ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΚΑΙ ΑΝΑΛΥΣΗ ΕΙΚΟΝΑΣ

ΕΡΓΑΣΤΗΡΙΑΚΕΣ ΑΣΚΗΣΕΙΣ, ΜΕΡΟΣ Α

ΓΕΩΡΓΙΟΣ ΠΕΤΡΑΚΗΣ

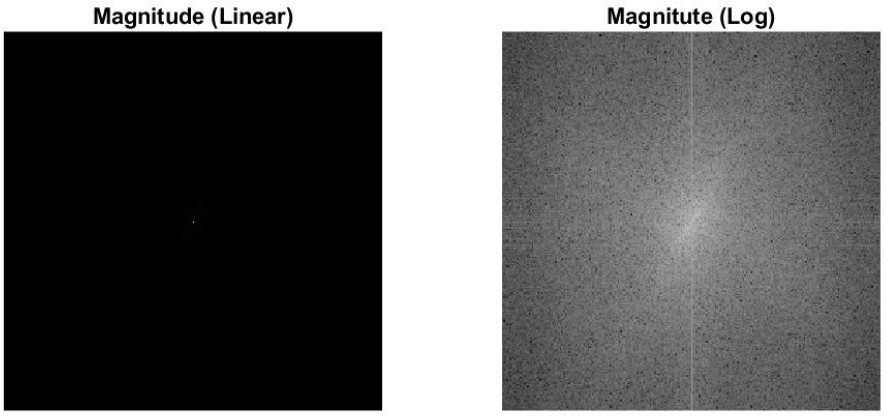
ΑΜ 1066516

5^{Ο} ΕΤΟΣ ΗΜΤΥ

Οι κώδικες για όλες τις ασκήσεις παρατίθενται σε μορφή στιγμιότυπων στο τέλος της αναφοράς

ΑΣΚΗΣΗ 1.1: Φιλτράρισμα στο πεδίο συχνοτήτων

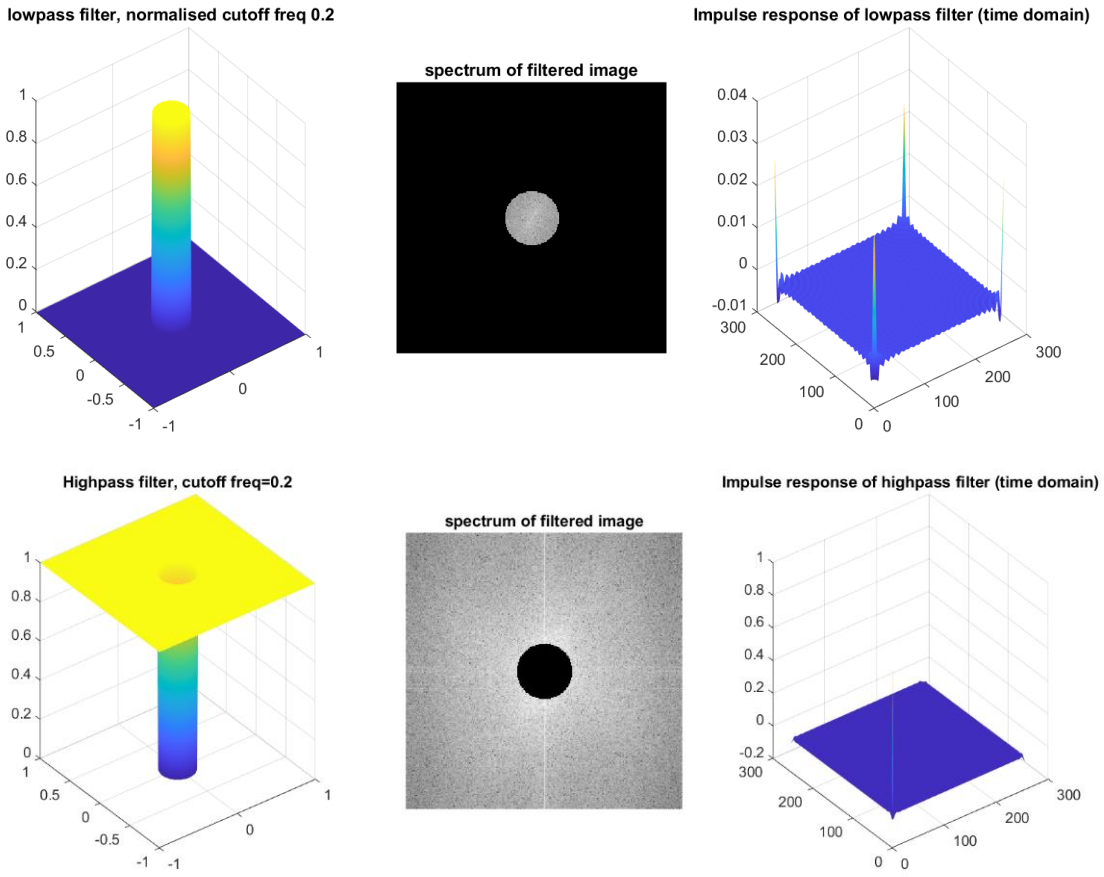
1) Γραμμική και Λογαριθμική απεικόνιση του πλάτους του 2D FFT της “aerial.tiff”



Σημείωση: Πριν την απεικόνιση γίνεται scaling μέσω της συνάρτησης mat2gray() και fftshift() για μεταφορά του συχνοτικού σημείου στο κέντρο της εικόνας. Σε σχόλια παρατίθεται και κώδικας κατά τον οποίο γίνεται η μεταφορά manually με πολλαπλασιασμό των στοιχείων της εικόνας με τον όρο (-1)^(x+y) πριν τον

υπολογισμό του fft. Για τον υπολογισμό του πλάτος φάσματος, αρχικά χρησιμοποιούμε την fft2() και έπειτα την συνάρτηση abs()

2-5) Φιλτράρισμα της εικόνας με κατωπερατό και ανωπερατό φίλτρο και απεικόνιση του φάσματος και της κρουστικής απόκρισης των φίλτρων



Επέλεξα να χρησιμοποιήσω ιδανικά φίλτρα τα οποία κατασκεύασα στη συχνότητα με χρήση της συνάρτησης freqspace και meshgrid. Αρχικά, δημιούργησα τα 2 διανύσματα f1 και f2 μεγέθους 256 το καθένα, όσο και οι 2 διαστάσεις της εικόνας, με την freqspace η οποία επιστρέφει κανονικοποιημένες τιμές συχνότητας στο διάστημα [-1,1]. Με την meshgrid επέκτεινα κατά στήλες/γραμμές τα διανύσματα για να δημιουργηθεί το 2D grid στο οποίο θα οριστεί και θα σχεδιαστεί το φίλτρο. Επέλεξα τυχαίο cutoff frequency και για τα δύο φίλτρα την (κανονικοποιημένη) συχνότητα 0.2 που ορίζεται ως απόσταση από το σημείο 0. Οπότε, για να κατασκευάσω τα φίλτρα θέτω τις τιμές μιας συνάρτησης h ίσες με 1 όπου η απόσταση sqrt(f1^2+f2^2) είναι μικρότερη του 0.2 για το κατωπερατό και 0 αλλού, και αντίστροφα για το ανωπερατό φίλτρο.

Για να εφαρμόσω τα φίλτρα απλώς πολλαπλασιάζω το πλάτος φάσματος με την συνάρτηση που υπολόγισα. Παρατηρώ ότι το φάσμα, εκτός από τα αποκομμένα μέρη τα οποία είναι εντελώς μαύρα, είναι φωτεινότερο σε σχέση με το φάσμα του πρώτου ερωτήματος. Πιστεύω ότι αυτό έχει να κάνει με το scaling που δέχεται η εικόνα του φάσματος από την imshow. Το φάσμα δεν είχε πριν απόλυτα μηδενικές τιμές, οπότε πιθανόν οι τιμές που παραμένουν ίδιες μετά την εφαρμογή του φίλτρου να ανατίθενται τώρα σε μεγαλύτερες τιμές εντάσεων.

Για τις κρουστικές αποκρίσεις υπολογίζω τον αντίστροφο μετασχηματισμό με την συνάρτηση ifft2. Επέλεξα να επιστρέψω τις κρουστικές αποκρίσεις στο σημείο (0,0) χρησιμοποιώντας ifftshift πριν τον υπολογισμό του αντίστροφου μετασχηματισμού, εφόσον το συχνοτικό σημείο των φίλτρων βρίσκεται εκ κατασκευής στο (0,0).

ΑΣΚΗΣΗ 1.2: Συμπίεση Εικόνας με χρήση του μετασχηματισμού DCT

Για εφαρμογή του μετασχηματισμού DCT ανά block εικόνας μεγέθους 32x32, αρχικά ορίζουμε τους πίνακες DCT 32x32 καλώντας την συνάρτηση dctmtx(32).

