Ψηφιακή Επεξεργασία Εικόνας

Εργασία:
Αλγόριθμος εξαγωγής ακμών του Canny (**Canny Edge Detection**) – Υλοποίηση.

Φοιτητές:

Μπιτζίδης Χαράλαμπος 57424

Α) Παρουσίαση του Αλγορίθμου :

Ο αλγόριθμος εξαγωγής ακμών του Canny είναι μια βέλτιστη τεχνική ανίχνευσης και δημιουργίας ακμών. Στόχο έχει να ανιχνεύσει ακμές σε μία εικόνα που θα του δώσουμε σαν είσοδο. Για να εκτελέσουμε τον αλγόριθμο του Canny , θα περάσουμε από τα εξής βήματα :

1. **Φιλτράρισμα της εικόνας** (blurrring).
2. **Προσδιορισμός της κλίσης της εικόνας.**
3. **Non-maximum Suppression.**
4. **Hysteresis Thresholding.**

Β) Περιγραφή του Αλγορίθμου :

Αρχικά, πριν να προχωρήσουμε στην εξήγηση των κεντρικών βημάτων του αλγορίθμου , χρειάζεται να διαβάσουμε την εικόνα και από το χρωματικό μοντέλο RGB να την μετατρέψουμε σε grayscale.

Επίσης η μετατροπή σε double τιμές χρειάζεται για την σωστή επεξεργασία των εικόνων.

Ο τύπος uint8 χρησιμεύει όταν θέλουμε να δείξουμε σε figures τις εικόνες που επιθυμούμε. Οπότε:

Κώδικας:
```
image = imread('Images/(...όνομα αρχείου εικόνας...).jpg');
image_gs=rgb2gray(image);
image_gs=double(image_gs);
```

α) **Φιλτράρισμα της εικόνας** :
Εφαρμοζουμε στην εικόνα ένα φίλτρο Gaussian , με το οποίο καταφέρνουμε να θολώσουμε την εικόνα. Η συγκεκριμένη διαδικασία γίνεται με τις παρακάτω εντολές στο Matlab :

Κώδικας:
```
kernel = fspecial('gaussian', [21 21], 1.6);
I_blurred = imfilter(image_gs, kernel);
```

β) **Προσδιορισμός της κλίσης της εικόνας :**

Για να προσδιορίσουμε την κλίση (theta) , αλλά και το πλάτος (magnitude) του κάθε εικονοστοιχείου (pixel) , θα χρειαστεί να εξάγουμε δύο εικόνες , οι οποίες θα προκύψουν από την εφαρμογή των μασκών Sobel . Οπότε δημιουργούμε τις μάσκες Sobel (x_mask και y_mask) , τις οποίες θα τις συνελίξουμε με την εικόνα που περάσαμε από το φίλτρο Gaussian.

Κώδικας:
```
x_mask = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
y_mask = [1, 2, 1; 0, 0, 0; -1, -2, -1];
```

Οι μάσκες x_mask και y_mask είναι προσανατολισμένες προς την οριζόντια και κατακόρυφη αντίστοιχα διεύθυνση , οπότε και ψάχνουν για κατακόρυφες και οριζόντιες αντίστοιχα ακμές.

Οπότε με την συνέλιξη προκύπτουν οι εικόνες x_dir και y_dir που έχουν μόνο τις κατακόρυφες και οριζόντιες αντίστοιχα ακμές :

Κώδικας:
```
x_dir=conv2(I_blurred, x_mask,'same');
y_dir=conv2(I_blurred, y_mask,'same');
```

Για τον προσδιορισμό των magnitude και theta για κάθε pixel της εικόνας , παίρνουμε τους παρακάτω γνωστούς τύπους:

Κώδικας:
```
magn=sqrt(x_dir.^2 + y_dir.^2);
theta=atan2(y_dir , x_dir);
```

Επίσης μετατρέπουμε την γωνία theta σε μοίρες :

Κώδικας:
```
theta=theta*180/pi;
```

γ) **Non-maximum Suppression :**

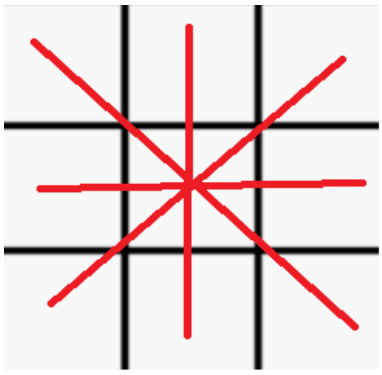
Αρχικά μετατρέπουμε τις αρνητικές γωνίες theta στις αντίστοιχες θετικές :

Κώδικας:

```
for i=1:lines
    for j=1:cols
        if (theta(i,j)<0)
            theta(i,j)=360+theta(i,j);
        end;
    end;
end;
```

Πρίν προχωρήσουμε στην διαδικασία της καταστολής των μη – μέγιστων τιμών , θα χρειαστεί να προσαρμόσουμε την γωνία theta καθενός pixel από ένα εύρος 0 έως 360 μοίρες στην κοντινότερη τιμή εξαρτάται από ποιο σημείο στο φάσμα των μοιρών βρίσκεται . Σκοπός μας είναι να λεπτύνουμε τις ακμές .Αυτό θα γίνει εάν σε κάθε pixel ελέγχουμε εάν το μέτρο του magnitude είναι το ίδιο με τα γειτονικά του pixel .

