ΤΕΧΝΟΛΟΓΙΚΟ ΕΚΠΑΙΔΕΥΤΙΚΟ ΙΔΡΥΜΑ ΣΕΡΡΩΝ
ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΑΠΟΚΑΤΑΣΤΑΣΗ ΘΟΡΥΒΟΠΟΙΗΜΕΝΩΝ ΕΙΚΟΝΩΝ ΑΠΟ ΠΡΟΣΘΕΤΙΚΟ ΘΟΡΥΒΟ ΜΕ ΧΡΗΣΗ ΧΩΡΙΚΩΝ ΦΙΛΤΡΩΝ

Αλτιντζής Γεώργιος - ΑΕΜ: 2352

Επιβλέπων: Χαράλαμπος Στρουθόπουλος - Καθηγητής

ΣΕΡΡΕΣ, ΝΟΕΜΒΡΙΟΣ 2011

ΠΕΡΙΕΧΟΜΕΝΑ

ΠΡΟΛΟΓΟΣ

Αυτή η πτυχιακή εργασία έχει ως σκοπό την δημιουργία λογισμικού για την μελέτη και παρουσίαση τεχνικών αποκατάστασης θορυβοποιημένων ψηφιακών εικόνων από διάφορες μορφές προσθετικού θορύβου. Οι τεχνικές αυτές (φίλτρα) αφορούν την επεξεργασία της εικόνας στο πεδίο του χώρου, δηλαδή την επεξεργασία χωρίς να προηγηθεί κάποιο είδος μετασχηματισμού (π.χ. Fourier). Αποτελείται από δύο κύρια μέρη. Το πρώτο είναι η δημιουργία βασικών μοντέλων θορύβου και το δεύτερο η δημιουργία βασικών ειδών φίλτρου. Παράλληλα παρουσιάζεται η βασική θεωρία που αφορά το εν λόγω αντικείμενο. Το λογισμικό δίνει την δυνατότητα εύκολης ρύθμισης των παραμέτρων των μοντέλων και την χρηστική παρουσίαση των αποτελεσμάτων.

Τελειώνοντας, θα ήθελα να ευχαριστήσω τον καθηγητή μου Στρουθόπουλο Χαράλαμπο για την πολύτιμη και καθοριστική συμβολή του από τη θέση του επιβλέποντα της πτυχιακής μου εργασίας, τους γονείς μου, Αναστάσιο και Γεωργία, για τους κόπους τους να σπουδάζω έως και σήμερα, τα αδέρφια μου, τους φίλους μου, τους συμφοιτητές μου που με στήριξαν ως φίλοι και ως συνάδελφοι στην φοιτητική μου πορεία και πάνω από όλους τον Θεό που με βοήθησε να φτάσω ως αυτό το σημείο.

ΕΙΣΑΓΩΓΗ

Τι είναι η Ψηφιακή Επεξεργασία Εικόνας (ΨΕΕ);

Η ψηφιακή επεξεργασία εικόνας (ΨΕΕ) αποτελεί έναν ευρύ επιστημονικό κλάδο που αναπτύχθηκε με την ραγδαία εξέλιξη των υπολογιστών. Τα υπερηχογραφήματα, οι μαγνητικές τομογραφίες, οι δορυφορικές φωτογραφίες κ.α. μπορούν να επεξεργαστούν ως ψηφιακές εικόνες.

Οι στόχοι της ΨΕΕ είναι οι εξής:

1) Η ψηφιοποίηση και κωδικοποίηση εικόνων με σκοπό την αποθήκευση, μετάδοση και εκτύπωσή τους.
2) Η βελτίωση και η αποκατάσταση των εικόνων με σκοπό την καλύτερη απεικόνισή τους.
3) Η ανάλυση και κατανόηση των εικόνων

Η ΨΕΕ συνεργάζεται με τους παρακάτω επιστημονικούς κλάδους:

1) Ψηφιακή Επεξεργασία Σημάτων (ΨΕΣ)
2) Ρομποτική όραση
3) Τεχνητή Νοημοσύνη
4) Αναγνώριση Προτύπων
5) Νευρωνικά Δίκτυα
6) Ασαφής Λογική
7) Κωδικοποίηση
8) Γραφικά Η/Υ

Η μετατροπή μιας εικόνας σε ψηφιακή μορφή ουσιαστικά είναι η μετατροπή ενός δυσδιάστατου αναλογικού σήματος σε ψηφιακό και απαιτεί τις διαδικασίες της δειγματοληψίας και του κβαντισμού. Από την μετατροπή αυτή με όρους της ψηφιακής επεξεργασίας σήματος προκύπτει, ένα δυσδιάστατο διακριτό σήμα πεπερασμένου μήκους και πλάτους. Στις περισσότερες των περιπτώσεων, μια ψηφιακή εικόνα είναι ένα ορθογώνιο, διαιρεμένο με γραμμές και στήλες σε ορθογώνιες περιοχές που κάθε μία έχει συγκεκριμένο χρώμα. Μια τέτοια περιοχή ονομάζεται στοιχείο της εικόνας ή εικονοστοιχείο. Στην αγγλική λέγεται

pixel ή *pel*, όρος ο οποίος προέρχεται από τη σύντμηση των λέξεων picture element.

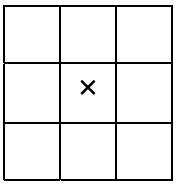
Ποια είναι τα είδη των ψηφιακών εικόνων;

Υπάρχουν τρία είδη ψηφιακών εικόνων που χαρακτηρίζονται από το πλήθος των χρωμάτων που περιέχουν:

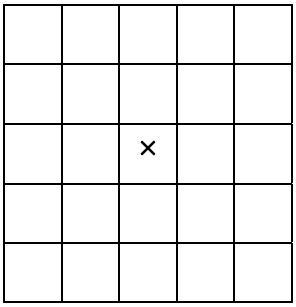
1) *Δυαδικές εικόνες* (*binary images*): Κάθε εικονοστοιχείο των εικόνων μπορεί να χρωματιστεί με ένα από δύο χρώματα.(συνήθως άσπρο ή μαύρο). Για κάθε εικονοστοιχείο απαιτείται ένα bit πληροφορίας, π.χ. με τιμή μηδέν (0) για το μαύρο και ένα (1) για λευκό. Οι εικόνες των εγγράφων που αποτελούνται μόνο_ από το χρώμα του χαρτιού και της μελάνης αναπαρίστανται σε δυαδική ψηφιακή μορφή.
2) *Εικόνες αποχρώσεων του γκρι* (*gray level images*): Κάθε εικονοστοιχείο των εικόνων μπορεί να χρωματιστεί με μία από τις αποχρώσεις του γκρι οι οποίες ξεκινούν από το μαύρο και καταλήγουν στο λευκό. Από αυτές τις αποχρώσεις συνήθως λαμβάνονται 256 αντιπροσωπευτικές που κωδικοποιούνται με τιμές 0,1,2….255. Η απόχρωση κάθε εικονοστοιχείου προφανώς απαιτεί πληροφορία ενός byte.
3) *Έγχρωμες εικόνες* (*color images*) στις οποίες κάθε εικονοστοιχείο χρωματίζεται με χρώματα που προέρχονται από την ανάμειξη των αποχρώσεων του κόκκινου, πράσινου και μπλε (*RGB*). Για κάθε ένα από τα τρία αυτά χρώματα λαμβάνονται 256 αποχρώσεις δηλαδή πληροφορία του ενός byte. Συνεπώς κάθε εικονοστοιχείο της έγχρωμης εικόνας, απαιτεί 3 bytes.

Τι ονομάζεται γειτονιά εικονοστοιχείων;

Μια ομάδα γειτονικών εικονοστοιχείων λέγεται γειτονιά. Σε μια γειτονιά $S_{M\times N}$ με M γραμμές και N στήλες μιας εικόνας διαστάσεων $J\times K$ υπάρχει ένα κεντρικό εικονοστοιχείο (j_c,k_c) όταν M, N είναι περιττοί αριθμοί. Η θέση των εικονοστοιχείων της S αναφέρονται συχνά, σχετικά με την θέση του κεντρικού εικονοστοιχείου της. Η πιο συνήθης γειτονιά είναι τριών (3) γραμμών και τριών (3) στηλών και λέγεται γειτονιά 3×3.



Γειτονιά 3×3



Γειτονιά 5×5

Γειτονιές 3×3 και 5×5 με τα κεντρικά τους εικονοστοιχεία.

Για παράδειγμα για N=M=3 ο πίνακας *S* είναι

w(-1,-1)	w(-1,0)	w(-1,1)
w(0,-1)	w(0,0)	w(0,1)
w(1,-1)	w(1,0)	w(1,1)

Οι συντελεστές της μάσκας και οι τιμές φωτεινότητας των εικονοστοιχείων μπορούν να εμπλακούν σε χρήσιμους υπολογισμούς για την επεξεργασία της εικόνας. Ο συνηθέστερος υπολογισμός δίνεται από την σχέση

$$A(j,k) = \sum_m \sum_n w(m,n) \cdot I(j+m, k+n)$$

όπου (j,k) εικονοστοιχείο της εικόνας *I*.

Αν (j_c,k_c) είναι το κεντρικό εικονοστοιχείο της γειτονιάς *S* της εικόνας $I_{J\times K}$ η τιμή *A* μπορεί να αποδοθεί ως τιμή φωτεινότητας του εικονοστοιχείου (j_c,k_c) μιας νέας εικόνας $I'_{J\times K}$. Αν αυτό εφαρμοσθεί για όλες τις γειτονιές της εικόνας *I*, τότε λέμε ότι η νέα εικόνα *I΄* προέκυψε από το φιλτράρισμα της *I* με την μάσκα *W.* Αν περιγράψουμε μια τέτοια πράξη με όρους της ψηφιακής επεξεργασίας σήματος τότε η εφαρμογή της μάσκας *W* σε όλα τα εικονοστοιχεία της εικόνας ισοδυναμεί με την έξοδο ενός γραμμικού και ανεξάρτητου από την μετατόπιση (LTI) συστήματος με απόκριση κρουστικής διέγερσης της μορφής

$$h(j,k) = \sum_m \sum_n w(m,n) \cdot \delta(j+m, k+n)$$

h=	w(1,1)	w(1,0)	w(1,-1)
	w(0,1)	w(0,0)	w(0,-1)
	w(-1,1)	w(-1,0)	w(-1,-1)

και θα δίνεται από τη σχέση *I΄= I**h*.

ΚΕΦΑΛΑΙΟ 1: Αποκατάσταση ψηφιακών εικόνων

Κατά την δημιουργία ή την επεξεργασία μιας ψηφιακής εικόνας προκαλούνται διάφορες αλλοιώσεις. Λέμε τότε ότι η εικόνα έχει υποβαθμιστεί Οι αλλοιώσεις αυτές οφείλονται κυρίως στις χρησιμοποιούμενες συσκευές. Για παράδειγμα κατά την λήψη μιας εικόνας και την μετατροπή της από αναλογική σε ψηφιακή έχει ως αποτέλεσμα την υποβάθμισή της. Η υποβάθμιση αυτή θεωρείται ως αποτέλεσμα της εφαρμογής θορύβου (θορυβοποίηση). Σχετικά παραδείγματα αποτελούν:

- Η καταγραφή μιας εικόνας με αισθητήρες (πχ CCD: charged coupled device)
- Η μετάδοση μιας εικόνας ηλεκτρονικά (όπως με το FAX)
- Η αποτύπωση μιας εικόνας σε film
- Η λήψη μιας εικόνας από δορυφόρους εξαιτίας της ατμόσφαιρας

Σε κάθε ένα από τα παραπάνω παραδείγματα αυτά που επηρεάζουν είναι οι κλιματολογικές συνθήκες (θερμοκρασία, υγρασία, φωτεινότητα, άσχημες καιρικές συνθήκες) κατά τη δημιουργία/λήψη των εικόνων, η ποιότητα των συσκευών των αισθητήρων και οι παρεμβολές από σήματα που χρησιμοποιούν τα ίδια κανάλια μετάδοσης.

Η αποκατάσταση μιας εικόνας αφορά την άρση ή μείωση της των αλλοιώσεων και πρέπει να εξυπηρετεί τον σκοπό της χρήσης που θα επακολουθήσει.

Η αποκατάσταση μιας εικόνας στοχεύει στην καλυτέρευση των ιδιοτήτων της είτε αυτές αφορούν την μείωση του θορύβου είτε την καλυτέρευση των χρωμάτων, της φωτεινότητας, της αντίθεσης κλπ. Η αποκατάσταση των εικόνων γίνεται συνήθως με το φιλτράρισμα τους στο πεδίο του χώρου ή στο πεδίο της συχνότητας και στόχοι είναι

α) η αποκατάσταση της εικόνας ώστε να είναι οπτικά αναγνωρίσιμη και

β) η άμεση αξιοποίηση των πληροφοριών της εικόνας.

