Τμήμα Μηχανικών Η/Υ και Πληροφορικής
Εργαστήριο Επεξεργασίας Σημάτων & Τηλεπικοινωνιών
Τομέας Υλικού και Αρχιτεκτονικής Υπολογιστών
Πανεπιστήμιο Πατρών

Εργαστήριο Ψηφιακής Επεξεργασίας Σημάτων

Σετ Βοηθητικών Ασκήσεων #3

(Εισαγωγή στην Ψηφιακή Επεξεργασία Εικόνας & Μηχανική Όραση)

Επιβλέπων Καθηγητής:
Δημήτριος Κοσμόπουλος

Σύνταξη – Επιμέλεια:
Αλέξανδρος-Οδυσσέας Φαρμάκης

Πάτρα, Εαρινό Εξάμηνο, 2022-23

Θέματα: Εισαγωγή στη MATLAB για ΨΕΣ

Θέμα 1ο) Μέθοδοι Φιλτραρίσματος Εικόνας

Το φιλτράρισμα και η επεξεργασία γενικώς μιας εικόνας συχνά είναι απαραίτητα για την αφαίρεση θορύβου, τη βελτίωση λεπτομερειών και πιθανώς την εξαγωγή κάποιας χρήσιμης πληροφορίας εντός αυτής. Συνεπώς, στην άσκηση αυτή ζητείται:
α) Χρησιμοποιώντας τις συναρτήσεις ***fspecial()*** και ***imfilter()*** της MATLAB, να εφαρμόσετε Gaussian φίλτρο στην εικόνα.
β) Να φιλτράρετε την ίδια εικόνα χρησιμοποιώντας τη συνάρτηση ***imnlmfilt()*** της MATLAB από το Image Processing Toolbox, το οποίο υλοποιεί φίλτρο μη-τοπικών μέσων (non-local means filter).
γ) Να εφαρμόσετε τα ίδια φίλτρα όπως προηγουμένως σε μια εκδοχή της εικόνας που σας δίνεται η οποία έχει μολυνθεί με Gaussian θόρυβο και να σχολιάσετε τα αποτελέσματά σας σε σχέση με την αρχική, μη θορυβοποιημένη εικόνα.

Λύση:

α) Αρχικά, να αναφέρουμε ότι η συνάρτηση ***fspecial()*** δημιουργεί ένα δισδιάστατο φίλτρο με τύπο που καθορίζουμε εμείς. Ορισμένοι τύποι φίλτρων έχουν προαιρετικές πρόσθετες παραμέτρους, με την αντίστοιχη σύνταξή τους. Ως έξοδο, επιστρέφει το πυρήνα συσχέτισης (correlation kernel), το οποίο βρίσκεται σε τέτοια μορφή ώστε να χρησιμοποιηθεί με τη συνάρτηση ***imfilter()***. Με άλλα λόγια, η ***fspecial()*** χρησιμεύει στη δημιουργία του φίλτρου, και τη πιθανή δημιουργία γραφήματός του αργότερα, ενώ η ***imfilter()*** χρησιμεύει στην εφαρμογή του σχεδιασμένου φίλτρου. Συνεπώς, ορίζουμε τις παραμέτρους μας εντός της ***fspecial()***, όπως την διασπορά *σ* της Gaussian κατανομής και το μέγεθος για το φίλτρο μας, και μετά περνάμε αυτό τον πυρήνα στην ***imfilter()***.

Ο κώδικας αυτού του ερωτήματος θα δοθεί συνολικά στο τέλος της άσκησης.

β) Τα φίλτρα μη-τοπικών μέσων χρησιμοποιούνται για την αφαίρεση θορύβου σε μια εικόνα και τη βελτίωση της υφής σε εφαρμογές μηχανικής όρασης. Μειώνει αποτελεσματικά το θόρυβο, λαμβάνοντας υπόψη την ομοιότητα μεταξύ περιοχών μιας εικόνας, και είναι κατάλληλη για θόρυβο που ακολουθεί Gaussian κατανομή. Πιο συγκεκριμένα όμως για τη συνάρτηση ***imnlmfilt()***, υπάρχει και η επιλογή της παραμέτρου 'DegreeOfSmoothing', το οποίο παίρνει θετικές τιμές και ελέγχει την ποσότητα εξομάλυνσης στην εικόνα εξόδου. Η παράμετρος αυτή αποτελεί την εκτίμηση της τυπικής απόκλισης του Gaussian θορύβου που υπάρχει στην εικόνα. Υψηλότερη τιμές τέτοιες οδηγούν σε υψηλότερη εξομάλυνση. Αν όμως δεν παρέχεται η τιμή αυτής της παραμέτρου, υπολογίζεται πριν από την εκτέλεση του φιλτραρίσματος, χρησιμοποιώντας τη συνέλιξη της εικόνας με μητρώο 3 x 3 το οποίο δεν είναι ευαίσθητο στο Laplacian της εικόνας, καθώς δομές όπως οι ακμές έχουν ισχυρά διαφορικά στοιχεία δεύτερης τάξης. Στην περίπτωση εικόνων RGB, το 'DegreeOfSmoothing' λαμβάνεται ως ο μέσος όρος των τυπικών αποκλίσεων του θορύβου που υπολογίζονται στα κανάλια.

Ο κώδικας αυτού του ερωτήματος θα δοθεί συνολικά στο τέλος της άσκησης.

γ) Παρατηρούμε πως, για την αποτελεσματική αφαίρεση του θορύβου τόσο με το Gaussian φίλτρο όσο το φίλτρο μη-τοπικών μέσων ότι πρέπει να μην υπερεκτιμήσουμε την διασπορά που ορίζουμε, ώστε να μην έχουμε τόση εξομάλυνση που να χαθεί η λεπτομέρεια της εικόνας μας.