Σε ένα grid 3x3 , δηλαδή σε ένα grid όπου λαμβάνουμε υπόψιν ένα κεντρικό Pixel και τα 8 γειτονικά του, είνα πιθανό να υπάρχουν 4 ακμές , όπως φαίνεται στην παρακάτω εικόνα:



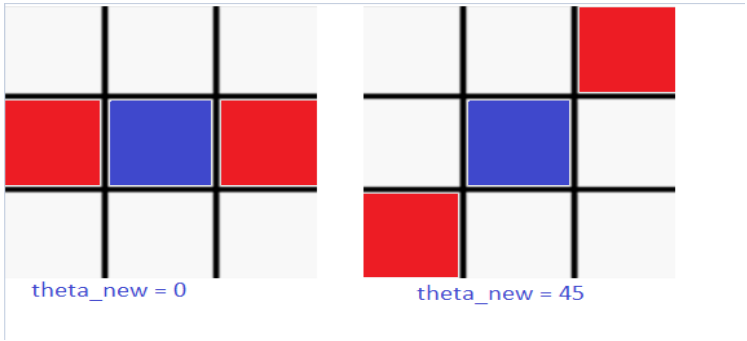
Εικόνα Β1: Απεικόνιση των 4 πιθανών κλίσεων που μπορεί να υπάρχει μία ακμή σε ένα grid 3χ3 .

Γι αυτό τον λόγο σε ένα εύρος των 0 έως 360 μοιρών δημιουργούμε 4 ομάδες (0 , 45 ,90 ,135) .

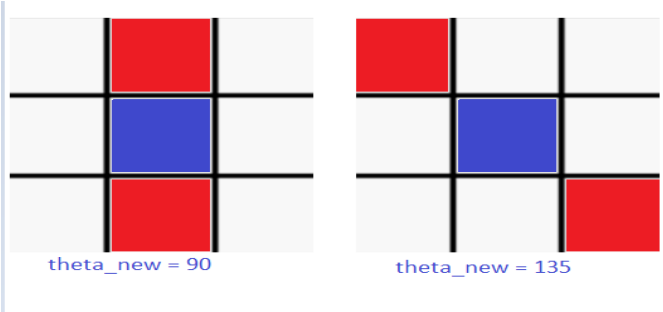
Κώδικας:

```
for i=1:lines
   for j=1:cols
        if ((theta(i,j) >= 0 ) && (theta(i,j) < 22.5) || (theta(i,j) >= 157.5) && (theta(i,j) < 202.5) ||
(theta(i,j) >= 337.5) && (theta(i,j) <= 360))
          theta_new(i,j) = 0;
       elseif ((theta(i,j) >= 22.5) && (theta(i,j) < 67.5) || (theta(i,j) >= 202.5) && (theta(i,j) < 247.5))
          theta_new(i,j) = 45;
       elseif ((theta(i,j) >= 67.5 && theta(i,j) < 112.5) || (theta(i,j) >= 247.5 && theta(i,j) < 292.5))
          theta_new(i,j) = 90;
       elseif ((theta(i,j) >= 112.5 && theta(i,j) < 157.5) || (theta(i,j) >= 292.5 && theta(i,j) < 337.5))
          theta_new(i,j) = 135;
       end;
   end;
end;
```

Στη συνέχεια με την διαδικασία της Non-maximum suppression ελέγχουμε σε αυτό το grid των 9 κελιών εάν υπάρχουν ακμές .Σύμφωνα με τις 4 παραπάνω πιθανές περιπτώσεις , ελέγχονται τα αντίστοιχα γειτονικά κελιά (χρωματίζονται κόκκινα) μαζί με το τωρινό κεντρικό κελί (χρωματίζεται μπλε).



Εικόνα Β2: Απεικόνιση συγκεκριμένης εξεταστέας ακμής , για τιμές της theta_new : 0 και 45 μοίρες.



Εικόνα Β3: Απεικόνιση συγκεκριμένης εξεταστέας ακμής , για τιμές της theta_new : 90 και 135 μοίρες.

Άρα με την διαδικασία αυτή ελέγχουμε εάν σε κάθε pixel ανάλογα με την κλίση που έχει , αν υπάρχει κάποια ακμή , ελέγχοντας και τα γειτονικά Pixel αν έχουν την ίδια τιμή magnitude στην διεύθυνση που ορίζει η γωνία theta_new. Ο αλγόριθμος που το κάνει αυτό :

Κώδικας:

```
for i=2:lines-1
   for j=2:cols-1
      if (theta_new(i,j)==0)
         BW(i,j) = (magn(i,j) == max([magn(i,j), magn(i,j+1), magn(i,j-1)]));
      elseif (theta_new(i,j)==45)
         BW(i,j) = (magn(i,j) == max([magn(i,j), magn(i+1,j-1), magn(i-1,j+1)]));
      elseif (theta_new(i,j)==90)
         BW(i,j) = (magn(i,j) == max([magn(i,j), magn(i+1,j), magn(i-1,j)]));
      elseif (theta_new(i,j)==135)
         BW(i,j) = (magn(i,j) == max([magn(i,j), magn(i+1,j+1), magn(i-1,j-1)]));
      end;
   end;
end;
```

Οπότε σε δημιουργούμε τον πίνακα BW, ο οποίος παίρνει την λογική τιμή 1 εάν το κεντρικό pixel είναι pixel ακμής και είναι όταν έχει μεγαλυτερη τιμή από τα δύο γειτονικά που "βλέπει" ανάλογα με την τιμή της theta_new.