ΚΕΦΑΛΑΙΟ 2: Μοντέλα θορύβου

Ο θόρυβος εμφανίζεται με δύο μορφές. Μία είναι ο προσθετικός και μία άλλη ο πολλαπλασιαστικός θόρυβος. Αν **I** είναι η αρχική εικόνα χωρίς θόρυβο και **n** ο θόρυβος, τότε η εικόνα με προσθετικό ή πολλαπλασιαστικό θόρυβο **Î** προκύπτει από τις παρακάτω σχέσεις:

Θορυβοποίηση με προσθετικό θόρυβο: **Î = I + n**

Θορυβοποίηση με πολλαπλασιαστικό θόρυβο: **Î = n · I**

Ο προσθετικός θόρυβος συνηθίζεται να έχει μέση τιμή ίση με το μηδέν και χαρακτηρίζεται από τη μεταβλητότητά του σ_n^2. Το αποτέλεσμα του θορύβου στις εικόνες μπορεί να χαρακτηριστεί από το λόγο σήματος προς θόρυβο (SNR, signal to noise ratio) που ορίζεται ως:

$$SNR = \frac{\sigma s}{\sigma n} = \sqrt{\frac{\sigma f^2}{\sigma n^2} - 1}$$

όπου σ_n^2 και σ_f^2 είναι οι μεταβλητότητες της εικόνας χωρίς θόρυβο και της εικόνας με θόρυβο, αντίστοιχα.

Ένας άλλος δείκτης είναι ο PSNR (peak to noise ratio) που ισούται με το λόγο του μέγιστου σήματος προς θόρυβο μεταξύ δύο εικόνων και μετριέται σε decibels. Μεγάλη τιμή PSNR σημαίνει λιγότερο θόρυβο στην εικόνα και αντίστροφα. Το PSNR ορίζεται ως:

$$PSNR = 10\log_{10}\left(\frac{V^2}{MSE}\right)$$

όπου το V ισούται με τη μέγιστη διακύμανση της αθορυβοποίητης εικόνας.

Αν πρόκειται για μια εικόνα αποχρώσεων του γκρι και η τιμή του κάθε εικονοστοιχείου κωδικοποιείται με 8 bits, θα είναι $V = 2^8 - 1 = 255$. Πιο γενικά, όταν η φωτεινότητα ισοδυναμεί με k bits τότε $V = 2^k - 1$. Για τις έγχρωμες εικόνες, το V καθορίζεται με τον ίδιο τρόπο, με τη διαφορά ότι το MSE είναι τώρα ίσο με τη μέση τιμή των τριών χρωματικών συνιστωσών. Συνεπώς για μια RGB εικόνα θα ισχύει:

$$PSNR = 10\log_{10}\left(\frac{V^2}{\frac{MSE(R)+MSE(G)+MSE(B)}{3}}\right)$$

Η συνάρτηση που περιγράφει μια υποβαθμισμένη/θορυβοποιημένη εικόνα στο πεδίο του χώρου είναι η ακόλουθη:

$$\hat{f}(k,j) = h(k,j) * f(k,j) + n(k,j)$$

όπου $\hat{f}$ η θορυβοποιημένη εικόνα, f η αρχική εικόνα, h η μοναδιαία κρουστική απόκριση και n ο θόρυβος που αλλοιώνει την εικόνα.

Ωστόσο, η συνέλιξη της αρχικής εξίσωσης της εικόνας f με τη μοναδιαία κρουστική απόκριση h έχει ως αποτέλεσμα την ίδια την f. Οπότε η εξίσωση του θορύβου n λειτουργεί μόνο προσθετικά στην εξίσωση της εικόνας f:

$$\hat{f}(k,j) = f(k,j) + n(k,j)$$

Ακολούθως θα αναφερθούμε σε μοντέλα θορύβου που δεν εξαρτώνται (είναι ασυσχέτιστα) με την τιμή ή την θέση των εικονοστοιχείων της εικόνας που προσβάλουν. Με την προϋπόθεση αυτή το μέγεθος που μας ενδιαφέρει για την περιγραφή του θορύβου στο πεδίο του χώρου είναι συνάρτηση πυκνότητας πιθανότητας των τιμών που προστίθενται στην τιμή του εικονοστοιχείου. Με άλλα λόγια οι τιμές του θορύβου θεωρούνται τιμές τυχαίας μεταβλητής και θεωρούμε την στατιστική συμπεριφορά της.

EXPONENTIAL NOISE

ΕΚΘΕΤΙΚΟΣ ΘΟΡΥΒΟΣ

Η συνάρτηση πυκνότητας πιθανότητας (σππ) του εκθετικού θορύβου δίνεται από την εξίσωση:

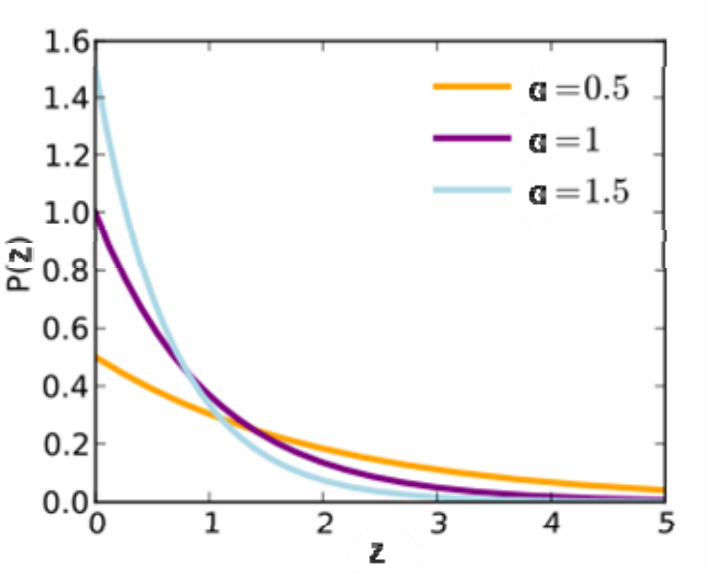
$$p(z) = \begin{cases} ae^{-az}, & a \geq 0 \\ 0, & a < 0 \end{cases}$$

όπου η παράμετρος α παίρνει τιμές μεγαλύτερες του μηδενός. Η μέση τιμή και η διασπορά που σχετίζονται με αυτήν την πυκνότητα ορίζονται ως:

$$\bar{z} = \frac{1}{a} \quad \text{και} \quad \sigma^2 = \frac{1}{a^2}$$

Αυτή η σππ αποτελεί μία ειδική περίπτωση της σππ τύπου Erlang και αντιστοιχεί στην περίπτωση όπου b = 1.

Ο εκθετικός εμφανίζεται σε απεικονίσεις που γίνονται με τη βοήθεια των ακτίνων laser.



UNIFORM DISTRIBUTION

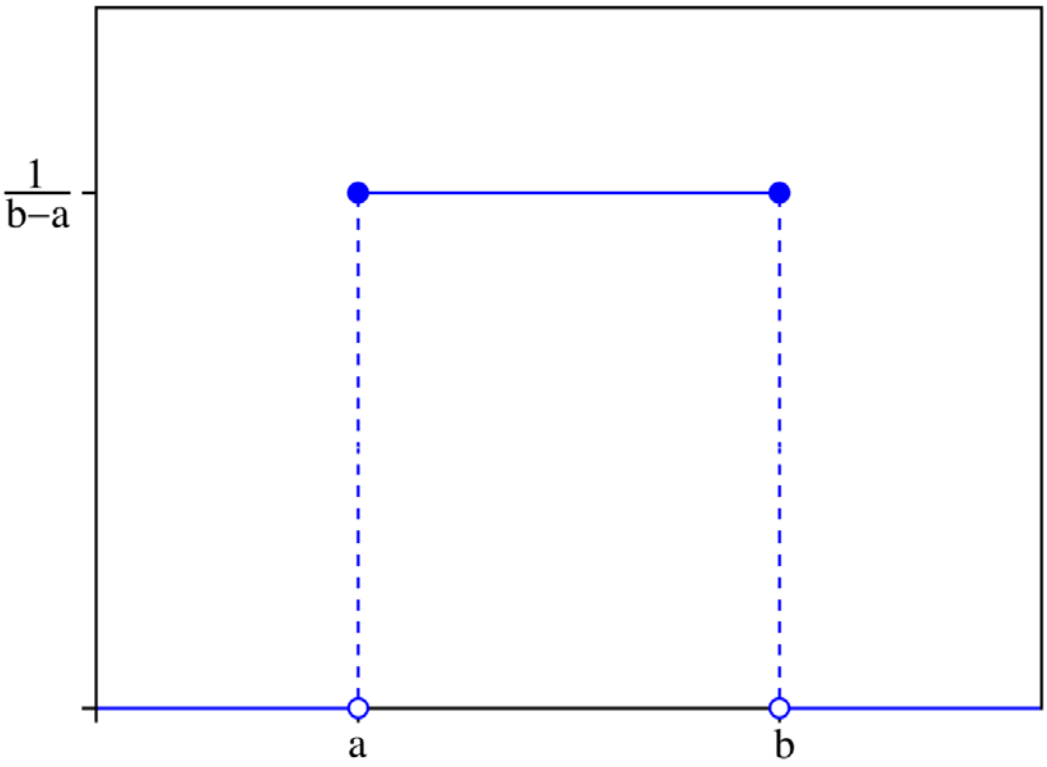
ΟΜΟΙΟΜΟΡΦΟΣ ΘΟΡΥΒΟΣ

Η συνάρτηση πυκνότητας πιθανότητας του ομοιόμορφου θορύβου δίνεται από την εξίσωση:

$$p(z) = \begin{cases} \dfrac{1}{b-\alpha}, \gamma\iota\alpha\ \alpha \le z \le b \\ 0, \alpha\lambda\lambda\iota\acute{\omega}\varsigma \end{cases}$$

Η μέση τιμή και η διασπορά που σχετίζονται με αυτήν την σππ ορίζονται ως:

$$\bar{z} = \frac{\alpha + b}{2} \quad \text{και} \quad \sigma^2 = \frac{(b-\alpha)^2}{12}$$



Η εμφάνιση του ομοιόμορφου θορύβου οφείλεται κυρίως στον κβαντισμό των αποχρώσεων σε διακριτά επίπεδα, δηλαδή κατά τη μετατροπή μιας αναλογικής εικόνας σε ψηφιακή.

Αντιμετωπίζεται καλύτερα όταν εφαρμοστούν τα εξής φίλτρα:

- το φίλτρο αριθμητικού μέσου (Arithmetic Mean filter)
- το φίλτρο αντι-αρμονικού μέσου (Contra-Harmonic Mean filter)
- το φίλτρο γεωμετρικού μέσου (Geometric Mean filter)
- το φίλτρο αρμονικού μέσου (Harmonic Mean filter)
- το φίλτρο μέσου σημείου (Midpoint filter)

BIPOLAR NOISE

ΔΙΠΟΛΙΚΟΣ ΘΟΡΥΒΟΣ

Η συνάρτηση πυκνότητας πιθανότητας του διπολικού θορύβου δίνεται από την εξίσωση:

$$p(z) = \begin{cases} p\alpha \text{ , για } z = \alpha \\ pb \text{ , για } z = b \\ 1 - p\alpha - pb \text{ , για } z = 0 \end{cases}$$

Αποτελεί τη γενικευμένη περίπτωση του διπολικού κρουστικού θορύβου με την διαφορά ότι οι τιμές του δεν προκαλούν την αλλοίωση της εικόνας που προκαλούν οι τιμές του τελευταίου με το να εμφανίζονται οι αρνητικές κρούσεις ως μαύρα σημεία πάνω στην εικόνα (ως κόκκοι πιπεριού) και οι θετικές κρούσεις ως λευκές κουκίδες (ως κόκκοι αλατιού).