Φιλτραρισμένη εικόνα από Gaussian θόρυβο με μέσο 0 και διασπορά 0,05 από την ***imfilter()***:

Φιλτραρισμένη εικόνα από Gaussian θόρυβο με μέσο 0 και διασπορά 0,05 από την ***imnlmfilt()***:

Ο κώδικας αυτού του ερωτήματος δίνεται μαζί με τα προηγούμενα στην επόμενη σελίδα.

Ο κώδικας της άσκησης είναι ο παρακάτω:

```
I = imread('acropolis.png');

% Δημιουργία θορυβοποιημένης εικόνας με μέσο 0 και διασπορά 0,05
noisy = imnoise(I, 'gaussian', 0, 0.05);

% Δημιουργία ενός Gaussian φίλτρου μέσω της fspecial()
variance = 1.0;     % Διασπορά της Gaussian κατανομής του φίλτρου
kernel_size = 3;    % Μέγεθος του πυρήνα συσχέτισης του φίλτρου
kernel = fspecial('gaussian', kernel_size, variance);

%======================== Original Image Filtered ========================%

figure(1);
imshow(I);
title('Original Image');

% Εφαρμογή του Gaussian φίλτρου μέσω της imfilter()
% Η επιλογή 'replicate' χρησιμοποιείται για την αποφυγή
% παραμορφώσεων στα όρια της εικόνας
imfiltered_o = imfilter(I, kernel, 'replicate');
figure(2);
imshow(imfiltered_o);
title('Filtered Original Image using imfilter()');

% Εναλλακτικά γίνεται και με προσθήκη της παραμέτρου 'DegreeOfSmoothing'
nlmfiltered_o = imnlmfilt(I);
figure(3);
imshow(nlmfiltered_o);
title('Filtered Original Image using imnlmfilt()');

%========================== Noisy Image Filtered ==========================%

figure(4);
imshow(noisy);
title('Noisy Image');

% Εφαρμογή του Gaussian φίλτρου μέσω της imfilter()
% Η επιλογή 'replicate' χρησιμοποιείται για την αποφυγή
% παραμορφώσεων στα όρια της εικόνας
imfiltered_n = imfilter(noisy, kernel, 'replicate');
figure(5);
imshow(imfiltered_n);
title('Filtered Noisy Image using imfilter()');

% Εναλλακτικά γίνεται και με προσθήκη της παραμέτρου 'DegreeOfSmoothing'
nlmfiltered_n = imnlmfilt(noisy);
figure(6);
imshow(nlmfiltered_n);
title('Filtered Noisy Image using imnlmfilt()');
```

Θέματα: Εισαγωγή στη MATLAB για ΨΕΣ

Θέμα 2ο) Εντοπισμός Χαρακτηριστικών Εικόνας

Η ανίχνευση άκρων και γωνιών είναι τεχνικές της ψηφιακής επεξεργασίας εικόνας που προσδιορίζουν και προβάλλουν τα όρια αυτά των αντικειμένων σε μια εικόνα. Χρησιμοποιούνται σε εφαρμογές υπολογιστικής όρασης και ανάλυσης εικόνας, όπως η αναγνώριση αντικειμένων, η παρακολούθηση και η διακριτοποίηση. Συνεπώς, στην άσκηση αυτή ζητείται:
α) Χρησιμοποιώντας τη συνάρτηση ***conv2()*** της MATLAB, εφαρμόστε κατάλληλα το τελεστή Sobel ώστε να υπολογίσετε την ισχύ και τον προσανατολισμό των ακμών της εικόνας.
β) Χρησιμοποιώντας τον αλγόριθμο Moravec, να εντοπίσετε τις γωνίες της εικόνας πρώτα με το άθροισμα των τετραγωνισμένων διαφορών και μετά με το άθροισμα των απόλυτων διαφορών. Ποια η διαφορά τους;
γ) Χρησιμοποιώντας την απόκριση Harris, να εντοπίσετε τις γωνίες της εικόνας και να συγκρίνετε τα αποτελέσματά σας με τον αλγόριθμο Moravec του προηγούμενου ερωτήματος.

Λύση:

α) Ο τελεστής Sobel αποτελεί ένα είδος τελεστή διακριτής διαφοροποίησης (discrete differentiation operator), το οποίο χρησιμοποιείται για την προσέγγιση της παραγώγου ενός διακριτού σήματος, δηλαδή έχει σχεδιαστεί για να υπολογίζει το ρυθμό μεταβολής σε κάθε σημείο του σήματος, ή της εικόνας όπως έχουμε στην προκειμένη περίπτωση.

Για να υπολογίσουμε τα ζητούμενα του ερωτήματος αυτού, πρέπει πρώτα να βρούμε τους ρυθμούς μεταβολής της εικόνας ως προς τους άξονες. Συνεπώς, πρέπει να εφαρμόσουμε τα εξής μητρώα:

$$\text{Άξονας x} \qquad \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \qquad \text{Άξονας y} \qquad \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

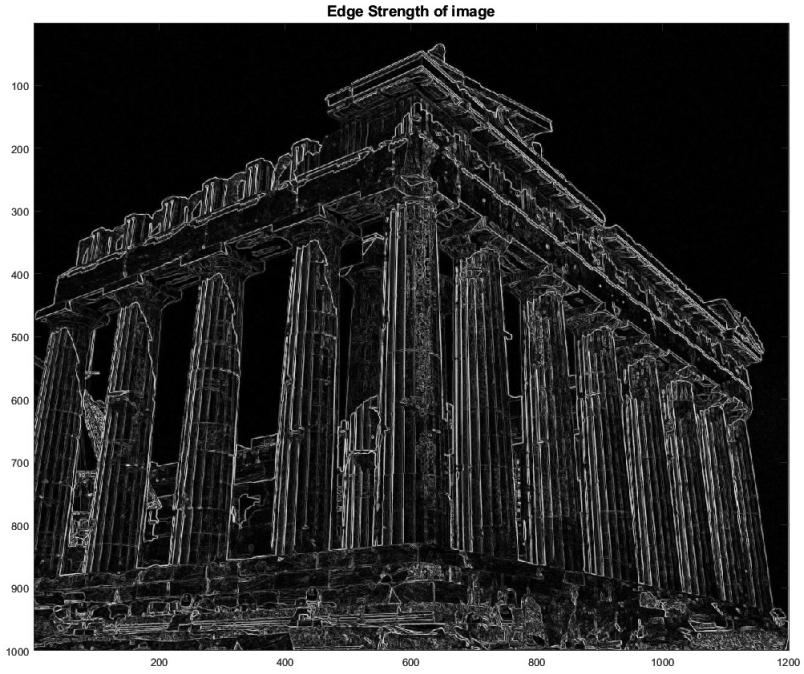
Αφού έχουμε υπολογίσει τις μερικές παραγώγους 1ης τάξης, ορίζουμε τις εξής ποσότητες:

$$\text{Edge Strength} = \sqrt{(x^2 + y^2)} \qquad \text{Edge Orientation} = \arctan\left(\frac{y}{x}\right)$$

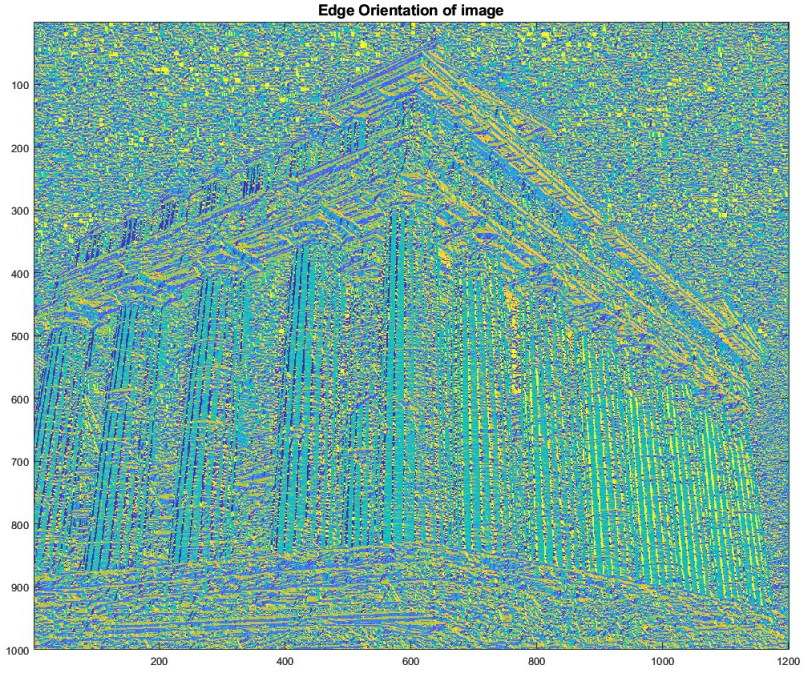
όπου x και y οι αντίστοιχες μερικοί παράγωγοι της εικόνας.

Στην επόμενη σελίδα φαίνονται τα γραφήματα της ισχύς και του προσανατολισμού των ακμών της εικόνας.

Ισχύς ακμών της εικόνας:



Προσανατολισμός ακμών της εικόνας:



Ο κώδικας αυτού του ερωτήματος θα δοθεί συνολικά στο τέλος της άσκησης.

β) Ο αλγόριθμος ανίχνευσης γωνίας του Moravec είναι ένας αλγόριθμος που χρησιμοποιείται για τον εντοπισμό γωνιών ή σημείων ενδιαφέροντος σε μια εικόνα. Στοχεύει στον εντοπισμό περιοχών σε μια εικόνα που έχουν σημαντικές διακυμάνσεις φωτεινότητας σε πολλαπλές κατευθύνσεις, οι οποίες τυπικά αντιστοιχούν σε γωνίες ή διασταυρώσεις μεταξύ διαφορετικών αντικειμένων ή δομών. Είναι σχετικά απλός στην υλοποίηση και έχει ανάγκη λίγων υπολογιστικών πόρων. Ωστόσο, έχει περιορισμούς στην ανίχνευση γωνιών παρουσία θορύβου και μη γωνιακών άκρων. Γενικώς, τα βασικά βήματα που απαιτούνται για την εφαρμογή του είναι τα εξής:

(1) Παραγώγιση της εισαγόμενης εικόνας με κάποιο τελεστή διακριτής διαφοροποίησης.
(2) Για κάθε εικονοστοιχείο στην εικόνα, ο αλγόριθμος υπολογίζει κάποιο άθροισμα διαφορών (συνήθως το SSD) μεταξύ των τιμών έντασης του εικονοστοιχείου και των γύρω γειτόνων του σε ένα μικρό παράθυρο. Το παράθυρο είναι συνήθως μια τετράγωνη περιοχή με κέντρο γύρω από το εικονοστοιχείο. Αυτό το άθροισμα διαφορών υπολογίζεται για όλες τις πιθανές διατάξεις του παραθύρου σε διαφορετικές κατευθύνσεις.
(3) Ο αλγόριθμος αξιολογεί μια συνάρτηση απόκρισης γωνίας για κάθε εικονοστοιχείο λαμβάνοντας υπόψη τις τιμές του αθροίσματος διαφορών που υπολογίστηκαν στο προηγούμενο βήμα. Η συνάρτηση απόκρισης μετρά την πιθανότητα ένα εικονοστοιχείο να είναι γωνία. Συνήθως ορίζεται ως η ελάχιστη τιμή του αθροίσματος διαφορών μεταξύ όλων των διατάξεων του παραθύρου. Μια μεγαλύτερη απόκριση υποδηλώνει μεγαλύτερη πιθανότητα ύπαρξης γωνίας.
(4) Εφαρμόζεται ένα κατώφλι στη συνάρτηση απόκρισης γωνιών για να φιλτράρει τις αδύναμες γωνίες και να διατηρεί μόνο τις ισχυρές. Αυτό το κατώφλι επιλέγεται εμπειρικά με βάση την επιθυμητή ευαισθησία του αλγορίθμου.
(5) Ο αλγόριθμος δίνει ως έξοδο την εικόνα με τις γωνίες που εντόπισε.