Κι άρα ο BW ανανεώνει τον πίνακα magnitude :

Κώδικας:

```
BW = BW.*magn;
```

δ) **Hysteresis Thresholding :**

Η διαδικασία της κατωφλίωσης με υστέρηση , στοχεύει στην παραπέρα μείωση του αριθμού των pixels των ακμών.

Χρησιμοποιούνται δύο κατώφλια : T_Low και T_High , σύμφωνα με τα οποία :

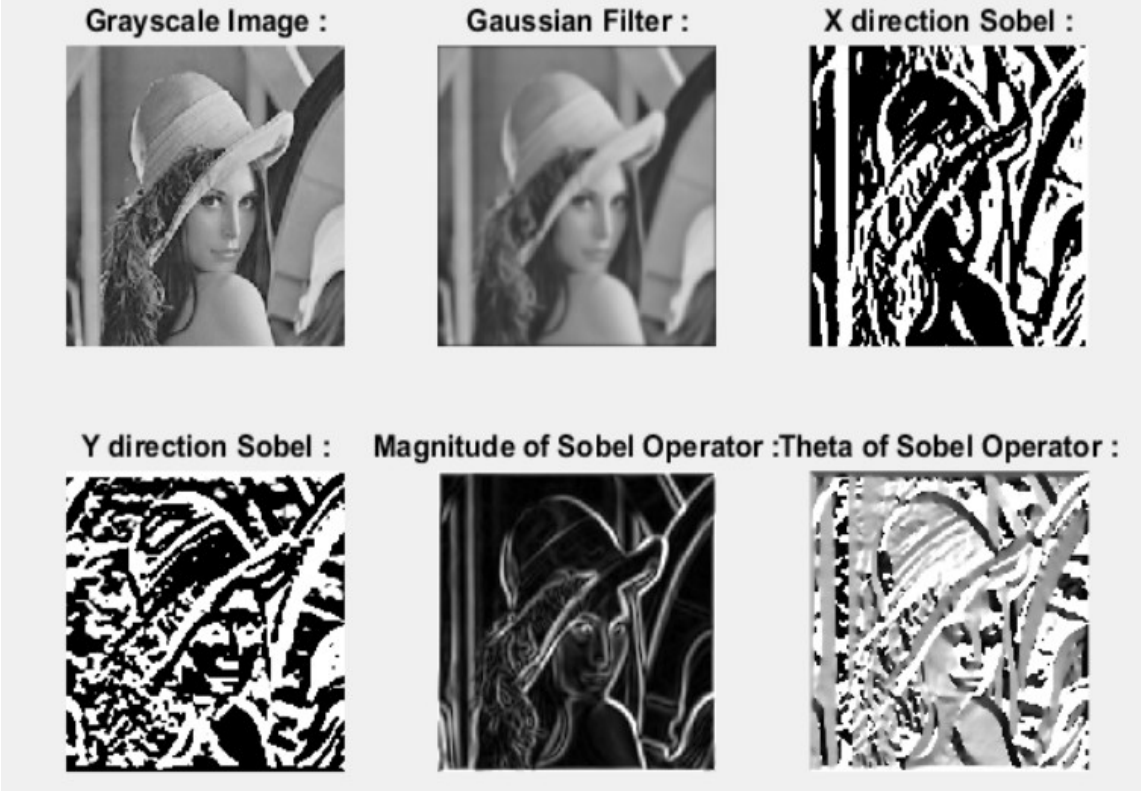
1. Αν BW < T_Low τότε I_ht = 0
2. Αν BW > T_High τότε I_ht = 1
3. Αν T_Low< BW < T_High τότε I_ht = 0 , εκτός αν BW > T_High για κάποιο γειτονικό pixel.