Προτού εφαρμόσω τον μετασχηματισμό, παρατηρώ ότι οι διαστάσεις της εικόνας (για πολύ λίγο) δεν είναι ακέραιο πολλαπλάσιο του μεγέθους του block, οπότε εφαρμόζω ένα resize της εικόνας στις διαστάσεις 256x256 με την συνάρτηση imresize. Αυτό το βήμα δεν είναι απαραίτητο ούτε αλλάζει την λειτουργικότητα του κώδικα, και θα μπορούσε να επιλυθεί με padding ακόμα και κατά την διαδικασία εφαρμογής του μετασχηματισμού ανά block.

Για την εφαρμογή του μετασχηματισμού ανά block, εντόπισα την συνάρτηση blockproc(), η οποία δέχεται ως όρισμα έναν πίνακα (την εικόνα), ένα μέγεθος μπλοκ και μία συνάρτηση η οποία δέχεται ως όρισμα ένα block_struct και την οποία εφαρμόζει ανά block εικόνας. Η συνάρτηση αυτή μας επιτρέπει να ορίσουμε τον πολλαπλασιασμό που γίνεται σε κάθε block εικόνας με τους πίνακες DCT και μας απαλλάσσει από το να «τεμαχίσουμε» την εικόνα manually για να εφαρμόσουμε τον μετασχηματισμό.

1. Εφαρμογή μεθόδου ζώνης για συμπίεση

Για την μέθοδο ζώνης ξεκινάμε ορίζοντας έναν πίνακα από άσους και έπειτα κάνοντας padding με μηδενικά στο μέγεθος 32x32. Το padding γίνεται με την

συνάρτηση padarray(), περνώντας επίσης την παράμετρο “post”, έτσι ώστε να κρατάμε κάθε φορά τους συντελεστές που βρίσκονται στο πάνω αριστερά μέρος του block, οι οποίοι γνωρίζουμε ότι αποτυπώνουν το μεγαλύτερο ποσοστό πληροφορίας/ενέργειας της εικόνας. Για να κρατάμε ένα ποσοστό συντελεστών στο διάστημα [5, 50]% ξεκινάμε από ένα block μεγέθους 4x4 και καταλήγουμε σε μέγεθος 21x21. Το ποσοστό των συντελεστών που διατηρούμε μετά από την εφαρμογή της μάσκας είναι R^2/32^2, όπου RxR το μέγεθος της μάσκας-ζώνης. Για εφαρμογή της μάσκας, χρησιμοποιούμε ξανά την συνάρτηση blockproc() στην εικόνα του μετασχηματισμού αυτή τη φορά.

Μετά την εφαρμογή της μάσκας, εφαρμόζουμε με παρόμοια διαδικασία τον αντίστροφο μετασχηματισμό για αποσυμπίεση, και έπειτα με την συνάρτηση immse() υπολογίζουμε την τιμή MSE μεταξύ της ανακατασκευασμένης εικόνας και της αρχικής εικόνας, για κάθε μέγεθος μάσκας, δηλαδή ποσοστό συμπίεσης.

2. Μέθοδος κατωφλίου

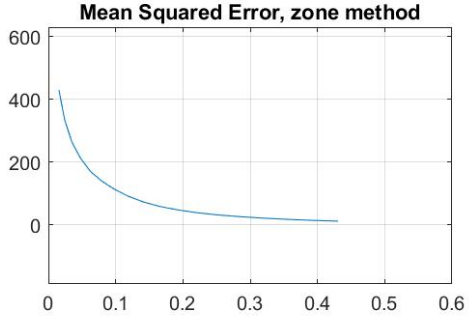
Η εφαρμογή αυτής της μεθόδου είναι απλούστερη. Για την συμπίεση μηδενίζουμε τις τιμές των συντελεστών DCT στην μετασχηματισμένη εικόνα, οι οποίες βρίσκονται κάτω από κάποιο κατώφλι. Αρχίζουμε από χαμηλό κατώφλι, όπου έχουμε χαμηλή συμπίεση, και φτάνουμε σε επίπεδο συμπίεσης κοντά στο 50% με υψηλό κατώφλι. Έπειτα ανακατασκευάζουμε την εικόνα με τον αντίστροφο μετασχηματισμό παρόμοια με την προηγούμενη περίπτωση και υπολογίζουμε την τιμή MSE σχετικά με την αρχική εικόνα..

Από τα αποτελέσματα παρατηρούμε ότι η μέθοδος κατωφλίου είναι σημαντικά αποδοτικότερη από την μέθοδο ζώνης. Παρουσιάζονται και ενδεικτικά δύο εικόνες συμπιεσμένες με τις διαφορετικές μεθόδους. Για ευκολότερη παρουσίαση, εμφανίζονται τα διαγράμματα ξανά σχεδιασμένα στο διάστημα 5%-20%, δηλαδή σε υψηλότερα ποσοστά συμπίεσης, και σχεδιασμένα στο ίδιο διάγραμμα.

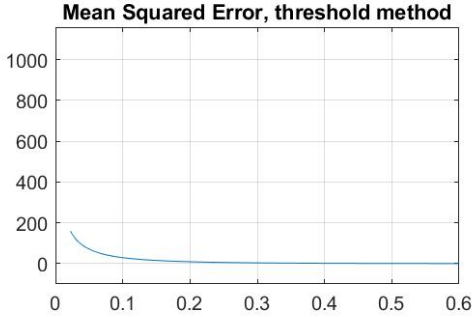
zone mask, r=0.43

threshold mask, r=0.022

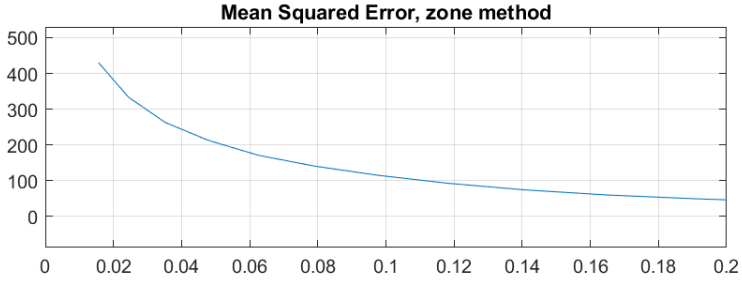
Mean Squared Error, zone method



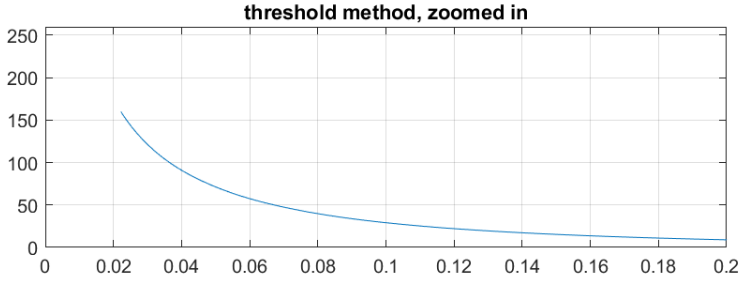
Mean Squared Error, threshold method



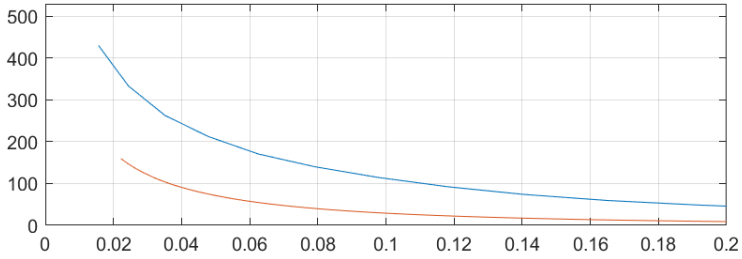
Mean Squared Error, zone method



threshold method, zoomed in



Zone: blue, Threshold: red



ΑΣΚΗΣΗ 1.3: ΒΕΛΤΙΩΣΗ ΕΙΚΟΝΑΣ – ΦΙΛΤΡΑΡΙΣΜΑ ΘΟΡΥΒΟΥ

Αρχικά, φορτώνω την εικόνα στην Matlab και έπειτα χρησιμοποιώ την συνάρτηση im2double για scaling στο διάστημα [0,1].

Έπειτα δημιουργείται ένα φίλτρο κινούμενου μέσου όρου με την συνάρτηση fspecial. Η εφαρμογή του φίλτρου κινούμενου μέσου όρου γίνεται με χρήση της συνάρτησης imfilter. Η εφαρμογή του αντίστοιχου φίλτρου μεσαίου γίνεται με την συνάρτηση medfilt2.

Στον κώδικα παρουσιάζονται τα αποτελέσματα για μέγεθος φίλτρου 3x3. Η αύξηση των διαστάσεων των φίλτρων προκαλούσε αλλοίωση της ποιότητας των αποτελεσμάτων, ώσπου τα φίλτρα μεγέθους 11x11 μείωναν το SNR των φιλτραρισμένων εικόνων σε επίπεδο χειρότερο από εκείνο των θορυβωδών εικόνων.

