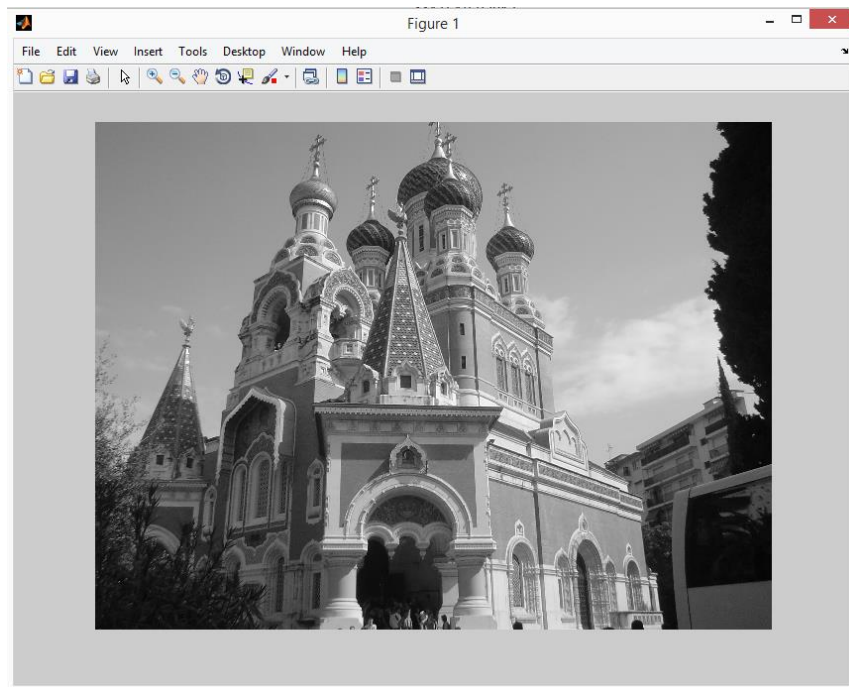
```

image=double(image);
figure(1);
imshow(uint8(image));
sobelx=[-1 -2 -1;0 0 0;1 2 1];
sobely=sobelx';
imagex = conv2(image,sobelx,'same');
imagey = conv2(image,sobely,'same');
image_new=sqrt(power(imagex,2)+power(imagey,2));
figure(2);
imshow(uint8(image_new));
end

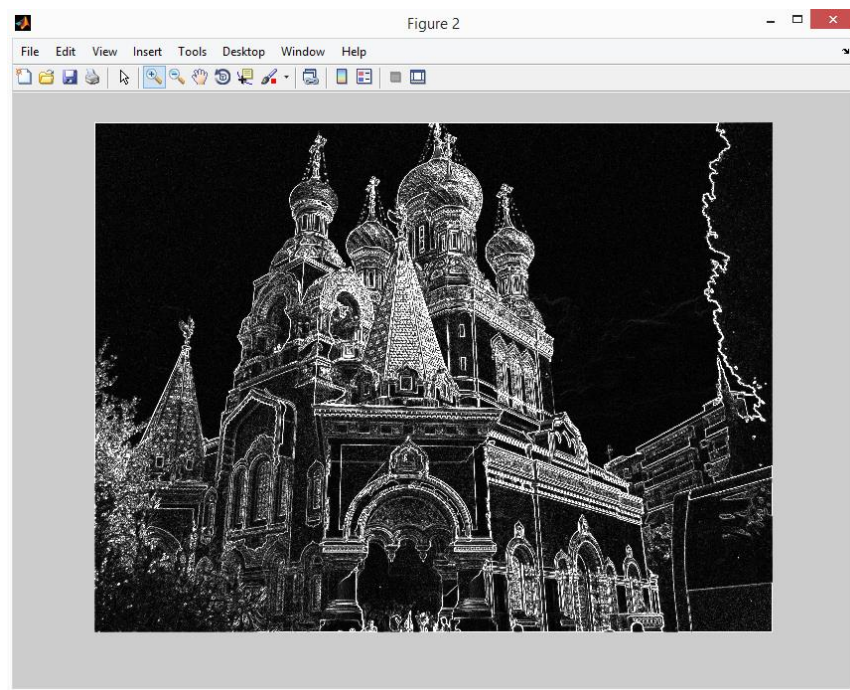
```