Οι τιμές αρχικοποιούνται :

T_Low = 0.075;
T_High = 0.175;

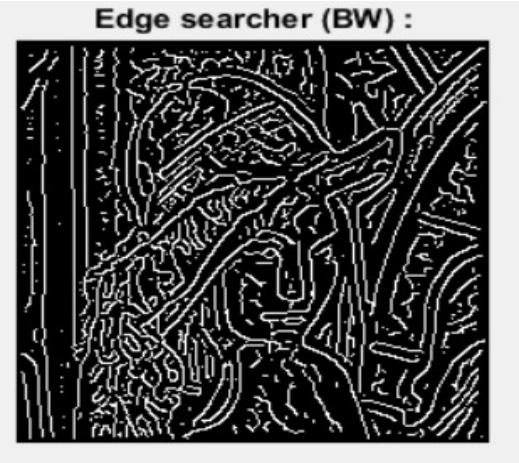
Γ) Αποτελέσματα :

Όσον αφορά την εικόνα Lenna.jpg :



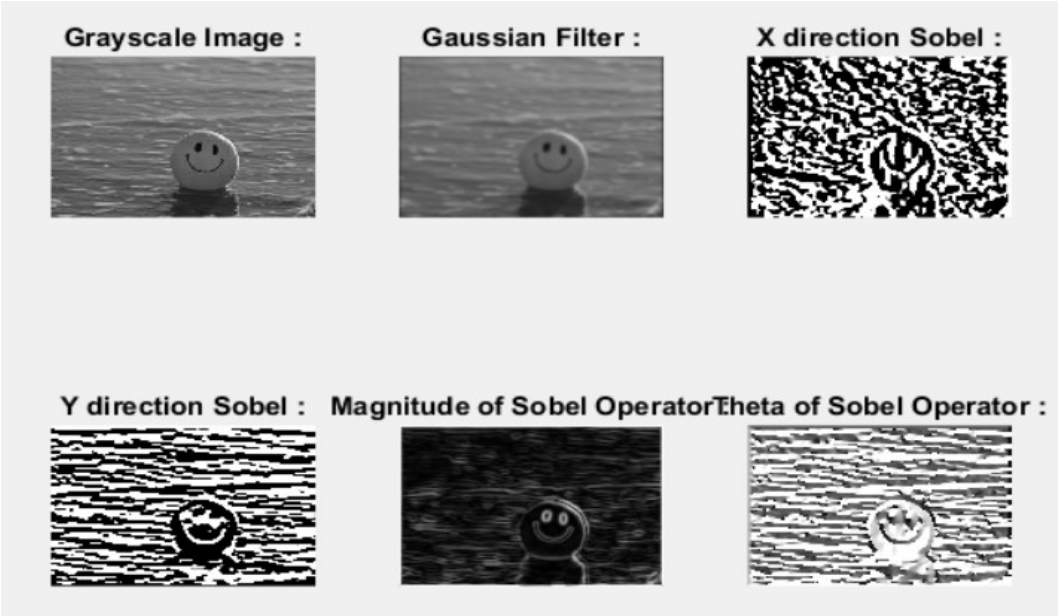
Εικόνα Γ1: Απεικόνιση ενδιάμεσων σταδίων.

Εικόνα Γ2: Αποτέλεσμα my_canny αλγορίθμου για την για την εικόνα Lenna.jpg .



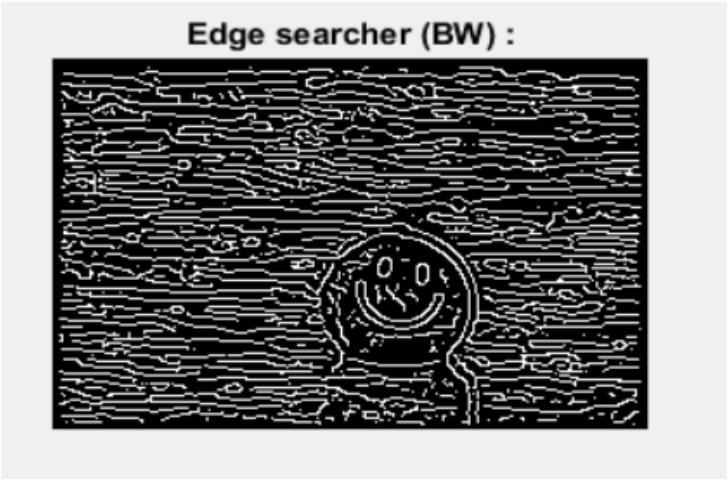
Εικόνα Γ3: Ο πίνακας BW.

Όσον αφορά την εικόνα emogi.jpg :