IMPULSE NOISE

ΚΡΟΥΣΤΙΚΟΣ ΘΟΡΥΒΟΣ (ΑΛΑΤΟΠΙΠΕΡΟΥ)

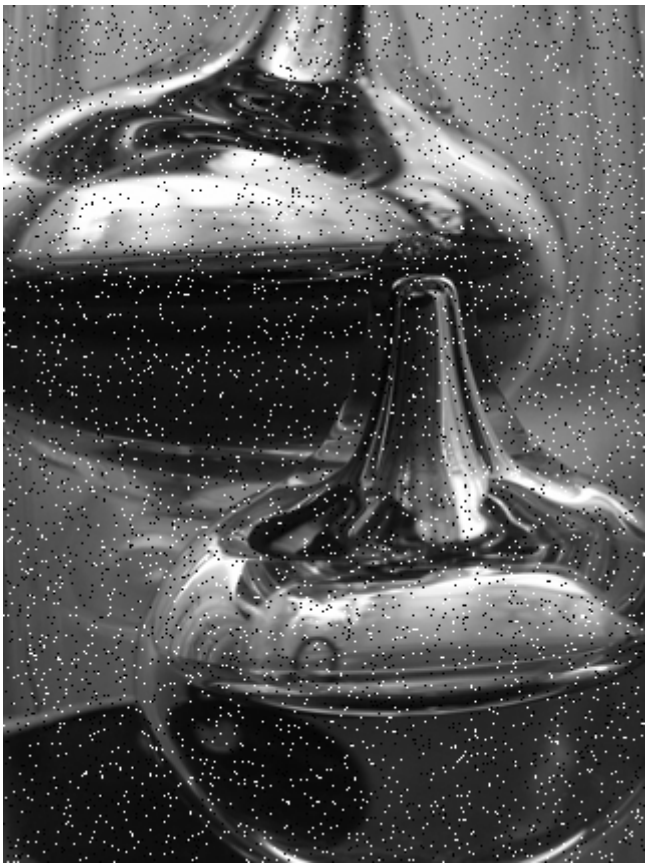
Η συνάρτηση πυκνότητας πιθανότητας του διπολικού κρουστικού θορύβου αλατοπίπερου δίνεται από την εξίσωση:

$$p(z) = \begin{cases} p\alpha \text{ , } \gamma\iota\alpha \ z = \alpha \\ pb \text{ , } \gamma\iota\alpha \ z = b \\ 1 - p\alpha - pb \text{ , } \gamma\iota\alpha \ z = 0 \end{cases}$$

Εάν είναι b > α, η ένταση b θα εμφανίζεται ως μία φωτεινή κουκίδα πάνω στην εικόνα. Αντίστροφα το επίπεδο α θα εμφανίζεται ως μία σκοτεινή κουκίδα. Εάν είτε το Pa είτε το Pb είναι ίσο με το μηδέν, ο θόρυβος ονομάζεται μονοπολικός. Εάν αντίθετα καμία από αυτές τις πιθανότητες δεν έχει μηδενική τιμή και ειδικότερα εάν είναι κατά προσέγγιση ίσες μεταξύ τους, οι τιμές του κρουστικού θορύβου θα μοιάζουν με κόκκους αλατοπίπερου που θα είναι κατανεμημένοι πάνω στην εικόνα με τυχαίο τρόπο. Για αυτό το λόγο, ο διπολικός κρουστικός θόρυβος είναι επίσης γνωστός και ως θόρυβος αλατοπίπερου (salt & pepper noise), ενώ άλλες εκφράσεις που χρησιμοποιούνται για να περιγράψουν αυτόν τον τύπο θορύβου είναι οι «θόρυβος αποβολής δεδομένων (data-drop-out noise)» και «θόρυβος ακμής ή ακίδας (spike noise)».

Οι τιμές του κρουστικού θορύβου μπορεί να είναι θετικές ή αρνητικές και λαμβάνονται υπόψη κατά την κλιμάκωση η οποία συνήθως αποτελεί τμήμα της ψηφιοποίησης της εικόνας. Επειδή η αλλοίωση της εικόνας λόγω της ύπαρξης κρουστικού θορύβου συνήθως είναι μεγάλη σε σχέση με την ισχύ του σήματος της εικόνας. Ως αποτέλεσμα, οι αρνητικές κρούσεις εμφανίζονται ως μαύρα σημεία πάνω στην εικόνα (ως κόκκοι πιπεριού), ενώ για τον ίδιο λόγο, οι θετικές κρούσεις εμφανίζονται ως λευκές κουκίδες (ως κόκκοι αλατιού). Για μία εικόνα των 8 bits αυτό τυπικά αντιστοιχεί στις τιμές α = 0 (μαύρο) και b = 255 (λευκό).

Εικόνα αλλοιωμένη από τον θόρυβο αλατοπίπερου:



Η εμφάνιση του κρουστικού θορύβου οφείλεται κυρίως σε συσκευές ανάγνωσης εικόνων με νεκρά εικονοστοιχεία, σε λάθη μετατροπής αναλογικών εικόνων σε ψηφιακές, σε λάθη στην μετάδοση δεδομένων και στη γρήγορη μεταγωγή δεδομένων εικόνας.

Αντιμετωπίζεται καλύτερα όταν εφαρμοστούν τα εξής φίλτρα:

- το φίλτρο περικεκομμένου μέσου (Alpha-Trimmed Mean filter) όταν ο θόρυβος συνδυάζεται με θόρυβο Gauss (Gaussian noise)
- το φίλτρο αντι-αρμονικού μέσου (Contra-Harmonic Mean filter) με θετική τιμή Q για την εξάλειψη του θορύβου «πιπεριού», με αρνητική τιμή Q για την εξάλειψη του θορύβου «αλατιού» και με Q = 0 για την μείωση και των 2 ειδών θορύβου
- το φίλτρο αρμονικού μέσου (Harmonic Mean filter) όταν υπερισχύει ο θόρυβος «αλατιού» έναντι του θορύβου «πιπεριού»
- το φίλτρο μεγίστου (Maximum Pixel Value filter) για την εξάλειψη του θορύβου «πιπεριού»
- το φίλτρο ενδιάμεσης τιμής (Median filter)

- το φίλτρο μέσου σημείου (Midpoint filter)
- το φίλτρο ελαχίστου (Minimum Pixel Value filter) για την εξάλειψη του θορύβου «αλατιού»

Αντιμετωπίζεται χειρότερα όταν εφαρμοστεί το φίλτρο γεωμετρικού μέσου (Geometric Mean filter) στο θόρυβο «πιπεριού».

GAUSSIAN (NORMAL) NOISE

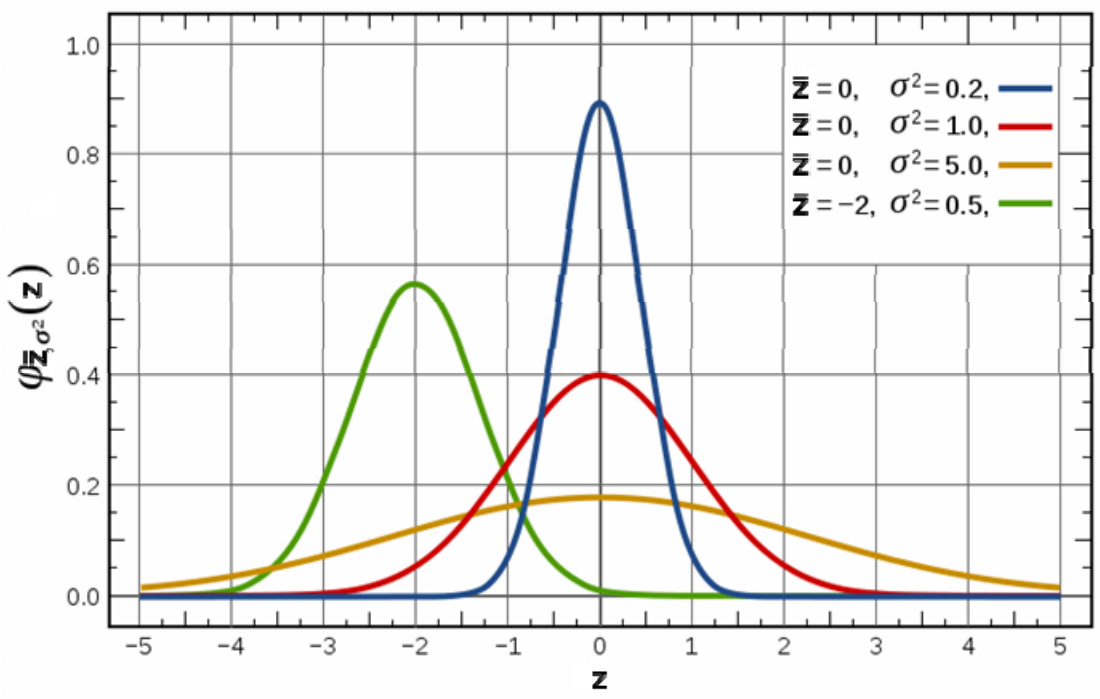
ΘΟΡΥΒΟΣ GAUSS (ΚΑΝΟΝΙΚΟΣ ΘΟΡΥΒΟΣ)

Η συνάρτηση πυκνότητας πιθανότητας μιας τυχαίας μεταβλητής z τύπου Gauss ορίζεται ως:

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(z-\bar{z})^2}{2\sigma^2}}$$

Η παράμετρος $\bar{z}$ είναι η μέση τιμή του z, ενώ το μέγεθος σ είναι η τυπική απόκλιση. Το τετράγωνο της τυπικής απόκλισης, σ^2, είναι γνωστό ως η μεταβλητότητα του z. Όταν η μεταβλητή περιγράφεται από την παραπάνω εξίσωση, περίπου το 70% των τιμών της βρίσκεται στην περιοχή τιμών $[(\bar{z}-\sigma),(\bar{z}+\sigma)]$ ενώ περίπου το 95% των τιμών της βρίσκεται στην περιοχή τιμών $[(\bar{z}-2\sigma),(\bar{z}+2\sigma)]$.

Ο θόρυβος Gauss εμφανίζεται σε μία εικόνα εξαιτίας παραγόντων όπως είναι ο θόρυβος των ηλεκτρονικών κυκλωμάτων καθώς και ο θόρυβος του αισθητήρα που οφείλεται στο φτωχό φωτισμό ή/και στην υψηλή θερμοκρασία.



Η εμφάνιση του θορύβου Gauss οφείλεται κυρίως σε αισθητήρες κατά την ανάγνωση εικόνων.

Αντιμετωπίζεται καλύτερα όταν εφαρμοστούν τα εξής φίλτρα:

- το φίλτρο περικεκομμένου μέσου (Alpha-Trimmed Mean filter) όταν ο θόρυβος συνδυάζεται με θόρυβο αλατοπίπερου (Impulse noise)
- το φίλτρο αριθμητικού μέσου (Arithmetic Mean filter)
- το φίλτρο αντι-αρμονικού μέσου (Contra-Harmonic Mean filter)
- το φίλτρο γεωμετρικού μέσου (Geometric Mean filter)
- το φίλτρο αρμονικού μέσου (Harmonic Mean filter)
- το φίλτρο ενδιάμεσης τιμής (Median filter)
- το φίλτρο μέσου σημείου (Midpoint filter)

RAYLEIGH NOISE

ΘΟΡΥΒΟΣ RAYLEIGH

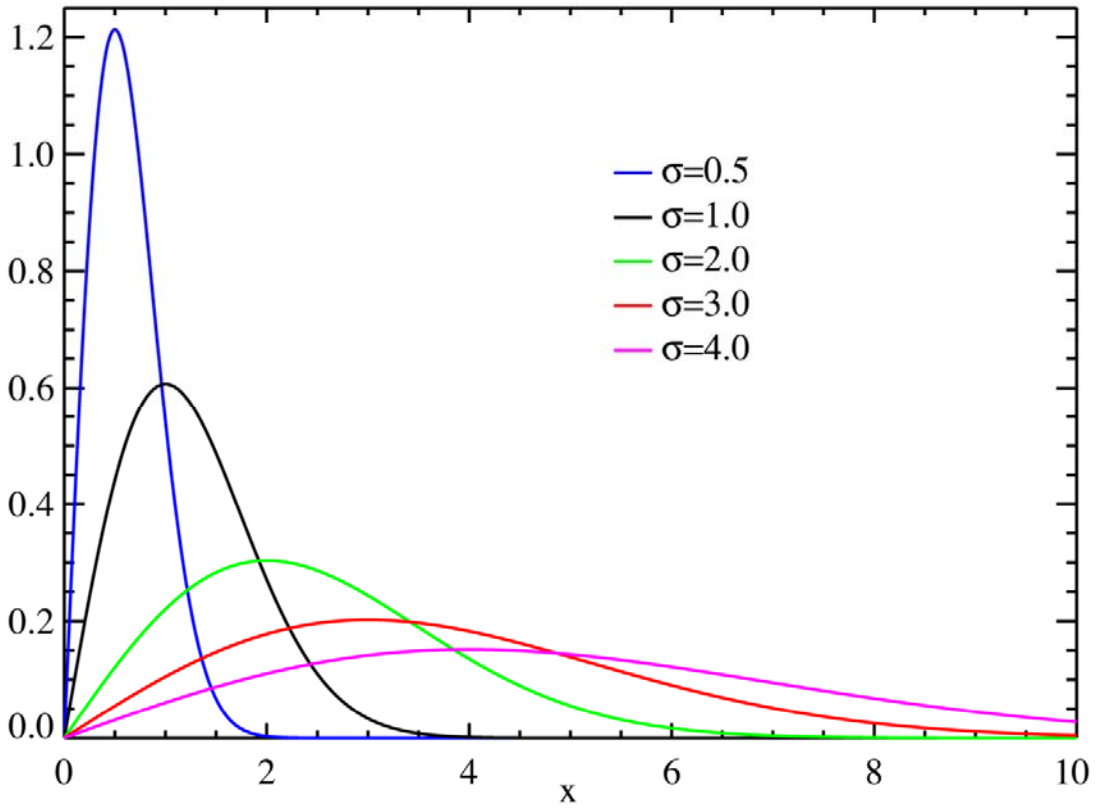
Η συνάρτηση πυκνότητας πιθανότητας του θορύβου Rayleigh δίνεται από την εξίσωση:

$$p(z) = \begin{cases} \frac{2}{b}(z-a)e^{-(z-a)^2/b} & , \gamma\iota\alpha \;\; z \geq \alpha \\ 0 & , \gamma\iota\alpha \;\; z \prec \alpha \end{cases}$$

Η μέση τιμή και η διασπορά που σχετίζονται με αυτήν την πυκνότητα ορίζονται ως:

$$\bar{z} = \alpha + \sqrt{\pi b/4} \quad \text{και} \quad \sigma^2 = \frac{b(4-\pi)}{4}$$

Ο θόρυβος Rayleigh βοηθάει στο χαρακτηρισμό φαινομένων θορύβου σε απεικονίσεις περιοχής.



