Άσκηση 1

Ψηφιακή επεξεργασία και ανάλυση εικόνας

Περιεχόμενα

Ο κώδικας μπορεί να βρεθεί εδώ:

https://github.com/GrigorisTzortzakis/Digital-image-processing-and-analysis/tree/main

1. Φιλτράρισμα στο πεδίο συχνοτήτων

1)

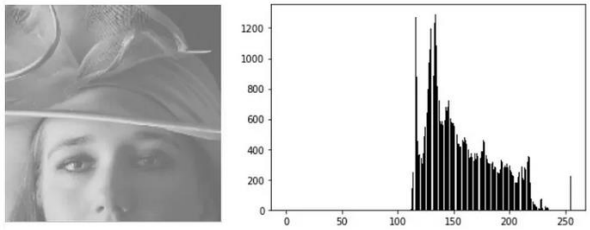
Αρχικά, ο γραμμικός μετασχηματισμός της περιοχής τιμών της εικόνας ή αλλιώς η τεχνική γνωστή ως contrast stretching, είναι μια απλή μέθοδος όπου επιχειρεί να βελτιώσει το contrast (αντίθεση χρωμάτων) της εικόνας επεκτείνοντας τις τιμές φωτεινότητας που μπορούν να λάβουν τα pixel της εικόνας. Με απλά λογία, «τεντώνουμε» το εύρος τιμών που λαμβάνουν τα pixel της εικόνας (ιστόγραμμα). Ειδικότερα, εφαρμόζουμε τον τύπο:

$$\boldsymbol{s = (r - rmin)\left(\frac{Imax - Imin}{rmax - rmin}\right) + Imin}$$

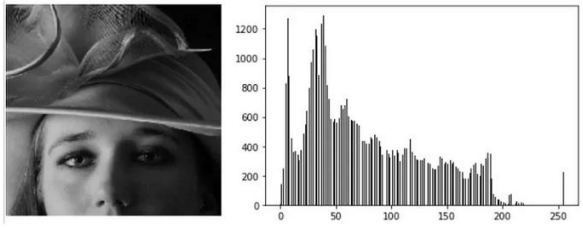
όπου:

- r= Τωρινή τιμή pixel
- rmin= Μικρότερη τιμή pixel που έχει η εικόνα
- rmax= Μεγαλύτερη τιμή pixel που έχει η εικόνα
- Imin=0
- Imax=255

Στις παρακάτω εικόνες μπορούμε να δούμε την λειτουργεία της τεχνικής.



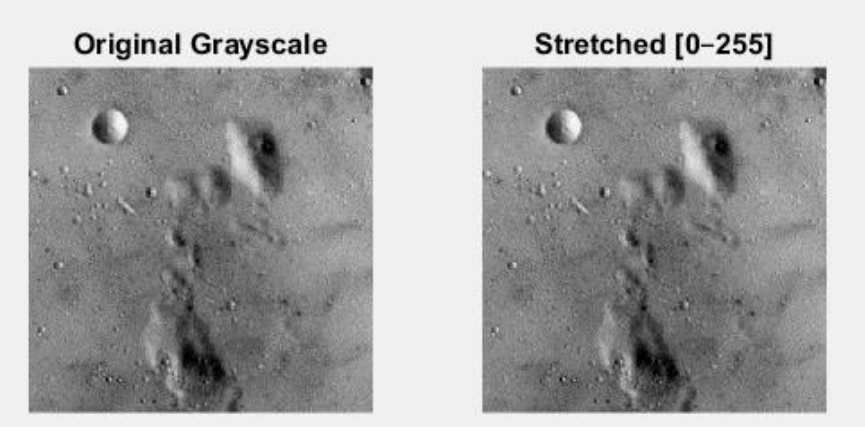
Input Image before Contrast Stretching with its histogram

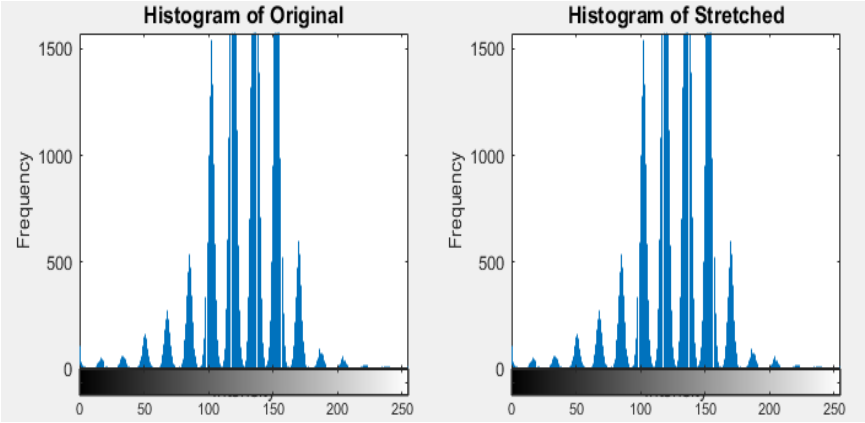


Input Image after Contrast Stretching with its histogram

Συνεχίζοντας, εφαρμόζουμε λοιπόν τα παραπάνω στην εικόνα moon και το αποτέλεσμα δεν είναι αυτό που περιμέναμε. Παρατηρούμε ότι η εικόνα δεν έχει υποστεί καμία αλλαγή σε σχέση με την αρχική. Αυτό συμβαίνει καθώς στην αρχική τιμή η ελάχιστη και μέγιστη τιμή είναι 0 και 255 αντίστοιχα, συνεπώς δεν αλλάζει κάτι όταν εμείς εφαρμόζουμε την συγκεκριμένη

κανονικοποιηση. Αυτό το συμπέρασμα επιβεβαιώνεται ευκολά από το ιστόγραμμα αλλά και από την βιβλιογραφία.

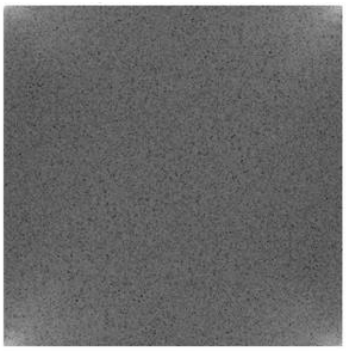




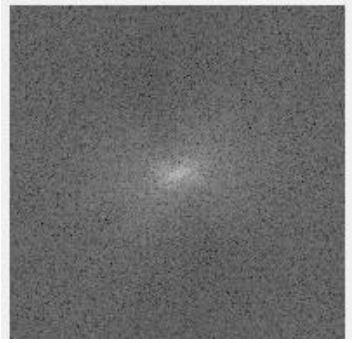
https://homepages.inf.ed.ac.uk/rbf/HIPR2/stretch.htm

https://samirkhanal35.medium.com/contrast-stretching-f25e7c4e8e33

Προχωρώντας στον μετασχηματισμό fourier, γνωρίζουμε ότι χωρίς το κεντράρισμα η ενέργεια εμφανίζεται στις άκρες του διαγράμματος. Με άλλα λογία, οι χαμηλές συχνότητες είναι στα άκρα του διαγράμματος και οι υψηλές στο κέντρο (η ενέργεια βρίσκεται στις χαμηλές επειδή αυτές καταλαμβάνουν το μεγαλύτερο πλήθος της εικόνας).



Αυτό που θέλουμε να κάνουμε εμείς είναι να αντιστρέψουμε τους ρόλους, δηλαδή να είναι οι χαμηλές συχνότητες στο κέντρο και οι υψηλές προς τα έξω. Για να φέρουμε αυτή τη μηδενική συχνότητα στο κέντρο, χρησιμοποιούμε μια απλή ιδιότητα του DFT. Αν πριν από τον υπολογισμό πολλαπλασιάσουμε κάθε στοιχείο της εικόνας με $(-1)^{x+y}$, τότε στην έξοδο του μετασχηματισμού η μηδενική συχνότητα μετατοπίζεται ακριβώς στο κέντρο.



2)

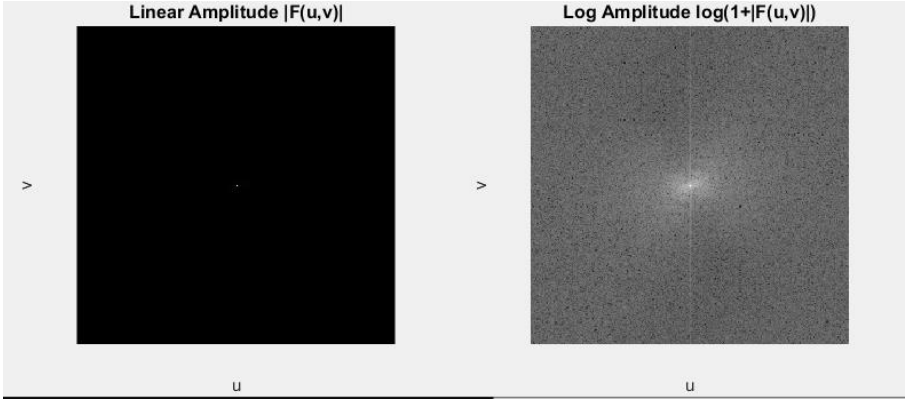
Ο 2D DFT είναι επέκταση του μονοδιάστατου DFT (1D DFT) σε δισδιάστατο πλέγμα. Ενώ ο μονοδιάστατος DFT μετασχηματίζει μια ακολουθία x[n] μήκους N σε:

$$\boldsymbol{X[k] = \sum_{n=0}^{N-1} x(n) e^{\frac{-j2\pi kn}{N}}}$$

ο δισδιάστατος DFT μετασχηματίζει μια M×N εικόνα f[m,n] σε:

$$\boldsymbol{F[u,v] = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f[m,n] e^{-j2\pi(\frac{um}{M}+\frac{un}{N})}}$$

Άρα, ο 2d μπορεί να θεωρηθεί απλώς ως η εφαρμογή του 1D δύο φορές, πρώτα σε κάθε γραμμή και μετά σε κάθε στήλη. Αναλυτικότερα, πρώτα αντιμετωπίζουμε κάθε γραμμή ως ξεχωριστό μονοδιάστατο σήμα και στην συνέχεια κάνουμε το ίδιο για κάθε στήλη.

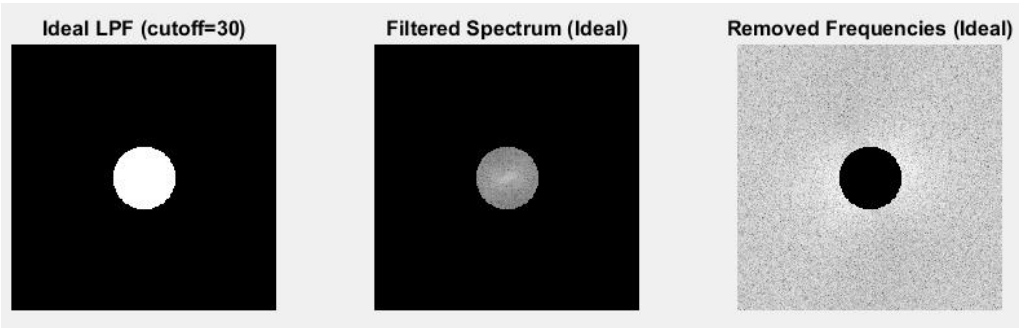


Στην αριστερή εικόνα, όπου απεικονίζεται το γραμμικό πλάτος |F(u,v)|, το φάσμα κυριαρχείται από τον όρο μηδενικής συχνότητας (DC component) στο κέντρο. Αυτός ο συντελεστής αντιπροσωπεύει το μέσο επίπεδο φωτεινότητας της εικόνας και έχει κατά πολύ μεγαλύτερο μέτρο από οποιονδήποτε άλλο συντελεστή του φάσματος. Ως αποτέλεσμα, σε μια γραμμική κλίμακα, όλα τα υπόλοιπα στοιχεία εμφανίζονται σκοτεινά ενώ μόνο το κεντρικό σημείο ξεχωρίζει ως λαμπρή κουκκίδα. Επεξηγηματικά, έχουμε αναφέρει παραπάνω πως το μεγαλύτερο ποσοστό της εικόνας αποτελείτε από χαμηλές συχνότητες, άρα αυτές έχουν το μεγαλύτερο ποσοστό ενέργειας και για αυτό βλέπουμε μόνο το κεντρικό σημείο.

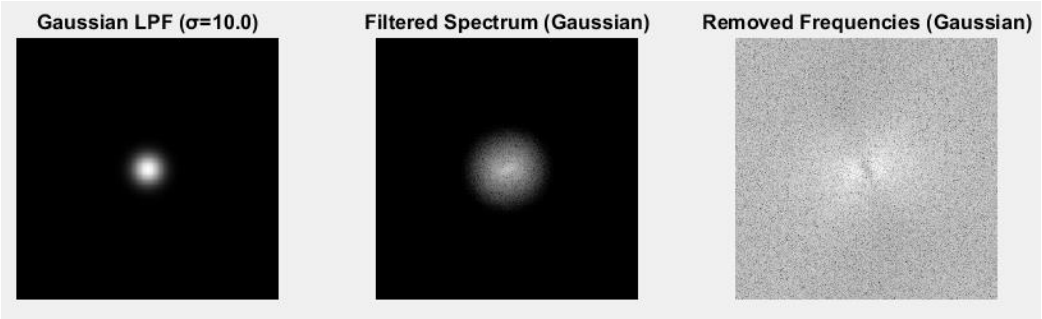
Στη δεξιά εικόνα βλέπουμε την λογαριθμική απεικόνιση. Το κεντρικό στοιχείο εξακολουθεί να είναι το πιο φωτεινό, αλλά τώρα αποκαλύπτονται και οι χαμηλές και υψηλές συχνότητες που ήταν "κρυμμένες" στη γραμμική απεικόνιση. Αυτό συμβαίνει καθώς παίρνοντας τον λογάριθμο της κάθε τιμής, μειώνουμε την τεραστία διαφορά ανάμεσα στις συχνότητες και έρχονται πιο «κοντά» στην ίδια κλίμακα.