Η συνάρτηση παίρνει σαν είσοδο το όνομα της εικόνας σαν αλφαριθμητικό χωρίς την κατάληξη .jpg, κάνει μετατροπή της εικόνας σε grayscale, μετατροπή αυτού που θα προκύψει σε double και εφαρμόζει τους τελεστές Sobel στην εικόνα, χρησιμοποιώντας την συνάρτηση conv2. Η καινούργια εικόνα image_new θα προκύψει από τις συνιστώσες imagex και imagey, εφαρμόζοντας την σχέση $\sqrt{(\text{imagex})^2 + (\text{imagey})^2}$, όπου κάθε μια συνιστώσα imagex και imagey, έχει προκύψει από την εφαρμογή των τελεστών sobelx και sobely, αντίστοιχα. Τέλος έχουμε ένα figure για την αρχική εικόνα, ένα για την τελική και επιστροφή του πίνακα image_new. Εκτελούμε συνάρτηση ως εξής: **erwthma5_1('church2');** και **erwthma5_1('San-Francisco');**. Τα αποτελέσματα πριν και μετά την εκτέλεση του κώδικα για την κάθε μια εικόνα φαίνονται παρακάτω:

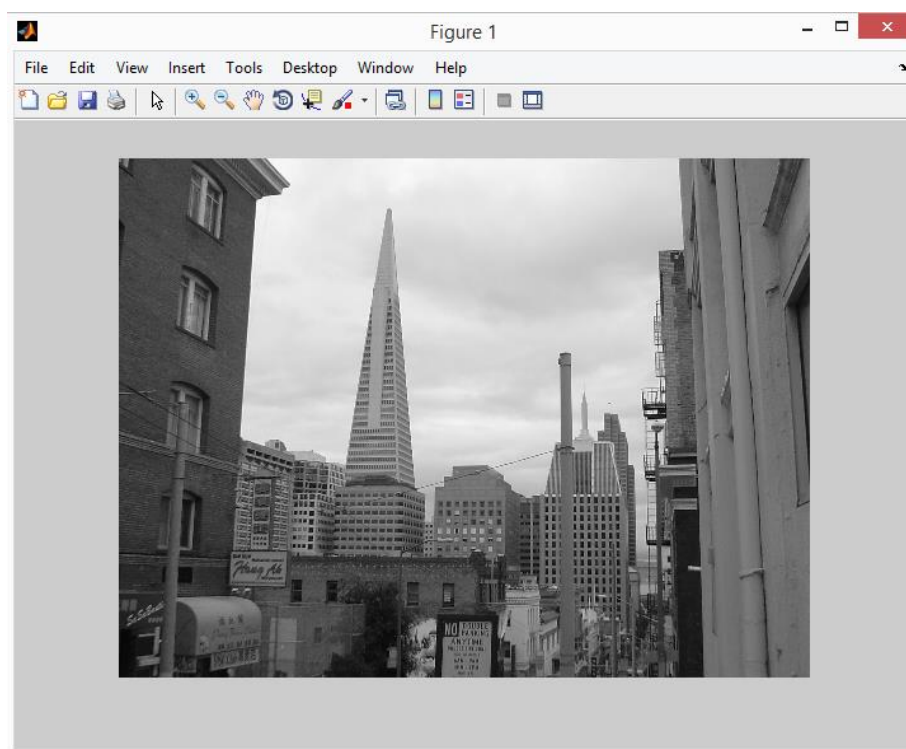
Πριν



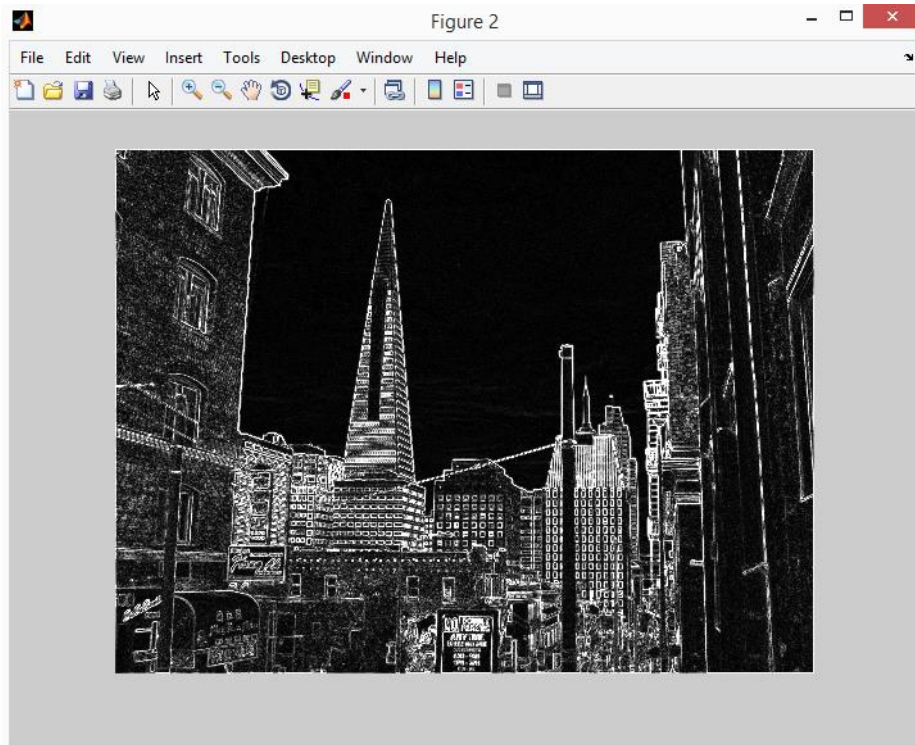