1. Λευκός θόρυβος

Προσθέτω λευκό θόρυβο μηδενικής μέσης τιμής και διασποράς σ=0.008. Η τιμή της διασποράς υπολογίστηκε πειραματικά.

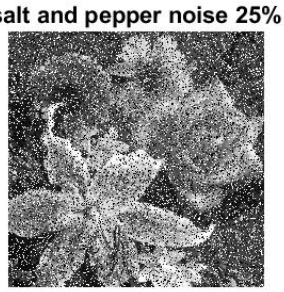
Τα αποτελέσματα εφαρμογής των φίλτρων παρουσιάζονται παρακάτω. Ενδεικτικά στο πρόγραμμα φαίνονται και οι τιμές SNR των φιλτραρισμένων εικόνων.

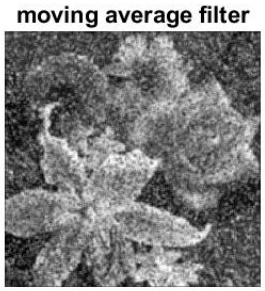
Η διαφορά μεταξύ των δύο φίλτρων δεν είναι ευδιάκριτη, ούτε καν με σύγκριση των SNR των δύο αποτελεσμάτων. Με οπτική παρατήρηση, συμπεραίνω ότι το φίλτρο κινούμενου μέσου όρου εξομαλύνει τον θόρυβο, χωρίς να επηρεάζει τόσο τις ακμές των αντικειμένων της εικόνας και χωρίς να προκαλεί απώλειες πληροφορίας. Αντίθετα, το φίλτρο κινούμενου μέσου φαίνεται να προκαλεί αλλοίωση των περιγραμμάτων των εικόνων. Έτσι, παρόλο που δείχνει να αφαιρεί τον θόρυβο, προκαλεί απώλεια πληροφορίας. Αυτό γίνεται εμφανέστερο από το λουλούδι κάτω αριστερά, όπου οι κηλίδες στο άνθος φαίνεται να συντίθενται μεταξύ τους μετά την εφαρμογή του φίλτρου. Η

2. Κρουστικός θόρυβος (Salt & Pepper)

Για την εφαρμογή του θορύβου σε ποσοστό 25% χρησιμοποιούμε ξανά την συνάρτηση imnoise. Έπειτα εφαρμόζουμε τα φίλτρα με παρόμοιο τρόπο όπως στο προηγούμενο παράδειγμα.

Εδώ οι διαφορές είναι εμφανείς. Το φίλτρο κινούμενου μέσου όρου είναι έχει αισθητά χειρότερη απόδοση από ότι το φίλτρο μέσου όρου.





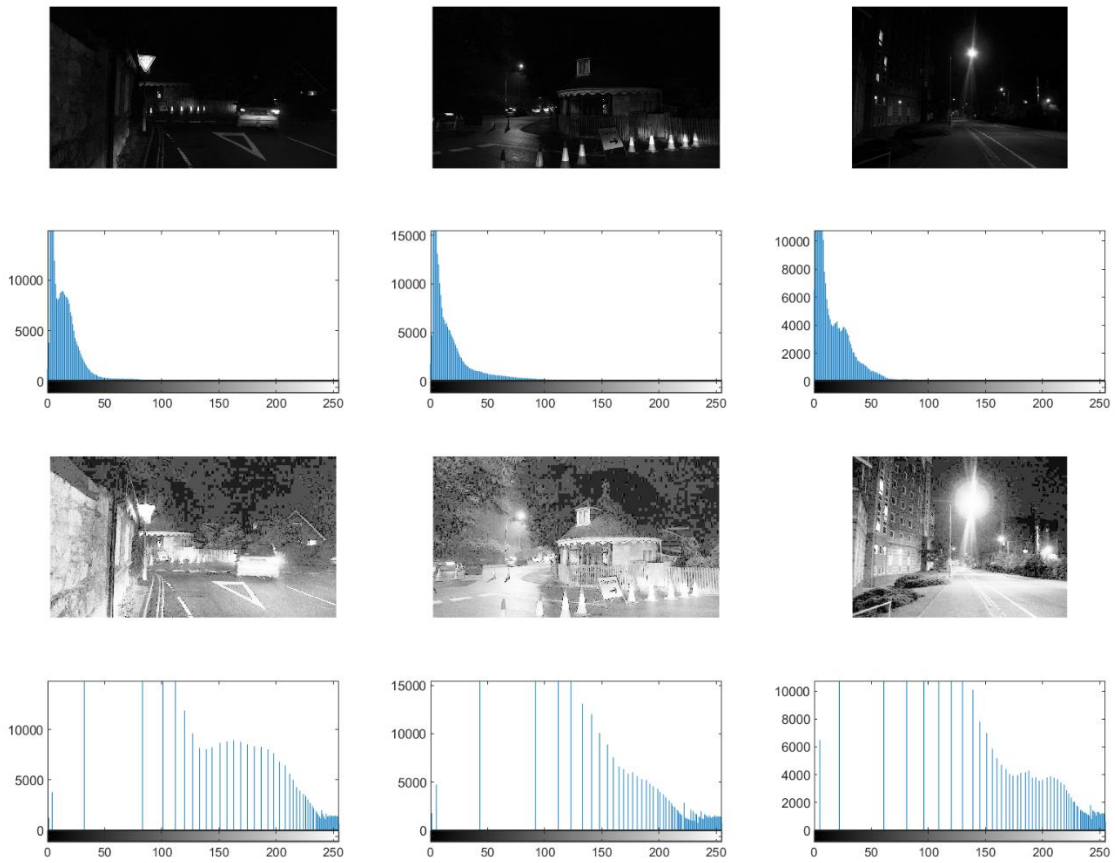
ΑΣΚΗΣΗ 1.4: ΒΕΛΤΙΩΣΗ ΕΙΚΟΝΑΣ – ΕΞΙΣΩΣΗ ΙΣΤΟΓΡΑΜΜΑΤΟΣ

1) Υπολογισμός Ιστογραμμάτων – Εφαρμογή ολικής εξίσωσης

Για τον υπολογισμό των ιστογραμμάτων χρησιμοποιήθηκε η συνάρτηση imhist, αφού πρώτα οι τιμές της έντασης των εικόνων μεταφέρθηκαν στο διάστημα [0 1] με την im2gray.

Για την εφαρμογή της ολικής εξίσωσης, αρχικά υπολογίστηκε η αθροιστική κατανομή των ιστογραμμάτων και έπειτα η τιμή του μετασχηματισμού T μεταξύ εντάσεων. Ο μετασχηματισμός T δέχεται ως όρισμα μία τιμή έντασης του ιστογράμματος και την αναθέτει σε μία νέα τιμή ώστε να σχηματίσει το εξισωμένο ιστόγραμμα.

Τα αποτελέσματα εμφανίζονται παρακάτω και έπειτα σχολιάζονται



Όπως ήταν αναμενόμενο, εφόσον το μεγαλύτερο μέρος των εικόνων καλύπτεται από σκούρο χρώμα, λόγω της έλλειψης φωτισμού, τα ιστογράμματα είναι έντονα συγκεντρωμένα προς τις χαμηλές εντάσεις.

Μετά την εφαρμογή του μετασχηματισμού, τόσο οι εικόνες όσο και τα διαγράμματα έχουν αλλάξει αρκετά μορφή. Τα διαγράμματα φαίνονται να έχουν εξαπλωθεί ομοιόμορφα σε όλο το εύρος τιμών εντάσεων. Η εφαρμογή την εξίσωσης στα ιστογράμματα έχει ενισχύσει τις αντιθέσεις των εικόνων και έχει φανερώσει λεπτομέρειες οι οποίες δεν ήταν πριν εμφανείς, λόγω χαμηλής έντασης.

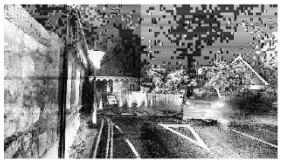
2) Τοπική εξίσωση ιστογραμμάτος

Για συντομία στην εφαρμογή της τοπικής εξίσωσης, χρησιμοποίησα την εξίσωση histeq της Matlab και την συνάρτηση blockproc για να την εφαρμόσω ανά block των εικόνων. Αντί για την συνάρτηση histeq, θα μπορούσε να δημιουργηθεί μία νέα συνάρτηση που θα δεχόταν ως είσοδο ένα block εικόνας και θα υπολόγιζε και θα εφάρμοζε τον μετασχηματισμό T, όπως ακριβώς στο προηγούμενο ερώτημα για την ολική εξίσωση ιστογράμματος.