3)

Για το συγκεκριμένο ερώτημα, υλοποιούμε ένα ιδανικό χαλημοπερατό φίλτρο καθώς και ένα gaussian. Το ιδανικό χαμηλοπερατό φίλτρο είναι το απλούστερο που μπορούμε να εφαρμόσουμε, καθώς το μόνο που κάνει είναι αποκοπή στην συχνότητα που εμείς επιλέγουμε.



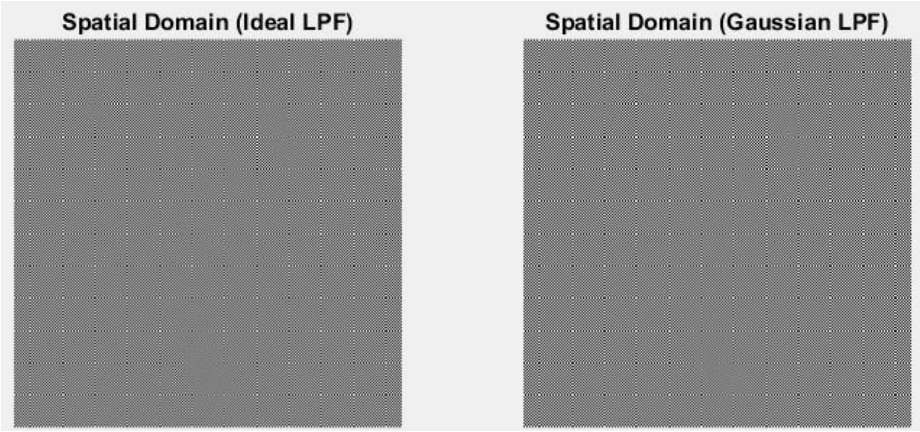
Στην παραπάνω εικόνα μπορούμε να διακρίνουμε ακριβώς την λειτουργία του. Ειδικότερα, βλέπουμε ότι έχει αφαιρέσει όλες τις συχνότητες που βρίσκονται άνω από το όριο που ορίσαμε και έχει απομείνει μόνο μια «σφαίρα». Το μόνο πρόβλημα που έχει, είναι ότι η αποκοπή είναι απότομη. Για αυτό τον λόγο, εφαρμόζουμε και gaussian φίλτρο.



Τώρα βλέπουμε πως αντί για μια απλή «σφαίρα», οι συχνότητες που έχουν απομείνει είναι πιο «θολωμένες», δηλαδή η μετάβαση είναι πιο ομαλή.

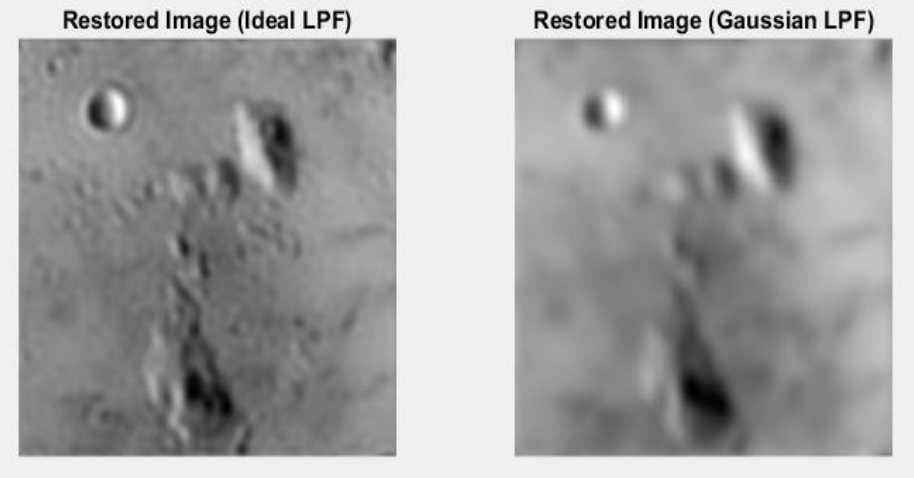
Τέλος, αν μεγαλώσουμε το όριο, τότε ο κύκλος θα είναι μεγαλύτερος, δηλαδή θα έχουμε κρατήσει παραπάνω συχνότητες. Αντίστοιχα, αν μικρύνουμε το όριο, τότε ο κύκλος θα είναι πιο μικρός.

4)

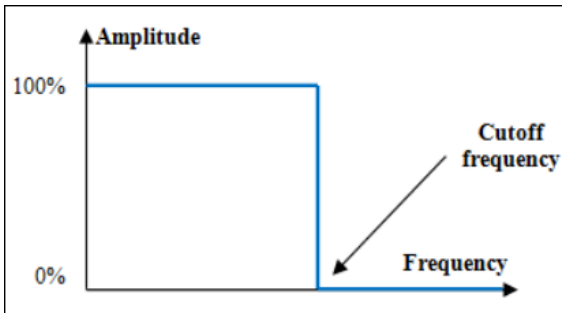


Το αποτέλεσμα είναι αρκετά διαφορετικό από αυτό που περιμέναμε, αφού οι εικόνες που βλέπουμε είναι αρκετά παραμορφωμένες. Αυτό οφείλετε στο ότι σε προηγούμενο ερώτημα εκτελέσαμε centering του φάσματος, δηλαδή πολλαπλασιάσαμε κάθε pixel με $(-1)^{x+y}$, ώστε να μεταφερθεί η μηδενική συχνότητα στο κέντρο. Αναλυτικότερα, έχουμε πολλαπλασιάσει κάθε pixel με +1 ή –1 σε ένα μοτίβο, έτσι ώστε η «μηδενική» συχνότητα να καταλήγει στο κέντρο του φάσματος. Όταν δεν αφαιρούμε αυτή την διαδικασία, τότε έχουμε ακόμα αυτές τις ετικέτες και η εικόνα φαίνεται σαν να έχει «μάσκα».

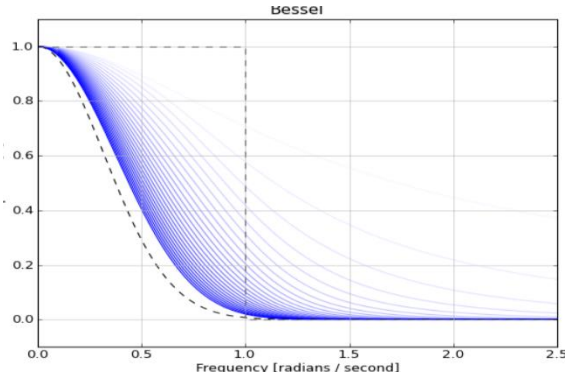
5)



Το ιδανικό φίλτρο έχει χάσει αρκετή λεπτομέρεια, το οποίο είναι αναμενόμενο, καθώς έχουμε αφαιρέσει τις υψηλές συχνότητες. Το αξιοσημείωτο γεγονός που παρατηρούμε ότι τα «κύματα» που έχουν σχηματιστεί σε μερικά σημεία π.χ γύρω από τον κρατήρα. Αυτό συμβαίνει επειδή η απότομη αποκοπή του φίλτρου δημιουργεί «άλμα» στην απόκριση συχνότητας του και αυτό μεταφράζεται σε συνάρτηση sinc στο spatial domain, η οποία αποσβένεται αργά.



Αντιθέτως, αυτό δεν συμβαίνει στο gaussian φίλτρο, επειδή η απόκριση συχνότητας του αποσβένει πιο ομαλά. Το αρνητικό είναι ότι χάνουμε αρκετή λεπτομέρεια και η εικόνα είναι ακόμα πιο «smooth».



2. Συμπίεση εικόνας με χρήση μετασχηματισμού DCT

1)

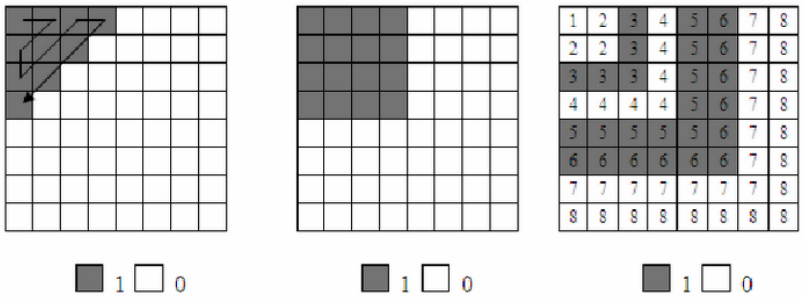
Η βασική ιδέα συμπίεσης DCT είναι ότι το μεγαλύτερο μέρος της «πληροφορίας» σε μια φυσική εικόνα βρίσκεται σε περιοχές ομαλής μεταβολής (χαμηλής συχνότητας). Συγκεκριμένα, ο μετασχηματισμός είναι μια γραμμική αλλαγή βάσης που αποσυσχετίζει τις τιμές των pixel, καθώς κάθε μπλοκ N×N εκφράζεται ως ένα σύνολο συνημιτονικών προτύπων διαφορετικών συχνοτήτων. Επεκτείνοντας, επειδή γειτονικά pixel σε μια φωτογραφία τείνουν να έχουν παρόμοιες τιμές, τα χαμηλής συχνότητας συνημίτονα συλλαμβάνουν το μεγαλύτερο μέρος της ενέργειας.

Συνεχίζοντας, διαιρώντας την εικόνα σε μη επικαλυπτόμενα μπλοκ N×N (N=32), καθένα από αυτά έχει σχετικά σταθερές στατιστικές ιδιότητες και μπορεί να επεξεργάζεται ανεξάρτητα. Σε κάθε μπλοκ B, ο DCT υπολογίζει τους συντελεστές:

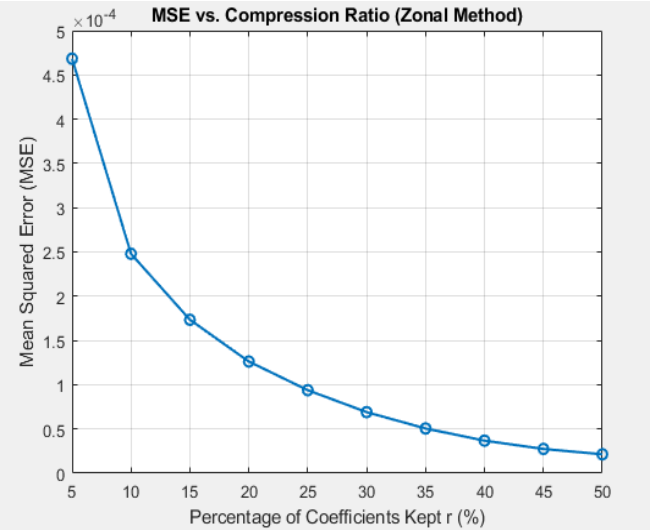
$$C_{u,v} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} B(x,y)\varphi_u(x)\varphi_v(y)$$

όπου κάθε φk είναι μια συνάρτηση βάσης συνημιτόνου με συχνότητα ανάλογη του k και ο συντελεστής (0,0) είναι ο μέσος όρος του μπλοκ.

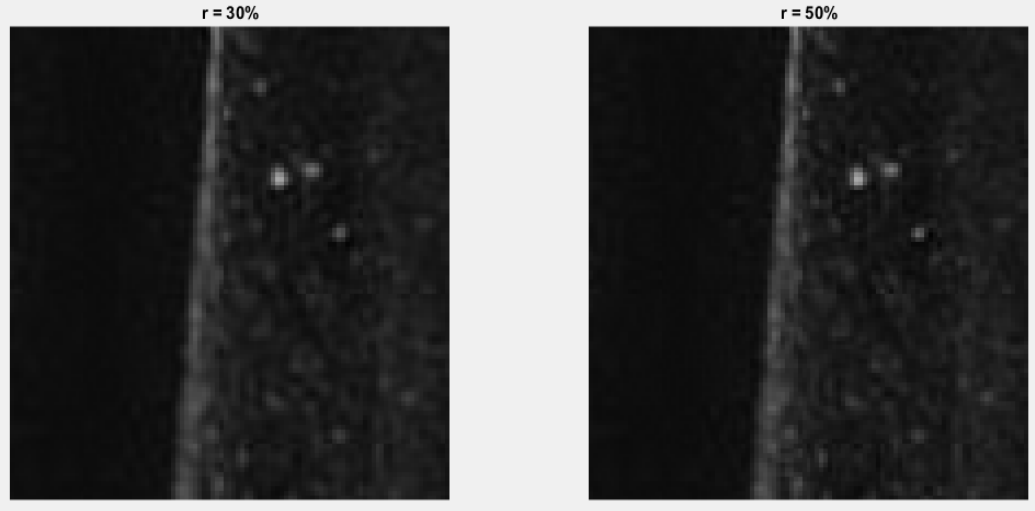
Προσθέτοντας, το φαινόμενο της συσσώρευσης ενέργειας σημαίνει ότι, τα μεγέθη |Cu,v| μειώνονται γρήγορα καθώς αυξάνεται το u+v. Πρακτικά, πάνω από το 90 % της συνολικής ενέργειας ενός μπλοκ βρίσκεται στους πρώτους λίγους δεκάδες συντελεστές. Η μέθοδος ζώνης αξιοποιεί αυτή την ιδιότητα με το να ορίζει εκ των προτέρων μια «ζιγκ-ζαγκ» πορεία στον πίνακα (u,v) και να διατηρεί μόνο τους πρώτους $K = [rN^2]$ συντελεστές σε αυτή την πορεία, μηδενίζοντας τους υπόλοιπους. Η παράμετρος r ελέγχει το ποσοστό των συντελεστών που διατηρούνται και τότε μικρότερο r σημαίνει μεγαλύτερη συμπίεση αλλά και μεγαλύτερη παραμόρφωση των συντελεστών που διατηρούνται.