επειδή $f(x;\sigma) = \frac{x}{\sigma^2} e^{-x^2/2\sigma^2}, \quad x \geq 0$

Αντιμετωπίζεται καλύτερα όταν εφαρμοστεί το φίλτρο αρμονικού μέσου (Harmonic Mean filter).

ERLANG (GAMMA) NOISE

ΘΟΡΥΒΟΣ ERLANG (GAMMA)

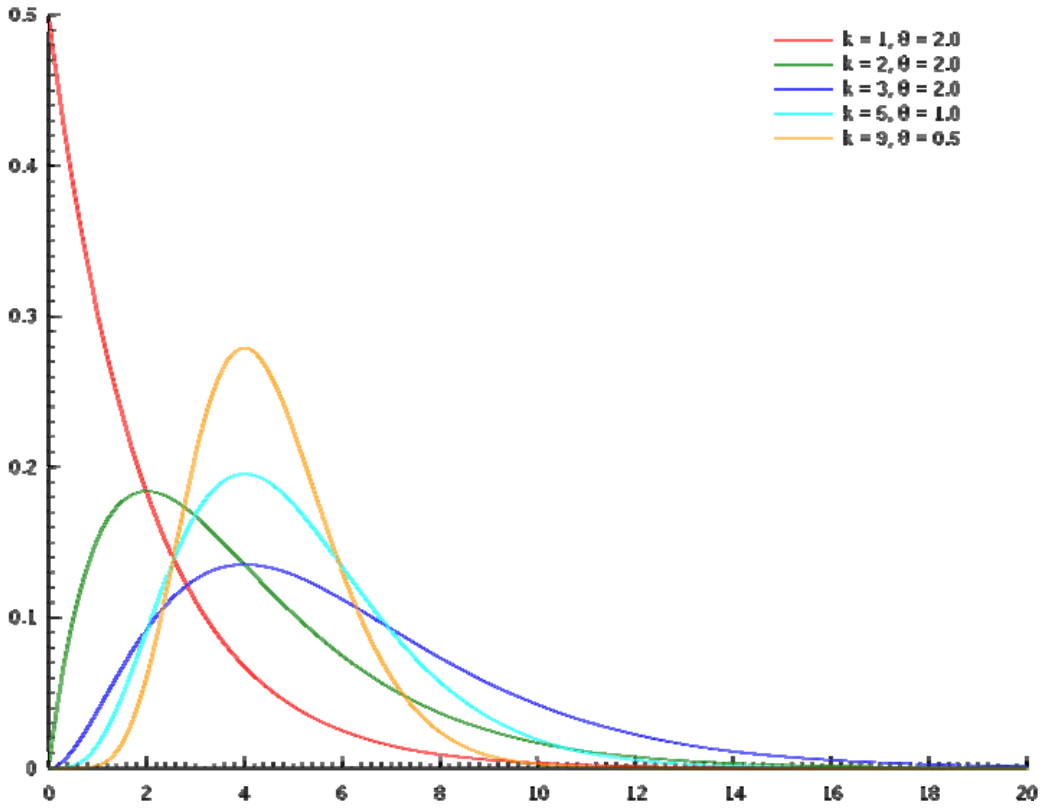
Η συνάρτηση πυκνότητας πιθανότητας του θορύβου Erlang / Gamma δίνεται από την εξίσωση:

$$p(z) = \begin{cases} \dfrac{\alpha^b z^{b-1}}{(b-1)!} e^{-\alpha z} & , \gamma\iota\alpha \ z \geq \alpha \\ 0 & , \gamma\iota\alpha \ z \prec 0 \end{cases}$$

όπου α>0 και b θετικός ακέραιος αριθμός. Η μέση τιμή και η διασπορά που σχετίζονται με αυτήν την πυκνότητα ορίζονται ως:

$$\bar{z} = b/\alpha \ \text{ και } \ \sigma^2 = \frac{b}{\alpha^2}$$

Ο θόρυβος Erlang βρίσκει εφαρμογές σε απεικόνιση με τη βοήθεια των ακτίνων laser.



επειδή $$f(x;k,\lambda) = \frac{\lambda^k x^{k-1} e^{-\lambda x}}{(k-1)!} \quad \text{for } x, \lambda \geq 0.$$

ΚΕΦΑΛΑΙΟ 3: Χωρικά φίλτρα

ALFA-TRIMMED MEAN FILTER

ΦΙΛΤΡΟ ΠΕΡΙΚΕΚΟΜΜΕΝΟΥ ΜΕΣΟΥ

Αν σε μία εικόνα διαγραφούν οι d/2 μικρότερες και οι d/2 μεγαλύτερες τιμές έντασης σε μια εικόνα στη γειτονιά S_{xy} και ότι χρησιμοποιείται το σύμβολο $g_r(s,t)$ για να αναπαρασταθούν τα υπόλοιπα mn – d εικονοστοιχεία που απομένουν, θα προκύψει το φίλτρο του περικεκομμένου μέσου το οποίο υπολογίζεται από τη μέση ένταση των υπόλοιπων εικονοστοιχείων:

$$\hat{f}(x,y) = \frac{1}{mn-d} \sum_{(s,t)\in S_{xy}} g_r(s,t)$$

.

όπου η τιμή του d κυμαίνεται από 0 έως mn-1.

Όταν το d λάβει την τιμή 0, το φίλτρο λειτουργεί ως φίλτρο αριθμητικού μέσου και όταν το d λάβει την τιμή mn – 1, μετατρέπεται στο φίλτρο διάμεσου. Για τις άλλες τιμές του d, το φίλτρο αυτό είναι ιδιαίτερα ωφέλιμο για περιπτώσεις που περιλαμβάνουν πολλαπλούς τύπους θορύβου αλατοπίπερου και θορύβου Gauss.

Επιδιόρθωση εικόνας με το φίλτρο περικεκομμένου μέσου:

ARITHMETIC MEAN FILTER

ΦΙΛΤΡΟ ΜΕΣΗΣ ΤΙΜΗΣ / ΑΡΙΘΜΗΤΙΚΟΥ ΜΕΣΟΥ

Το φίλτρο μέσης τιμής λειτουργεί αντικαθιστώντας τη φωτεινότητα του κάθε εικονοστοιχείου με τη μέση φωτεινότητα σε μια γειτονιά του στην εικόνα. Αυτό πρακτικά σημαίνει ότι αν μια εικόνα έχει σταθερές τιμές φωτεινότητας, το φίλτρο μπορεί και μειώνει αθροιστικά τον θόρυβο χωρίς να επηρεάζει σε μεγάλο βαθμό τις ακμές της εικόνας. Το φίλτρο μέσης τιμής ορίζεται ως:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t)$$

και αποτελεί τον μέσο όρο των τιμών του πλήθους m**x**n μιας γειτονιάς εικονοστοιχείων.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει με την τιμή 92 η οποία είναι και ο μέσος όρος στο τρέχων παράθυρο αφού (22+77+48+150+77+158+0+77+219) / (3*3) = 92.

22	77	48
150	77	158
0	77	219

Επιδιόρθωση εικόνας με το φίλτρο της μέσης τιμής:


CONTRA-HARMONIC MEAN FILTER

ΦΙΛΤΡΟ ΑΝΤΙ-ΑΡΜΟΝΙΚΟΥ ΜΕΣΟΥ

Το φίλτρο αντι-αρμονικού μέσου ορίζεται ως:

$$\hat{f}(x,y) = \frac{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q+1}}{\sum_{(s,t)\in S_{xy}} g(s,t)^{Q}}$$

.

Ο παραπάνω συντελεστής Q είναι γνωστός ως η τάξη του φίλτρου. Το φίλτρο λειτουργεί εξαίσια όσον αφορά την μείωση ή την εξαφάνιση του θορύβου αλατοπίπερου. Εάν οι τιμές της τάξης Q είναι θετικές, το φίλτρο μπορεί να εξαλείψει το θόρυβο πιπεριού. Από την άλλη πλευρά, εάν οι τιμές της τάξης Q είναι αρνητικές, το φίλτρο μπορεί να εξαλείψει το θόρυβο αλατιού. Ωστόσο, δεν υπάρχει η δυνατότητα να πραγματοποιηθούν οι παραπάνω διαδικασίες ταυτόχρονα. Όταν το Q λάβει την τιμή 1, το φίλτρο λειτουργεί ως φίλτρο αριθμητικού μέσου και ως φίλτρο αρμονικού μέσου όταν λάβει την τιμή -1.

Επιδιόρθωση εικόνας με το φίλτρο αντι-αρμονικού μέσου:

GAUSSIAN FILTER

ΦΙΛΤΡΟ GAUSS

Το φίλτρο Gauss αποτελεί ένα φίλτρο του οποίου η κρουστική απόκριση είναι μια συνάρτηση Gauss. Από μαθηματική άποψη, ένα φίλτρο Gauss τροποποιεί το σήμα εισόδου με συνέλιξη με τη συνάρτηση Gauss. Αυτός ο μετασχηματισμός είναι επίσης γνωστός ως το μετασχηματισμός Weierstrass. Το φίλτρο Gauss είναι ένα κατωδιαβατό φίλτρο και προκαλεί την θάμπωση της εικόνας.

Η κρουστική απόκριση του φίλτρου Gauss στη μία διάσταση δίνεται από την σχέση:

$$g(x) = \sqrt{\frac{a}{\pi}} \cdot e^{-a \cdot x^2}$$

ή, με την τυπική απόκλιση ως παράμετρο, από την σχέση:

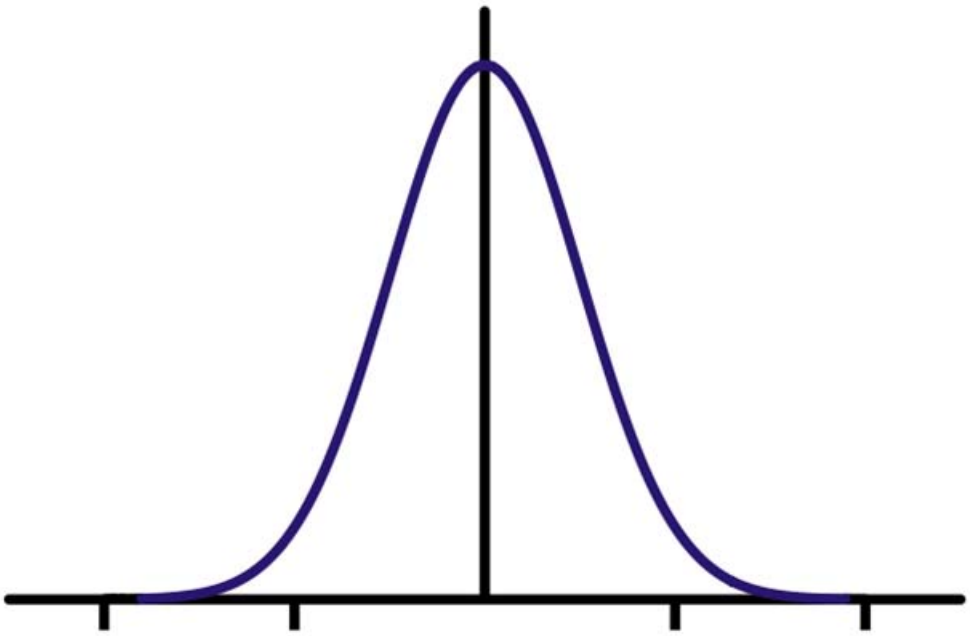
$$g(x) = \frac{1}{\sqrt{2 \cdot \pi} \cdot \sigma} \cdot e^{-\frac{x^2}{2\sigma^2}}$$

Η κρουστική απόκριση του φίλτρου Gauss στις δύο διαστάσεις είναι το γινόμενο δύο τέτοιων φίλτρων Gauss -ενός ανά διεύθυνση- και δίνεται από την σχέση:

$$g(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

όπου x είναι η απόσταση από τον οριζόντιο άξονα, y είναι η απόσταση από τον κάθετο άξονα, και σ είναι η τυπική απόκλιση της κατανομής Gauss.