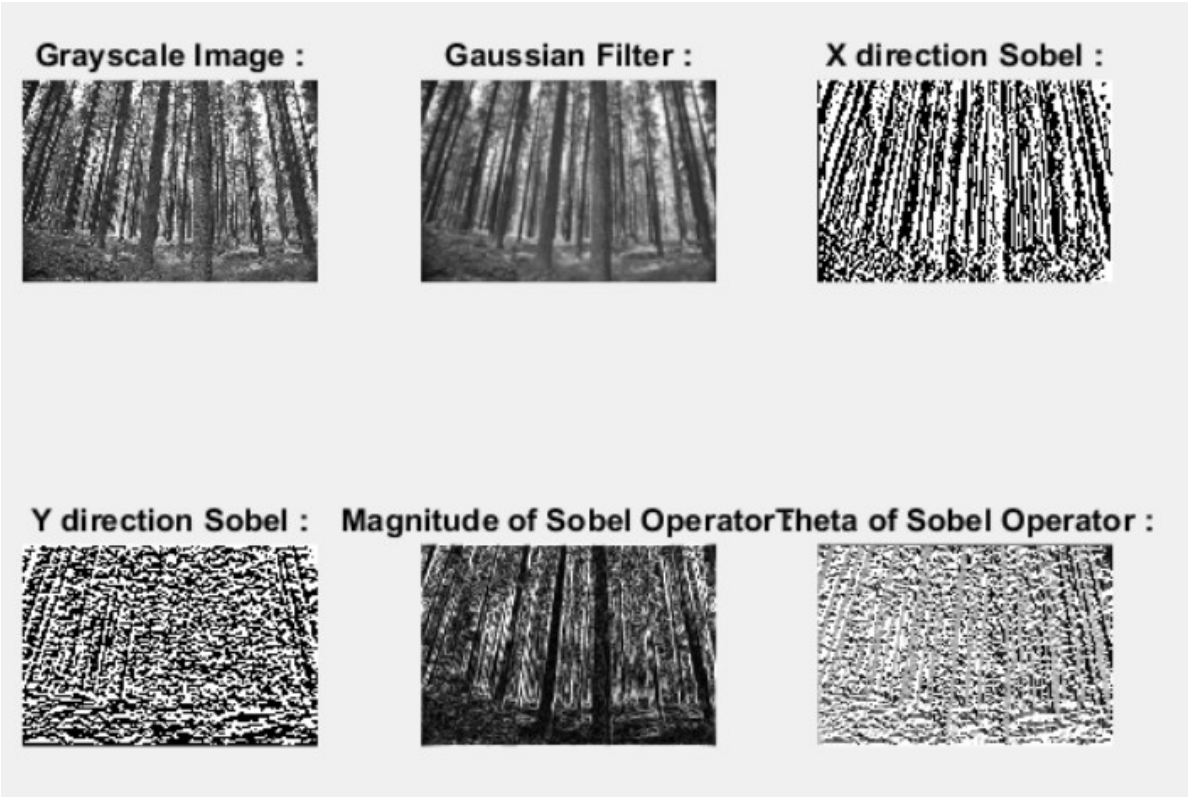
Εικόνα Γ4: Απεικόνιση ενδιάμεσων σταδίων.

Εικόνα Γ5: Αποτέλεσμα my_canny αλγορίθμου για την εικόνα emogi.jpg .



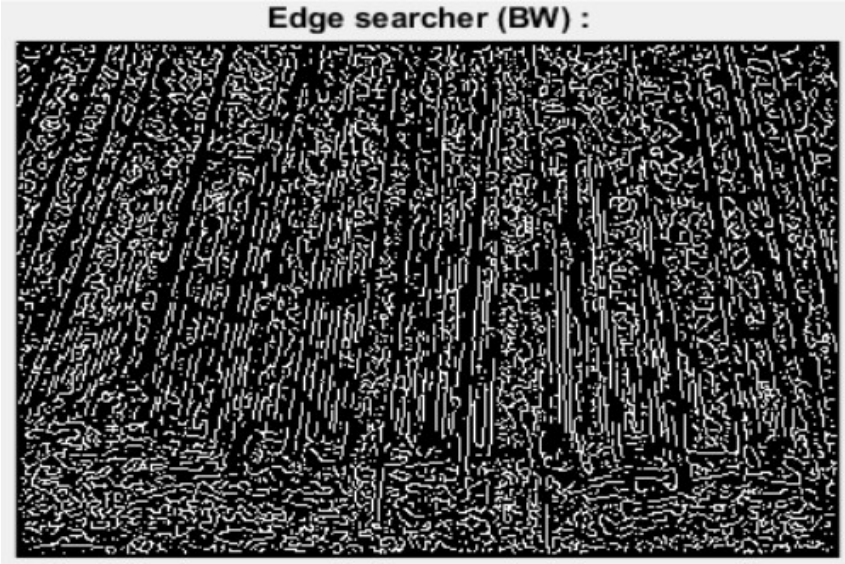
Εικόνα Γ6: Ο πίνακας BW.

Όσον αφορά την εικόνα forest.jpg :



Εικόνα Γ7: Απεικόνιση ενδιάμεσων σταδίων.

Εικόνα Γ8: Αποτέλεσμα my_canny αλγορίθμου για την εικόνα forest.jpg .



Εικόνα Γ9: Ο πίνακας BW.

Δ) Σύγκριση με παρόμοιες τεχνικές :

Για την αξιολόγηση του αλγορίθμου που σχεδιάσαμε , θα τον συγκρίνουμε με τον αντίστοιχο αλγόριθμο Canny Edge Detection που υπάρχει έτοιμος στις βιβλιοθήκες του Matlab. Για να εφαρμόσουμε τον έτοιμο αλγόριθμο του Matlab δημιουργούμε μια ρουτίνα (matlab_canny) που ως είσοδο θα παίρνει την εικόνα image (στην οποία θέλουμε να ανιχνεύσουμε ακμές) και ως έξοδο θα βγάζει μια εικόνα με τις ακμές που μπόρεσε να ανιχνεύσει.