Σχετικά με το μέγεθος των «παραθύρων» που εφαρμόζω την τοπική εξίσωση ιστογράμματος, παρατήρησα ότι τα καλύτερα αποτελέσματα είχαν τα μεγέθη που ήταν είτε πλατιά (μεγάλος αριθμός στοιχείων στον άξονα x) είτε ψηλά (μεγάλος αριθμός στοιχείων στον άξονα y). Η διαφορά που παρατήρησα μεταξύ των 2 διαφορετικών περιπτώσεων είναι ότι στην πρώτη περίπτωση αναδεικνύονται καλύτερα οι λεπτομέρειες των αντικειμένων της εικόνας, όμως στην δεύτερη περίπτωση η συνολική ποιότητα της εικόνας είναι καλύτερη. Τέλος, τετραγωνικά παράθυρα (ή παράθυρα με περίπου ίσες τις 2 διαστάσεις) προκαλούσαν artefacts και «αποτύπωναν» τα περιγράμματα τους έντονα στην τελική εικόνα.

Παρακάτω παρουσιάζονται τα αποτελέσματα για παράθυρο μεγέθους [4 400]



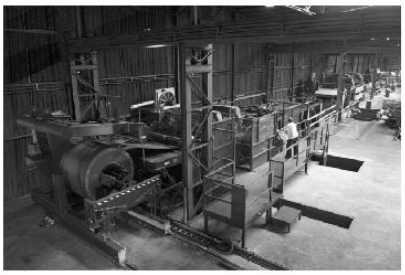
Τα αποτελέσματα δεν είναι ιδιαίτερα εμφανή εδώ, αλλά η τοπική εξίσωση παρόλο που προκαλεί θόρυβο στην τελική εικόνα, φανερώνει περισσότερες λεπτομέρειες αντικειμένων σε σχέση με την ολική εξίσωση

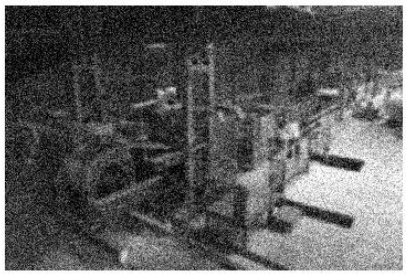
ΑΣΚΗΣΗ 1.5: Αποκατάσταση εικόνας (Wiener Deconvolution)

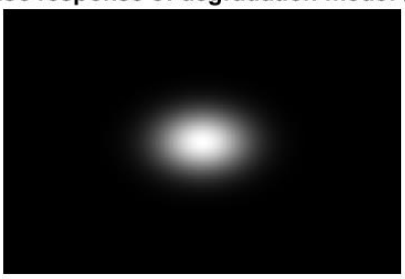
Με την συνάρτηση imfilter παράγουμε gaussian φίλτρο με διασπορά 2. Η συνάρτηση imfilter δίνει φίλτρο το οποίο είναι ήδη κεντραρισμένο στο (0,0) στη συχνότητα, οπότε προτού εμφανίσουμε τα αποτελέσματα της εφαρμογής του φίλτρου υποβάθμισης, θα πραγματοποιήσουμε ifftshift στις φιλτραρισμένες εικόνες, παρόλο που δεν πραγματοποιήσαμε ποτέ fftshift. Το shift αυτό είναι απαραίτητο λόγω της μετατόπισης φάσης που προκαλεί το φίλτρο, η οποία μεταφράζεται σε χωρική μετατόπιση από τον ifft2. Η αιτιολόγηση αυτή δίνεται και στον κώδικα υπό μορφή σχολίων.

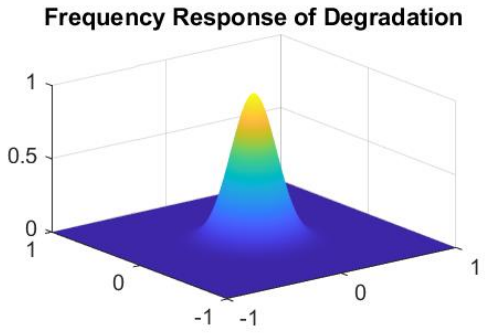
Έπειτα εφαρμόζουμε προσθετικό λευκό θόρυβο, διασποράς σ=0.015 μετρώντας πειραματικά τον λόγο SNR.

Η αρχική εικόνα, η θορυβώδης εικόνα, η point spread function και το φάσμα του φίλτρου φαίνονται παρακάτω









Όπως αναφέρθηκε παραπάνω, φαίνεται ότι η point spread function είναι μετατοπισμένη στον χώρο και το φάσμα είναι κεντραρισμένο στη συχνότητα. Η μετατόπιση αυτή στο χώρο παράγει μία γραμμική μετατόπιση φάσης στη συχνότητα, η οποία μεταφέρεται και στην εικόνα κατά την εφαρμογή του φίλτρου.

1. Φίλτρο Wiener + Inverse Filter

Για την κατασκευή του φίλτρου Wiener υπολογίζουμε τα απαραίτητα φάσματα και τις φασματικές πυκνότητες, της θορυβώδους εικόνας και του θορύβου. Η φασματική πυκνότητα της αρχικής εικόνας εκτιμάται από την διαφορά της αντίστοιχης υποβαθμισμένης μείον της φασματικής πυκνότητας του θορύβου. Οι παραπάνω φασματικές πυκνότητες χρησιμοποιούνται για την κατασκευή του

φίλτρου στη συχνότητα, το οποίο έπειτα πολλαπλασιάζεται με την θορυβώδης εικόνα. Παρατίθεται σύντομο κομμάτι κώδικα, όπου φαίνεται η εξίσωση του φίλτρου στη συχνότητα.

```
%kataskeui filtrou wiener
%ipologismos twn fasmatikwn piknotitwn
n_fact=420*630; %gia diairesh me plithos pixel
N=fft2(additive_noise); %noise fft
G=fft2(noisy); %degraded+noisy image fft
F=fft2(img); %original image fft
Pn=abs(N.*N)./n_fact; %noise PSD
Pg=abs(G.*G)./n_fact; %degraded+noisy image PSD
%Pf=abs(F.*F); %original image PSD
Pf=Pg-Pn; %estimated original image PSD
Ph=abs(H.*H)./n_fact; %degradation function PSD


%W=Sh./(Sh+Pn./Ps);
W=Pf./(Pf+Pn);

F=W.*G;
wiener_img=ifft2(F);
```

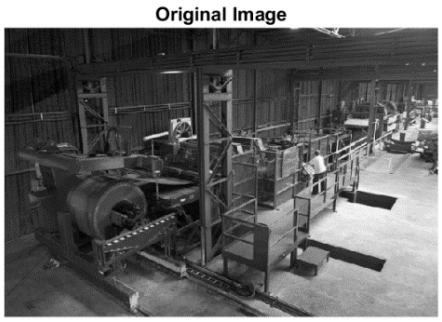
Έπειτα, κατασκευάζεται το αντίστροφο φίλτρο, όπως φαίνεται παρακάτω. Το πρόβλημα της διαίρεσης με το μηδέν αντιμετωπίζεται αντικαθιστώντας τα στοιχεία inf της Matlab με την μέγιστη τιμή του πίνακα που δεν είναι άπειρη.

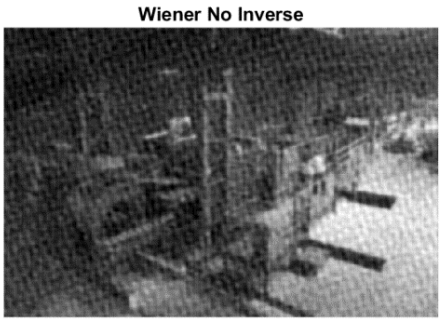
```
%efarmogi inverse filter
IF=1./H;
IF(isinf(IF))=max(IF(~isinf(IF))); %gia diaireseis me to mhden.
% antikathistw ta inf me ton megalitero arithmo tou pinaka pou den einai inf
F_final=IF.*F;
final_img=ifft2(F_final);
```

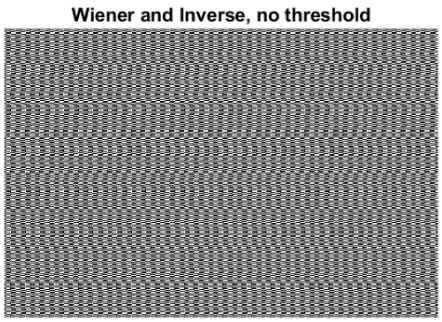
Τέλος εφαρμόζεται και αυτό το φίλτρο στην εικόνα πολλαπλασιαστικά και υπολογίζεται ο αντίστροφος μετασχηματισμός.

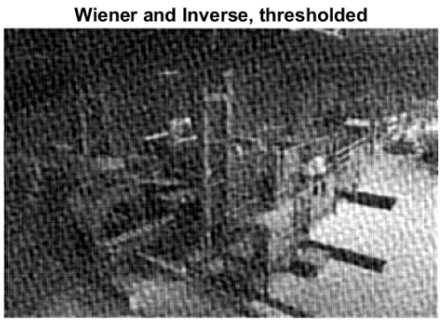
Η εφαρμογή κατωφλίου γίνεται όπως στον τύπο των διαφανειών του μαθήματος στην εξίσωση του αντίστροφου φίλτρου. Το κατώφλι υπολογίστηκε επίσης

πειραματικά στην τιμή 2. Τα αποτελέσματα φαίνονται παρακάτω.