Παρατηρούμε ότι η εκδοχή που χρησιμοποιεί το SSD (δηλαδή υπολογίζει το άθροισμα των τετραγωνισμένων διαφορών) πρακτικά ενισχύει τις διαφορές μεταξύ των εντάσεων των γειτονικών εικονοστοιχείων. Αυτή η ενίσχυση ανταποκρίνεται περισσότερο σε απότομες αλλαγές έντασης, όπως αυτές που συμβαίνουν στις γωνίες. Ως αποτέλεσμα, η έκδοση SSD τείνει να είναι καλύτερο στον εντοπισμό γωνιών. Από την άλλη όμως, η εκδοχή που χρησιμοποιεί το SAD (δηλαδή υπολογίζει το άθροισμα των απόλυτων διαφορών), καθιστά τον αλγόριθμο λιγότερο ευαίσθητο στο πρόσημο των αλλαγών φωτεινότητας. Τυπικά, οι ακμές έχουν μια σταδιακή μεταβολή στις τιμές φωτεινότητας, με αποτέλεσμα να υπάρχουν μικρές αλλά σταθερές διαφορές μεταξύ γειτονικών εικονοστοιχείων, κάτι που μπορεί να χρησιμεύσει για τον εντοπισμό αυτών.

Ο κώδικας αυτού του ερωτήματος θα δοθεί συνολικά στο τέλος της άσκησης.

γ) Ο αλγόριθμος ανίχνευσης γωνίας του Harris είναι ένας αλγόριθμος που χρησιμοποιείται για τον εντοπισμό γωνιών ή σημείων ενδιαφέροντος σε μια εικόνα, το οποίο σε αντίθεση με τον αλγόριθμο του Moravec, ενσωματώνει ένα μέτρο τόσο της αλλαγής της φωτεινότητας όσο και της κατανομής των γωνιών στον χώρο, το οποίο οδηγεί σε μεγαλύτερη ακρίβεια εντοπισμού πραγματικής γωνίας στην εικόνα. Γενικώς, τα βασικά βήματα που απαιτούνται για την εφαρμογή του είναι τα εξής:

(1) Παραγώγιση της εισαγόμενης εικόνας με κάποιο τελεστή διακριτής διαφοροποίησης.
(2) Για κάθε εικονοστοιχείο, υπολογίζονται τα γινόμενα των παραγώγων τετραγωνίζοντας τις παραγώγους που βρήκαμε στο προηγούμενο βήμα, και κρατάμε τα γινόμενά τους.
(3) Υπολογίζεται η συνάρτηση απόκρισης γωνίας *R* για κάθε εικονοστοιχείο χρησιμοποιώντας τον ακόλουθο τύπο:

$$R = det(M) - k \cdot (trace(M))^2$$

όπου det(M) είναι η ορίζουσα του τανυστή *M*, το trace(M) είναι το ίχνος του τανυστή *M* και το *k* είναι μια εμπειρική σταθερά (συνήθως βρίσκεται μεταξύ 0,04 και 0,06). Η συνάρτηση απόκρισης γωνίας καταγράφει την πιθανότητα ένα εικονοστοιχείο να είναι γωνία με βάση τις ιδιοτιμές του *M*.
(4) Εφαρμόζεται ένα κατώφλι στη συνάρτηση απόκρισης γωνιών για να φιλτράρει τις αδύναμες γωνίες και να διατηρεί μόνο τις ισχυρές. Αυτό το κατώφλι επιλέγεται εμπειρικά με βάση την επιθυμητή ευαισθησία του αλγορίθμου.
(5) Ο αλγόριθμος δίνει ως έξοδο την εικόνα με τις γωνίες που εντόπισε.

Ενδεικτικά, δίνεται το γράφημα που συγκρίνει τις γωνίες που εντοπίζονται από το Harris (δίνονται με κόκκινο) και από το Moravec με τη χρήση των SSD (δίνονται με μπλε) για *h_thresh* = 0,1 και *m_thresh* = 0,5. Όσες γωνίες εντοπίζονται από κοινού εμφανίζονται με πράσινο.

Ο κώδικας αυτού του ερωτήματος δίνεται μαζί με τα προηγούμενα στις επόμενες σελίδες.