Στο βήμα της ανακατασκευής, παίρνουμε κάθε μπλοκ 32×32 από το οποίο έχουν μηδενιστεί οι περισσότεροι συντελεστές και εφαρμόζουμε αντίστροφη DCT. Με απλά λόγια, συναρμολογούμε ξανά τα κύματα συνημιτόνου που αντιστοιχούν στους λίγους συντελεστές που κρατήσαμε. Μόλις τελειώσουμε αυτή την διαδικασία σε όλα τα μπλοκ, τα επανατοποθετούμε στις αρχικές τους θέσεις στο πλέγμα της εικόνας.



Η απότομη πτώση του MSE μεταξύ 5 % και 20 % διατήρησης συντελεστών αντανακλά το γεγονός ότι οι πολύ χαμηλές συχνότητες της DCT μεταφέρουν το μεγαλύτερο μέρος της ενέργειας κάθε μπλοκ εικόνας. Επαναφέροντας μόνο αυτούς τους λίγους συντελεστές, ο συμπιεστής ανακτά τις μεγάλες διαβαθμίσεις φωτεινότητας και τις ομαλές μεταβάσεις, εξαλείφοντας το μεγαλύτερο μέρος του μέσου τετραγωνικού σφάλματος. Το σφάλμα συνεχίζει να μειώνεται και στο 50 περίπου ο αλγόριθμος έχει συγκλείσει.



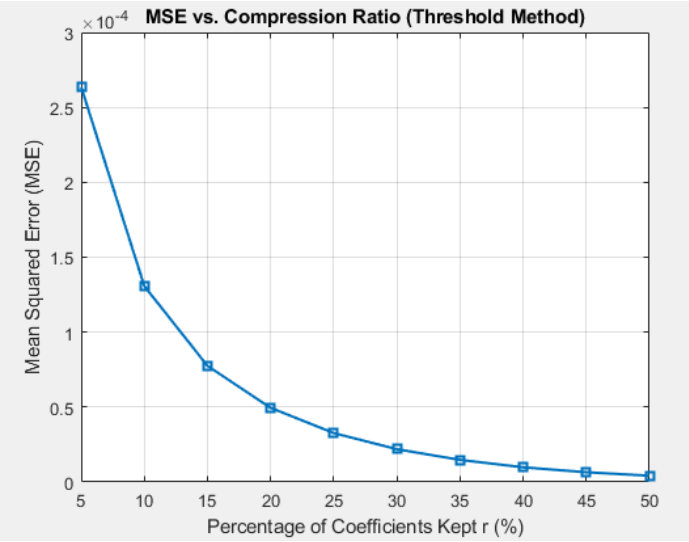
Κάνοντας μεγέθυνση στις ανακατασκευασμένες εικόνες, βλέπουμε το ποσοστό που κάθε συμπίεση έχει επηρεάσει την εικόνα. Με 10%, έχουμε εμφανή υποβάθμιση, αφού είναι θολωμένη και δεν υπάρχει σχεδόν καθόλου λεπτομέρεια. Με 30%, έχουμε και πάλι απώλεια λεπτομερειών και "blurring" άλλα η ποιότητα είναι σε γενικά πλαίσια αποδεκτή. Τέλος, με 50% οι διάφορες είναι πρακτικά ελάχιστες, με πολύ μικρή παραμόρφωση στις λεπτομέρειες.

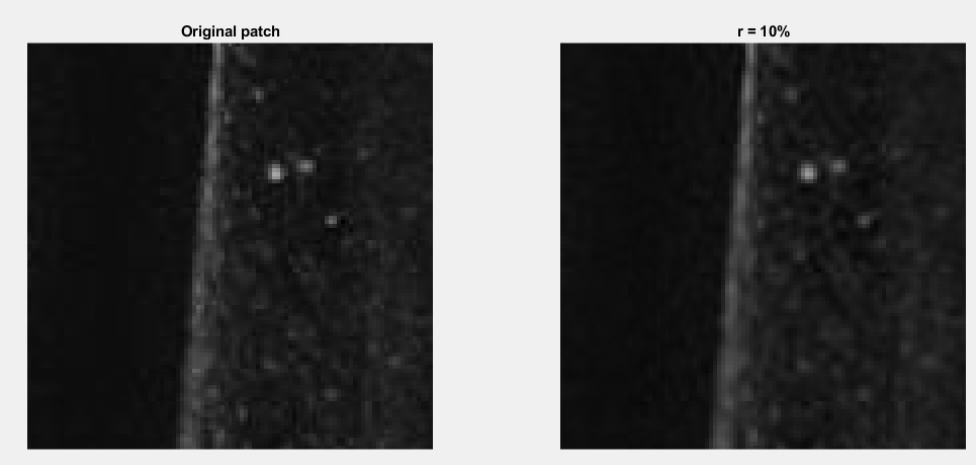
2)

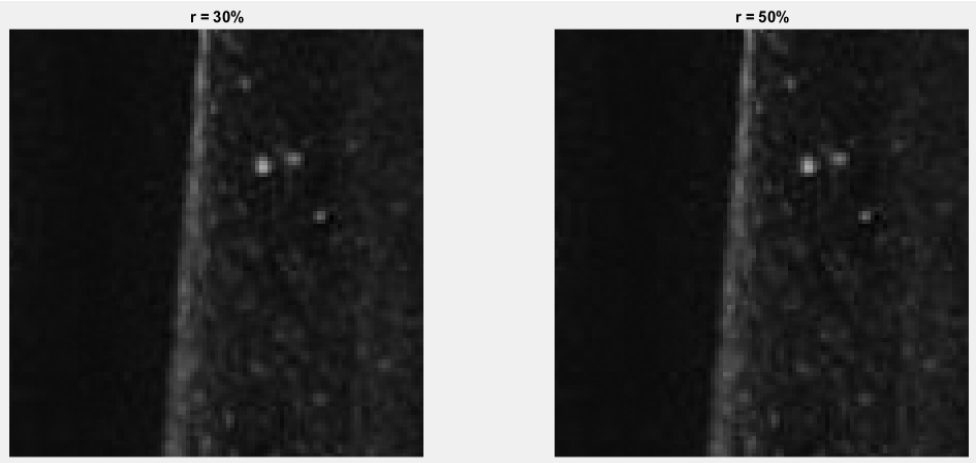
Στη μέθοδο κατωφλιού (thresholding), αντί να διατηρούμε σταθερά τις πρώτες χαμηλές συχνότητες, επιλέγουμε τους συντελεστές που έχουν απόλυτο πλάτος μεγαλύτερο από το όριο που θέσαμε, δηλαδή εκείνους που πραγματικά «κουβαλούν» περισσότερη ενέργεια σε κάθε μπλοκ. Μετά τον υπολογισμό της DCT σε κάθε 32×32 μπλοκ, διατάσσουμε τους συντελεστές κατά φθίνουσα τάξη απόλυτης τιμής $|\boldsymbol{C_{u,v}}|$. Για ένα δεδομένο r, κρατάμε τους πρώτους $\boldsymbol{K} = [\boldsymbol{rN^2}]$ μεγαλύτερους σε απόλυτο μέγεθος συντελεστές και μηδενίζουμε όλους τους υπόλοιπους. Για παράδειγμα, έστω ότι έχουμε 8 συντελεστές [20, 5, -3, 1, -1, 0, 0, 0] και θέλουμε r=50%, δηλαδή να κρατήσουμε 4 από αυτούς. Τότε, βλέπουμε την απολυτή τιμή τους και ορίζουμε ως κατώφλι την 4η μεγαλύτερη (1). Οτιδήποτε είναι πάνω από αυτό το όριο το κρατάμε και όλα τα υπόλοιπα μηδενίζονται.

Ουσιαστικά, το κατώφλι τ επιλέγεται έτσι ώστε το πλήθος των $|\boldsymbol{C_{u,v}}| \geq \boldsymbol{\tau}$ σε κάθε μπλοκ να ισούται με K. Αυτό σημαίνει ότι για διαφορετικά μπλοκ, η τοποθέτηση και το σχήμα της «μάσκας» στον πίνακα συχνοτήτων αλλάζει προσαρμοστικά στις τοπικές ιδιότητες της εικόνας π.χ. σε περιοχές με έντονες γωνίες ή υφή θα διατηρηθούν περισσότεροι συντελεστές υψηλών συχνοτήτων, ενώ σε ομαλές περιοχές το κατώφλι θα περιορίσει πολύ τα δεδομένα.

Η ανακατασκευή γίνεται με τον ίδιο τρόπο με την προηγουμένη μέθοδο όπως και ο υπολογισμός σφάλματος, δηλαδή είναι ο μέσος όρος τετραγώνων των διαφορών ανάμεσα στην αρχική και στην ανακατασκευασμένη εικόνα.







Όπως είδαμε και από την θεωρία, τα αποτελέσματα είναι πολύ καλυτέρα από αυτά της προηγουμένης μεθόδου. Ακόμα και με μόλις 10% των τιμών, έχουμε διατηρήσει λεπτομέρειες και η εικόνα γενικά είναι ευκρινής. Τέλος, από το σφάλμα βλέπουμε ότι πρακτικά με 30% των τιμών, έχουμε μια σχετικά «καθαρή» εικόνα.

3. Βελτίωση εικόνας - Φιλτράρισμα θορύβου

1)

Αρχικά, ας δούμε πως λειτουργούν τα συγκεκριμένα φίλτρα. Το moving average βρίσκει τον μέσο όρο των γειτονικών pixel και αναθέτει αυτή την τιμή π.χ Έστω ότι έχουμε τις τιμές [10, 12, 25, 20] που βρίσκονται στις θέσεις 1, 2, 3, 4 αντίστοιχα και το φίλτρο έχει παράθυρο 3. Για την θέση 1 δεν βρίσκουμε νέα τιμή, καθώς για να βρούμε τον μέσο όρο πρέπει να έχουμε προηγουμένη και επόμενη τιμή. Η θέση 2 θα έχει νέα τιμή $\frac{10+12+25}{3} = \mathbf{15.66}$, η θεση 3 $\frac{12+25+20}{3} = \mathbf{19}$ και η θέση 4 θα έχει την ίδια με πριν. Το νέο αποτέλεσμα είναι [10, 15.66, 19, 25].

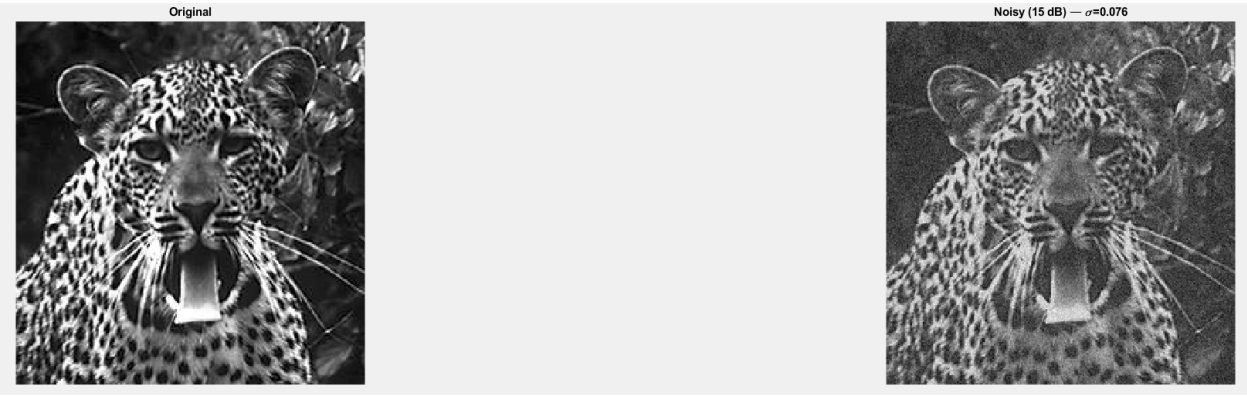
Παρατηρούμε πως οι τιμές έχουν έρθει πολύ πιο κοντά σε σχέση με πριν, δηλαδή έχουν μικρότερη διαφορά αναμεσά τους. Τότε, καταλαβαίνουμε ότι το συγκεκριμένο φίλτρο ουσιαστικά κάνει blurring, αφού δεν έχουμε μεγάλες αυξομειώσεις στις τιμές (άρα έχουμε χαμηλή συχνότητα).

Προχωρώντας, το median φίλτρο βάζει τις τιμές με βάση αύξουσα σειρά και διαλέγει την μεσαία π.χ Έστω ότι έχουμε τις τιμές [10, 12, 25, 20] που βρίσκονται στις θέσεις 1, 2, 3, 4 αντίστοιχα και το φίλτρο έχει παράθυρο 3. Για την θέση 1 και 4 ισχύει το ίδιο με την προηγουμένη περίπτωση, η θέση 2 είναι $\boldsymbol{\alpha\upsilon\xi ουσα\ σειρα} = \mathbf{10, 12, 25} = \boldsymbol{\eta\ τιμη\ στην\ μεση}$ =12 και η θέση 3 είναι $\mathbf{12, 20, 25} = \mathbf{20}$. Το νέο αποτέλεσμα είναι [10, 12, 20, 20].