Όπως αναμενόταν, η εφαρμογή του αντίστροφου φίλτρου χωρίς κατώφλι προκαλεί αλλοίωση του περιεχομένου της εικόνας. Μετά την χρήση κατωφλίου στο φίλτρο, παρατηρούμε ότι η εικόνα έχει ορισμένα artefacts αλλά οι αντιθέσεις φαίνονται καλύτερα από ότι στην υποβαθμισμένη εικόνα.

3) Wiener Deconvolution

Για την υλοποίηση της αποσυνέλιξης, χρησιμοποίησα την συνάρτηση deconvwnr της Matlab. Η συνάρτηση αυτή δέχεται την εικόνα, την PSF της υποβάθμισης (κρουστική απόκριση) και το NSR του θορύβου.

Τα αποτελέσματα εμφανίζονται παρακάτω

Η αποσυνέλιξη είναι αποτελεσματική στην αποκατάσταση της εικόνας, αλλά έχει ελαφρώς χειρότερη απόδοση από ότι η προηγούμενη μέθοδος. Αυτό συμβαίνει επειδή η αποσυνέλιξη δεν εκμεταλλεύεται πληροφορία σχετικά με την φασματική πυκνότητα του θορύβου, αλλά αντιμετωπίζει το σύστημα υποβάθμισης ως έναν γραμμικό μετασχηματισμό της αρχικής εικόνας. Βέβαια, η μέθοδος αυτή είναι περισσότερο χρήσιμη σε πραγματικά σενάρια, όπου δεν έχουμε ουσιώδης πληροφορία για την φασματική πυκνότητα του θορύβου.

Στην περίπτωση όπου η φασματική πυκνότητα του θορύβου δεν είναι γνωστή, την υπολογίζουμε από ένα παράθυρο του φάσματος της θορυβώδους εικόνας σε υψηλές συχνότητας.

```
%Stin periptwsi pou den einai gnwsto to PSD, to ipologizoume apo ena
%parathiro ths PSD ths degraded eikonas stis ipsiles sixnotites
%efoson to sixnotiko shmeio vrisketai sto (0,0) theloume na ipologisoume
%thn mesh timh se ena parathiro peripou sto meso tou pinaka (ipsiles
%sixnotites)
mid_x=round(size(Pg,1)/2);
mid_y=round(size(Pg,2)/2);
%se ena parathiro 20x20 ipologizoume tin mesi timi olwn twn stoixeiwn
Pn_unknown_mean=mean(Pg(mid_x-10:mid_x+10,mid_y-10:mid_y+10),'all')
%gia sigrisi me tin mesi timi tou gnwstou noise PSD
Pn_known_mean=mean(Pn,'all')
%efarmogi twn wiener-inverse filtering kai deconvolution
Pn(:)=Pn_unknown_mean;
```

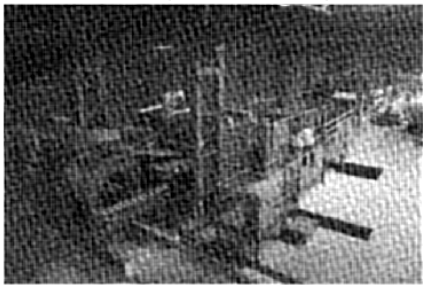
Εφόσον τα φάσματα που υπολογίσαμε πριν δεν είχαν κεντραριστεί, παίρνουμε τιμές γύρω από το κέντρο της φασματικής πυκνότητας της υποβαθμισμένης εικόνας και υπολογίζουμε τη μέση τιμή τους. Αργότερα, ακολουθούμε ακριβώς τις ίδιες

διαδικασίες με την προηγούμενη περίπτωση, απλώς η φασματική πυκνότητα του θορύβου έχει παντού την τιμή που υπολογίσαμε με τον παραπάνω τρόπο. Για λόγους σύγκρισης, κατά την εκτέλεση του προγράμματος υπολογίζεται και η μέση τιμή της φασματικής πυκνότητας από το πραγματικό φάσμα του θορύβου. Οι τιμές διαφέρουν συνήθως στο δεύτερο ή τρίτο δεκαδικό ψηφίο.

Τα αποτελέσματα με άγνωστη την φασματική πυκνότητα του θορύβου φαίνονται παρακάτω

Wiener and Inverse, no threshold

Wiener and Inverse, thresholded, unknown Pn



Η απόδοση φαίνεται να είναι αρκετά ικανοποιητική, συγκριτικά και με την περίπτωση του γνωστού φάσματος θορύβου.

ΑΣΚΗΣΗ 1.6: Ανίχνευση Ακμών

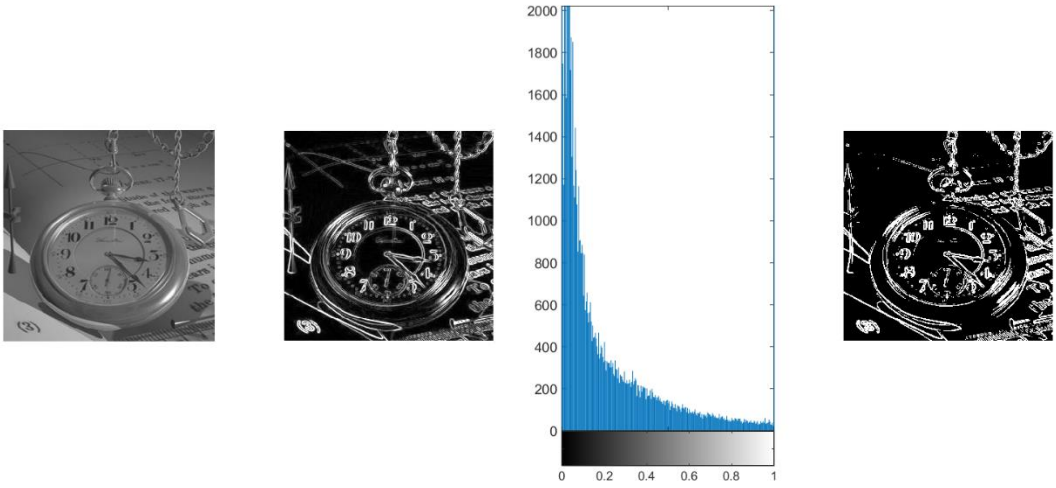
1, 2) Μάσκες Sobel και ολική κατωφλίωση

Ορίζω τους πίνακες των μασκών και εφαρμόζω δισδιάστατη συνέλιξη για τον υπολογισμό των gradient images κατά την διεύθυνση των 2 αξόνων. Από τις τιμές σε κάθε άξονα, χρησιμοποιώ τους παρακάτω τύπους για να λάβω τις τιμές και τις γωνίες των παραγώγων.

$$mag = \sqrt{(gradientx)^2 + (gradienty)^2}$$

$$angle = \arctan\left(\frac{gradienty}{gradientx}\right)$$

Παρακάτω εμφανίζονται η αρχική εικόνα, η εικόνα του πλάτους που παίρνω από τους υπολογισμούς μετά την συνέλιξη και το ιστόγραμμα του πλάτους. Το ιστόγραμμα χρησιμοποιείται για να μαντέψουμε μία τιμή για το κατώφλι που θα θέσουμε στο πλάτος για να ξεχωρίσουμε μεταξύ στοιχείων που ανήκουν και όχι σε ακμές.



Από το ιστόγραμμα παρατηρώ ότι η διάμεσος βρίσκεται περίπου στο διάστημα [0, 0.3]. Επίσης παρατηρώ από την αρχική εικόνα ότι πολλές ακμές διακόπτονται ή δεν ανιχνεύονται, οπότε επιλέγω μια σχετικά χαμηλότερη τιμή στο 0.1 για το κατώφλι. Το αποτέλεσμα φαίνεται στην δεξιότερη εικόνα, η οποία αποτελείται μόνο από εντάσεις 0 και 1.

3) Μετασχηματισμός Hough

Προσπάθησα να εφαρμόσω αναλυτικά τον μετασχηματισμό Hough της εικόνας, αντί να εφαρμόσω την συνάρτηση της Matlab.

Αρχικά, όρισα το πεδίο ρθ, χωρίζοντας τις γωνίες από -90 έως 90 μοίρες και υπολογίζοντας την μέγιστη δυνατή τιμή του ρ ως την μέγιστη διαγώνιο του xy πεδίου. Έπειτα δημιούργησα έναν πίνακα «συσσωρευτή» με τις διαστάσεις του χώρου ρθ.