Ο κώδικας της άσκησης είναι ο παρακάτω:

```
% Τελεστές Sobel για τους άξονες x και y
sobel_x = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
sobel_y = [-1, -2, -1; 0, 0, 0; 1, 2, 1];

image = imread('acropolis.png');
I = rgb2gray(image);
I = double(I)/255;  % Κανονικοποίηση των τιμών φωτεινότητας των
                    % εικονοστοιχείων για ευκολότερες πράξεις

figure(1);
imagesc(I);
colormap gray;
axis image;
title("Original image (grayscale)");


% Συνέλιξη του τελεστή Sobel κατά τον άξονα x
f_x = conv2(I, sobel_x, 'same');
figure(2);
imagesc(f_x);
colormap gray;
axis image;
title("∂I(x, y) / ∂x");

% Συνέλιξη του τελεστή Sobel κατά τον άξονα y
f_y = conv2(I, sobel_y, 'same');
figure(3);
imagesc(f_y);
colormap gray;
axis image;
title("∂I(x, y) / ∂y");

% Υπολογισμός ισχύς και προσανατολισμού των ακμών
edge_strength = sqrt(f_x.^2 + f_y.^2);
edge_orientation = atan2(f_y, f_x);

figure(4);
imagesc(edge_strength);
colormap gray;
axis image;
title('Edge Strength of image');

figure(5);
imagesc(edge_orientation);
axis image;
title('Edge Orientation of image');


%================ Moravec Corner Detection Algorithm Part ================%
```

```
% Καθορισμός του μεγέθους του παραθύρου για τον εντοπισμό γωνιών
w_size_m = 3;

% Αρχικοποίηση των μητρώων της απόκρισης γωνίας με μηδενικά
response_ssd = zeros(size(I));
response_sad = zeros(size(I));

% Επανάληψη για κάθε εικονοστοιχείο της εικόνας
% SSD -> Sum of Squared Differences
% SAD -> Sum of Absolute Differences
for y = 1:size(I, 1)
    for x = 1:size(I, 2)
        % Υπολογισμός του SSD και του SAD σε διαφορετικές κατευθύνσεις
        dx_window = f_x(max(1, y-floor(w_size_m/2)) : min(size(I, 1), ...
            y+floor(w_size_m/2)), max(1, x-floor(w_size_m/2)) : ...
            min(size(I, 2), x+floor(w_size_m/2)));
        dy_window = f_y(max(1, y-floor(w_size_m/2)) : min(size(I, 1), ...
            y+floor(w_size_m/2)), max(1, x-floor(w_size_m/2)) : ...
            min(size(I, 2), x+floor(w_size_m/2)));
        ssd_values = sum(dx_window(:).^2 + dy_window(:).^2);
        sad_values = sum(abs(dx_window(:)) + abs(dy_window(:)));

        % Αποθήκευση της απόκρισης γωνίας
        response_ssd(y, x) = ssd_values;
        response_sad(y, x) = sad_values;
    end
end

% Κατώφλι Moravec
m_thresh = 0.5;

% Αρχική εικόνα με τις ανιχνευμένες γωνίες χρησιμοποιώντας Moravec (SSD)
% και τυπωμένες ως κόκκινες τελείες
corners_ssd = response_ssd > m_thresh * max(response_ssd(:));
figure(6);
imshow(I);
title('Moravec Corner Detection (using SSD)');
hold on;
[x, y] = find(corners_ssd);
plot(y, x, 'r.', 'MarkerSize', 3);
hold off;

% Αρχική εικόνα με τις ανιχνευμένες γωνίες χρησιμοποιώντας Moravec (SAD)
% και τυπωμένες ως κόκκινες τελείες
corners_sad = response_sad > m_thresh * max(response_sad(:));
figure(7);
imshow(I);
title('Moravec Corner Detection (using SAD)');
hold on;
[x, y] = find(corners_sad);
```

```
plot(y, x, 'r.', 'MarkerSize', 3);
hold off;

%====================== Harris Corner Detection Part ====================%

% Καθορισμός του μεγέθους του παραθύρου για τον εντοπισμό γωνιών
w_size_h = 3;

% Αρχικοποίηση του παραθύρου για τη γωνιακή απόκριση
window = ones(w_size_h);

% Υπολογισμός των γινομένων του gradient
Ixx = f_x .* f_x;
Iyy = f_y .* f_y;
Ixy = f_x .* f_y;

% Υπολογισμός του αθροίσματος των τετραγώνων
Sxx = conv2(Ixx, window, 'same');
Syy = conv2(Iyy, window, 'same');
Sxy = conv2(Ixy, window, 'same');

% Ορισμός σταθεράς Harris, ορίζουσας και ίχνους του σχηματιζόμενου τανυστή
k = 0.04;
det_M = Sxx .* Syy - (Sxy).^2;
trace_M = Sxx + Syy;
harris_response = det_M - k * (trace_M).^2;


% Κατώφλι Harris
h_thresh = 0.1;

% Αρχική εικόνα με τις ανιχνευμένες γωνίες χρησιμοποιώντας την
% απόκριση Harris και τυπωμένες ως κόκκινες τελείες
corners_harris = harris_response > h_thresh * max(harris_response(:));
figure(13);
imshow(I);
title('Harris Corner Detection');
hold on;
[x, y] = find(corners_harris);
plot(y, x, 'r.', 'MarkerSize', 3);
hold off;

%==================== Corner Detection Comparison Part ==================%

% Σύγκριση των Harris και SSD Moravec
figure(9);
imshow(I);
title('Harris (red) vs SSD Moravec (blue) Corner Detections');
hold on;
[x1, y1] = find(corners_harris);
plot(y1, x1, 'r.', 'MarkerSize', 3);
```

```
[x2, y2] = find(corners_ssd);
plot(y2, x2, 'b.', 'MarkerSize', 3);

% Εύρεση των κοινών σημείων τα οποία τυπώνονται με πράσινο
overlap_points = intersect([x1, y1], [x2, y2], 'rows');
[x3, y3] = deal(overlap_points(:, 1), overlap_points(:, 2));
plot(y3, x3, 'g.', 'MarkerSize', 3);
hold off;

% Σύγκριση των Harris και SAD Moravec
figure(10);
imshow(I);
title('Harris (red) vs SAD Moravec (blue) Corner Detections');
hold on;
[x1, y1] = find(corners_harris);
plot(y1, x1, 'r.', 'MarkerSize', 3);
[x2, y2] = find(corners_sad);
plot(y2, x2, 'b.', 'MarkerSize', 3);

% Εύρεση των κοινών σημείων τα οποία τυπώνονται με πράσινο
overlap_points = intersect([x1, y1], [x2, y2], 'rows');
[x3, y3] = deal(overlap_points(:, 1), overlap_points(:, 2));
plot(y3, x3, 'g.', 'MarkerSize', 3);
hold off;
```

Θέματα: Εισαγωγή στη MATLAB για ΨΕΣ

Θέμα 3ο) Διακριτοποίηση βάσει περιοχής (Region-based Segmentation)

Η διακριτοποίηση βάσει περιοχής είναι μια τεχνική στην ψηφιακή επεξεργασία εικόνας που έχει στόχο τη τμηματοποίηση μιας εικόνας σε σημαντικές περιοχές ή αντικείμενα με βάση ορισμένες ιδιότητές τους. Μια προσέγγιση που χρησιμοποιείται είναι αυτή της μεθόδου ανάπτυξης περιοχής.