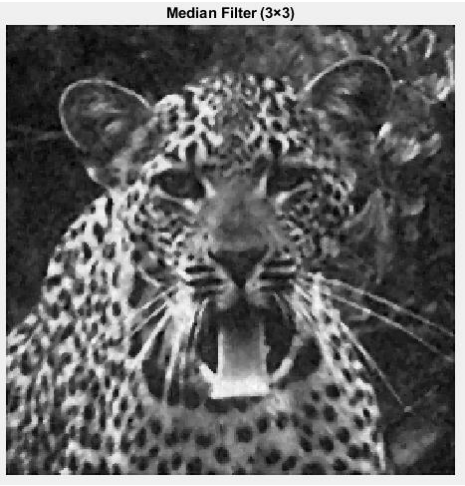
Παρατηρούμε πως τώρα οι τιμές και πάλι διαφέρουν αρκετά μεταξύ τους, άρα έχουν διατηρηθεί οι υψηλές συχνότητες.

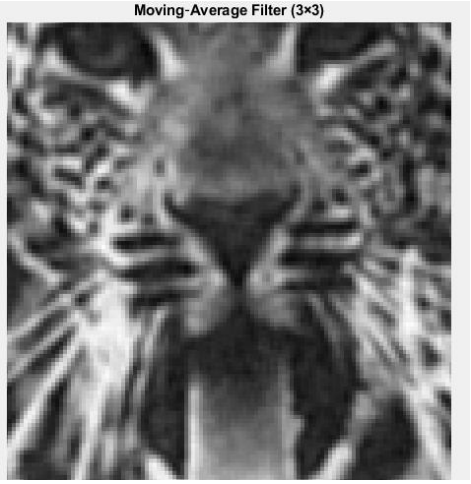
Συμπερασματικά, το κάθε φίλτρο αποδίδει καλυτέρα σε διαφορετικά είδη θορύβου. Το mean average είναι καλό σε περιπτώσεις όπου ο θόρυβος είναι τυχαίος (gauss), αφού βρίσκει τον μέσο όρο και έτσι δεν μας ενδιαφέρει η τυχαιοτητα. Το mean είναι καλό για κρουστικό θόρυβο ή για θορυβο που θα έχει μεγάλους outlier, αφού όπως είδαμε από την λειτουργία του δεν θα επιλεχθούν ποτέ οι ακραίες τιμές, ενώ με τον μέσο όρο θα μας «χαλάσει» η ακρίβεια του αποτελέσματος.

Συνεχίζοντας με το πείραμα, προσθέτουμε τον θόρυβο που ζητείτε και έχουμε τα παρακάτω αποτελέσματα:



Βλέπουμε ότι η αρχική εικόνα έχει υποστεί κλασσική υποβάθμιση λευκού θορύβου, αφού είναι εμφανής σε όλα τα σημεία τα «grain».

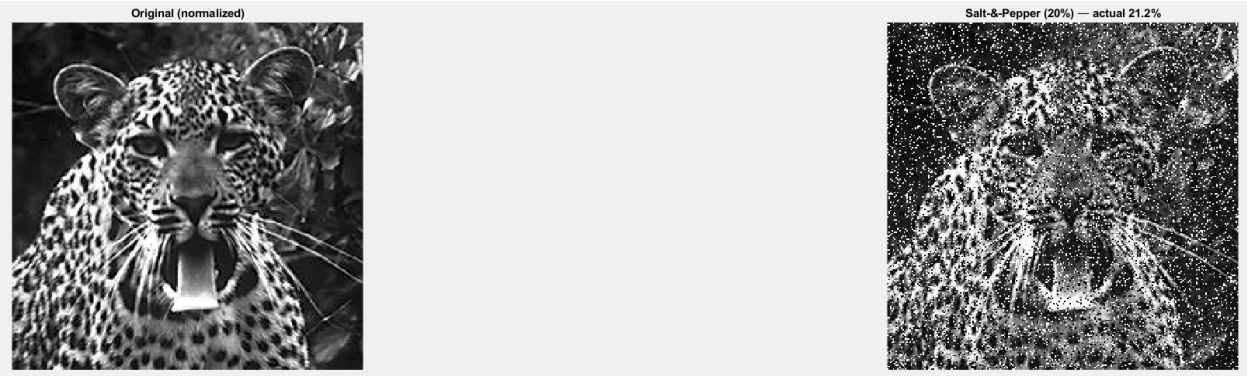




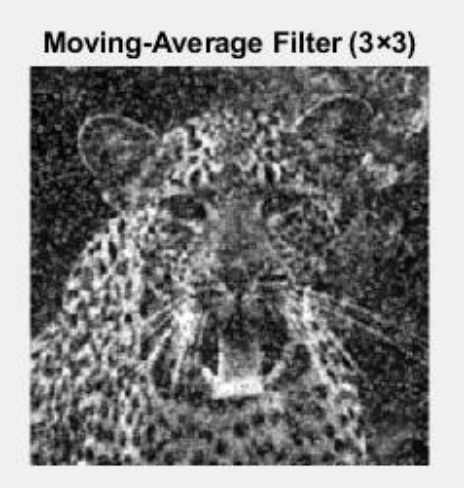
```
Noisy-image SNR: 15.02 dB (target: 15 dB)
Output SNR (moving average): 14.62 dB
Output SNR (median):         14.93 dB
```

Τα αποτελέσματα επιβεβαιώνουν τα θεωρητικά μας συμπεράσματα, αφού πράγματι το moving average είναι blurred, αλλά επειδή ακριβώς έχει αφαιρέσει τις υψηλές συχνότητες, τότε αφαίρεσε και τον θόρυβο με επιτυχία (ξέρουμε ότι αυτός βρίσκεται κυρίως σε εκείνες τις περιοχές). Αντιθέτως, το median έχει διατηρήσει τις λεπτομέρειες καλυτέρα αλλά ταυτόχρονα έχει ακόμα «grain».

2)



Εφαρμόζουμε τον θόρυβο που ζητείτε (salt and pepper) και βλέπουμε ότι η εικόνα πρακτικά έχει καταστραφεί.



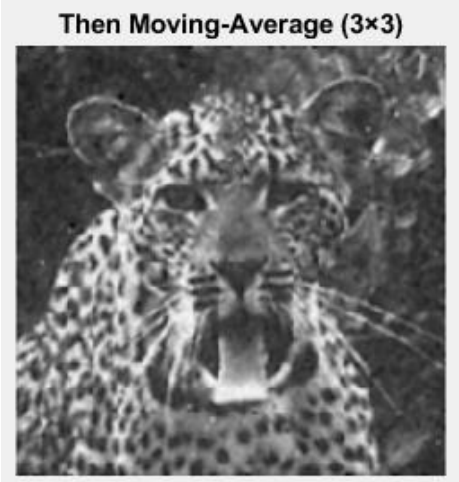
```
Actual noise ratio: 21.19% (target: 20%)
Output SNR (moving average): 10.02 dB
Output SNR (median):         13.31 dB
```

Τα αποτελέσματα επιβεβαιώνουν και πάλι την θεωρία, αφού το moving average φίλτρο δεν κατάφερε να απομακρύνει τον θόρυβο εξαιτίας των outlier. Πρακτικά, αυτά χαλάνε τον μέσο όρο και έτσι δεν γίνεται να απομακρύνουμε τον θόρυβο. Αντιθέτως, το median κατάφερε με επιτυχία να αποκαταστήσει την εικόνα, αφήνοντας μόνο ελάχιστα υπολείμματα θορύβου σε αυτή.

3)

Εφαρμόζουμε τον θόρυβο που ζητείτε και βλέπουμε ότι η εικόνα δεν είναι πλέον εμφανής.

Για να απομακρύνουμε τον θόρυβο με επιτυχία, πρώτα θα εφαρμόσουμε median και μετά moving average φίλτρο. Με αυτή την σειρά, πρώτα θα διώξουμε τον salt and pepper θόρυβο και στην συνέχεια μπορούμε να απομακρύνουμε και τον gaussian. Αν κάνουμε το αντίθετο, όπως είδαμε και στο προηγούμενο ερώτημα, δεν θα έχει φύγει ο κρουστικός θόρυβος και δεν θα είναι πλέον σε θέση το median να τον αντιμετωπίσει μαζί με τον gauss (πρώτα θα έχουμε χαλάσει την εικόνα λόγω των outlier με το moving average φίλτρο).



Βλέπουμε ότι τα αποτελέσματα δεν είναι όσο καλά ήταν στα προηγούμενα ερωτήματα, αλλά είναι λογικό εφόσον έχουμε 2 θορύβους μαζί. Με το πρώτο φίλτρο απομακρύναμε τον κρουστικό θόρυβο και με τον δεύτερο διώξαμε το grain και κάναμε smooth μερικές άσπρες κουκίδες.

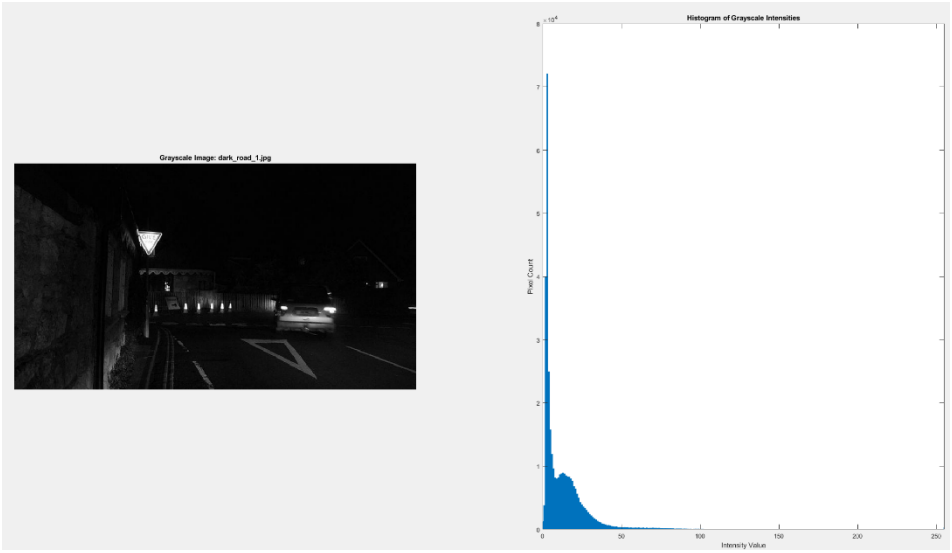
4. Βελτίωση εικόνας - Εξίσωση ιστογράμματος

1)

Το ιστόγραμμα πρακτικά μας δείχνει την κατανομή τιμών φωτεινότητας στα pixel, δηλαδή πόσα λαμβάνουν μια συγκεκριμένη τιμή (0-255 για την περίπτωση μας).

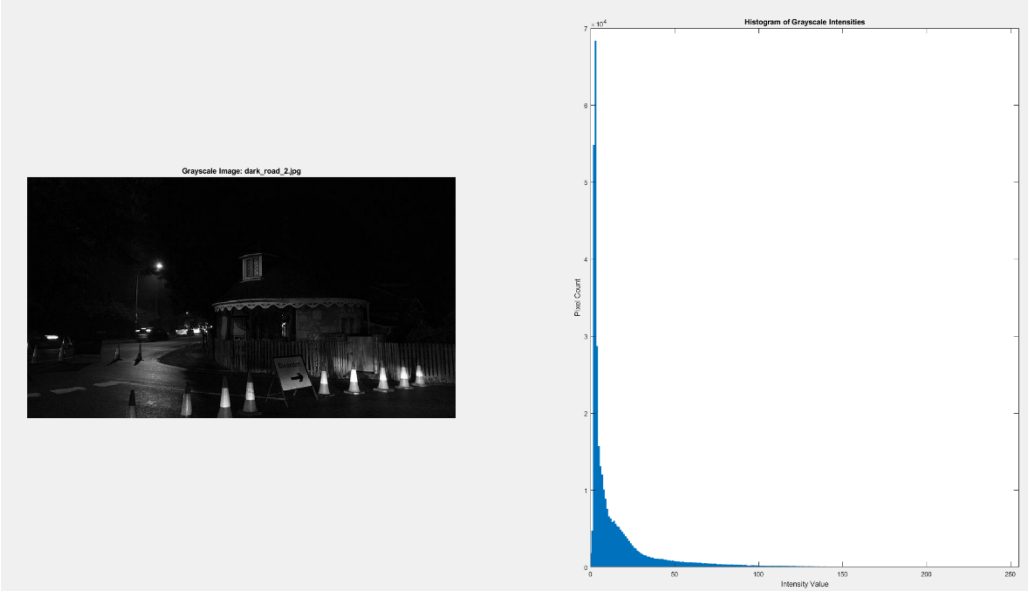
Προχωρώντας, ας δούμε τα αρχικά ιστογράμματα των εικόνων:

dark 1:



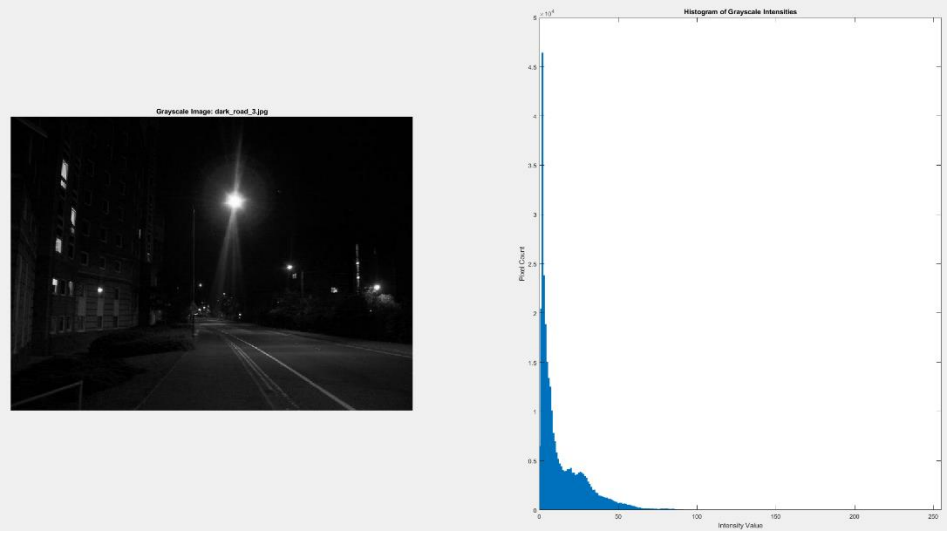
Περισσότερο από το 70% των pixels έχει ένταση κάτω από 30, γεγονός που αντικατοπτρίζει το σκοτάδι της επιφάνειας του δρόμου και των γύρω τοίχων. Ένα μικρό "κύμα" μεσαίων τόνων μεταξύ περίπου 20 και 40 οφείλεται στο πίσω μέρος του αυτοκινήτου και στο ζωγραφισμένο βέλος λωρίδας, ενώ μια πάνω από ένταση 50 παράγεται από την φωτεινή πινακίδα "GIVE WAY" και κάποια ελάχιστα φωτεινά σημεία. Αυτή η έντονη κλίση προς τα αριστερά και οι πρακτικά ανύπαρκτες φωτεινές κορυφές φανερώνουν πολύ χαμηλή συνολική αντίθεση, καθιστώντας δύσκολη τη διάκριση σημάτων ή εμποδίων εκτός από τα λίγα έντονα φωτισμένα στοιχεία (υπενθυμίζουμε ότι οι χαμηλές τιμές είναι σκοτεινές και οι υψηλές φωτεινές).

dark 2:



Η συσσώρευση των pixels κάτω από ένταση 25 είναι ακόμη εντονότερη, ωστόσο, το ιστόγραμμα παρουσιάζει ευρύτερη κατανομή μεσαίων τόνων από περίπου 30 έως 80, λόγω των φωτισμένων περιοχών. Σε σύγκριση με την πρώτη εικόνα, προσφέρει ελαφρώς καλύτερη ορατότητα των στοιχείων του δρόμου, παρότι παραμένει κυρίως σκοτεινή.

dark 3:



Παρουσιάζει μια κάπως πιο ισορροπημένη κατανομή εντάσεων. Παρότι η συσσώρευση στα αριστερά διατηρείται, η κορυφή κοντά στο μηδέν είναι χαμηλότερη, υποδεικνύοντας λιγότερα εντελώς μαύρα pixels και το ιστόγραμμα εμφανίζει πιο ομαλή κατανομή από περίπου 20 έως 60. Αυτό οφείλεται στο έντονο φως στο κέντρο που καταλαμβάνει μεγάλη περιοχή και άρα πολλά pixel.

2)

Αρχικά, η συγκεκριμένη τεχνική προσπαθεί να αλλάξει την κατανομή του ιστογράμματος. Με άλλα λόγια, ο σκοπός της είναι η κατανομή των pixel σε όλες τις τιμές φωτεινότητας. Δεν πρέπει να μας μπερδεύει με το ερώτημα 1, όπου απλά αυξήσαμε το δυναμικό εύρος, δηλαδή απλά θέσαμε ελάχιστη και μέγιστη τιμή. Ο σκοπός μας τώρα δεν είναι να βελτιώσουμε οπτικά την εικόνα, άλλα να αυξήσουμε το contrast, ώστε να είναι ορατές περιοχές που αρχικά ήταν σκοτεινές.

Ειδικότερα, πρώτα, υπολογίζεται το ιστόγραμμα της εικόνας h(rk) που μετρά πόσα pixel έχουν κάθε γκρι επίπεδο rk. Αυτό κανονικοποιείται διαιρώντας με τον συνολικό αριθμό pixel N, δίνοντας τη συνάρτηση μάζας πιθανότητας:

$$\boldsymbol{p(r_k) = \frac{h(r_k)}{N}}$$

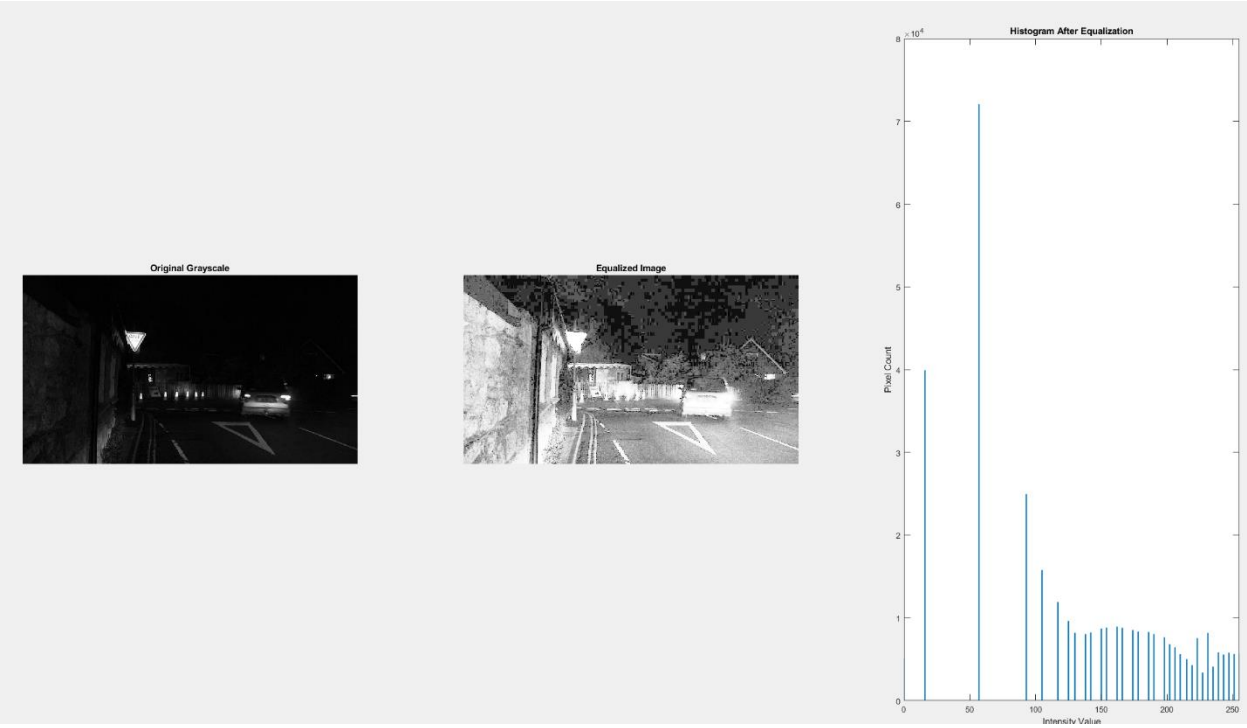
Έπειτα, σχηματίζεται η CDF:

$$s(r_k) = \sum_{j=0}^{k} p(r_j)$$

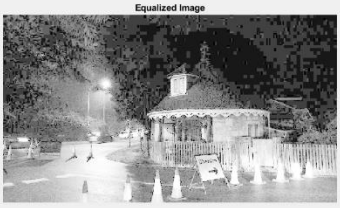
Εφόσον s(rk) είναι μη φθίνουσα και κυμαίνεται από 0 έως 1, την κλιμακώνουμε στο πλήρες εύρος εξόδου [0,L−1] (όπου L=256 για 8-bit) με:

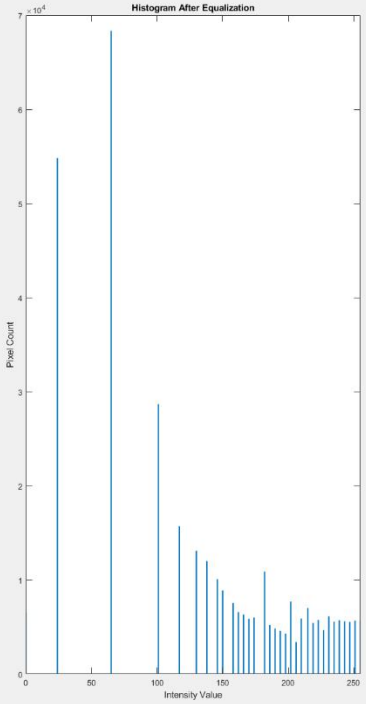
$$T(r_k) = round((L-1)s(r_k))$$

Αυτή η T είναι ο πίνακας αναζήτησης, δηλαδή κάθε αρχική τιμή rk αντικαθίσταται από το T(rk). Οι σκοτεινές περιοχές, όπου πολλά pixel συγκεντρώνονται σε χαμηλά επίπεδα, «τεντώνονται» προς τα πάνω, ενώ φωτεινές περιοχές μπορεί να «τραβηχτούν» προς τα κάτω.

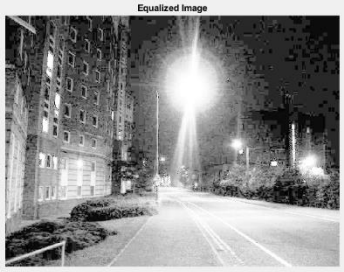


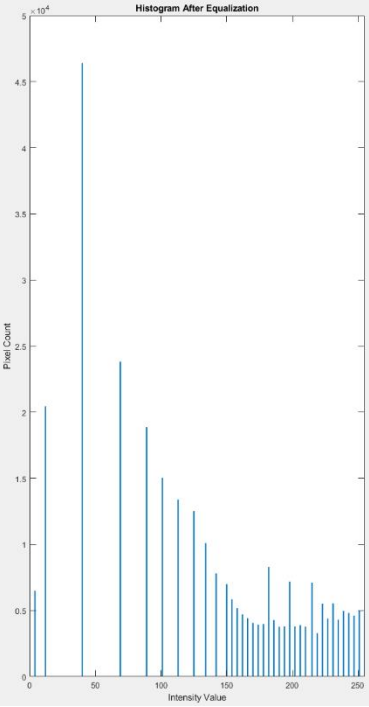

Original Grayscale


Equalized Image


Histogram After Equalization
x 10^4
Pixel Count
Intensity Value
0
1
2
3
4
5
6
7
0
50
100
150
200
250


Original Grayscale


Equalized Image


Histogram After Equalization
x 10^4
Pixel Count
Intensity Value
0
0.5
1
1.5
2
2.5
3
3.5
4
4.5
5
0
50
100
150
200
250

Από τα αποτελέσματα βλέπουμε ότι πράγματι η τεχνική δούλεψε και το contrast έχει αυξηθεί. Οι σκοτεινές περιοχές είναι πλέον ορατές και βλέπουμε από το ιστόγραμμα ότι έχουν μεταφερθεί πολλά pixel σε φωτεινές περιοχές. Επίσης, μπορούμε να δούμε ότι ο θόρυβος έχει αυξηθεί σημαντικά σε όλες τις εικόνες και έχουμε σε πολλά σημεία artifact.

https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html

3)

Η τοπική (ή προσαρμοστική) εξίσωση ιστογράμματος βελτιώνει την αντίθεση προσαρμόζοντας την συνάρτηση μετασχηματισμού σε μικρές, γειτονικές περιοχές της εικόνας. Η εικόνα χωρίζεται αρχικά σε ένα πλέγμα ορθογωνίων "tile" (π.χ. 8×8 ή 16×16) και για κάθε ένα από αυτά υπολογίζεται το δικό του ιστόγραμμα και η τοπική CDF. Με την αντιστοίχιση των τιμών των pixel σύμφωνα με αυτή την τοπική CDF, οι σκοτεινές και φωτεινές περιοχές κάθε τμήματος "τεντώνονται" ανεξάρτητα, έτσι ώστε ακόμη και λεπτομέρειες να γίνονται πιο ευδιάκριτες.

