ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Αναφορά 4ης Εργαστηριακής Άσκησης

«WIENER & CLSR RESTORATION FILTERS»

Καραμαΐλής Παντελής 2016030040
Βλάχος Κωνσταντίνος 2016030042
Γαλάνης Μιχάλης 2016030036

ΠΕΡΙΕΧΟΜΕΝΑ

Οι σύνδεσμοι για τις παρακάτω ενότητες είναι διαδραστικοί.
Πιέστε πάνω στο επιθυμητό τμήμα της άσκησης για τη μετάβαση σε αυτό.

ΕΙΣΑΓΩΓΗ

Σκοπός Εργαστηρίου

Σκοπός της άσκησης αυτής είναι η μελέτη δύο τεχνικών αποκατάστασης μιας εικόνας όταν αυτή έχει υποστεί θόρυβο και θόλωση. Οι τεχνικές αυτές ονομάζονται **Wiener Filter** (Least Mean Square Filter) και **CLSR Filter** (Constrained Least Squares Restoration) και χρησιμοποιούν μια προσέγγιση της εικόνας για να αποκαταστήσουν την αρχική.

1 – WIENER FILTER RESTORATION

Στο πρώτο ερώτημα μας ζητήθηκε να εφαρμόσουμε τη τεχνική Wiener.

Επισκόπηση & Θεωρητική Ανάλυση

Ουσιαστικά η τεχνική αυτή βασίζεται στη μέθοδο των ελαχίστων τετραγώνων και σε πίνακες συσχέτισης των συναρτήσεων της εικόνας και του θορύβου. Πιο συγκεκριμένα αποσκοπούμε στην ελαχιστοποίηση της διαφοράς της αρχικής εικόνας $f(x,y)$ με την προσέγγισή της $\hat{f}(x,y)$. Ο μετασχηματισμός της προσέγγισης αυτής ισούται με:

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma\left[\frac{S_n(u,v)}{S_f(u,v)}\right]}\right] G(u,v)$$

όπου $S_f(u,v)$ και $S_n(u,v)$ είναι το φάσμα ισχύ των $f_e(x,y)$ και $n_e(x,y)$ αντίστοιχα.

Υλοποίηση

Αρχικά διαβάζουμε μια εικόνα της επιλογής μας που βρίσκεται σε ικανοποιητική κατάσταση. Εάν είναι έγχρωμη τη μετατρέπουμε σε ασπρόμαυρη. Εφαρμόζουμε τρία διαφορετικά φίλτρα gaussian (**`imnoise`**) με μέση τιμή θορύβου ίση με μηδέν και διασπορά 0.001, 0.005, 0.01 αντίστοιχα, για να μελετήσουμε και να συγκρίνουμε πολλές περιπτώσεις. Ύστερα εφαρμόζουμε φίλτρο average (**`imfilter`** & **`fspecial`**) με μάσκα [2 2] και στις τρεις εικόνες.

Για κάθε τιμή της διασποράς θορύβου υπολογίσαμε ξεχωριστά το $H(u,v)$ (μετασχηματισμός Fourier του φίλτρου), $H^*(u,v)$ (συζυγής μετασχηματισμός Fourier του φίλτρου), S_n(φάσμα ισχύ), S_f(διασπορά θορύβου) και $G(u,v)$ (μετασχηματισμός Fourier της εικόνας που «παρατηρούμε»). Θέσαμε τη τιμή του γ ίση με 10^7 για να υπάρχει ισορροπημένη φωτεινότητα στις εικόνες μας.

Τέλος, υπολογίζουμε τις 3 προσεγγισμένες εικόνες με βάση τη σχέση που αναφέρθηκε προηγουμένως στο πεδίο της συχνότητας και για να μπορέσουμε να τις αναπαραστήσουμε, υπολογίζουμε τους αντίστροφους μετασχηματισμούς τους.

Μετάβαση στον κώδικα MATLAB

2 – CLSR FILTER RESTORATION

Στο δεύτερο μέρος της άσκησης χρησιμοποιούμε το φίλτρο αποκατάστασης CLSR.