Μετά



Πριν



Μετά



Στην συνέχεια για την ολική κατωφλίωση των παραπάνω αποτελεσμάτων που προέκυψαν από την συνάρτηση `erwthma5_1()`, δημιουργήθηκε η συνάρτηση `erwthma5_2()`, η οποία καλεί την συνάρτηση `erwthma5_1()`, μετατρέπει τον πίνακα που επιστρέφει σε `uint8` λόγω του ότι το εύρος τιμών του δεν είναι από 0 έως 255 και εφαρμόζει την ολική κατωφλίωση κάνοντας προσπέλαση κάθε `pixel` της εικόνας. Όποιο `pixel` είναι πάνω από το `threshold` που έχουμε θέσει, άρα είναι πιο φωτεινό, στην περίπτωση μας κυρίως οι ακμές, του θέτουμε τιμή 255, δηλαδή λευκό, ενώ τα υπόλοιπα `pixels` που είναι κάτω από αυτό το `threshold` τους θέτουμε τιμή 0, δηλαδή μαύρο. Καλούμε την συνάρτηση `erwthma5_2()` μία φορά για κάθε εικόνα περνώντας σαν παράμετρο το όνομα της εικόνας, όπως φαίνεται παρακάτω

`erwthma5_2('church2');` και `erwthma5_2('San-Francisco');` .

Ο κώδικας την συνάρτησης φαίνεται παρακάτω:

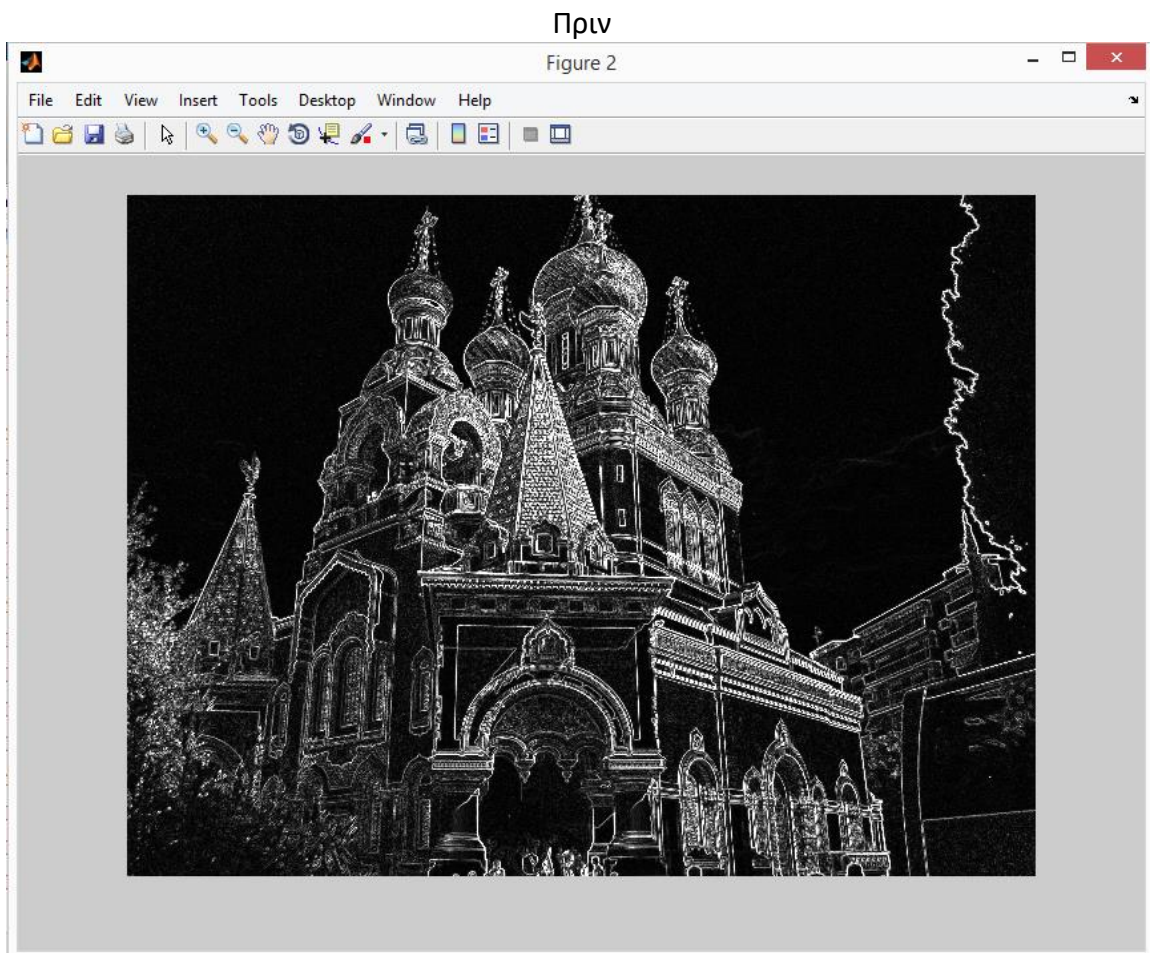
```
function [] = erwthma5_2(name)
image=erwthma5_1(name);
image=uint8(image);
for i=1:size(image,1)
    for j=1:size(image,2)
        if(image(i,j)<100)
            image(i,j)=0;
        else
            image(i,j)=255;
        end
    end
end
```