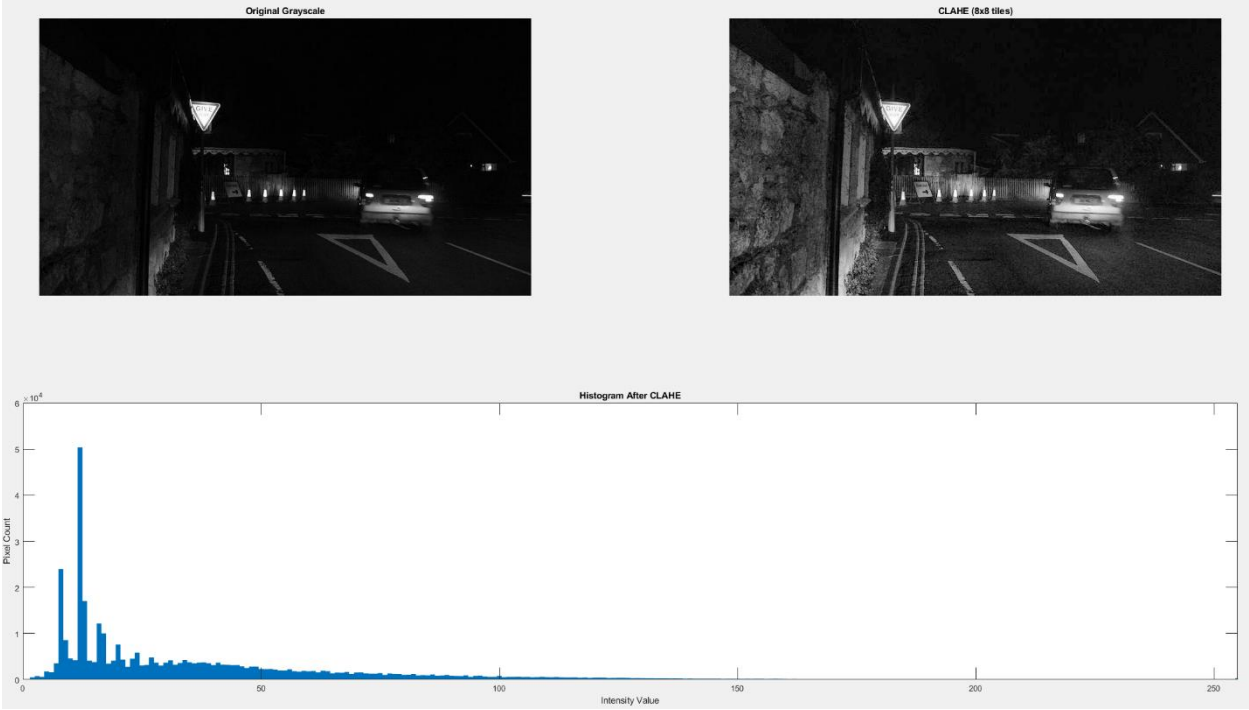
Επιλέγουμε την μέθοδο CLAHE (Contrast Limited Adaptive Histogram Equalization). Αυτή ξεκινά διαιρώντας την εικόνα σε ένα πλέγμα μικρών, μη επικαλυπτόμενων tile. Μέσα σε καθένα από αυτά, υπολογίζεται ανεξάρτητα το τοπικό ιστόγραμμα, επιτρέποντας στον αλγόριθμο να προσαρμόζεται σε διαφορές φωτεινότητας και αντίθεσης που διαφέρουν από το ένα τμήμα στο άλλο.

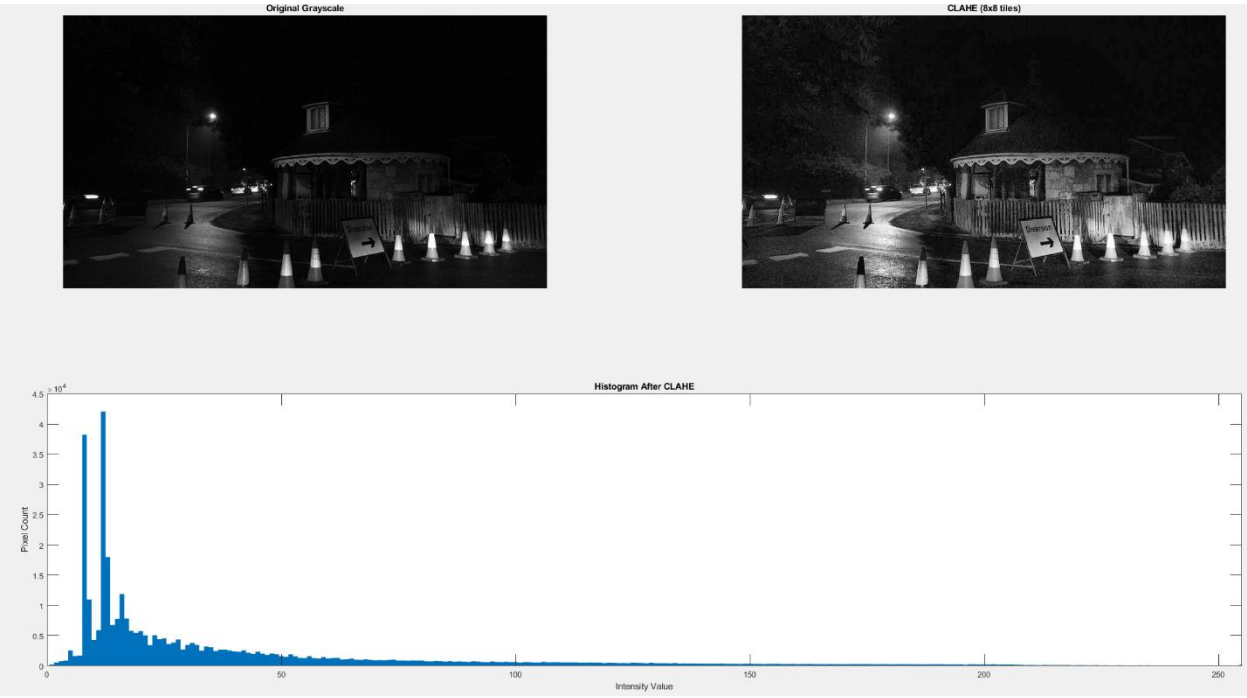
Προσθέτοντας, κύρια βελτίωση σε σχέση με την απλή μέθοδο είναι η εισαγωγή ενός ορίου (clip limit) στο ιστόγραμμα κάθε tile. Σε πολύ σκοτεινές περιοχές, κυριαρχούν λίγες τιμές έντασης, δημιουργώντας έντονες κορυφές που αν εξισωθούν πλήρως, θα μετατρέψουν τον θόρυβο σε υπερβολικά artifacts. Η CLAHE αντιμετωπίζει αυτό το πρόβλημα "περιορίζοντας" το ύψος στο προκαθορισμένο όριο και οι πλεονάζουσες τιμές που αφαιρούνται ανακατανέμονται ομοιόμορφα σε όλα τα επίπεδα. Αφού γίνει αυτό, τότε υπολογίζεται η cdf για κάθε περιοχή.

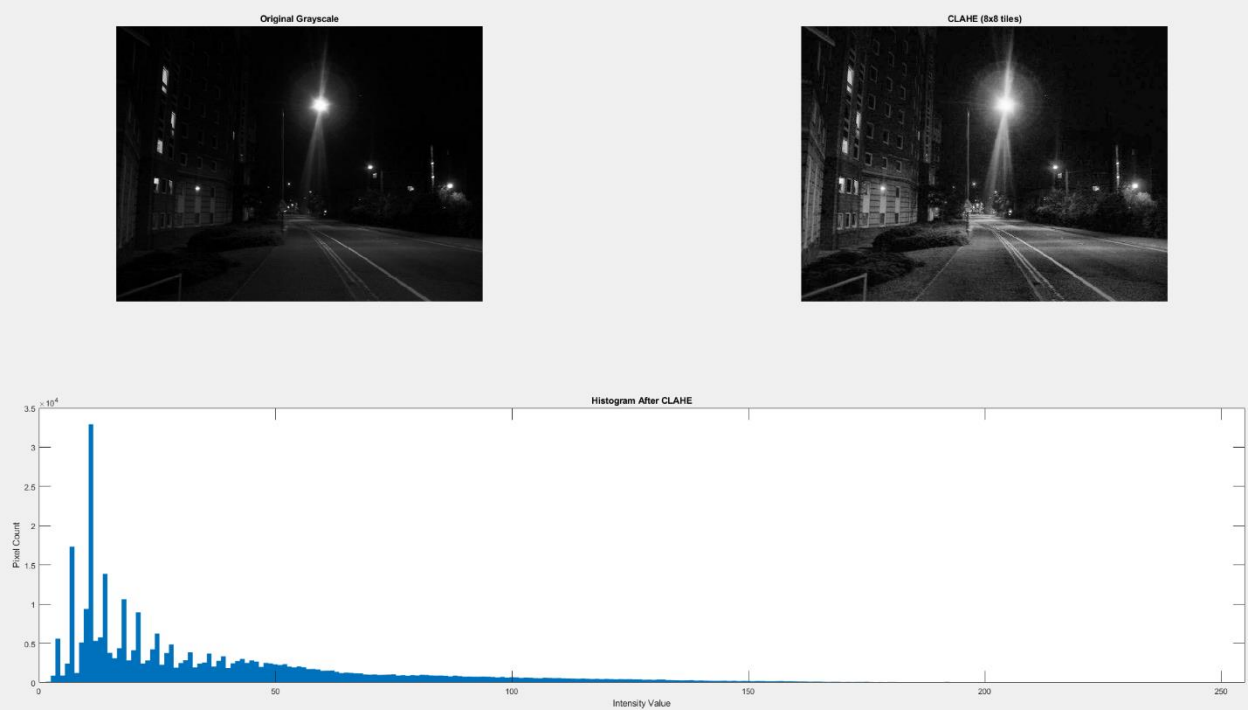
Επίσης, για να αποφύγουμε το «blockiness» που εμφανίζεται, εφαρμόζουμε διγραμμική παρεμβολή ώστε να μην υπάρχουν ασάφειες σε float τιμές.

Τέλος, η επιλογή ενός πλέγματος 8×8 αντανακλά μια ισορροπία μεταξύ στατιστικής αξιοπιστίας, κλίμακας χαρακτηριστικών και υπολογιστικής αποδοτικότητας. Τα tile πρέπει να είναι αρκετά μεγάλα ώστε να περιέχουν αρκετά pixel για ένα αξιόπιστο ιστόγραμμα, αλλά και αρκετά μικρά ώστε να ανταποκρίνονται σε ουσιαστικές τοπικές διακυμάνσεις. Άρα, όσο μεγαλύτερο

μέγεθος παραθύρου έχουμε, τόσο πιο πολύ τα αποτελέσματα μας θα μοιάζουν με την ολική εξίσωση ιστογράμματος, ενώ με μικρότερο έχουμε περισσότερες ανεξάρτητες περιοχές, χωρίς αυτό να σημαίνει ότι όσο πιο μικρό είναι τόσο καλύτερο είναι.







Τα αποτελέσματα είναι πολύ καλυτέρα από αυτά του προηγούμενο ερωτήματος. Όχι μόνο αυξήσαμε το contrast με επιτυχία και είναι εμφανής οι σκοτεινές περιοχές, άλλα ταυτόχρονα έχουμε καταφέρει να έχουμε ελάχιστο θόρυβο και σχεδόν καθόλου artifact. Αξιοσημείωτο γεγονός είναι ότι το ιστόγραμμα δεν ακολουθεί ομοιόμορφη κατανομή σε όλο του το φάσμα όπως με την κλασσική μέθοδο, άλλα αυτό είναι και ο παράγοντας για τα καλυτέρα αποτελέσματα μας. Αφού δεν είμαστε αναγκασμένοι να το εξισώσουμε ολόκληρο, τότε μπορούμε πραγματικά να αυξήσουμε το contrast χωρίς να καταστρέψουμε την εικόνα. Το μόνο μειονέκτημα αυτής της μεθόδου είναι ότι απαιτεί περισσότερο χρόνο και πολυπλοκότητα από μια απλή εξίσωση ιστογράμματος.

5. Αποκατάσταση εικόνας - Αποσυνέλιξη

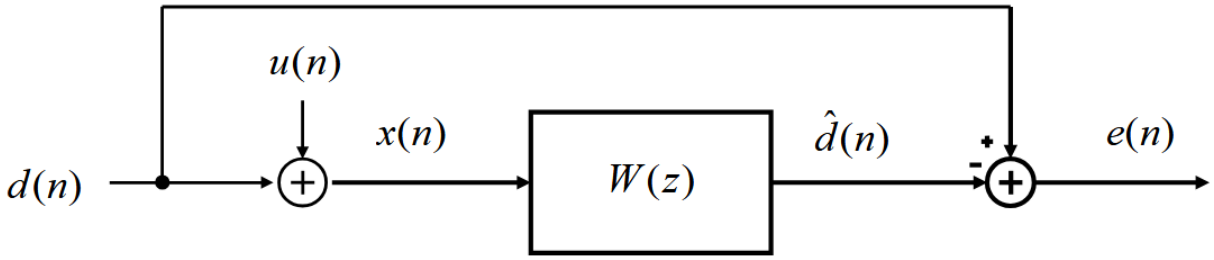
Μέρος Α

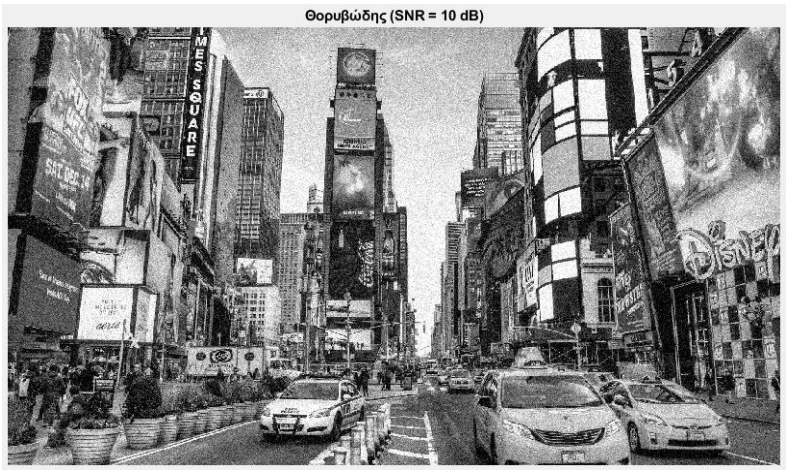
1)