Επισκόπηση & Θεωρητική Ανάλυση

Η μέθοδος αυτή βασίζεται στην επιλογή ενός πίνακα Q, με τον οποίο ελαχιστοποιούμε συναρτήσεις της μορφής $\|Q_f\|^2$. Ο πίνακας έχει μεγάλη επίδραση στην αποτελεσματικότητα της τεχνικής, για αυτό απαιτείται μια καλή εκτίμησή του. Ο Μετασχηματισμός Fourier που θα προκύψει αυτή τη φορά ισούται με:

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2 + \gamma[P(u,v)]}\right] G(u,v)$$

Όπου $P(u,v)$ ο μετασχηματισμός Fourier του πίνακα:

$$P(x,y) = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

τον οποίο επεκτείναμε σε διαστάσεις M x N όπου M = X+2 και N = Y+2. Όπου x,y οι διαστάσεις της αρχικής εικόνας. Με έναν αλγόριθμο που αναλύεται παρακάτω, βρίσκουμε τη βέλτιστη τιμή του γ έτσι ώστε να ικανοποιείται το κριτήριο $\|g - HF\|^2 = \|n\|^2$. Ξεκινάμε με ένα διάνυσμα r ώστε:

$$r = g - H\hat{f}$$

Διαλέγουμε αρχικά μια αρχική τιμή για το γ και υπολογίζουμε μια εκτίμηση για το $\|n\|^2$ χρησιμοποιώντας τη σχέση:

$$\|n\|^2 = (M-1)(N-1)[\sigma_n^2 + \bar{\eta}_e^2]$$

Με σ_n^2, $\bar{\eta}_e^2$ η διασπορά θορύβου και το τετράγωνο της μέσης τιμής του θορύβου αντίστοιχα. Υπολογίζουμε κατόπιν το $\hat{F}(u,v)$ και βρίσκουμε τον αντίστροφο μετασχηματισμό του. Υπολογίζουμε το διάνυσμα r και την ποσότητα $\varphi(\gamma) = \|r\|^2$. Αναλόγως με το αν η τιμή αυτή βρίσκεται ανάμεσα στις τιμές $\|n\|^2 - \alpha$ και $\|n\|^2 + \alpha$ ο αλγόριθμος τερματίζει ή επαναλαμβάνεται με α μια μεταβλητή που καθορίζει την ακρίβεια. Πλέον έχουμε την τελική προσέγγιση για την ανακτημένη εικόνα.

Υλοποίηση

Διαβάζουμε την ίδια εικόνα με προηγουμένως και εφαρμόζουμε τα ίδια φίλτρα Average και Gaussian (θορύβου) με τις ίδιες παραμέτρους. Δημιουργούμε τον πίνακα P και τον επεκτείνουμε όπως περιεγράφηκε παραπάνω. Υπολογίζουμε και πάλι καθεμία από τις απαραίτητες μεταβλητές της εξίσωσης και επιπλέον τις ποσότητες r, $\varphi(\gamma)$, $\|n\|^2$. Η υλοποίηση του παραπάνω αλγορίθμου ήταν απλή και επιτεύχθηκε με ένα βρόγχο επανάληψης while. Υπενθυμίζουμε ότι ο αλγόριθμος χρησιμοποιήθηκε συνολικά τρεις φορές, μια για κάθε διασπορά θορύβου στην εικόνα που επιλέξαμε.

Μετάβαση στον κώδικα MATLAB

ΑΠΟΤΕΛΕΣΜΑΤΑ & ΣΥΜΠΕΡΑΣΜΑΤΑ

Παρακάτω παρουσιάζουμε τα αποτελέσματα, όπως αυτά παράχθηκαν από το Matlab.

ΑΡΧΙΚΗ ΕΙΚΟΝΑ GRAYSCALE

ΕΙΚΟΝΑ ΜΕ ΘΟΡΥΒΟ GAUSSIAN $(\sigma^2 = 0.001)$

ΕΙΚΟΝΑ ΜΕ ΘΟΡΥΒΟ GAUSSIAN $(\sigma^2 = 0.005)$

ΕΙΚΟΝΑ ΜΕ ΘΟΡΥΒΟ GAUSSIAN $(\sigma^2 = 0.01)$

WIENER FILTER
$(\sigma^2 = 0.001)$
MSE = 1135

WIENER FILTER
$(\sigma^2 = 0.005)$
MSE = 660.53

WIENER FILTER
$(\sigma^2 = 0.01)$
MSE = 658.21

CLSR FILTER
($\sigma^2 = 0.001$)
MSE = 821.14

CLSR FILTER
($\sigma^2 = 0.005$)
MSE = 498.81

CLSR FILTER
($\sigma^2 = 0.01$)
MSE = 431.85

Σύμφωνα με τις τιμές των τετραγωνικών σφαλμάτων παρατηρούμε ότι η τεχνική CLSR έχει αποκαταστήσει πιο αποτελεσματικά την εικόνα σε κάθε περίπτωση της διασποράς θορύβου.