Σχήμα ενός τυπικού φίλτρου Gauss:



GEOMETRIC MEAN FILTER

ΦΙΛΤΡΟ ΓΕΩΜΕΤΡΙΚΟΥ ΜΕΣΟΥ

Το φίλτρο γεωμετρικού μέσου ορίζεται ως:

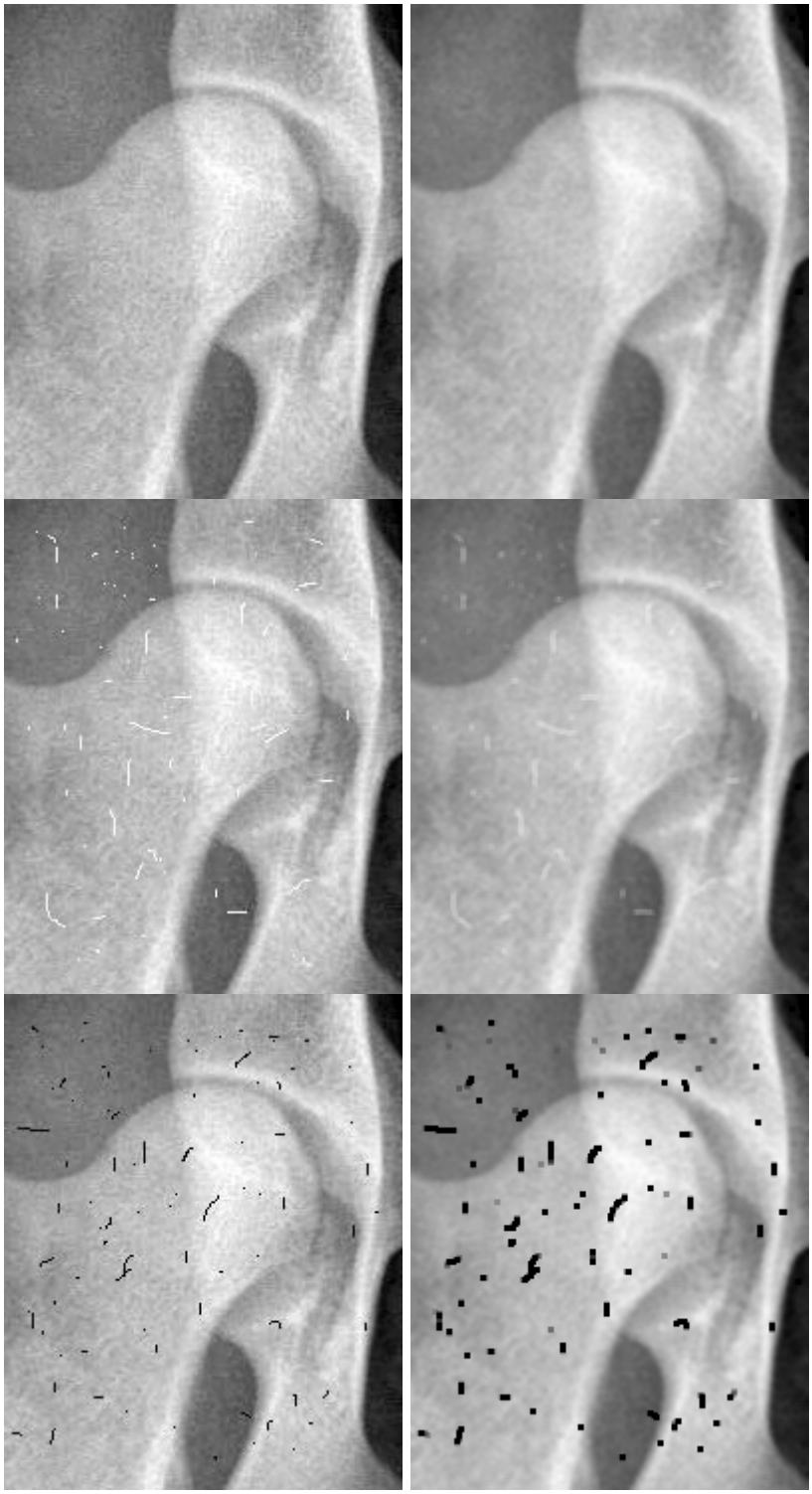
$$\hat{f}(x,y) = \left[\prod_{(s,t)\in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}.$$

όπου m*n μια γειτονιά εικονοστοιχείων.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει με την τιμή 34 αφού (22*77*48*150*77*158*1*77*219) ^ [1/(3*3)] ~= 34,7.

22	77	48
150	77	158
1	77	219

Επιδιόρθωση εικόνων με το φίλτρο της μέσης τιμής:



HARMONIC MEAN FILTER

ΦΙΛΤΡΟ ΑΡΜΟΝΙΚΟΥ ΜΕΣΟΥ

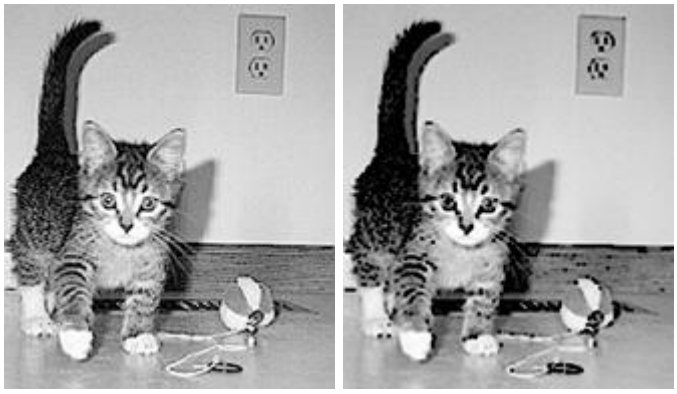
Το φίλτρο αρμονικού μέσου ορίζεται ως:

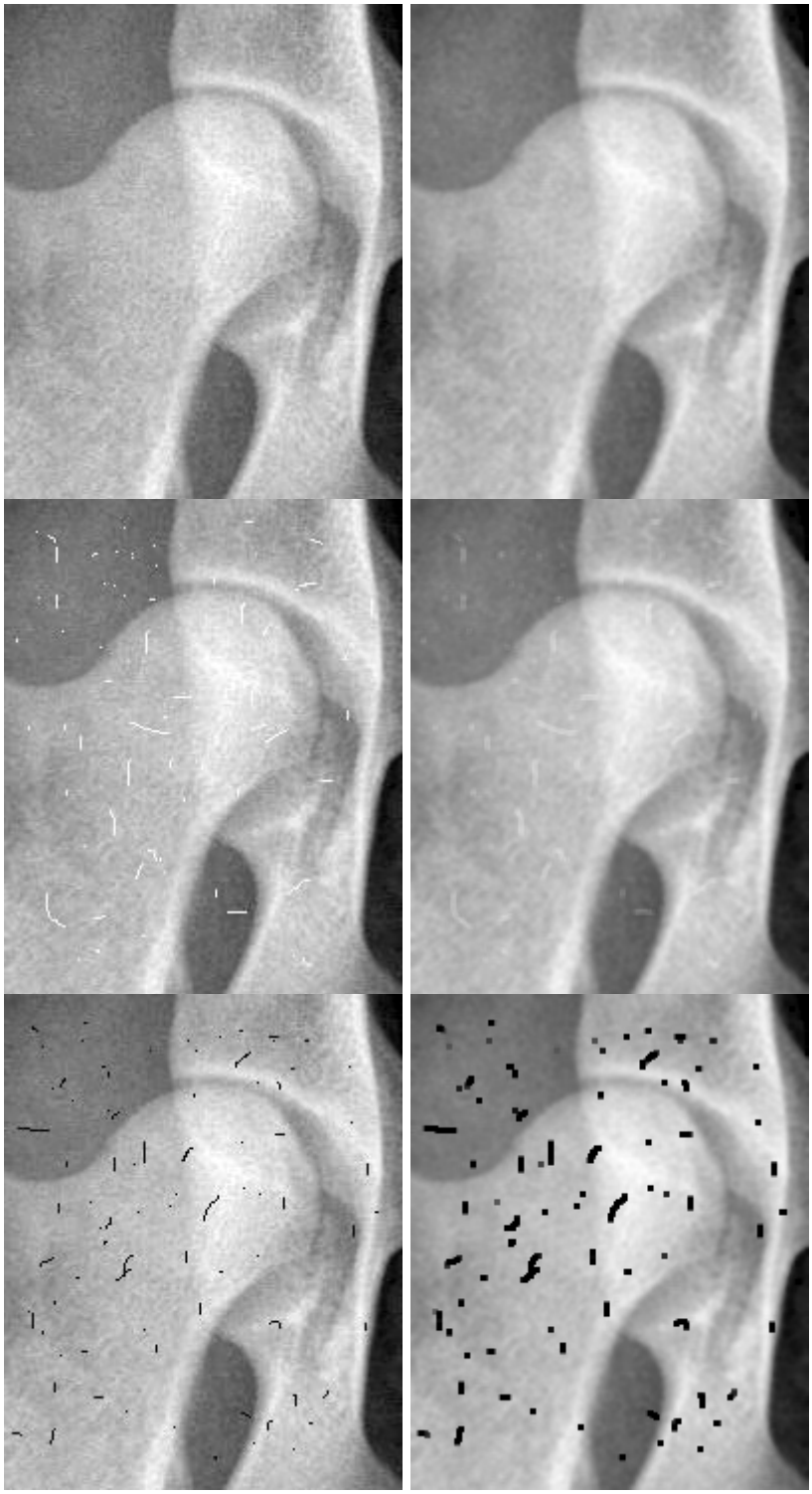
$$\hat{f}(x,y) = \frac{mn}{\sum_{(s,t)\in S_{xy}} \frac{1}{g(s,t)}}$$

όπου m×n μια γειτονιά εικονοστοιχείων.

Όσο πιο μεγάλο είναι το παράθυρο του φίλτρου τόσο περισσότερο θολώνει η αποκαταστημένη εικόνα και τόσο καλυτερεύει η αποκατάσταση της εικόνας. Το φίλτρο αρμονικού μέσου αποκαθιστά αποτελεσματικά εικόνες αλλοιωμένες από θόρυβο αλατοπίπερου όταν υπερισχύει ο θόρυβος «αλατιού» έναντι του θορύβου «πιπεριού». Επίσης, αποκαθιστά αποτελεσματικά εικόνες αλλοιωμένες τον θόρυβο Gauss.

Επιδιόρθωση εικόνων με το φίλτρο αρμονικού μέσου:





LOCAL ADAPTIVE FILTER

ΠΡΟΣΑΡΜΟΖΟΜΕΝΟ ΦΙΛΤΡΟ ΜΕΙΩΣΗΣ ΤΟΠΙΚΟΥ ΘΟΡΥΒΟΥ

Έστω S_{kj} η περιοχή στην οποία θα εφαρμοσθεί το φίλτρο. Για να εφαρμοστεί το φίλτρο θα πρέπει να ληφθούν υπόψη οι ακόλουθοι τέσσερις παράμετροι:

1. η τιμή g(k,j) της θορυβοποιημένης εικόνας στη θέση (k,j)
2. η τιμή της διασποράς σ_η^2 του θορύβου που παραμορφώνει την αρχική εικόνα f(k,j) οδηγώντας στην εικόνα g(k,j)
3. η τιμή του τοπικού μέσου m_L των εικονοστοιχείων της περιοχής S_{kj}
4. η τιμή της διασποράς σ_L^2 των εικονοστοιχείων της περιοχής S_{kj}

Το φίλτρο θα είναι χρήσιμο να ακολουθεί τους παρακάτω κανόνες:

- Όταν η τιμή της διασποράς σ_η^2 ισούται με το μηδέν, το φίλτρο πρέπει να επιστρέφει την τιμή της συνάρτησης g(k,j). Σε αυτήν και μόνο, την ιδιαίτερη περίπτωση, υπάρχει πλήρης απουσία θορύβου. Οπότε η θορυβοποιημένη εικόνα g(k,j) είναι ίδια με την αρχική εικόνα f(x,y).
- Όταν η τιμή της τοπικής διασποράς παρουσιάζεται αυξημένη σε σχέση με την τιμή της διασποράς σ_η^2, το φίλτρο πρέπει να επιστρέφει μία τιμή κοντά σε αυτήν της f(k,j). Εάν η τοπική διασπορά έχει υψηλή τιμή, υπάρχει σημαντική πιθανότητα να υπάρχουν ακμές τις οποίες και δεν πρέπει να καταστρέψει το φίλτρο.
- Όταν οι τιμές των δύο διακυμάνσεων ισούνται μεταξύ τους, το φίλτρο θα πρέπει να επιστρέφει την αριθμητική μέση τιμή των εικονοστοιχείων στην περιοχή S_{kj}. Τέτοια κατάσταση μπορεί να προκύψει αν η τοπική περιοχή έχει τις ίδιες ιδιότητες με την εικόνα στο σύνολό της. Αυτό σημαίνει ότι ο τοπικός θόρυβος μπορεί να μειωθεί με την αντικατάσταση των τιμών των εικονοστοιχείων από την μέση τιμή τους.

Μία έκφραση για τον υπολογισμό της τιμής της συνάρτησης $\hat{f}(k,j)$ με την βοήθεια των τριών παραπάνω κανόνων, μπορεί να γραφεί ως:

$$\hat{f}(k,j) = g(k,j) - \frac{\sigma_\eta^2}{\sigma_L^2}\left[g(k,j) - m_L\right].$$

Από τις τέσσερις παραμέτρους του φίλτρου, η τιμή της διασποράς του θορύβου σ_η^2 δεν είναι γνωστή και θα πρέπει είτε να υπάρχει γνώση του θορύβου για να της αποδοθεί κάποια τιμή είτε να εκτιμηθεί. Όλες οι υπόλοιποι παράμετροι προκύπτουν από τα εικονοστοιχεία της περιοχής S_{kj} σε κάθε θέση (k,j) στην οποία και κεντράρεται το παράθυρο του φίλτρου. Κάτι που ισχύει και δεν τονίζεται ιδιαίτερα είναι ότι ισχύει $\sigma_\eta^2 \geq \sigma_L^2$. Το φίλτρο θεωρεί τον θόρυβο προσθετικό.