Για κάθε στοιχείο που έχει τοποθετηθεί προηγουμένως κατά την ολική κατωφλίωση σε ακμή, υπολογίζω κάθε δυνατή ευθεία που περνάει από αυτό το σημείο και αυξάνω το αντίστοιχο σημείο που αντιστοιχεί στην ευθεία αυτή στον «συσσωρευτή». Έτσι, μετά το τέλος του ελέγχου κάθε σημείου ακμής, θα έχω πόσα σημεία περιέχει κάθε δυνατή ευθεία του χώρου ρθ. Ταξινομώντας κατά φθίνων αριθμό σημείων, μπορούμε να επιλέξουμε να σχεδιάσουμε τις ευθείες που

διέρχονται από τον μεγαλύτερο αριθμό ακμών, και είναι πιθανότερο να συμπληρώνουν ακμές που λείπουν από την εικόνα. Για την εξαγωγή των ευθειών από τον ταξινομημένο πίνακα συσσωρευτή, λαμβάνουμε την θέση των 100 πρώτων ευθειών και εξάγουμε τις συντεταγμένες στον χώρο ρθ, οι οποίες ορίζουν και τις αντίστοιχες ευθείες. Έπειτα, για κάθε τιμή του οριζόντιου άξονα, σχεδιάζουμε κάθε μία από αυτές τις ευθείες, υπολογίζοντας την αντίστοιχη συντεταγμένη y από τις τιμές ρ, θ και x. Το αποτέλεσμα του μετασχηματισμού αυτού φαίνεται παρακάτω.

Οι πρώτες γραμμές που σχεδιάζονται είναι όλες κοντά στο περίγραμμα της εικόνας και είναι κατακόρυφες ή οριζόντιες. Αν επιλέξουμε να σχεδιάσουμε τις πρώτες 100 το αποτέλεσμα φαίνεται παρακάτω. Είναι πιθανόν η εικόνα να μην έχει σχεδιαστεί σε σωστή κλίμακα και σχετική θέση με τα αποτελέσματα του μετασχηματισμού ή αντίστροφα. Η υποψία αυτή προέρχεται από το γεγονός ότι υπάρχουν ευθείες του μετασχηματισμού που είναι παράλληλες με τους δείκτες του ρολογιού (οι οποίοι περιέχουν και πολλά σημεία ακμών στην εικόνα του πλάτους), όμως οι ευθείες αυτές δεν ευθυγραμμίζονται σωστά με τους δείκτες.

ΠΑΡΑΡΤΗΜΑ – ΚΩΔΙΚΕΣ MATLAB

ΑΣΚΗΣΗ 1.1

```
clc;
clear;
close all;


%img=imread("moon.jpg");
img=imread("aerial.tiff");
gray_img=double(im2gray(img));
%gray_img=double(rgb2gray(img));
% for i=1:height(gray_img)
%     for j=1:width(gray_img)
%         gray_img(i,j)=gray_img(i,j).*(-1)^(i+j);
%     end
% end

tr_img=fft2(gray_img);
tr_img=fftshift(tr_img); %metafora sixnotikou simeiou sto (0,0)
%task1: apeikonisi platous fasmatos se grammiki kai logarithmiki klimaka
lin_mag=abs(tr_img);
lin_mag=mat2gray(lin_mag); %scaling sto [0 1]
figure(1);
set(gcf, 'Position', [100 600 700 400]);
subplot(1,2,1);
imshow(lin_mag);
title("Magnitude (Linear)");

log_mag=abs(log(tr_img+1));
log_mag=mat2gray(log_mag);
subplot(1,2,2);
imshow(log_mag);
title("Magnitute (Log)");

%task2: dhmiourgia 2d lpf kai filtrarisma
[f1,f2]=freqspace(256,'meshgrid'); %xrhsimopoioume meshgrid gia na dhmiourghsoume to
filtro
z=zeros(256,256); %gia na kathorisoume to cutoff freq
for i=1:256
    for j=1:256
        z(i,j)=sqrt(f1(i,j).^2+f2(i,j).^2); %ipologizoume to metro tou 2D dianismatos
f tis sixnotitas
    end
end
h=zeros(256,256); %to filtro
for u=1:256
    for v=1:256
        if z(u,v)<=0.2 %normalised cutoff frequency 0.2
            h(u,v)=1;
        else
            h(u,v)=0;
        end
    end
end

figure(2);
set(gcf, 'Position', [100 100 1100 400]);
subplot(131);
surf(f1,f2,h);
shading interp;
title("lowpass filter, normalised cutoff freq 0.2");
filtered_img_low=tr_img.*h; %sineliksi ston xrono => pollaplasiasmos stin sixnotita
log_mag=abs(log(filtered_img_low+1));
log_mag=mat2gray(log_mag);
subplot(132)
imshow(log_mag);
title("spectrum of filtered image");

ir_low=ifft2(ifftshift(h)); %metaferw to sixnotiko simeio pisw sto (0,0) protou
efarmosw ifft
subplot(133);
```

```
surf(ir_low);
shading interp;
title("Impulse response of lowpass filter (time domain)");

%highpass filter
hp=ones(256,256)-h; %dhmiourgia ipsiperatou filtrou stin sixnotita me xrisi tou
prohgoumenou filtrou
figure(3);
set(gcf, 'Position',[700 600 1200 400]);
subplot(131);
surf(f1,f2,hp);
shading interp;
title('Highpass filter, cutoff freq=0.2')
filtered_img_high=tr_img.*hp; %efarmogi filtrou
log_mag=abs(log(filtered_img_high+1));
log_mag=mat2gray(log_mag);
subplot(132);
imshow(log_mag,[]);
title('spectrum of filtered image');
ir_high=ifft2(ifftshift(hp));
%ir_high=ifftshift(ir_high);
subplot(133);
surf(ir_high);
title('Impulse response of highpass filter (time domain)');
shading interp;

filtered_img_low=ifftshift(filtered_img_low);
final_img_low=abs(ifft2(filtered_img_low));
filtered_img_high=ifftshift(filtered_img_high);
final_img_high=abs(ifft2(filtered_img_high));

figure(4);
set(gcf, 'Position', [1200 100 600 400]);
subplot(133);
imshow(final_img_high,[]);
title('high-filtered image');
subplot(132);
imshow(final_img_low,[]);
title("low-filtered image");
subplot(131);
imshow(img,[]);
title("starting image");

for i=1:4
    saveas(figure(i),num2str(i)+".png")
end
```

ΑΣΚΗΣΗ 1.2

```
clc;
clear;
close all;

file=load("barbara.mat");
img=file.barbara;
img=imread("lenna.jpg");
img=imresize(img,[256 256]); %mikro transform stin eikona gia na ehoume akrives fit
twn 32x32 block anti gia padding
img=double(rgb2gray(img));
%xrisimopoiw thn blockproc sinartisi gia na efarmosw ton DCT matrix ana
%block 32x32 ths eikonas
dct_m=dctmtx(32); %dhmiourgia pinaka metasxhmatismou DCT megethous 32x32
dct_fun=@(block_struct) dct_m * block_struct.data * dct_m';
dct_im=blockproc(img,[32 32],dct_fun); %efarmogi DCT ana block
%methodos zwnhs: dhmiourgw thn zwnh kai meta kanw padding gia na ftiaksw
%tin maska
msevalues_zone=[];
msevalues_threshold=[];
r_zone=[];
for r=4:21
    zone=ones(r,r);
    r_zone=[r_zone r^2/32^2]; %pososto sintelestwn pou kratame se kathe iteration. ta
plots tha ginoun se auton ton aksona
    mask=padarray(zone,[32-r 32-r],'post');
```

```
    %discard bits according to mask. Se kathe 32x32 block kratame ta bits
    %pou orizei h maska
    compressed_img=blockproc(dct_im,[32 32], @(block_struct) mask.*
block_struct.data);
    inversedct= @(block_struct) dct_m'*block_struct.data*dct_m;
    reconstructed_img=blockproc(compressed_img,[32 32],inversedct);
    msevalues_zone=[msevalues_zone immse(reconstructed_img, img)];
end
%endeiktika kanoume imshow to r=50%
figure(1);
set(gcf,'Position',[100 300 800 500]);
subplot(221);
imshow(reconstructed_img,[]);
title("zone mask, r=0.43");

r_threshold=[]; %pososta
for threshold=0:0.1:80
    compressed_img=dct_im;
    compressed_img(abs(compressed_img)<threshold)=0;
    remaining_coeffs=nnz(compressed_img);
    r_threshold=[r_threshold remaining_coeffs/(256*256)];
    inversedct= @(block_struct) dct_m'*block_struct.data*dct_m;
    reconstructed_img=blockproc(compressed_img,[32 32],inversedct);
    msevalues_threshold=[msevalues_threshold immse(reconstructed_img, img)];
end

subplot(222);
imshow(reconstructed_img,[]);
ttl=sprintf("threshold mask, r=%1.3f",r_threshold(end));
title(ttl);

subplot(223);
plot(r_zone,msevalues_zone);
axis([0.0 0.6 min(msevalues_zone)-200 max(msevalues_zone)+200]);
grid on;
title("Mean Squared Error, zone method");

subplot(224);
plot(r_threshold,msevalues_threshold);
axis([0.0 0.6 -100 max(msevalues_threshold)+1000]);
grid on;
title("Mean Squared Error, threshold method");

figure(2);
set(gcf,'Position',[1000 300 600 700]);
subplot(311);
plot(r_zone,msevalues_zone);
axis([0.0 0.2 min(msevalues_zone)-100 max(msevalues_zone)+100]);
grid on;
title("Mean Squared Error, zone method");
subplot(312);
plot(r_threshold,msevalues_threshold);
axis([0.0 0.2 0 max(msevalues_threshold)+100]);
grid on;
title("threshold method, zoomed in");
subplot(313);
plot(r_zone,msevalues_zone);
grid on;
hold on;
plot(r_threshold(length(r_threshold):-
1:1),msevalues_threshold(length(msevalues_threshold):-1:1));
axis([0.0 0.2 0 max(msevalues_zone)+100]);
title("Zone: blue, Threshold: red");
%apo to teleutaio diagramma katalavainoume oti h methodos katwfliou gia tin
%epilogi tis maskas xreiazetai perissotera pixels gia na ftasei se epidosi
%tin maska zwnhs

for i=1:2
    saveas(figure(i),"fig"+num2str(i)+".png");
end
```