ΠΑΡΑΡΤΗΜΑ – ΚΩΔΙΚΑΣ

Σε περίπτωση που χρειαστεί ο κώδικας τον παραθέτουμε παρακάτω:

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% BASIC INFORMATION                                   %
% Course: Digital Image Processing - Lab 4            %
% Deadline: 07-05-2019                                %
% LAB31239720:  Pantelis Karamailis, 2016030040       %
%               Kostantinos Vlachos, 2016030042       %
%               Mixalis Galanis,     2016030036       %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

%Clearing things up
clear all;
close all;
clearvars;
%Gathering Image and converting it to grayscale(Resolution: 510x343)
original_image = imread('image.jpg');
grayscale_image = rgb2gray(original_image); %Converting to grayscale
imwrite(grayscale_image, 'grayscale_image.jpg');
%Generating Gaussian Noise with 3 different noise
noises = [0.001 0.005 0.01];
noised_image_1 = imnoise(grayscale_image,'gaussian',0,noises(1));
noised_image_2 = imnoise(grayscale_image,'gaussian',0,noises(2));
noised_image_3 = imnoise(grayscale_image,'gaussian',0,noises(3));
%Blurring image with average filter
imshow(grayscale_image);
h = fspecial('average',[2 2]);
filtered_image_1 = imfilter(noised_image_1,h);
filtered_image_2 = imfilter(noised_image_2,h);
filtered_image_3 = imfilter(noised_image_3,h);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 1 - WIENER RESTORATION FILTER                        %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
%Calculating Fourier Transforms of filter
[horizontal_pixels, vertical_pixels] = size(grayscale_image);
H = fft2(h,horizontal_pixels,vertical_pixels);
H_conj = conj(H);
%Calculating Fourier Transforms of Degraded Image
G_1 = fft2(filtered_image_1);
G_2 = fft2(filtered_image_2);
G_3 = fft2(filtered_image_3);
S_f = abs(fft2(grayscale_image)).^2;
%Calculating Fourier Transforms of Estimated Image
gamma = 10^7;
F_1_1 = (H_conj ./ (abs(H).^2 + gamma .* (noises(1) ./ S_f))).* G_1;
F_1_2 = (H_conj ./ (abs(H).^2 + gamma .* (noises(2) ./ S_f))).* G_2;
F_1_3 = (H_conj ./ (abs(H).^2 + gamma .* (noises(3) ./ S_f))).* G_3;
%Estimated Image
IFT_1_1 = ifft2(F_1_1);
IFT_1_2 = ifft2(F_1_2);
IFT_1_3 = ifft2(F_1_3);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% 2 - CONSTRAINED LEAST SQUARE RESTORATION FILTER     %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
P = [0 -1 0; -1 4 -1;0 -1 0];
M = horizontal_pixels + 2;
N = vertical_pixels + 2;
P_Extended = zeros(M,N);
P_Extended((M/2-1):(M/2+1),(N/2-1):(N/2+1)) = P;
P_F_T = fft2(P_Extended, M, N);
H = fft2(h,M,N);
H_conjugate = conj(H);

G_1 = fft2(filtered_image_1,M,N);
gamma_1 = 5;
norm_est_1 = norm((M-1)*(N-1)*(0 + noises(1)));
a_1 = norm_est_1 * 0.02;
while(1)
    F_1 = (H_conjugate./(abs(H).^2 + gamma_1*(abs(P_F_T).^2))).*G_1;
    f_1 = ifft2(F_1);
    R_1 = G_1 - H.*F_1;
    r_1 = ifft2(R_1);
    f_g_1 = norm(r_1);
    if(f_g_1<(norm_est_1 - a_1))
        gamma_1 = gamma_1 + 0.05 * gamma_1;
    end
    if (f_g_1>(norm_est_1 + a_1))
        gamma_1 = gamma_1 - 0.05 * gamma_1;
    end
    if((norm_est_1 - a_1) < f_g_1 && f_g_1 < (norm_est_1 + a_1))
        break;
    end
end