Επειδή σπάνια υπάρχει ακριβή γνώση της τιμής της διασποράς σ_η^2 είναι πιθανόν η συνθήκη $\sigma_\eta^2 \geq \sigma_L^2$ να παραβιάζεται στην εφαρμογή του φίλτρου. Αυτός είναι και ο λόγος που θα πρέπει να ενσωματωθεί ένας έλεγχος στην όποια πραγμάτωση της παραπάνω εξίσωσης, με τέτοιο τρόπο ώστε εάν διαπιστωθεί ότι $\sigma_\eta^2 \succ \sigma_L^2$, ο λόγος τους να ισούται με μονάδα. Αυτό το γεγονός καθιστά το φίλτρο μη γραμμικό. Παρόλα αυτά, δεν επιτρέπει να παρουσιαστούν περίεργα αποτελέσματα (όπως π.χ. αρνητικές τιμές έντασης εξαρτώμενες από την τιμή του m_L) και τα οποία θα όφειλαν την ύπαρξή τους σε ενδεχόμενη έλλειψη γνώσης σχετικά με την τιμή της διασποράς του θορύβου της εικόνας. Τέλος, αν υποτεθεί ότι εφαρμόζεται τεχνική στην οποία να επιτρέπεται η εμφάνιση αρνητικών τιμών έντασης, με την επανακλιμάκωση των τιμών της έντασης στο τέλος αυτής, αυτό ίσως να οδηγούσε στον αφανισμό της δυναμικής της περιοχής της εικόνας.

MAXIMUM PIXEL VALUE FILTER

ΦΙΛΤΡΟ ΜΕΓΙΣΤΟΥ

Το φίλτρο μεγίστου ορίζεται ως:

$$\hat{f}(x,y) = \max_{(s,t)\in S_{xy}} \{g(s,t)\}$$

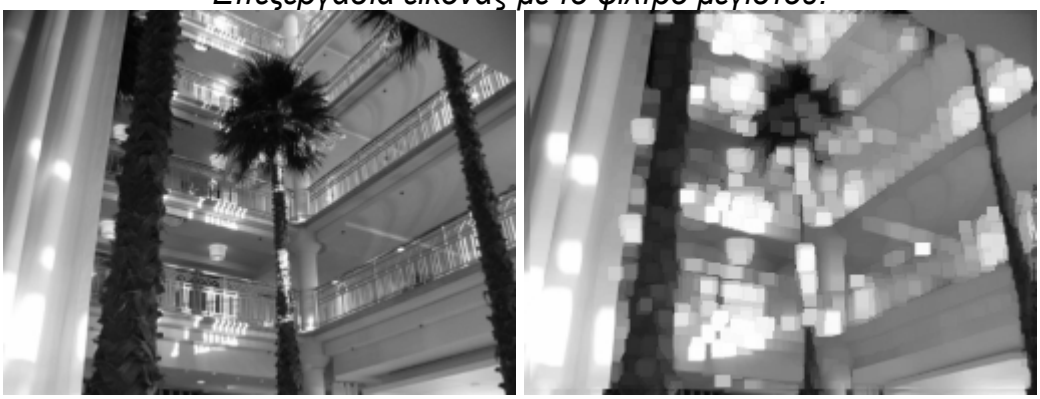
.

Αυτό το φίλτρο είναι χρήσιμο για την εύρεση των πιο φωτεινών σημείων μιας εικόνας. Επιπλέον, επειδή ο θόρυβος πιπεριού έχει πολύ μικρές τιμές, αυτό το φίλτρο προκαλεί την ελάττωσή του κατά τη διαδικασία επιλογής της μέγιστης τιμής του εικονοστοιχείου που βρίσκεται εντός της περιοχής Sxy.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει με την τιμή 219 η οποία είναι και φωτεινότερη τιμή στο τρέχων παράθυρο.

22	77	48
150	77	158
0	77	219

Επεξεργασία εικόνας με το φίλτρο μεγίστου:



MEDIAN FILTER

ΦΙΛΤΡΟ ΕΝΔΙΑΜΕΣΗΣ ΤΙΜΗΣ

Το πιο γνωστό φίλτρο στατιστικής διάταξης είναι το φίλτρο ενδιάμεσης τιμής που αντικαθιστά την τιμή ενός εικονοστοιχείου με την τιμή διάμεσου των επιπέδων έντασης που ανήκουν στην γειτονιά αυτού του εικονοστοιχείου. Το φίλτρο αυτό, που αρχικά αναπτύχθηκε από τον Tukey, ορίζεται ως:

$$\hat{f}(x,y) = \underset{(s,t)\in S_{xy}}{median}\{g(s,t)\}$$

.

Στη διαδικασία υπολογισμού της τιμής της διαμέσου, περιλαμβάνεται και η τιμή της έντασης του εικονοστοιχείου που βρίσκεται στη θέση (x,y). Τα φίλτρα διαμέσου είναι πολύ δημοφιλή, επειδή για ορισμένους τύπους τυχαίου θορύβου, προσφέρουν άριστες δυνατότητες μείωσης του θορύβου με αξιοσημείωτα μικρότερο βαθμό θόλωσης σε σχέση με τα γραμμικά φίλτρα εξομάλυνσης του ίδιου μεγέθους. Τα φίλτρα διαμέσου είναι ιδιαίτερα αποτελεσματικά στην παρουσία τόσο μονοπολικού όσο και διπολικού κρουστικού θορύβου. Στην πραγματικότητα τα φίλτρα διαμέσου προσφέρουν άριστα αποτελέσματα για εικόνες οι οποίες έχουν αλλοιωθεί από αυτόν τον τύπο θορύβου.

Το φίλτρο της ενδιάμεσης τιμής χρησιμοποιείται για την εξομάλυνση των ακμών μιας εικόνας. Δεν επηρεάζει τις βηματικές συναρτήσεις και τις συναρτήσεις κλιμάκωσης αλλά εξομαλύνει τις συναρτήσεις του απλού και του τριγωνικού παλμού.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα παραμείνει στην τιμή 77 η οποία είναι και η ενδιάμεση τιμή στο τρέχων παράθυρο (0, 22, 48, 77, 77, 77, 158, 219).

22	77	48
150	77	158
0	77	219

MEDIAN ADAPTIVE FILTER

ΠΡΟΣΑΡΜΟΖΟΜΕΝΟ ΦΙΛΤΡΟ ΔΙΑΜΕΣΟΥ

Το φίλτρο διαμέσου λειτουργεί πολύ καλά εάν η πυκνότητα του κρουστικού θορύβου δεν είναι πολύ μεγάλη (προκύπτει ότι λειτουργεί καλύτερα όταν το ποσοστό του κρουστικού θορύβου δεν ξεπερνάει το 20%). Σε αντίθεση με το φίλτρο διαμέσου, το προσαρμοζόμενο φίλτρο διαμέσου επιδιώκει να μην καταστρέψει τις λεπτομέρειες τις εικόνας που δεν αποτελούν κρουστικό θόρυβο. Επίσης, το προσαρμοζόμενο φίλτρο διαμέσου δεν αποκαθιστά την εικόνα στηριζόμενο στο αρχικό μέγεθος του παραθύρου εφαρμογής του αλλά το αυξάνει όσο χρειάζεται μέχρι ένα ορισμένο όριο.

Τα βήματα του προσαρμοζόμενου φίλτρου διαμέσου είναι τα εξής:

Φάση Α:

έστω $A_1 = z_{med} - z_{min}$ και $A_2 = z_{med} - z_{max}$.

Αν ισχύει $A_1 > 0$ και $A_2 < 0$ τότε εκτελείται η φάση Β.

Αλλιώς, μεγαλώνει το μέγεθος του παραθύρου.

>Όσο το μέγεθος του παραθύρου δεν υπερβαίνει τις διαστάσεις ενός μεγέθους S_{max} επαναλαμβάνεται η φάση Α.

Αλλιώς επιστρέφεται η τιμή z_{med}.

Φάση Β:

έστω $B_1 = z_{jk} - z_{min}$ και $B_2 = z_{jk} - z_{max}$.

Αν ισχύει $B_1 > 0$ και $B_2 < 0$ τότε επιστρέφεται η τιμή z_{jk}.

Αλλιώς επιστρέφεται η τιμή z_{med}.

όπου:

z_{min} η μικρότερη τιμή απόχρωσης στην περιοχή του παραθύρου

z_{max} η μεγαλύτερη τιμή απόχρωσης στην περιοχή του παραθύρου

z_{med} η διάμεση τιμή απόχρωσης στην περιοχή του παραθύρου

z_{jk} η τιμή απόχρωσης του σημείου (j,k) στην περιοχή του παραθύρου

S_{max} το μέγιστο επιτρεπτό μέγεθος του παραθύρου

Ο παραπάνω αλγόριθμος στοχεύει σε τρία πράγματα:

1. να αφαιρέσει τον κρουστικό θόρυβο αλατοπίπερου,
2. να μειώσει άλλους τύπους θορύβου (όπως η λέπτυνση) και
3. να αραιώσει τα περιγράμματα των αντικειμένων στην εικόνα.

Για τον αλγόριθμο, οι αποχρώσεις της εικόνας με τιμές z_{min} και z_{max} υπέχουν κάθε φορά θέσεις κρουστικού θορύβου. Κατά την φάση Α, ελέγχεται εάν η τιμή z_{med} είναι τιμή κρουστικού θορύβου. Εφόσον η τιμή z_{med} είναι μικρότερη της τιμής z_{max} και μεγαλύτερη της τιμής z_{min} δεν μπορεί να αποτελεί τιμή κρουστικού τύπου. Κατά την φάση Β, ελέγχεται εάν η τιμή z_{med} είναι τιμή κρουστικού τύπου. Ομοίως, αν $B_1 > 0$ και $B_2 < 0$ τότε η τιμή z_{jk} είναι μικρότερη της τιμής z_{max} και μεγαλύτερη της τιμής z_{min} και για αυτό δεν μπορεί να αποτελεί τιμή κρουστικού τύπου. Όπως φαίνεται από τα πολλά στάδια του αλγορίθμου η εικόνα αλλοιώνεται όσο το δυνατόν λιγότερο κατά την αποκατάστασή της.

MIDPOINT FILTER

ΦΙΛΤΡΟ ΜΕΣΟΥ ΣΗΜΕΙΟΥ

Το φίλτρο μέσου σημείου υπολογίζει το μέσο όρο της μέγιστης και της ελάχιστης τιμής της περιοχής της εικόνας στην οποία εφαρμόζεται το φίλτρο:

$$\hat{f}(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right].$$

Το φίλτρο αυτό συνδυάζει τόσο στατιστική διάταξη όσο και υπολογισμούς μέσου όρου και λειτουργεί με τον καλύτερο δυνατό τρόπο για την περίπτωση τυχαία κατανεμημένου θορύβου όπως είναι ο θόρυβος Gauss, ή ο ομοιόμορφος θόρυβος.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει στην τιμή 109 η οποία είναι και η μέση τιμή της φωτεινότερης τιμής 219 και της σκοτεινότερης τιμής 0.

22	77	48
150	77	158
0	77	219

Επεξεργασία εικόνας με το φίλτρο μέσου σημείου:

MINIMUM PIXEL VALUE FILTER

ΦΙΛΤΡΟ ΕΛΑΧΙΣΤΟΥ

Το φίλτρο ελαχίστου ορίζεται ως:

$$\hat{f}(x,y) = \min_{(s,t)\in S_{xy}} \{g(s,t)\}$$

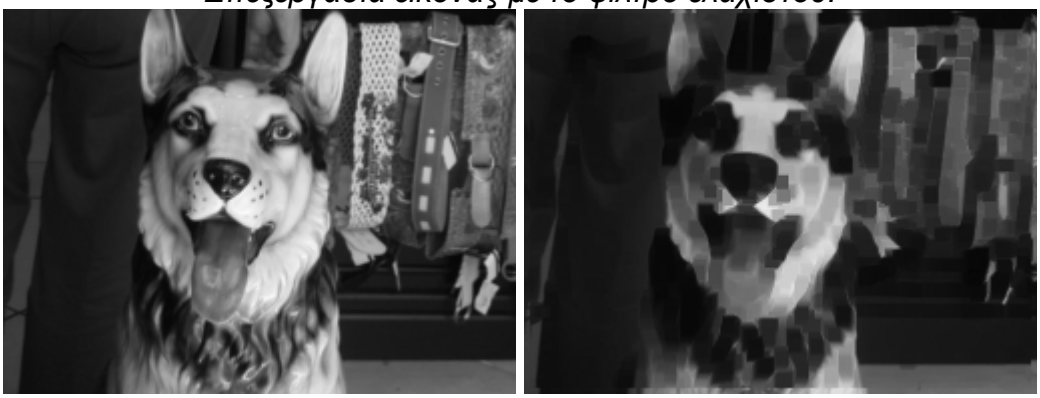
.