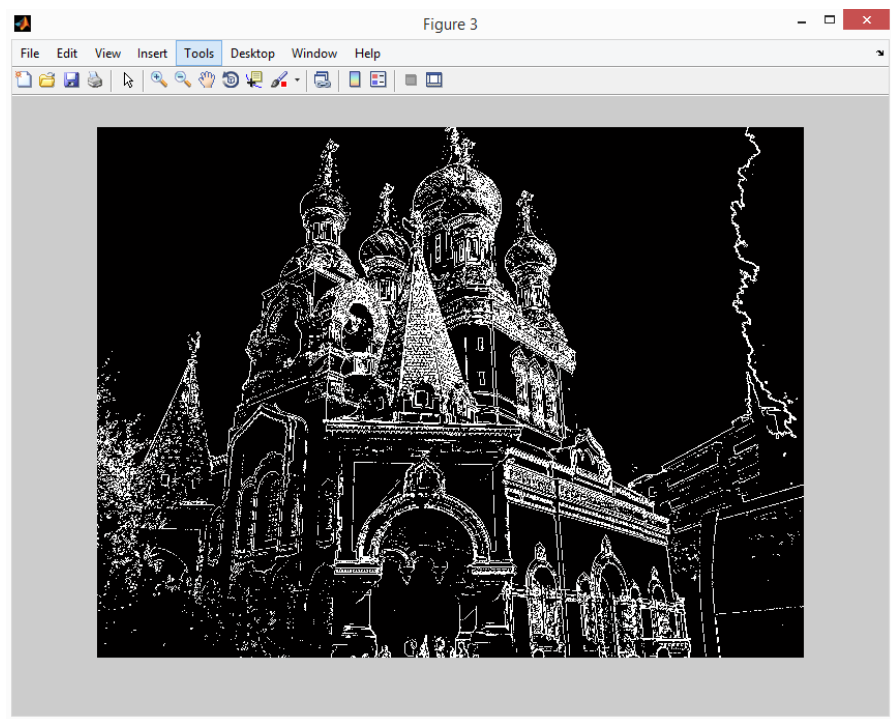
```
end  
end  
figure(3);  
imshow(uint8(image));  
end
```

Σαν κατώφλι χρησιμοποιήθηκε η τιμή έντασης 100, μετά από δοκιμές.

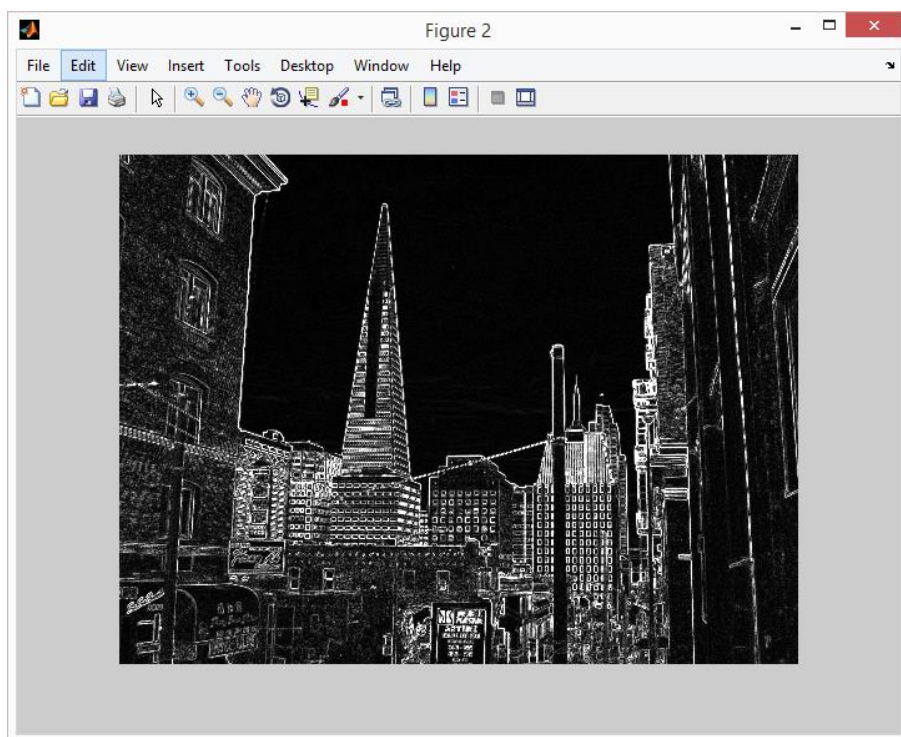
Αποτελέσματα ολικής κατωφλίωσης πριν και μετά την κλήση της συνάρτησης, για κάθε εικόνα:



Μετά



Πριν



Μετά