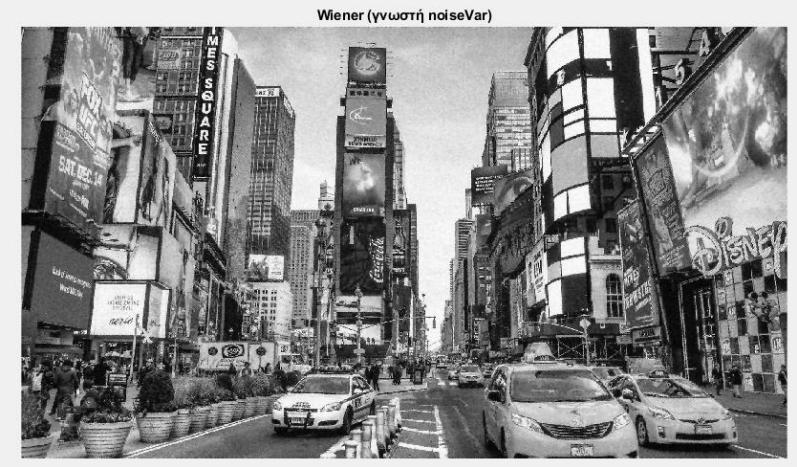
Αρχικά, το φίλτρο wiener απομακρύνει τον θόρυβο ελαχιστοποιώντας το σφάλμα στην έξοδο του. Αυτή περιγράφεται από την εξίσωση $\boldsymbol{d'(n) = x(n) + u(n)}$, όπου το d(n) είναι το σήμα εισόδου και u(n) είναι ο θόρυβος. Ο σκοπός μας είναι να ελαχιστοποιήσουμε το μέσο τετραγωνικό σφάλμα $\boldsymbol{E[(d(n) - d'(n))^2]}$ και συγκεκριμένα, αυτός είναι ο ορισμός για το βέλτιστο φίλτρο wiener. Ο τρόπος με τον οποίο γίνονται οι υπολογισμοί είναι με την εξίσωση wiener hopf:

$$\boldsymbol{R_{xx} * h = r_{dx} => h = R_{xx}^{-1} r_{dx}}$$

όπου το $\boldsymbol{R_{xx}}$ είναι το μητρώο αυτοσυσχέτισης του σήματος που έχει τον θόρυβο και $\boldsymbol{r_{dx}}$ είναι το διάνυσμα ετεροσυσχέτισης μεταξύ της «καθαρής» εισόδου και αυτής που περιέχει θόρυβο. Παρατηρούμε ότι για να λύσουμε αυτή την εξίσωση, πρέπει να γνωρίζουμε την ισχύ του θορύβου ώστε να βρούμε το μητρώο αυτοσυσχέτισης R_{xx}. Εφόσον έχουμε αυτή την γνώση για το συγκεκριμένο ερώτημα, η λύση του προβλήματος είναι απλή.



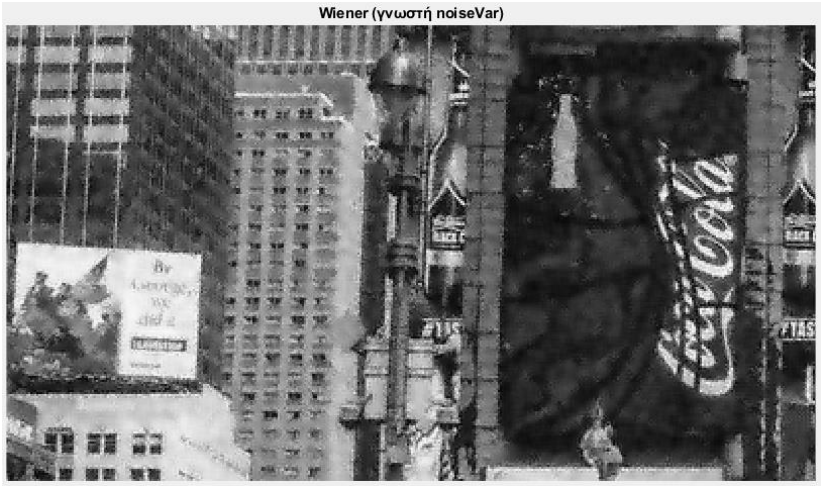




Από τα αποτελέσματα βλέπουμε ότι το φίλτρο έχει απομακρύνει μεγάλο ποσοστό του θορύβου, ενώ ταυτόχρονα έχει διατηρήσει τις λεπτομέρειες τις εικόνας πλήρως. Συμπερασματικά, ενώ δεν έχει διώξει εντελώς τον θόρυβο (δεν είναι ο σκοπός του wiener), τον έχει ελαττώσει σε αποδεκτό επίπεδο διατηρώντας παράλληλα όλες τις λεπτομερείς τις εικόνας χωρίς να υπάρχει καθόλου blurring ή κάποια άλλη παραμόρφωση.

2)

Εφόσον δεν γνωρίζουμε τον θόρυβο, τότε πρέπει να κάνουμε την εκτίμηση του, καθώς είδαμε παραπάνω ότι είναι αναγκαίο να τον γνωρίζουμε, ώστε να λύσουμε τις εξισώσεις του φίλτρου. Για λογούς απλότητας, το μόνο που αλλάζουμε στο συγκεκριμένο πρόβλημα είναι ότι δεν βάζουμε ως όρισμα την ισχύ του θορύβου στην built in συνάρτηση της matlab και τότε αυτή κάνει την εκτίμηση αυτόματα.



Συγκρίνοντας τα αποτελέσματα με αυτά του προηγουμένου ερωτήματος, παρατηρούμε ότι είναι αρκετά πιο blurred. Αυτό συμβαίνει επειδή η εκτίμηση του θορύβου δεν είναι πάντα ακριβής, με αποτέλεσμα το φίλτρο να εκτελεί μεγαλύτερες διορθώσεις από ότι χρειάζεται. Σε γενικές γραμμές, ο θόρυβος έχει απομακρυνθεί με επιτυχία και υπάρχει ελάχιστο blurring, το οποίο μάλιστα είναι ελάχιστα ορατό.

Μέρος Β

1)

Αρχικά, οφείλουμε να κατανοήσουμε τι είναι το psf. Όταν έχουμε ένα φίλτρο, ουσιαστικά κάνουμε συνέλιξη αυτού με την εικόνα. Αναλυτικότερα, περνάει κάθε pixel με το kernel του φίλτρου. Η psf περιγράφει λοιπόν ένα σημείο, δηλαδή το πως το kernel επηρεάζει ένα μόνο pixel. Όταν εφαρμόζουμε λοιπόν αυτό τον μετασχηματισμό, εννοούμε ότι απλά φιλτράρουμε την εικόνα.

Συνεχίζοντας, εφόσον μιλάμε για συνέλιξη, είναι λογικό να υποθέσουμε ότι το σύστημα μας είναι LTI. Τότε, για να βρούμε την κρουστική απόκριση αρκεί να βάλουμε ως είσοδο στο σύστημα μας της συνάρτηση δ. Συγκεκριμένα, αυτό ισχύει από τις ιδιότητες που έχουμε μάθει στα σήματα και συστήματα, επειδή είναι:

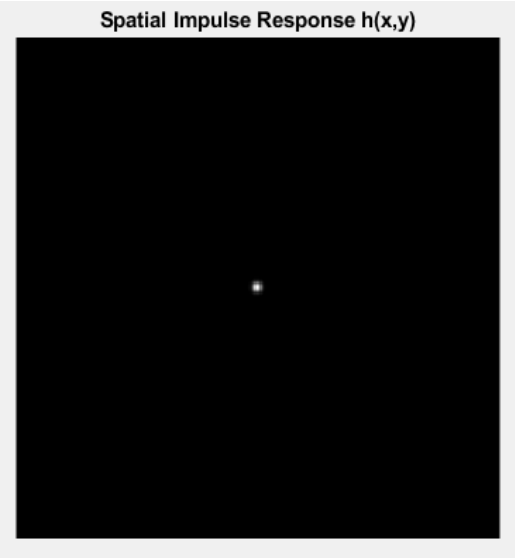
$$\boldsymbol{Y[m,n] = (h * x)[m,n] = \sum_{k,l} h[k,l]x[m-k,n-l]}$$

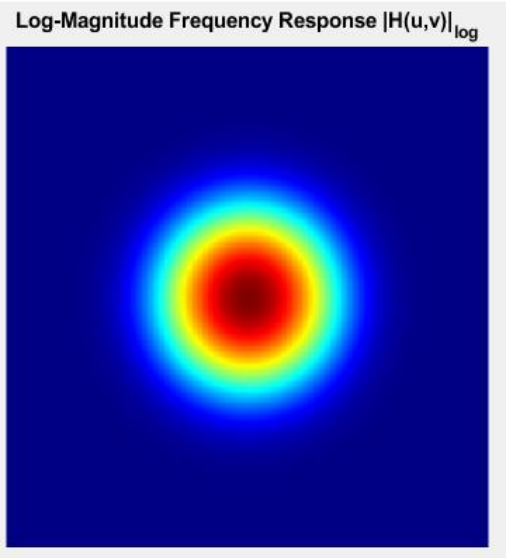
Και με είσοδο δ είναι:

$$\boldsymbol{\sum_{k,l} h[k,l]\delta[m-k,n-l] = h[k,l]}$$

Γνωρίζουμε από ιδιότητες ότι για να λυθεί αυτό το άθροισμα βρίσκουμε τα σημεία μηδενισμού της δ, τα οποία είναι m=k και n=l. Τότε, λύνουμε το άθροισμα και ισχύει ότι είναι η κρουστική συνάρτηση για τους περιορισμούς που θέσαμε.

Για την απόκριση συχνότητας, το μόνο που έχουμε να κάνουμε είναι να μεταφέρουμε στον χώρο της συχνότητας την κρουστική απόκριση. Επιλέγουμε να το κάνουμε αυτό με τον μετασχηματισμοί fourier.





Από τα αποτελέσματα βλέπουμε ότι το φίλτρο του αγνώστου συστήματος είναι low pass (blurring), καθώς στο πεδίο του χώρου υπάρχει μόνο μια εμφανής λεπτή κουκίδα και στο πεδίο της συχνότητας οι μονές που περνάνε είναι οι χαμηλές συχνότητες (το κόκκινο δείχνει ποιες περνάνε και το μπλε είναι το σημείο αποκοπής).

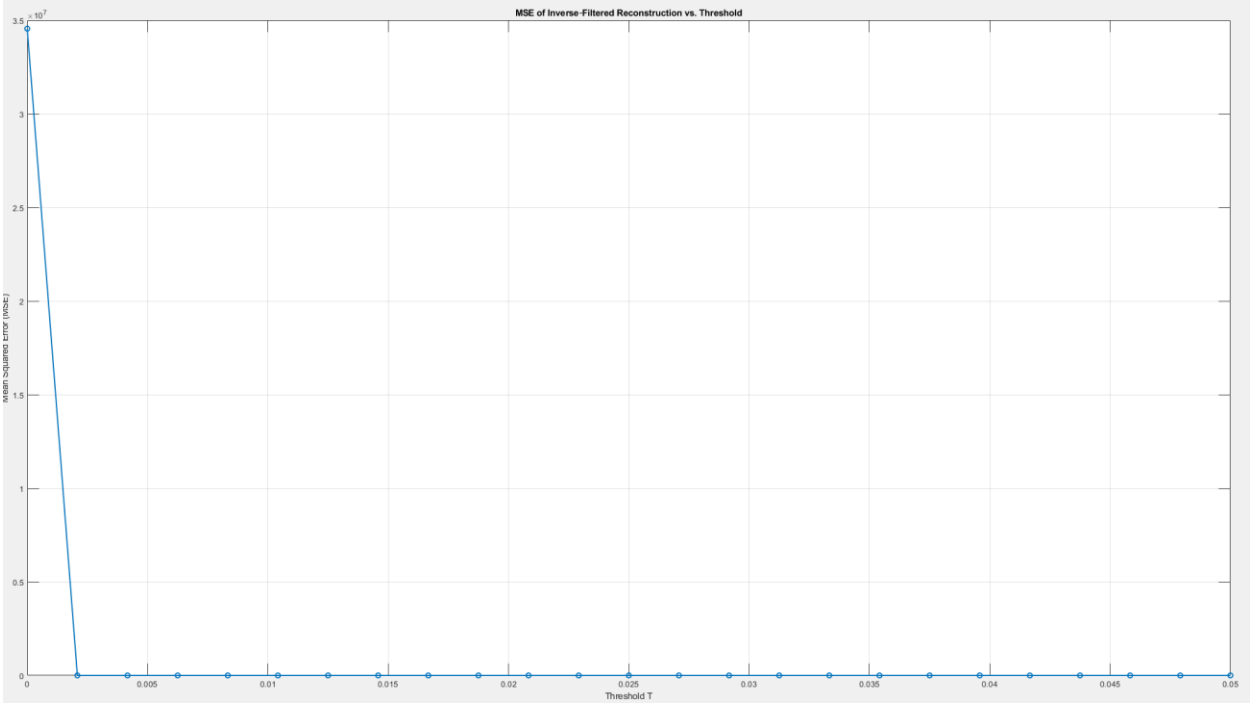
2)

Γνωρίζουμε ότι η θόλωση της εικόνας είναι συνέλιξη αυτής με το kernel του φίλτρου, δηλαδή είναι $\boldsymbol{b}(\boldsymbol{u},\boldsymbol{v}) = \boldsymbol{k}(\boldsymbol{u},\boldsymbol{v}) * \boldsymbol{s}(\boldsymbol{u},\boldsymbol{v})$, όπου b είναι η εικόνα μετά το φιλτράρισμα, k είναι το kernel, s είναι η αρχική εικόνα και u,v είναι όλες οι συχνότητες της εικόνας. Συνεχίζοντας, ξέρουμε ότι η συνέλιξη στο πεδίο του χρόνου είναι γινόμενο στο πεδίο της συχνότητας, άρα είναι $B(u,v) = K(u,v)S(u,v)$.