Αυτό το φίλτρο είναι χρήσιμο για την εύρεση των πιο σκοτεινών σημείων μιας εικόνας. Επιπλέον, επειδή ο θόρυβος αλατιού έχει πολύ μεγάλες τιμές, αυτό το φίλτρο προκαλεί την ελάττωσή του κατά τη διαδικασία επιλογής της ελάχιστης τιμής του εικονοστοιχείου που βρίσκεται εντός της περιοχής Sxy.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει με την τιμή 0 η οποία είναι και σκοτεινότερη τιμή στο τρέχων παράθυρο.

22	77	48
150	77	158
0	77	219

Επεξεργασία εικόνας με το φίλτρο ελαχίστου:



VARIOUS WEIGHTS FILTER

ΦΙΛΤΡΟ ΔΙΑΦΟΡΩΝ ΒΑΡΩΝ

Αποτελεί τη γενικευμένη εκδοχή του φίλτρου της μέσης τιμής καθώς η τιμή του κάθε εικονοστοιχείου της ορισθείσας γειτονιάς δεν λαμβάνει ως βάρος την τιμή 1 αλλά εδώ τα βάρη μπορούν να δεχθούν οποιεσδήποτε δεκαδικές τιμές από το -255 έως το 255. Άλλη διαφορά του παρόντος φίλτρου με το φίλτρο της μέσης τιμής είναι ότι το παράθυρο εφαρμογής του δεν μπορεί να είναι μόνο τετράγωνο αλλά μπορεί να είναι και ορθογώνιο με κάθε πλευρά να κυμαίνεται από 1 έως 7 εικονοστοιχεία.

Το φίλτρο «διαφόρων βαρών» ορίζεται ως:

$$\hat{f}(x,y) = \frac{1}{mn} \sum_{(s,t)\in S_{xy}} g(s,t) * W(m,n)$$

.

και αποτελεί τον μέσο όρο των τιμών του πλήθους m*n των γινομένων των εικονοστοιχείων μιας γειτονιάς με ορισμένα βάρη ανάλογα με τη θέση των εικονοστοιχείων στο παράθυρο του φίλτρου.

Για παράδειγμα, στην παρακάτω γειτονιά 3x3 εικονοστοιχείων αποχρώσεων του γκρι η τιμή 77 του κεντρικού εικονοστοιχείου θα αλλάξει με την τιμή 100 η οποία είναι και ο μέσος όρος των γινομένων του τρέχοντος παραθύρου αριστερά με τα βάρη στον πίνακα δεξιά αφού (22*1+77*1+48*1+150*1+77*2+158*1+0*1+77*1+219*1) / (3*3) ~= 100,56.

22	77	48
150	77	158
0	77	219

1	1	1
1	2	1
1	1	1

ΒΙΒΛΙΟΓΡΑΦΙΑ

- Ψηφιακή Επεξεργασία Εικόνας (3η έκδοση), Rafael C. Gonzalez, Richard E. Woods, ISBN: 9789604182558
- Ψηφιακή Επεξεργασία & Ανάλυση Εικόνας (2η έκδοση) , Ν. Παπαμάρκος, ISBN: 9789609273138
- Σημειώσεις θεωρίας του μαθήματος «ΨΗΦΙΑΚΗ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ» (Δρ. Χαράλαμπος Π. Στρουθόπουλος / Σέρρες, Δεκέμβριος 2009) του τμήματος Πληροφορικής και Επικοινωνιών του ΤΕΙ Σερρών
- Διαδίκτυο:
  - *wikipedia:*
    - http://en.wikipedia.org/wiki/Erlang_distribution
    - http://en.wikipedia.org/wiki/Exponential_distribution
    - http://en.wikipedia.org/wiki/Gaussian_filter
    - http://en.wikipedia.org/wiki/Gaussian_noise
    - http://en.wikipedia.org/wiki/Image_noise
    - http://en.wikipedia.org/wiki/Monte_Carlo_method
    - http://en.wikipedia.org/wiki/Rayleigh_distribution
    - http://en.wikipedia.org/wiki/Salt_and_pepper_noise
    - http://en.wikipedia.org/wiki/Uniform_distribution
  - http://gilbert-ap186.blogspot.com/2009/09/activity-18-noise-models-and-basic.html
  - www.massey.ac.nz/~mjjohnso/notes/59731/presentations/Adaptive_Median Filtering.doc
  - http://jrvitug.blogspot.com/2009/10/activity-18-noise-model-and-basic-image.html
  - www.cs.su.ac.th/~kanawong/courses/517483/ppt/Chapter%205.ppt
  - *digimizer.com:*
    - http://www.digimizer.com/manual/m-image-filtergeomean.php
    - http://www.digimizer.com/manual/m-image-filterharmonicmean.php
    - http://www.digimizer.com/manual/m-image-filtermax.php
    - http://www.digimizer.com/manual/m-image-filtermean.php

- http://www.digimizer.com/manual/m-image-filtermedian.php
- http://www.digimizer.com/manual/m-image-filtermid.php
- http://www.digimizer.com/manual/m-image-filtermin.php

- http://www.eng.ucy.ac.cy/cpitris/courses/ECE626/Presentations/Lecture5.pdf
- https://www8.cs.umu.se/kurser/TDBC30/VT05/material/lecture5.pdf

ΠΑΡΑΡΤΗΜΑ 1: Πηγαίος κώδικας

Μοντέλα θορύβου

EXPONENTIAL NOISE

ΕΚΘΕΤΙΚΟΣ ΘΟΡΥΒΟΣ

```
//----------- Noise: Exponential / Εκθετικός θόρυβος -----------
image imnoiseExponential( image im, int a )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;

  memalloc( im2 ) ;
  for( j = 0; j < im.J; j++ )
    for( k = 0; k < im.K; k++ )
      im2.pel[ j][ k] =
        setNewGrayValue( im.pel[ j][ k], exponential_rand( a ) );

  return im2 ;
}
//-----------------------------------------------------------------------

//----------- Συνάρτηση Exponential -----------
// Για τον υπολογισμό χρησιμοποιήθηκε μια συνάρτηση παραγωγής τυχαίων
// αριθμών εκθετικής κατανομής στο διάστημα (0,1)
inline int exponential_rand( int b )
{
  float a = b / 5000.0 ;
  return floor( (-1.0 /a*1.0) * log( 1.0 - 0.99 * rand() / RAND_MAX +
0.01 ) + 0.5 ) ;
}
//-----------------------------------------------------------------------
```

UNIFORM DISTRIBUTION

ΟΜΟΙΟΜΟΡΦΟΣ ΘΟΡΥΒΟΣ

```
//----------- Noise: Uniform Distribution / Ομοιόμορφος θόρυβος --------
/*
 Εκτέλεση της συνάρτησης τυχαίων αριθμών rand()
 για την επιλογή τυχαίων τιμών απόχρωσης από δοσμένο εύρος
 και πρόσθεση της κάθε τιμής με αυτή της αρχικής εικόνας
*/
image imnoiseUniformDistribution( image im, int lowGrayValue, int
highGrayValue )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, GrayRange ;

  GrayRange = highGrayValue - lowGrayValue + 1 ;

  memalloc( im2 ) ;
  for( j = 0 ; j < im.J; j++ )
    for( k = 0; k < im.K; k++ )
      im2.pel[ j][ k] = setNewGrayValue( im.pel[ j][ k], ( GrayRange *
rand() / RAND_MAX ) - abs( lowGrayValue ) * 1.0 ) ;

  return im2 ;
}
//-------------------------------------------------------------------
```

BIPOLAR NOISE

ΔΙΠΟΛΙΚΟΣ ΘΟΡΥΒΟΣ

```
//----------- Noise: Bipolar / Διπολικός θόρυβος -----------
image imnoiseBipolar( image im, int grayValueA, int grayValueB, float
grayValueAPossibility, float grayValueBPossibility )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;

  memalloc( im2 ) ;

  for( j = 0; j < im.J;  j++ )
    for( k = 0; k < im.K; k++ )
    {
       im2.pel[ j][ k] = setNewGrayValue( im.pel[ j][ k], bipolar_rand(
grayValueA, grayValueB, grayValueAPossibility, grayValueBPossibility ) )
;
    }
  return im2 ;
}
//--------------------------------------------------------------------------

//----------- Συνάρτηση Bipolar -----------
// Για τον υπολογισμό χρησιμοποιήθηκε μέθοδος Monte Carlo
inline int bipolar_rand( int a, int b, float pa, float pb )
{
  float x;
  x = 1.0 * rand() / RAND_MAX ;
  if     ( x < pa )      return a ;
  else if( x > 1 - pb )  return b ;
  else       return 0 ;
}
//--------------------------------------------------------------------------
```

IMPULSE NOISE

ΚΡΟΥΣΤΙΚΟΣ ΘΟΡΥΒΟΣ (ΑΛΑΤΟΠΙΠΕΡΟΥ)

```
//----------- Noise: Impulse (Salt & Pepper) / Κρουστικός θόρυβος
//(αλατοπίπερου) -----------
/*
 Γίνεται αντιγραφή όλων των εικονοστοιχείων της αρχικής εικόνας και
 ακολουθείται τυχαία αντικατάσταση τους με απόχρωση μαύρου(0) ή
 λευκού(255) βάσει ποσοστού που δίνεται από τον χρήστη
*/
image imnoiseImpulse( image im, float possibility )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;

  memalloc( im2 ) ;

  for( j = 0; j < im.J;  j++ )
    for( k = 0; k < im.K; k++ )
    {
       im2.pel[ j][ k] = impulse_rand( im.pel[ j][ k], possibility ) ;
    }
  return im2 ;
}
//-------------------------------------------------------------------------

//----------- Συνάρτηση Impulse -----------
// Για τον υπολογισμό χρησιμοποιήθηκε μέθοδος Monte Carlo
inline int impulse_rand( unsigned char cur, float p )
{
  float x;
  x = 1.0 * rand() / RAND_MAX ;
  if( x < p )  return 255 * ( rand() % 2 ) ;  // θορυβοποίηση της
εικόνας με τιμή απόχρωσης 0 ή 255
  else   return cur ;
}
//-------------------------------------------------------------------------
```

GAUSSIAN (NORMAL) NOISE

ΘΟΡΥΒΟΣ GAUSS (ΚΑΝΟΝΙΚΟΣ ΘΟΡΥΒΟΣ)

```
//----------- Noise: Gaussian / Θόρυβος Gauss -----------
image imnoiseGauss( image im, int mean, int sigma )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;

  memalloc( im2 ) ;

  for( j = 0; j < im.J; j++ )
    for( k = 0; k < im.K; k++ )
      im2.pel[ j][ k] = setNewGrayValue( im.pel[ j][ k], gauss_rand(
mean, sigma ) ) ;

  return im2 ;
}
//-------------------------------------------------------------------

//----------- Συνάρτηση Gauss -----------
// Για τον υπολογισμό χρησιμοποιήθηκε μέθοδος Monte Carlo
inline int gauss_rand( int mean, int sigma )
{
  float x, y, p ;

  do
  {
    x = 6.0 * rand() / RAND_MAX - 3 ;
    y = 1.0 * rand() / RAND_MAX ;
    p = exp( -x * x ) ;
  }
  while( y > p ) ;

  return floor( x * sigma + mean + 0.5 ) ;
}
//-------------------------------------------------------------------
```

RAYLEIGH NOISE

ΘΟΡΥΒΟΣ RAYLEIGH

```
//----------- Noise: Rayleigh / Θόρυβος Rayleigh -----------
image imnoiseRayleigh( image im, int offset, int sigma )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;

  memalloc( im2 ) ;

  for( j = 0; j < im.J; j++ )
    for( k = 0; k < im.K; k++ )
      im2.pel[ j][ k] = setNewGrayValue( im.pel[ j][ k], rayleigh_rand(
offset, sigma ) );

  return im2 ;
}
//---------------------------------------------------------------------

//----------- Συνάρτηση Rayleigh -----------
// Για τον υπολογισμό χρησιμοποιήθηκε μια συνάρτηση παραγωγής τυχαίων
// αριθμών κατανομής Rayleigh στο διάστημα (0,1)
inline int rayleigh_rand( int offset, int sigma )
{
  return floor( offset + sigma * sqrt( -2.0 * log( 0.99 * rand() /
RAND_MAX + 0.01 ) ) + 0.5 ) ;
}
//---------------------------------------------------------------------
```