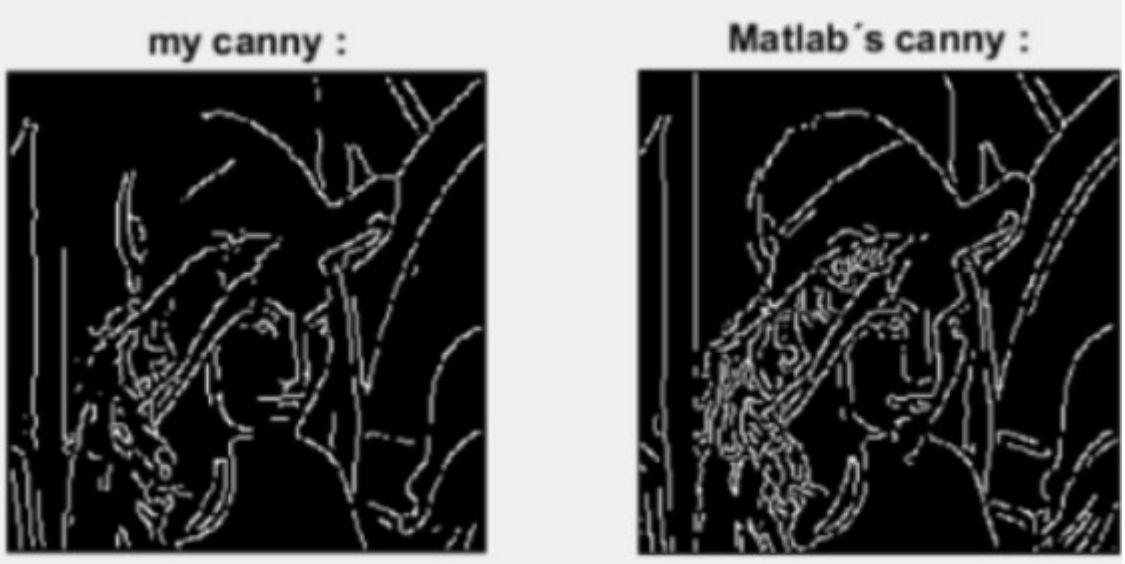
Κώδικας:
I_mine= matlab_cannny(image);

Μέσα στην ρουτίνα αυτήν , θα καλούμε την συνάρτηση edge με ορίσματα την εικόνα σε grayscale και τον τύπο του edge detection που επιθυμούμε (στην περίπτωσή μας Canny).

Κώδικας:
image_gs_matlb = rgb2gray(image);
I_mtlb = edge (image_gs_mtlb,'Canny');

Μερικά από τα αποτελέσματα της τεχνικής αυτής παρουσιάζονται παρακάτω μαζί με τα αντίστοιχα αποτελέσματα της τεχνικής my_canny :

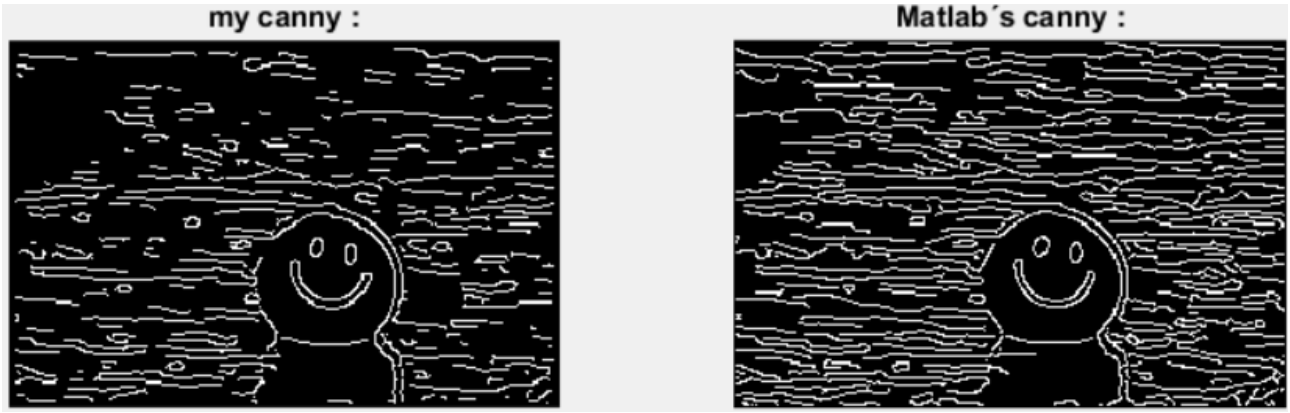
Όσον αφορά την εικόνα Lenna.jpg:



Εικόνα Δ1: Τα αποτελέσματα των αλγορίθμων my_canny και matlab_canny , για την εικόνα Lenna.jpg .

Εικόνα Δ2: Απεικόνιση των διαφορών των εικόνων I_mine και I_mtlb στο ίδιο figure .