ΑΣΚΗΣΗ 1.3

```
clc;
clear;
close all;

img=imread("flower.png");
img=im2double(im2gray(img));
noisy_img_gaussian=imnoise(img,'gaussian',0,0.008);

%task 1: ipologismos SNR gia na vroume to mean value manually
sn=snr(img,noisy_img_gaussian-img)

%ipologizoume oti gia na exoume mean SNR iso me 15dB, efarmozoume gaussian
%thorivo me tipiki apoklisi ?=0.008

%sxediazoume moving average filter
dim=3;
avg=fspecial('average',dim); %arxika 3x3

%apply to image
filtered_avg=imfilter(noisy_img_gaussian,avg);
filtered_median=medfilt2(noisy_img_gaussian, [dim dim]); %apply median filter

%gia elegxo twn snr meta ta filtra
[peak,snratiofiltered_avg]=psnr(filtered_avg,img);
[peak,snratiofiltered_median]=psnr(filtered_median,img);
%parathrw oti h auksisi twn diastasewn twn filtrwn
%xeirotereuei tin poiothta ths eikonas mexri opou to 11x11 filtra prokaloun
%meiwsi tou snr sxetika me tin thorivwdis eikona
%apo optiki parathrhsh to filtro moving average fainetai na diathrei
%perissoteres plhrofories kai na tholwnei ligotero tin eikona parolo pou h
%filtrarismeni eikona me to median filtro ehei megalitero SNR

figure(1);
set(gcf,'Position',[100 500 800 450]);
subplot(221);
imshow(img);
title("original image");
subplot(222);
imshow(noisy_img_gaussian);
title("image with gaussian white noise std=0.064");
subplot(223);
imshow(filtered_avg);
title("moving average filtered image");
subplot(224);
imshow(filtered_median);
title("median filtered image");

%task2: kroustikos thorivos
noisy_img_imp=imnoise(img,'salt & pepper', 0.25);
filtered_avg=imfilter(noisy_img_imp,avg,'conv');
filtered_median=medfilt2(noisy_img_imp);

figure(2);
set(gcf,'Position',[900 500 800 450]);
subplot(221);
imshow(img);
title("original image");
subplot(222);
imshow(noisy_img_imp,[]);
title("salt and pepper noise 25%");
subplot(223);
imshow(filtered_avg,[]);
title("moving average filter");
subplot(224);
imshow(filtered_median,[]);
title("moving median filter");

for i=1:2
    saveas(figure(i),"fig"+num2str(i)+".png");
end
```

ΑΣΚΗΣΗ 1.4

```
clc;
clear;
close all;

img1=imread("dark_road_1.jpg");
img2=imread("dark_road_2.jpg");
img3=imread("dark_road_3.jpg");
images={img1 img2 img3};

%task 1: ipologismos kai emfanisi histograms
figure(1)
set(gcf,"Position",[500 100 1200 900]);
hists={};
for i=1:1:3
    images{i}=im2gray(images{i});
    hists{i}=imhist(images{i}); %gia na ta apothikeusw
    subplot(4,3,i);
    imshow(images{i},[])
    subplot(4,3,3+i);
    imhist(images{i})
end

eqimages={};
for i = 1:1:3
    curr_img=images{i};
    cdf=cumsum(hists{i})/sum(hists{i}); %h athroistiki sinartisi
    L=256; %diaforetikes times
    T=uint8((L-1)*cdf); %transform function se uint8 opws oi eikones
    %xrisimopoioume tin T gia na kanoume map ta pixels se alla intensities
    %analoga me to intensity tous
    for j=1:1:size(curr_img,1)
        for k=1:1:size(curr_img,2)
            curr_img(j,k)=T(curr_img(j,k)+1); %+1 giati to array ksekina apo 1
        end
    end
    eqimages{i}=curr_img;
    subplot(4,3,i+6);
    imshow(eqimages{i},[]);
    subplot(4,3,i+9);
    imhist(eqimages{i});
end

%gia sintomia kwdika sto local histogram equalization entopisa tin sinartisi
%histeq ths matlab thn opoia xrisimopoiw mazi me thn sinartisi blockproc gia na thn
%efarmosw ana block opws kai sto metasxhmatismo DCT. Anti gia thn histeq
%function tha mporouse na dhmiourgithei mia nea sinartisi pou tha ekane ana
%block oti ginetai sto teleutaio for loop diladi tha ipologize kai tha
%efarmoze ton metasxhmatismo T me eisodo ena block eikonas. H sinartisi
%auti tha mporouse na xrisimopoihthei anti gia thn histeq me thn blockproc

%ston aksona y diathrw tin sinoliki eikona kaliteri alla ston aksona x
%emfanizw perissoteres leptomereies sta katw merh tis eikonas
figure(2)
set(gcf,'Position',[300 100 1200 900]);
window_size=[4 400];
%xrisimopoiw parathiro pou ehei megalitero ipsos kai mikro platos
%(san na deigmatolhptw sixnotera ston orizontio aksona)
%giati anamenw oi "leptomereies" kai to contrast tis eikonas na vrisketai
%kata tin dieuthinsi tou x. Epishs oles oi eikones ehoun ton skouro ourano
%sto panw meros tous. Ta pixel auta theloume na ta apomonwsoume apo tis leptomereies
tou dromou
%kai twn kthriwn gia na ehoume kalitero dinato equalisation se ena window.
block_fun=@(block_struct) histeq(block_struct.data);
local_eq_images={};
for i=1:1:3
    curr_img=images{i};
    %xrisimopoiw simmetriko padding gia na metavallw oso ligotero ginetai
    %to histogram kathe block

local_eq_images{i}=blockproc(curr_img,window_size,block_fun,'PadPartialBlocks',true,'P
adMethod','symmetric');
    %kanw crop tis eikones pisw sto arxiko tous megethos gia na min
```

```
    %emfanizetai to padding
    local_eq_images{i}=local_eq_images{i}(1:size(curr_img,1),1:size(curr_img,2));

    subplot(2,3,i);
    imshow(images{i},[]);
    subplot(2,3,i+3);
    imshow(local_eq_images{i},[]);
end

for i=1:2
    saveas(figure(i),"fig"+num2str(i)+".png");
end
```