α) Εξηγήστε το τρόπο λειτουργίας της μεθόδου ανάπτυξης περιοχής για την διακριτοποίηση βάσει περιοχής.

β) Να γράψετε κατάλληλο πρόγραμμα MATLAB το οποίο να εφαρμόζει την προαναφερόμενη μέθοδο. Πειραματιστείτε με διαφορετικές τιμές για το εύρος κατωφλίου και σημεία έναρξης και σημειώστε τα αποτελέσματά σας.

Λύση:

α) Η μέθοδος ανάπτυξης περιοχής αποτελεί μια επαναληπτική διαδικασία που ξεκινά από κάποιο εικονοστοιχείο ή περιοχή, το οποίο ονομάζουμε seed, και επεκτείνει βάσει της συγκέντρωσης των γειτονικών εικονοστοιχείων ή περιοχών που πληρούν ορισμένα κριτήρια ομοιότητας. Η βασική ιδέα πίσω από την ανάπτυξη της περιοχής είναι ότι γειτονικά εικονοστοιχεία ή περιοχές με παρόμοια χαρακτηριστικά, όπως χρώμα, φωτεινότητα, υφή ή άλλα χαρακτηριστικά, είναι πιθανό να ανήκουν στο ίδιο αντικείμενο ή περιοχή. Μια τέτοια περιοχή μεγαλώνει όσο συνεχίζουν τα νέα άμεσα γειτονικά εικονοστοιχεία να πληρούν τις συνθήκες που έχουν τεθεί. Έστω για παράδειγμα το παρακάτω 5 x 5 μητρώο.

$$\begin{pmatrix} 7 & 5 & 4 & 8 & 5 \\ 6 & 7 & 6 & 2 & 5 \\ 4 & 5 & 3 & 6 & 4 \\ 7 & 7 & 4 & 7 & 9 \\ 8 & 6 & 5 & 8 & 5 \end{pmatrix}$$

Έστω ότι θέλαμε να εφαρμόσουμε την τεχνική αυτή, ξεκινώντας από το πιο κεντρικό στοιχείο του μητρώου (εδώ το 3), για κατώφλια $T = 2$, $T = 3$ και $T = 4$. Τα αποτελέσματα αυτών θα ήταν τα εξής:

$T = 2$

$$\begin{pmatrix} 7 & 5 & 4 & 8 & 5 \\ 6 & 7 & 6 & 2 & 5 \\ 3 & 3 & 3 & 6 & 4 \\ 7 & 7 & 3 & 7 & 9 \\ 8 & 6 & 3 & 8 & 5 \end{pmatrix}$$

$T = 3$

$$\begin{pmatrix} 7 & 3 & 3 & 8 & 3 \\ 3 & 7 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 7 & 7 & 3 & 7 & 9 \\ 8 & 3 & 3 & 8 & 5 \end{pmatrix}$$

$T = 4$

$$\begin{pmatrix} 3 & 3 & 3 & 8 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 3 \\ 3 & 3 & 3 & 3 & 9 \\ 8 & 3 & 3 & 8 & 5 \end{pmatrix}$$

Παρατηρούμε πως σε κάθε περίπτωση τα στοιχεία που η (απόλυτη) διαφορά τους με το επιλεγμένο στοιχείο ανάπτυξης είναι καθαρά μεγαλύτερο από το ορισμένο κατώφλι έχουμε ουσιαστικά τη δημιουργία ενός “συνόρου”, ενώ στα υπόλοιπα βλέπουμε δημιουργείται μια “περιοχή”. Συνεπώς, η επιλογή κατάλληλου κατωφλίου είναι πολύ σημαντική και ποικίλλει ανά εφαρμογή και περίπτωση.

Η επιλογή των κριτηρίων ανάπτυξης δεν εξαρτάται μόνο από το ίδιο το συγκεκριμένο πρόβλημα, αλλά και τον τύπο των δεδομένων της εικόνας που χρησιμοποιείται, όπως RGB ή grayscale. Η γενική διαδικασία ανάπτυξης σταματά όταν συνεχίζουν τα γειτονικά εικονοστοιχεία να πληρούν τις συνθήκες ανάπτυξης, και για να αυξηθεί η ικανότητα της ανάπτυξης της περιοχής συχνά λαμβάνει υπόψη τα κριτήρια που σχετίζονται με την γενικότερη φύση των εικόνων και των αντικειμένων, όπως το μέγεθος και το σχήμα. Τα βήματα εφαρμογής για τη μέθοδο ανάπτυξης της περιοχής είναι τα εξής:

(1) Η εικόνα σαρώνεται γραμμή ανά γραμμή ώστε να μη βρεθούν αποδιδόμενα εικονοστοιχεία.
(2) Παίρνουμε το εικονοστοιχείο που βρέθηκε ως κέντρο και ελέγχονται τα γειτονικά του με αυτό ένα προς ένα. Αν η διαφορά τους είναι μικρότερη από το προκαθορισμένο όριο, τότε συγχωνεύονται.
(3) Με τα πρόσφατα συγχωνευμένα εικονοστοιχεία ως κέντρο, το βήμα 2 ανιχνεύεται μέχρι την περιοχή δεν μπορεί να γίνει περαιτέρω επέκταση.
(4) Επιστρέφοντας στο βήμα 1, συνεχίζεται η σάρωση έως ότου δεν βρίσκονται τα εικονοστοιχεία που δεν μπορεί να τους αποδοθεί νέα τιμή και η διαδικασία ανάπτυξης τελειώνει.