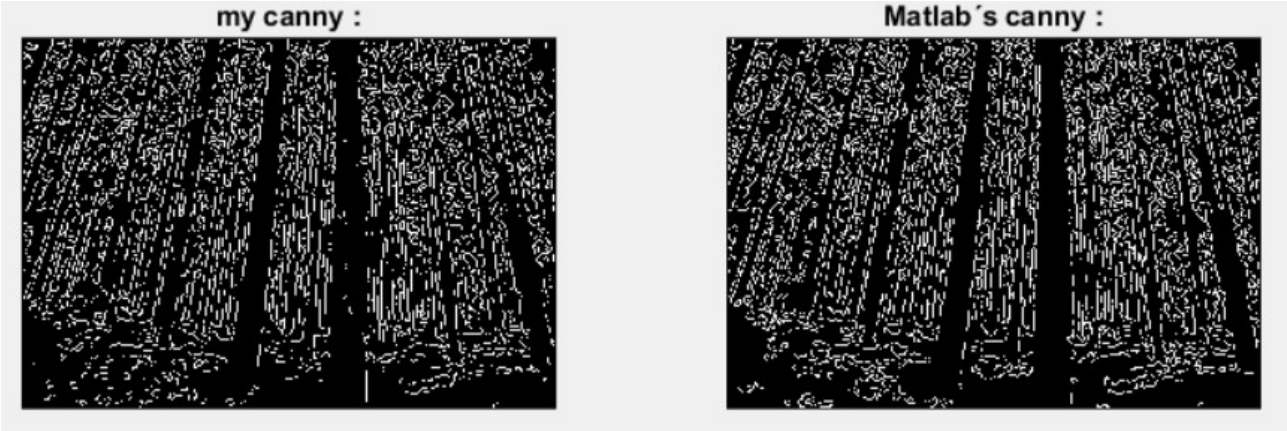
Όσον αφορά την εικόνα emogi.jpg :



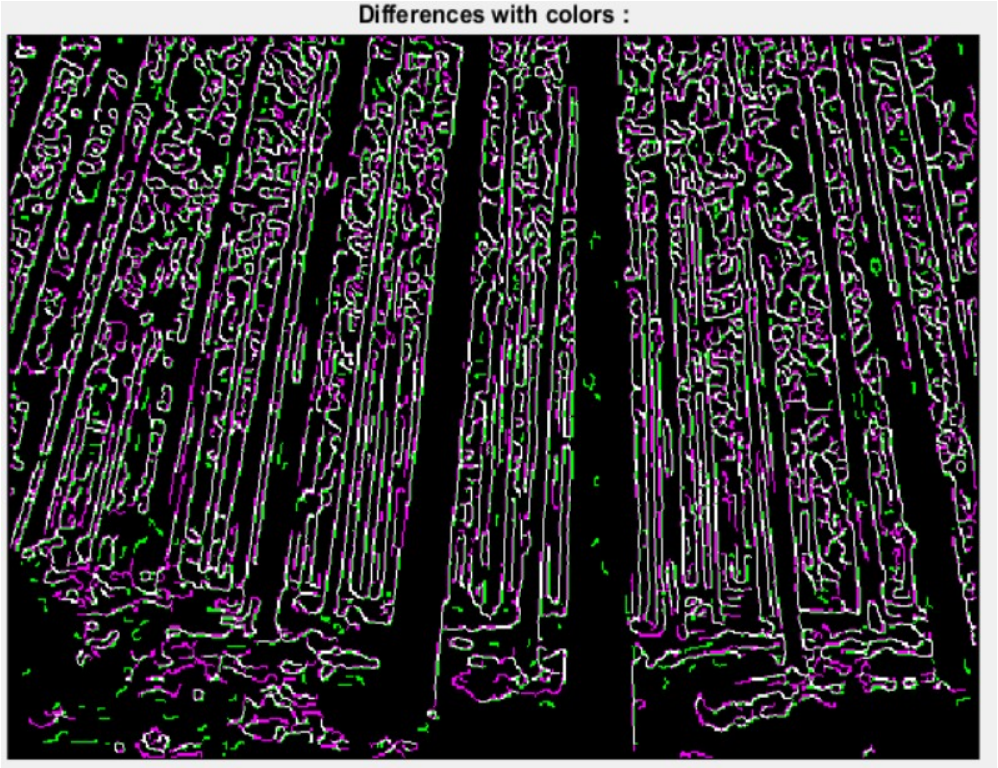
Εικόνα Δ3: Τα αποτελέσματα των αλγορίθμων my_canny και matlab_canny , για την εικόνα emogi.jpg .

Εικόνα Δ4: Απεικόνιση των διαφορών των εικόνων I_mine και I_mtlb στο ίδιο figure .

Όσον αφορά την εικόνα forest.jpg :



Εικόνα Δ5: Τα αποτελέσματα των αλγορίθμων my_canny και matlab_canny , για την εικόνα forest.jpg .



Εικόνα Δ6: Απεικόνιση των διαφορών των εικόνων I_mine και I_mtlb στο ίδιο figure .

Από τις παραπάνω εικόνες φαίνονται οι διαφορές των δύο εικόνων στο περίπου. Οι εικόνες με τα μωβ και πράσινα Pixel , απεικονίζουν τις διαφορές στην επιτυχία των αλγορίθμων. Τα μωβ Pixel είναι οι ακμές που εντόπισε ο matlab_canny και όχι ο my_canny. Με τα πράσινα pixel υποδεικνύεται το αντίθετο. Παρατηρείται ότι ο my_canny “'έχασε” λιγότερα Pixel από αυτά που “'έχασε” ο matlab_canny.