ΑΣΚΗΣΗ 1.5

```
clc;
clear;
close all;

img=imread("factory.jpg");
img=im2double(rgb2gray(img)); %grayscale kai scaling sto [0 1] me thn im2double

gfilter=fspecial('gaussian', size(img), 2); %dhmiourgia smoothing filter me fspecial
img_filtered=imfilter(img,gfilter,'replicate','same'); %filtrarisma eikonas

noisy=imnoise(img_filtered,'gaussian',0,0.015);
additive_noise=noisy-img_filtered;
snr(img,additive_noise) %gia manual elegxo tou SNR sta 10db
%epeidi to filtro einai kentrarismeno, protou
%ipologisoume ton FFT tou pragmatopoioume fftshift gia na to feroume sto
%(0,0). Meta tha ksanaxreiastei na kanoume fftshift to transfer function
%gia na kanoume visualize (to fftshift einai aneksartito diladi apo tin
H=fft2(gfilter); %transfer function of degradation model

figure(1);
set(gcf,'Position', [100 500 700 450]);
subplot(221);
imshow(img);
subplot(222);
imshow(noisy);
subplot(223);
imshow(fftshift(abs(H)),[]);
title("Impulse response of degradation model (PSF)");
subplot(224);
[f1,f2]=freqspace(size(H),'meshgrid');
surf(f1,f2,fftshift(abs(H)));
shading interp;
title("Frequency Response of Degradation");

%kataskeui filtrou wiener
%ipologismos twn fasmatikwn piknotitwn
n_fact=420*630; %gia diairesh me plithos pixel
N=fft2(additive_noise); %noise fft
G=fft2(noisy); %degraded+noisy image fft
F=fft2(img); %original image fft
Pn=abs(N.*N)./n_fact; %noise PSD
Pg=abs(G.*G)./n_fact; %degraded+noisy image PSD
%Pf=abs(F.*F); %original image PSD
Pf=Pg-Pn; %estimated original image PSD
Ph=abs(H.*H)./n_fact; %degradation function PSD


%W=Sh./(Sh+Pn./Ps);
W=Pf./(Pf+Pn);

F=W.*G;
wiener_img=ifft2(F);

%efarmogi inverse filter
IF=1./H;
IF(isinf(IF))=max(IF(~isinf(IF))); %gia diaireseis me to mhden.
% antikathistw ta inf me ton megalitero arithmo tou pinaka pou den einai inf
```

```
F_final=IF.*F;
final_img=ifft2(F_final);

%efarmogi threshold
threshold=2.0;
thresholded_values=threshold*abs(H).*IF;
IF(abs(IF)>threshold)=thresholded_values(abs(IF)>threshold); %efarmogi threshold
F_thresholded=IF.*F;
final_img_thresholded=ifft2(F_thresholded);
%o pollaplasiasmos me to degradation PSF to opoio einai kentrarismeno (diladi den
%ehei tin arxi tou sto (0,0)) prokalei mia metatopisi fasis sti sixnotita h
%opoia metafrazetai ws metatopish kai otan epistrefoume sto time domain.
%Gia auton ton logo parolo pou den xrisimopoihsame fftshift kata ton
%ipologismo kapoiou fasmatos pragmatopoioume twra shift logw ths
%metatopisis fasis, gia na mporoume na optikopoihsoume swsta thn eikona
%mesw ths imshow()
final_img=ifftshift(final_img);
final_img_thresholded=ifftshift(final_img_thresholded);

figure(2);
set(gcf,'Position',[700 100 1000 800],'Name','Noise PSD known');
subplot(221);
imshow(img);
title("Original Image");
subplot(222);
imshow(wiener_img);
title("Wiener No Inverse");
subplot(223);
imshow(final_img);
title("Wiener and Inverse, no threshold");
subplot(224);
imshow(final_img_thresholded);
title("Wiener and Inverse, thresholded");

%Wiener deconvolution.

%dokimasa na ftiaksw to wiener deconvolution filter alla den paragei ta
%epithimita apotelesmata opote xrisimopoihsa thn sinartisi tis matlab
% Wd=Pf.*conj(H)./(Pf.*Ph + Pn);
% Wd(isinf(Wd))=max(Wd(~isinf(Wd)));
% F_deconv=Wd.*F;
% img_deconv=ifft2(F_deconv);
% img_deconv=ifftshift(img_deconv);

img_deconv=deconvwnr(noisy,gfilter,1/10); %auto

figure(3)
set(gcf,'Position',[1000 100 800 500],'Name','Noise PSD known');
imshow(img_deconv,[]);
title("Wiener Deconvolution");



%Stin periptwsi pou den einai gnwsto to PSD, to ipologizoume apo ena
%parathiro ths PSD ths degraded eikonas stis ipsiles sixnotites
%efoson to sixnotiko shmeio vrisketai sto (0,0) theloume na ipologisoume
%thn mesh timh se ena parathiro peripou sto meso tou pinaka (ipsiles
%sixnotites)
mid_x=round(size(Pg,1)/2);
mid_y=round(size(Pg,2)/2);
%se ena parathiro 20x20 ipologizoume tin mesi timi olwn twn stoixeiwn
Pn_unknown_mean=mean(Pg(mid_x-10:mid_x+10,mid_y-10:mid_y+10),'all')
%gia sigrisi me tin mesi timi tou gnwstou noise PSD
Pn_known_mean=mean(Pn,'all')
%efarmogi twn wiener-inverse filtering kai deconvolution
Pn(:)=Pn_unknown_mean;

F=W.*G;
wiener_img=ifft2(F);

%efarmogi inverse filter
IF=1./H;
IF(isinf(IF))=max(IF(~isinf(IF))); %gia diaireseis me to mhden.
% antikathistw ta inf me ton megalitero arithmo tou pinaka pou den einai inf
F_final=IF.*F;
```

```
final_img=ifft2(F_final);

%efarmogi threshold
threshold=2.0;
thresholded_values=threshold*abs(H).*IF;
IF(abs(IF)>threshold)=thresholded_values(abs(IF)>threshold); %efarmogi threshold
F_thresholded=IF.*F;
final_img_thresholded=ifft2(F_thresholded);
%o pollaplasiasmos me to degradation PSF to opoio einai kentrarismeno (diladi den
%ehei tin arxi tou sto (0,0)) prokalei mia metatopisi fasis sti sixnotita h
%opoia metafrazetai ws metatopish kai otan epistrefoume sto time domain.
%Gia auton ton logo parolo pou den xrisimopoihsame fftshift kata ton
%ipologismo kapoiou fasmatos pragmatopioume twra shift logw ths
%metatopisis fasis, gia na mporoume na optikopoihsoume swsta thn eikona
%mesw ths imshow()
final_img=ifftshift(final_img);
final_img_thresholded=ifftshift(final_img_thresholded);

figure(4);
set(gcf,'Position', [100 500 700 450],'Name','Noise PSD estimated from degraded
image');
subplot(121);
imshow(final_img);
title("Wiener and Inverse, no threshold");
subplot(122);
imshow(final_img_thresholded);
title("Wiener and Inverse, thresholded, unknown Pn");

for i=1:4
    saveas(figure(i),"fig"+num2str(i)+".png");
end
```

ΑΣΚΗΣΗ 1.6

```
clc;
clear;
close all;

img=imread("clock.jpg");
img=im2double(rgb2gray(img));
%orizoume ta sobel kernels
sobel_y=[-1 -2 -1; 0 0 0; 1 2 1];
sobel_x=[-1 0 1; -2 0 2; -1 0 1];

grad_x=conv2(sobel_x,img);
grad_y=conv2(sobel_y,img);

mag=sqrt(grad_x.^2+grad_y.^2); %gradient image
angle=atan(grad_y./grad_x); %gradient angle

figure(1)
set(gcf,'Position',[100 400 1200 500]);
subplot(141);
imshow(img);
subplot(142);
imshow(mag);
subplot(143);
imhist(mag);
%thresholding
%apo to istogramma ths gradient image parathrw oti h diamesos vrisketai se
%xamhles times pixel intensity. opote ipothetw oti ena katallhlo threshold
%pithanon na vrisketai sto diasthma [0.0,0.3]
%kai me optiki parathrhsh parathrw oti
%exw polles akmes oi opoies eite diakoptontai eite den anixneuontai
%katholou (pithanotata logw xamilou contrast eikonas) opote epilegw mia
%xamili timi gia ot threshold

thresh=0.1*max(mag,[],'all');
mag(mag>=thresh)=1.0;
mag(mag<thresh)=0.0;
subplot(144);
imshow(mag);
```

```
% Hough transform
% orizoume to ??-plane
theta=linspace(-pi/2,pi/2,180);
d = sqrt(size(img,1)^2 + size(img,2)^2); %megisti timi tou r einai h diagwnios tou x,y
plane
r=linspace(-d,d,2*d);

%ara h ipodiairesi tou xwrou tha einai h ekshs
acc_cells=zeros(length(theta),length(r));

%ipologismos gia kathe eutheia tou r-theta plane posa pixels twn akmwn
%vriskontai panw se autes tis eutheies. oi eutheies me ta perissotera
%epivevaiwmena stoixeia tha einai autes pou tha sxediastoun

for i=1:size(mag,1)
    for j=1:size(mag,2)
        if mag(i,j)>0 %an einai shmeio akmhs
            for t=1:length(theta)
                R=i*cos(theta(t))+j*sin(theta(t)); %ipologizw kathe eutheia pou
dierxetai apo auto to simeio
                [~,indexofclosest]=min(abs(R-r)); %stroggilopoiw stin kontinoteri
ipodiairesi tou r
                acc_cells(t,indexofclosest)=acc_cells(t,indexofclosest)+1;
                %auksanw to cell kathe eutheias pou periexei auto to simeio
                %akmis
            end
        end
    end
end

num_of_lines=100;
[~,indices]=maxk(acc_cells(:),num_of_lines);
[theta_lines,r_lines]=ind2sub(size(acc_cells),indices);

%times
ts=theta(theta_lines);
rs=r(r_lines);

figure(2);
imshow(img);
hold on;

for i=1:num_of_lines
    x=1:size(img,1);
    y=(rs(i)-x*cos(ts(i)))/sin(ts(i));
    plot(x,y,'r');
end

for i=1:2
    saveas(figure(i),"fig"+num2str(i)+".png");
end
```