Η παραπάνω μέθοδος πρέπει να σαρωθεί πρώτα, η οποία έχει σχετικά μεγάλη εξάρτηση σχετικά με την επιλογή της αφετηρίας ανάπτυξης της περιοχής. Για να αντιμετωπιστεί αυτό το πρόβλημα, η μέθοδος μπορεί να βελτιωθεί ως εξής:
(1) Ορίζουμε το όριο της διαφοράς στο μηδέν. Επεκτείνουμε την περιοχή με την παραπάνω μέθοδο και συγχωνεύουμε τα εικονοστοιχεία με την ίδια τιμή.
(2) Λαμβάνουμε μια μέση διαφορά μεταξύ όλων των γειτονικών περιοχών και συγχωνεύουμε τις γειτονικές περιοχές με την ελάχιστη διαφορά.
(3) Ορίζουμε τα κριτήρια τερματισμού και συγχωνεύουμε τις περιοχές με τη σειρά τους, επαναλαμβάνοντας τη λειτουργία στο βήμα 2 έως ότου εκπληρωθούν τα κριτήρια τερματισμού, οπότε η διαδικασία ανάπτυξης είναι ολοκληρώνεται.

Όταν υπάρχει μια περιοχή με αργά μεταβαλλόμενες τιμές στην εικόνα, η παραπάνω μέθοδος μπορεί να προκαλέσει τη συγχώνευση διαφορετικών, “ξένων” περιοχών και την παραγωγή σφαλμάτων. Για την αντιμετώπιση αυτού του προβλήματος, αντί να χρησιμοποιήσουμε την τιμή του νέου εικονοστοιχείου για να συγκρίνουμε με την τιμή του εικονοστοιχείου γειτονιάς, η περιοχή τιμών του νέου εικονοστοιχείου θα συγκριθεί με την τιμή του κάθε γειτονικού εικονοστοιχείου. Για μια περιοχή εικόνας με *N* εικονοστοιχεία, η τιμή σε grayscale είναι:

$$m=\frac{1}{N}\sum_{R} f(x,y)$$

Η σύγκριση των εικονοστοιχείων είναι:

$$\max_{R} |f(x,y)-m|<T$$

όπου *T* το κατώφλι (threshold).

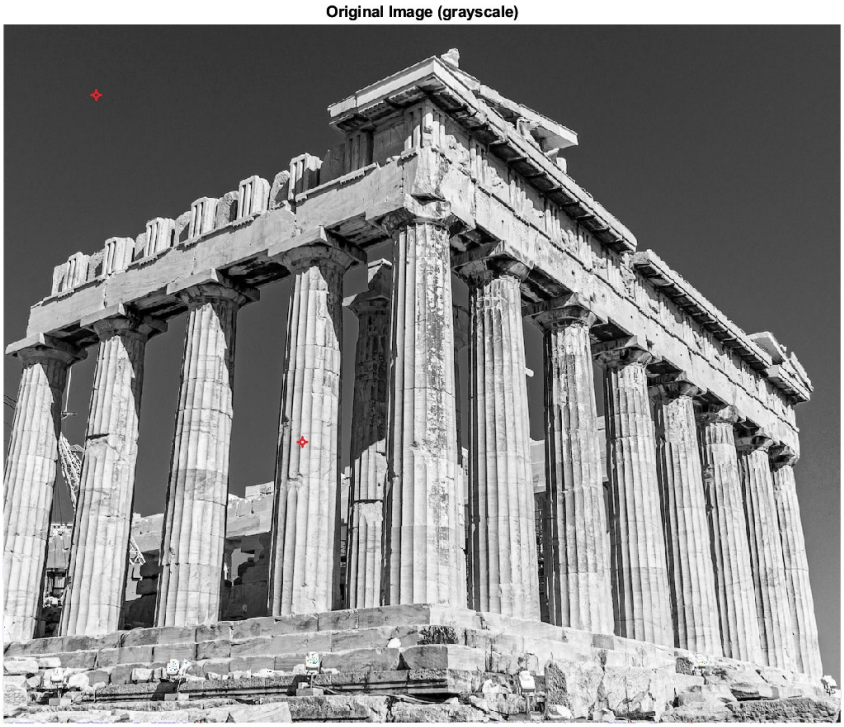
Αν η περιοχή είναι ομοιόμορφη, η αλλαγή του grayscale στην περιοχή πρέπει να είναι όσο μικρότερη δυνατόν. Αν η περιοχή είναι μη μέση (non-mean), που αποτελεί τη γενική κατάσταση, και αποτελείται από δύο μέρη, η αναλογία αυτών των δύο εικονοστοιχείων στο *R* είναι q_1 και q_2 αντίστοιχα, και οι grayscale τιμές είναι m_1 και m_2. Τότε ο μέσος όρος της περιοχής είναι $q_1 m_1 + q_2 m_2$ και για το εικονοστοιχείο με grayscale τιμή m_1, και η διαφορά μεταξύ των δύο τιμών είναι $s_m = m_1 - (q_1 m_1 + q_2 m_2)$. Μπορούμε να δούμε ότι η πιθανότητα σωστής απόφασης τιμής είναι:

$$P(T)=\frac{1}{2}[P(|T-s_m|)+P(|T+s_m|)]$$

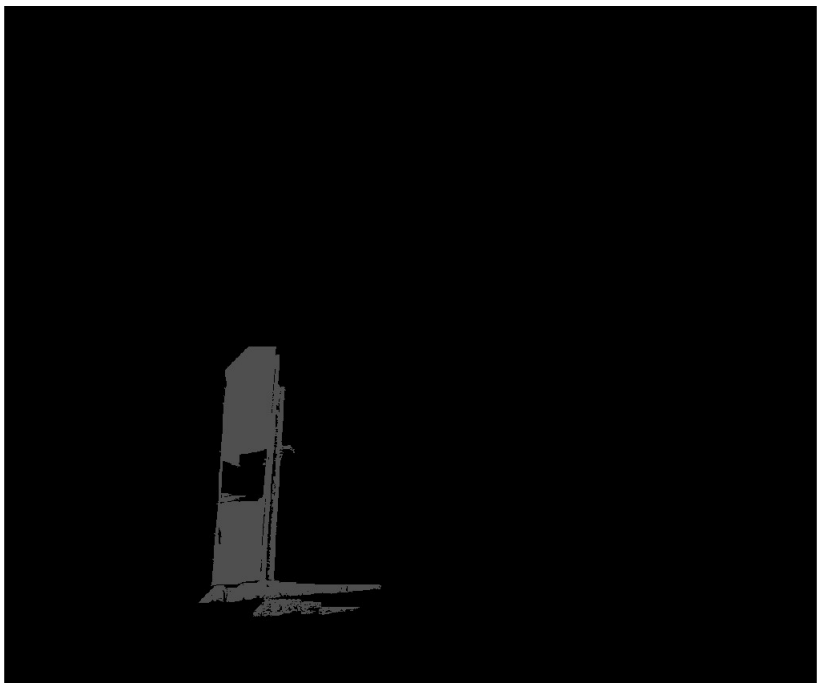
Αυτό υποδηλώνει ότι η grayscale διαφορά μεταξύ διαφορετικών εικονοστοιχείων θα πρέπει να είναι όσο το δυνατόν μεγαλύτερη όταν λαμβάνεται υπόψη η μέση grayscale τιμή.

β) Στις επόμενες παρουσιάζονται μερικές έξοδοι της παρακάτω εικόνας, όπου με κόκκινο έχουμε σημειώσει τα εικονοστοιχεία που αντιστοιχούν στα (x, y) = (112, 85) και (x, y) = (362, 505) για κατώφλια *thresh* = 0,4444 και *thresh* = 0,4888 το καθένα.

Αρχική εικόνα:



Εικόνα για (x, y) = (362, 505) και *thresh* = 0,4444:



Εικόνα για (x, y) = (362, 505) και *thresh* = 0,4888:

Εικόνα για (x, y) = (362, 505) και *thresh* = 0,4444:

Εικόνα για (x, y) = (362, 505) και *thresh* = 0,4888:

Ο κώδικας της άσκησης δίνεται στις επόμενες σελίδες.

Ο κώδικας της άσκησης είναι ο παρακάτω:

```
image = imread('acropolis.png');
I = rgb2gray(image);
figure(1);
imshow(I);
title('Original Image (grayscale)');
I = double(I)/255;  % Κανονικοποίηση των τιμών φωτεινότητας των
                    % εικονοστοιχείων για ευκολότερες πράξεις
[M, N] = size(I);   % Λήψη μεγέθους εικόνας (σειρές και στήλες)
[y, x] = getpts;    % Λήψη σημείου εκκίνησης region growing από διπλό
                    % κλικ πάνω στην αρχική εικόνα
x1 = round(x);      % Στρογγυλοποίηση στην τετμημένη (άξονας x)
y1 = round(y);      % Στρογγυλοποίηση στην τεταγμένη (άξονας y)
seed = I(x1, y1);   % Αποθήκευση του gray value του σημείου έναρξης
                    % για το region growing
Y = zeros(M, N);    % Αρχικοποίηση μητρώου ίδιου μεγέθους με την αρχική
                    % εικόνα στην οποία θα εγγραφεί το αποτέλεσμα
Y(x1, y1) = seed;   % Ορισμός χρώματος του σημείου στο Y ίδιο
                    % με το χρώμα του επιλεγμένου εικονοστοιχείου
sum = seed;         % Αποθήκευση του αθροίσματος των gray value των
                    % σημείων που πληρούν τις region growing συνθήκες
suit = 1;           % Αποθήκευση του πλήθους των σημείων που πληρούν τις
                    % region growing συνθήκες της περιοχής (suitable)
count = 1;          % Καταγραφή του αριθμού των νέων σημείων που πληρούν
                    % τα κριτήρια στα οκτώ σημεία γύρω από ένα σημείο
                    % κάθε φορά
thresh = 0.4888;    % Κατώφλι

while count > 0
    s = 0;          % Καταγραφή του αθροίσματος των gray value των νέων
                    % σημείων που πληρούν τα κριτήρια στα οκτώ σημεία
                    % γύρω από ένα σημείο
    count = 0;
    for i = 1:M
        for j = 1:N
            % Έλεγχος του αν το σημείο έχει το ίδιο gray value με το seed
            if Y(i, j) == seed
                % Προσδιορισμός του αν αυτό το σημείο βρίσκεται στο όριο
                if (i-1) > 0 && (i+1) < (M+1) && (j-1) > 0 && (j+1) < (N+1)
                    % Προσδιορισμός του αν τα γύρω σημεία πληρούν
                    % τις συνθήκες πεδίου. Τα u, v αποτελούν offset
                    for u = -1:1
                        for v = -1:1
                            % Κρίνεται αν υπάρχει ή όχι στο πίνακα
                            % εξόδου Y και είναι ένα σημείο που
                            % ικανοποιεί τη συνθήκη
                            if (Y(i+u, j+v) == 0 && ...
                                    abs(I(i+u, j+v) - seed) <= thresh)
                                % Σύμφωνα με τις δύο παραπάνω συνθήκες,
                                % το σημείο που αντιστοιχεί στη θέση
```

```
                                % στο Y ορίζεται στο χρώμα του seed
                                Y(i+u, j+v) = seed;
                                count = count + 1;
                                % Το gray level αυτού του σημείου
                                % αθροίζεται στο s
                                s = s + I(i+u, j+v);
                            end
                        end
                    end
                end
            end
        end
    end

    sum = sum + s;      % Πρόσθεση του s στο άθροισμα του grayscale σημείου
    seed = sum / suit;  % Υπολογισμού του νέου μέσου όρου grayscale
end

figure(2);
imshow(Y);
title('Segmented image');
```