Για μία σύγκριση με μεγαλύτερη ακρίβεια , δημιουργήθηκε μία τρίτη εικόνα , η οποία έχει λευκά pixels δηλαδή λογική τιμή 1 ή μέγιστη τιμή pixel δηλαδή 255 στα σημεία που διαφέρουν οι I_mine και I_mtlb, η εικόνα I_diff . Η εικόνα αυτή ουσιαστικά μας δείχνει το ποσοστό ομοιότητας μεταξύ των 2 εικόνων , άρα και το πόσο κοντά είναι η τεχνική my_canny στην έτοιμη ρουτίνα του Matlab ,αν θεωρήσουμε την δεύτερη ως απόλυτα σωστή .
Αυτή η τεχνική μετράει τον αριθμό των λευκών pixel στην εικόνα I_diff και σε σχέση με τα συνολικά Pixel αυτής , βγάζει το ποσοστό αποτυχίας της τεχνικής my_canny δηλαδ'η πόσο απέχει από την τεχνική matlab_canny . Ομοίως προκύπτει και το ποσοστό επιτυχίας , δηλαδή το ποσοστό-βαθμός ομοιότητας των δύο τεχνικών.

Ε) Συμπεράσματα :

Στην προηγούμενη παράγραφο δείξαμε τα αποτελέσματα από την σύγκριση των 2 εικόνων από την εφαρμογή της συνάρτησης imshowpair().

Γενικά ο αλγόριθμος my_canny φαίνεται να τα πηγαίνει αρκετά καλά αφού μπορεί να εντοπίσει και ακμές που είναι δύσκολο να εντοπιστούν. Κάποιες φορές αποκλίνει από την τεχνική matlab_canny , αλλά αυτό μπορεί να διορθωθεί από την καλύτερη επιλογή των low και high thresholds (T_Low και T_High).\

Επίσης με την δεύτερη στρατηγική σύγκρισης των αποτελεσμάτων , με την εικόνα I_diff , ακολουθουν τα αποτελέσματα για τις παραπάνω περιπτώσεις :

```
Comparing the two final Images ...

Printing the differences :
Differences printed with imshowpair()!

Find Differences with a 3rd image I_diff :
Number of total pixels: 50625
Number of white pixels: 4802
Number of black pixels: 45823
Percentage of failure : 9.4854
Percentage of matching : 90.5146
```

Εικόνα Ε1: Απεικόνιση της σύγκρισης των εικόνων I_mine και I_mtlb της Lenna.jpg.

```
Comparing the two final Images ...

Printing the differences :
Differences printed with imshowpair()!

Find Differences with a 3rd image I_diff :
Number of total pixels: 50325
Number of white pixels: 8345
Number of black pixels: 41980
Percentage of failure : 16.5822
Percentage of matching : 83.4178
```

Εικόνα Ε2: Απεικόνιση της σύγκρισης των εικόνων I_mine και I_mtlb της emogi.jpg.

```
Comparing the two final Images ...

Printing the differences :
Differences printed with imshowpair()!

Find Differences with a 3rd image I_diff :
Number of total pixels: 173500
Number of white pixels: 37403
Number of black pixels: 136097
Percentage of failure : 21.5579
Percentage of matching : 78.4421
```

Εικόνα Ε3: Απεικόνιση της σύγκρισης των εικόνων I_mine και I_mtlb της forest.jpg.

Σχολιασμός :

Τα ποσοστά φαίνονται αρκετά υψηλά , αν και στις εικόνες που προέκυψαν από το imshowpair() , φαίνεται να υπάρχουν αρκετές διαφορές .Αυτό ίσως εξηγείται από το γεγονός ότι στο ποσοστό επιτυχίας συμπεριλαμβάνονται και τα pixel που είναι μαύρα και στις δύο εικόνες , στα οποία δεν αναμέναμε να βρούμε ακμή. Οπότε όσο περισσότερες οι ακμές και λιγότερα τα σημεία που δεν υπάρχουν ακμές , τόσο πιο αντιπροσωπευτικό θα είναι το ποσοστό ομοιότητας των 2 εικόνων. Μια λύση σε αυτό ίσως ήταν να μην συμπεριλάβουμε τα μαυρα κοινά pixel στον ποσοστό ομοιότητας έτι ώστε να έχουμε ένα πιο κοντά στην πραγματικότητα νούμερο.

ΣΤ) Βιβλιογραφία :

1. Ν. Παπαμάρκος, Ψηφιακή Επεξεργασία και Ανάλυση Εικόνας, Έκδοση 3η+ , Εκδόσεις : Προέρχεται από Αυτοέκδοση , Ξάνθη.
2. https://www.mathworks.com/?s_tid=gn_logo
3. https://www.mathworks.com/matlabcentral/answers/index
4. https://www.youtube.com/watch?v=17cOHpSaqi0
5. https://www.youtube.com/watch?v=sRFM5IEqR2w
6. https://www.google.com/search?q=images&source=lmns&bih=641&biw=1396&hl=el&ved=2ahUKEwjYh-m2varqAhVRNRoKHahLDVIQ_AUoAHoECAEQAA