Η τεχνική αντίστροφου φίλτρου προσπαθεί να ανακτήσει την αρχική εικόνα από τον παραπάνω υπολογισμό, δηλαδή κάνουμε $S(\boldsymbol{u},\boldsymbol{v}) = \frac{\boldsymbol{B}(\boldsymbol{u},\boldsymbol{v})}{\boldsymbol{K}(\boldsymbol{u},\boldsymbol{v})}$. Το πρόβλημα που προκύπτει είναι ότι το K μπορεί να είναι πολύ μικρό για μερικές τιμές και

έτσι να λαμβάνουμε πολύ μεγάλο αποτέλεσμα. Για αυτό τον λόγο, χρησιμοποιούμε το threshold, ώστε να αποκλείουμε πολύ μικρές τιμές που θα μας χαλάνε το αποτέλεσμα. Πρακτικά, κάνουμε:

$$\boldsymbol{S(u,v) = \frac{B(u,v)}{K(u,v)}, \text{ όταν } K(u,v) \geq T} \text{ και } \boldsymbol{S(u,v) = 0 \text{ όταν } K(u,v) \leq T}$$



Βλέπουμε ότι αρχικά χωρίς το όριο, το σφάλμα ανακατασκευής είναι τεράστιο για τον λόγο που εξηγήσαμε παραπάνω. Ακόμα και με πολύ μικρές τιμές ορίου, το σφάλμα ελαττώνεται δραματικά και πρακτικά μηδενίζεται με όριο περίπου το 0.002. Όμως, το αποτέλεσμα που βλέπουμε δεν είναι μια τέλεια ανακατασκευή. Αυτό συμβαίνει επειδή έχουμε αποκλείσει πολλές συχνότητες και άρα έχουμε χάσει πληροφορία. Συμπερασματικά, το μηδενικό σφάλμα δεν σημαίνει ότι έχουμε το βέλτιστο οπτικά αποτέλεσμα.

3)

Έχει απαντηθεί στο προηγούμενο ερώτημα.

6. Ανίχνευση ακμών

1)

Το συγκεκριμένο φίλτρο χρησιμοποίει 3χ3 μάσκες για την εύρεση ακμών. Συγκεκριμένα, έχει μια 3χ3 μάσκα G_x για τις οριζόντιες ακμές και μια 3χ3 μάσκα G_y για τις κάθετες ακμές οι οποίες είναι αντίστοιχα:

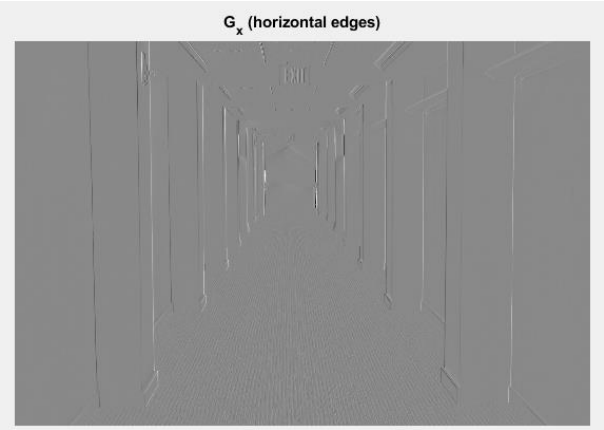
$$\boldsymbol{G_x} = \begin{matrix} \mathbf{-1} & \mathbf{0} & \mathbf{+1} \\ \mathbf{-2} & \mathbf{0} & \mathbf{+2} \\ \mathbf{-1} & \mathbf{0} & \mathbf{+1} \end{matrix} \qquad \boldsymbol{G_y} = \begin{matrix} \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{+1} & \mathbf{+2} & \mathbf{+1} \end{matrix}$$

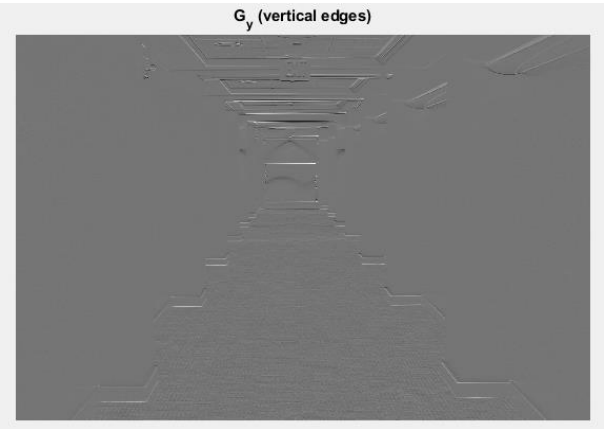
Στην συνέχεια, πολλαπλασιάζουμε κάθε pixel της εικόνας με το αντίστοιχο της μάσκας. Για παράδειγμα, έστω ότι έχουμε την εικόνα $\boldsymbol{S} = \begin{matrix} \mathbf{25} & \mathbf{35} & \mathbf{45} \\ \mathbf{30} & \mathbf{40} & \mathbf{50} \\ \mathbf{35} & \mathbf{45} & \mathbf{55} \end{matrix}$.
Πολλαπλασιάζουμε όλα τα στοιχεία του μητρώου με το αντίστοιχο του G_x και αθροίζοντας το αποτέλεσμα έχουμε 80. Κάνουμε το ιδιο και για G_y και έχουμε αποτέλεσμα 40. Αυτά τα νούμερα μας δείχνουν την κατεύθυνση της ακμής.

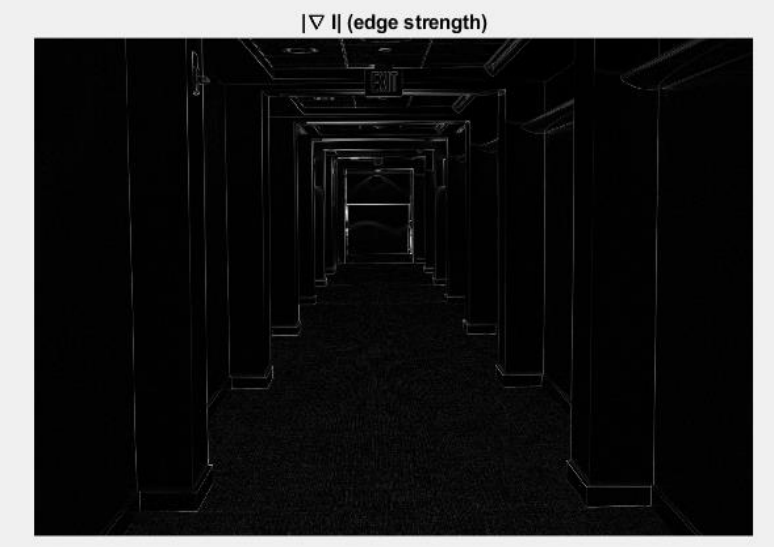
Τέλος, συνδυάζουμε τα αποτελέσματα ώστε να λάβουμε την δύναμη της ακμής:

$$|\boldsymbol{G}| = \sqrt{\boldsymbol{G_x^2 + G_y^2}} = \sqrt{\mathbf{80^2 + 40^2}} = \mathbf{89.4}$$

https://homepages.inf.ed.ac.uk/rbf/HIPR2/sobel.htm





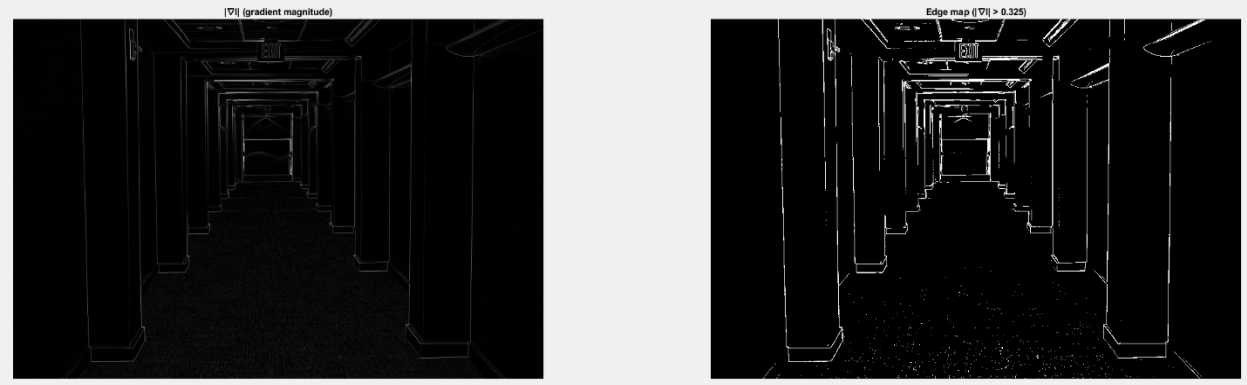


Βλέπουμε ότι τα φίλτρα λειτουργούν σωστά και εντοπίζονται οι ακμές όπως πρέπει. Όπως είδαμε στο μάθημα, το G_x τελικά βρίσκει τις κάθετες ακμές και το G_y τις οριζόντιες (είναι το αντίθετο από ότι περιμέναμε διαισθητικά). Το τελικό αποτέλεσμα συνδυάζει ότι έχουμε βρει και βλέπουμε ότι πράγματι έχει βρει τις ακμές.

2)

Με την ολική κατοφλίωση, μετατρέπουμε την εικόνα μόνο σε έναν χάρτη ακμών. Συγκεκριμένα, θέτουμε ένα όριο για το τελικό G που υπολογίζουμε και οποιαδήποτε τιμή είναι εντός αυτού, μετατρέπεται σε 1 (για overlapping τιμές που είναι σε πολλαπλά 3χ3 παράθυρα φροντίζει ο αλγόριθμος να τα βάλει σε μια μόνο περιοχή). Ότι είναι κάτω από το όριο γίνεται 0.

Για την επιλογή κατωφλιού, επιλέγουμε την μέθοδο Otsu. Αυτή έχει στόχο να βρει αυτόματα το ιδανικό σημείο διαχωρισμού που χωρίζει ένα σύνολο τιμών σε δύο ξεκάθαρες κατηγορίες. Η μια από αυτές είναι “εμφανή” και η άλλη “μη εμφανή” pixel. Αναλυτικότερα, ο αλγόριθμος εξετάζει το ιστόγραμμα και υποθέτει ότι αυτό μπορεί να περιγραφεί ως μίγμα δύο κατηγοριών. Για κάθε πιθανή τιμή κατωφλίου, «σπάει» το ιστόγραμμα σε μία «κάτω» και μία «πάνω» κατηγορία. Κατόπιν, μετρά πόσο «σφιχτά» είναι συγκεντρωμένη η κάθε κατηγορία γύρω από τον δικό της μέσο όρο (αυτή η μέτρηση ονομάζεται inner class διασπορά). Σκοπός είναι να επιλεγεί το κατώφλι που κάνει και τις δύο κατηγορίες όσο το δυνατόν πιο συμπαγείς, δηλαδή να ελαχιστοποιεί το άθροισμα των ενδο-διασπορών τους ενώ ταυτόχρονα μεγιστοποιεί την διασπορά αναμεσά στις δυο κλασσεις (δηλαδή να μην μοιάζουν όσο το δυνατόν περισσότερο).



Βλέπουμε ότι οι ακμές είναι ακόμα πιο εμφανής με την συγκεκριμένη μέθοδο, δείχνοντας ότι βρήκε με επιτυχία τα classes και τις διασπορές που ορίσαμε.

Bonus

Αρχικά, εφαρμόζεται ένας ανιχνευτής ακμών (το έχουμε κάνει στο προηγούμενο ερώτημα) στην αρχική εικόνα. Αυτές οι ακμές είναι υποψήφιες για να ανήκουν σε μία ή περισσότερες ευθείες και κάθε ακμή «ψηφίζει» για όλες τις ευθείες που θα μπορούσαν να την περιέχουν. Συνεχίζοντας, γράφουμε την ευθεία στη μορφή πολικών συντεταγμένων ως:

$$\boldsymbol{\rho = xcos\theta + ysin\theta}$$

όπου θ είναι η γωνία της ευθείας σε σχέση με τον άξονα x και ρ είναι η κάθετη απόσταση από την αρχή των αξόνων. Για ένα δεδομένο (x, y), καθώς η θ διατρέχει το διάστημα 0–180°, οι αντίστοιχες τιμές ρ σχηματίζουν μια ημιτονοειδή καμπύλη στον χώρο παραμέτρων (ρ, θ). Ένας πίνακας οργανώνεται με διακριτά κελιά για θ και ρ, δηλαδή κάθε φορά που ένα σημείο ακμής ψηφίζει, αυξάνει την τιμή σε όλα τα κελιά ($\boldsymbol{\theta_i}$, $\boldsymbol{p_j}$) κατά μήκος της καμπύλης του.

Αφού συμπληρωθεί ο πίνακας, αναζητούμε αυτές τις κορυφές των οποίων οι ψήφοι υπερβαίνουν ένα επιλεγμένο όριο. Κάθε ανιχνευμένη κορυφή αντιπροσωπεύει μια υποψήφια ευθεία στην εικόνα. Επιστρέφοντας αυτές τις παραμέτρους στο πεδίο της εικόνας, μπορούμε να σχεδιάσουμε την ευθεία ή πιο πρακτικά, να εξάγουμε τα τμήματά της. Επεκτείνοντας, εντοπίζουμε ποια αρχικά σημεία ακμής συνέβαλαν στην κορυφή (δηλαδή βρίσκονται κοντά στην ευθεία) και υπολογίζουμε τα άκρα του τμήματος βρίσκοντας τις ελάχιστες και μέγιστες προβολές κατά μήκος της διεύθυνσης της ευθείας.

https://medium.com/@shantanuparab99/hough-transform-3e5f6875b9b8


Original + Hough Lines (red)
EXIT