G_2 = fft2(filtered_image_2,M,N);
gamma_2 = 5;
```

```
norm_est_2 = norm((M-1)*(N-1)*(0 + noises(2)));
a_2 = norm_est_2 * 0.02;
while(1)
    F_2 = (H_conjugate./(abs(H).^2 + gamma_2*(abs(P_F_T).^2))).*G_2;
    f_2 = ifft2(F_2);
    R_2 = G_2 - H.*F_2;
    r_2 = ifft2(R_2);
    f_g_2 = norm(r_2);
    if(f_g_2<(norm_est_2 - a_2))
        gamma_2 = gamma_2 + 0.05 * gamma_2;
    end
    if (f_g_2>(norm_est_2 + a_2))
        gamma_2 = gamma_2 - 0.05 * gamma_2;
    end
    if((norm_est_2 - a_2) < f_g_2 && f_g_2 < (norm_est_2 + a_2))
        break;
    end
end

G_3 = fft2(filtered_image_3,M,N);
gamma_3 = 5;
norm_est_3 = norm((M-1)*(N-1)*(0 + noises(3)));
a_3 = norm_est_3 * 0.02;
while(1)
    F_3 = (H_conjugate./(abs(H).^2 + gamma_3*(abs(P_F_T).^2))).*G_3;
    f_3 = ifft2(F_3);
    R_3 = G_2 - H.*F_3;
    r_3 = ifft2(R_3);
    f_g_3 = norm(r_3);
    if(f_g_3<(norm_est_3 - a_3))
        gamma_3 = gamma_3 + 0.05 * gamma_3;
    end
    if (f_g_3>(norm_est_3 + a_3))
        gamma_3 = gamma_3 - 0.05 * gamma_3;
    end
    if((norm_est_3 - a_3) < f_g_3 && f_g_3 < (norm_est_3 + a_3))
        break;
    end
end

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% SUMMARY - COMPARING IMAGES AND CALCULATING MSE's    %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Displaying and comparing noise images
subplot(4,1,1); imshow(grayscale_image); title("ORIGINAL GRAYSCALE IMAGE");
subplot(4,3,4); imshow(noised_image_1); title("AVERAGE FILTERED - GAUSSIAN
NOISED IMAGE (0.001)");
subplot(4,3,5); imshow(noised_image_2); title("AVERAGE FILTERED - GAUSSIAN
NOISED IMAGE (0.005)");
subplot(4,3,6); imshow(noised_image_3); title("AVERAGE FILTERED - GAUSSIAN
NOISED IMAGE (0.01)");
subplot(4,3,7); imshow(uint8(IFT_1_1)); title("WIENER AVERAGE FILTERED -
GAUSSIAN NOISED IMAGE (0.001)");
```

```
subplot(4,3,8); imshow(uint8(IFT_1_2)); title("WIENER AVERAGE FILTERED - GAUSSIAN NOISED IMAGE (0.005)");
subplot(4,3,9); imshow(uint8(IFT_1_3)); title("WIENER AVERAGE FILTERED - GAUSSIAN NOISED IMAGE (0.01)");
subplot(4,3,10); imshow(uint8(f_1)); title("CLSR AVERAGE FILTERED - GAUSSIAN NOISED IMAGE (0.001)");
subplot(4,3,11); imshow(uint8(f_2)); title("CLSR AVERAGE FILTERED - GAUSSIAN NOISED IMAGE (0.005)");
subplot(4,3,12); imshow(uint8(f_3)); title("CLSR AVERAGE FILTERED - GAUSSIAN NOISED IMAGE (0.01)");


%Calculating Mean Squared Errors
[horizontal_pixels,vertical_pixels] = size(grayscale_image);
mse_1 = immse(grayscale_image, uint8(IFT_1_1));
mse_2 = immse(grayscale_image, uint8(IFT_1_2));
mse_3 = immse(grayscale_image, uint8(IFT_1_3));
mse_4 = immse(grayscale_image, uint8(f_1(2:end-1, 2:end-1)));
mse_5 = immse(grayscale_image, uint8(f_2(2:end-1, 2:end-1)));
mse_6 = immse(grayscale_image, uint8(f_3(2:end-1, 2:end-1)));
```

Μετάβαση στα αποτελέσματα της άσκησης