ERLANG (GAMMA) NOISE

ΘΟΡΥΒΟΣ ERLANG (GAMMA)

```
//----------- Noise: Erlang (Gamma) / Θόρυβος Erlang (Gamma) -----------
image imnoiseErlang( image im, float a, int b )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k ;
  long int f ;

  memalloc( im2 ) ;

  f = factorial( b - 1 ) ;

  for( j = 0; j < im.J; j++ )
    for( k = 0; k < im.K; k++ )
      im2.pel[ j][ k] = setNewGrayValue( im.pel[ j][ k], erlang_rand( a,
b, f, im.pel[ j][ k] ) ) ;

  return im2 ;
}
//---------------------------------------------------------------------

//----------- Συνάρτηση Erlang (Gamma) -----------
// Για τον υπολογισμό χρησιμοποιήθηκε ο μαθηματικός τύπος του θορύβου
int erlang_rand( float a, int b, long int f, unsigned char cur )
{
  return floor( pow( a, b ) * pow( cur, b - 1 ) * exp( - a * cur ) / f )
;
}
//---------------------------------------------------------------------
```

factorial

```
//----------- Συνάρτηση εύρεσης παραγοντικού -----------

inline long int factorial( int n )
{
  if( 1 >= n )  return( 1 ) ;
  else          n = n * factorial( n - 1 ) ;
  return ( n ) ;
}
//-------------------------------------------------------------------------
```

image

```
myheader1.h
...
struct image
{
        int J, K, bppel ;
        unsigned char **pel ;
        bool empty;
};
```

memalloc

```
void memalloc( image &im )
{
  int j ;

  if( NULL == im.pel )  // memory has not been free when im.pel != NULL
  {
    im.pel = new unsigned char * [ im.J] ;
    for( j = 0; j < im.J; j++ ) im.pel[ j] = new unsigned char [ im.K] ;
  }
}
//---------------------------------------------------------------------
```

setNewGrayValue

```
//----------- Απόδοση τιμής απόχρωσης θορύβου -----------
/*
 Η απόδοση τιμής στη νέα απόχρωση πραγματοποιείται ύστερα από έλεγχο
υπέρβασης του εύρους
 των δυνατών τιμών απόχρωσης [0, 255]:
 * όταν η τιμή της νέας απόχρωσης υπερβαίνει το ανώτερο όριο των 255   ,
λαμβάνει την τιμή 255
 * όταν η τιμή της νέας απόχρωσης υπολείπεται του κατώτερου ορίου του 0,
λαμβάνει την τιμή 0
 * όταν η τιμή της νέας απόχρωσης ανήκει στο διάστημα [0, 255]   , δεν
μεταβάλλεται
*/
unsigned char setNewGrayValue( int currentGrayValue, float
noiseGrayValue )
{
  int newGrayValue ;
  int uncheckedNewGrayValue = currentGrayValue + (int) noiseGrayValue ;

  if( 255 < uncheckedNewGrayValue )
    newGrayValue = 255 ;
  else if( 0 > uncheckedNewGrayValue )
    newGrayValue = 0 ;
  else
    newGrayValue = uncheckedNewGrayValue ;

  return newGrayValue ;
}
//---------------------------------------------------------------------------
```

Χωρικά φίλτρα

ALFA-TRIMMED MEAN FILTER

ΦΙΛΤΡΟ ΠΕΡΙΚΕΚΟΜΜΕΝΟΥ ΜΕΣΟΥ

```
//----------- Filter: Alpha-Trimmed Mean / Φίλτρο Περικεκομμένου μέσου -
----------
image imfilterAlphaTrMean( image im, int d, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, m, r, c, sum ;
  bool flag ;
  unsigned char swap, *g ;

  memalloc( im2 ) ;
  g = new unsigned char [ size*size] ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          g[ r + c*size] = im.pel[ j + size/2 - r][ k + size/2 - c] ;
        }
      }

      // Αριθμητική ταξινόμηση των τιμών απόχρωσης από την μικρότερη
στην μεγαλύτερη
      flag = true ;
      while( flag )
      {
        flag = false ;
        for( m = 0; m < size*size; m++ )
        {
          if( g[ m] > g[ m + 1] )
          {
            flag    = true ;
            swap    = g[ m] ;
            g[ m]   = g[ m+1] ;
            g[ m+1] = swap ;
      } } }


      // Το άθροισμα δεν περιέχει τις d/2 χαμηλότερες και τις d/2
υψηλότερες τιμές απόχρωσης
      sum = 0 ;
      for( m = d/2; m < size*size - d/2; m++ )  sum = sum + g[ m] ;

      im2.pel[ j][ k] = sum / ( size*size - (d/2) *2 ) ;
    }
  }
```

```
  delete[] g ;

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//-----------------------------------------------------------------------
```

ARITHMETIC MEAN FILTER

ΦΙΛΤΡΟ ΜΕΣΗΣ ΤΙΜΗΣ / ΑΡΙΘΜΗΤΙΚΟΥ ΜΕΣΟΥ

```
//----------- Filter: Arithmetic Mean (Spatial Averaging) / Φίλτρο
Αριθμητικού μέσου -----------
image imfilterSpatialAveraging( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c, sum ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
    for( k = size/2; k < im.K - size/2; k++ )
    {
      sum = 0 ;
      for( r = 0; r < size; r++ )
        for( c = size-1; c >= 0; c-- )
          sum = sum + im.pel[ j + size/2 - r][ k + size/2 - c] ;
      im2.pel[ j][ k] = sum / ( size * size ) ;
    }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

CONTRA-HARMONIC MEAN FILTER

ΦΙΛΤΡΟ ΑΝΤΙ-ΑΡΜΟΝΙΚΟΥ ΜΕΣΟΥ

```
//----------- Filter: Contra-Harmonic Mean / Φίλτρο Αντι-Αρμονικού μέσου
-----------
image imfilterContraharmonicMean( image im, float Q, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur ;
  long double num, den ;  // num = numerator, den = denominators

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
    for( k = size/2; k < im.K - size/2; k++ )
    {
      num = 0.0 ;  den = 0.0 ;
      for( r = 0; r < size; r++ )
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( 0 != cur )
          {
            num = num + pow( cur, Q + 1.0 ) ;
            den = den + pow( cur, Q ) ;
        } }
      im2.pel[ j][ k] = unsigned ( num / den ) ;
    }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

GAUSSIAN FILTER

ΦΙΛΤΡΟ GAUSS

```
// N x N discrete values of a 2D Gaussian filter are computed
//---------------------------------------------------------------------------
-------
float** gauss2d( int N, float v )
{
  int r, c ;
  float x,y, sum = 0.0000 ;

  float** res = new float* [ N];  for( r = 0; r < N; r++ )  res[ r] =
new float [ N];

  if( N % 2 )
  for( r = 0; r < N; r++ )  for( c = 0; c < N; c++ )
  {
    x = (float)( c-N/2 ) ;  y = (float)( r-N/2 ) ;
    res[ r][ c] = exp( -( x*x+y*y )/( 2*v*v ) )/( 2*M_PI*v*v );
  }
  else
  for( r = 0; r < N; r++ )  for( c = 0; c < N; c++ )
  {
    x = (float)c -(float)(N-1)/2 ;  y = (float)r - (float)(N-1)/2 ;
    res[ r][ c] = exp( -( x*x+y*y )/( 2*v*v ) )/( 2*M_PI*v*v ) ;
  }

  for( r = 0; r < N; r++ )  for( c = 0; c < N; c++ )  sum += res[ r][ c]
;
  for( r = 0; r < N; r++ )  for( c = 0; c < N; c++ )  res[ r][ c] /= sum
;

  return res ;
}
//---------------------------------------------------------------------------
```

GEOMETRIC MEAN FILTER

ΦΙΛΤΡΟ ΓΕΩΜΕΤΡΙΚΟΥ ΜΕΣΟΥ

```
//----------- Filter: Geometric Mean / Φίλτρο Γεωμετρικού μέσου --------
---
image imfilterGeometricMean( image im, int size )
{
  // ΥΠΟΛΟΓΙΣΜΟΣ ΓΙΝΟΜΕΝΟΥ ΚΑΘΕ ΟΡΟΥ ΜΕΤΑ ΤΟΝ ΥΠΟΛΟΓΙΣΜΟ ΤΗΣ ΡΙΖΑΣ ΤΟΥ

  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur ;
  long double p ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      p = 1 ;
      for( r = 0; r < size; r++ )
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( 0 != cur )  p = p * pow( cur, 1.0/(size*size) ) ;
      }
      im2.pel[ j][ k] = p ;
  } }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//-------------------------------------------------------------------------
```

HARMONIC MEAN FILTER

ΦΙΛΤΡΟ ΑΡΜΟΝΙΚΟΥ ΜΕΣΟΥ

```
//----------- Filter: Harmonic Mean / Φίλτρο Αρμονικού μέσου -----------
image imfilterHarmonicMean( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur ;
  long double sum ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
    for( k = size/2; k < im.K - size/2; k++ )
    {
      sum = 0 ;
      for( r = 0; r < size; r++ )
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( 0 != cur )  sum = sum + 1.0 / cur ;
          else    // επειδή (αριθμός διάφορος του απείρου)/μηδέν =
άπειρο = μέγιστη τιμή = λευκό
          {       // οπότε σταματάει ο βρόχος επανάληψης και δίνεται
τέτοια τιμή στο sum
            sum = -1 ;  // ώστε να δημιουργηθεί λευκό εικονοστοιχείο
στην νέα εικόνα
            break ;
          }
        }
      if( -1 != sum )  im2.pel[ j][ k] = size*size / sum ;
      else             im2.pel[ j][ k] = 255 ;
    }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//-----------------------------------------------------------------------
```

LOCAL ADAPTIVE FILTER

ΠΡΟΣΑΡΜΟΖΟΜΕΝΟ ΦΙΛΤΡΟ ΜΕΙΩΣΗΣ ΤΟΠΙΚΟΥ ΘΟΡΥΒΟΥ

```
//----------- Filter: Local Adaptive / Προσαρμοζόμενο φίλτρο μείωσης
τοπικού θορύβου -----------
image imfilterLocalAdapt( image im, double sH2, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int i, j, k, r, c, sum ;
  bool flag ;
  double sL2 ;
  unsigned char swap, m, mL, *g ;

  memalloc( im2 ) ;
  g = new unsigned char [ size*size] ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      sum = 0 ;
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          g[ r + c*size] = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          sum = sum + g[ r + c*size] ;
        }
      }


/*
      // Αριθμητική ταξινόμηση των τιμών απόχρωσης από την μικρότερη
στην μεγαλύτερη
      flag = true ;
      while( flag )
      {
        flag = false ;
        for( i = 0; i < size*size; i++ )
        {
          if( g[ i] > g[ i + 1] )
          {
            flag    = true ;
            swap    = g[ i] ;
            g[ i]   = g[ i+1] ;
            g[ i+1] = swap ;
      } } }

      mL = g[ size*size/2] ;  // Το μεσαίο στοιχείο
*/
      mL = sum / ( size*size ) ;  // Μέση τιμή
```

```
      // Υπολογισμός της μέσης τιμής των αποχρώσεων των εικονοστοιχείων
της αλλοιωμένης εικόνας
      m = sum * 1.0 / ( size*size ) ;

      // Υπολογισμός της διακύμανσης (διασποράς) των αποχρώσεων των
εικονοστοιχείων της αλλοιωμένης εικόνας
      for( i = 0; i < size*size; i++ )  sL2 = sL2 + pow( ( g[ i] - m ),
2 ) ;  sL2 = sL2 * 1.0 / ( size*size ) ;

      im2.pel[ j][ k] = im.pel[ j][ k] - ( sH2/sL2 ) * ( im.pel[ j][ k]
- mL ) ;
    }
  }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

MAXIMUM PIXEL VALUE FILTER

ΦΙΛΤΡΟ ΜΕΓΙΣΤΟΥ

```
//----------- Filter: Maximum Pixel Value / Φίλτρο Μεγίστου -----------
image imfilterMaximumPixelValue( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur, max ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      max = 0 ;
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( cur > max )  max = cur ;
      } }
      im2.pel[ j][ k] = max ;
  } }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

MEDIAN FILTER

ΦΙΛΤΡΟ ΕΝΔΙΑΜΕΣΗΣ ΤΙΜΗΣ

```
//----------- Filter: Median / Φίλτρο Ενδιάμεσης τιμής -----------
image imfilterMedian( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, m, r, c ;
  bool flag ;
  unsigned char swap, *g ;

  memalloc( im2 ) ;
  g = new unsigned char [ size*size] ;
  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          g[ r + c*size] = im.pel[ j + size/2 - r][ k + size/2 - c] ;
        }
      }

      // Αριθμητική ταξινόμηση των τιμών απόχρωσης από την μικρότερη
στην μεγαλύτερη
      flag = true ;
      while( flag )
      {
        flag = false ;
        for( m = 0; m < size*size; m++ )
        {
          if( g[ m] > g[ m + 1] )
          {
            flag    = true ;
            swap    = g[ m] ;
            g[ m]   = g[ m+1] ;
            g[ m+1] = swap ;
      } } }

      im2.pel[ j][ k] = g[ size*size/2] ;  // Το μεσαίο στοιχείο
    }
  }

  delete[] g ;

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

MEDIAN ADAPTIVE FILTER

ΠΡΟΣΑΡΜΟΖΟΜΕΝΟ ΦΙΛΤΡΟ ΔΙΑΜΕΣΟΥ

```
//----------- Filter: Median Adaptive / Προσαρμοζόμενο φίλτρο διαμέσου -
----------
image imfilterMedianAdapt( image im, int size, int maxsize )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c, i, A1, A2, B1, B2, size2 ;
  bool flag, flag2 ;
  unsigned char cur, swap, Zmax, Zmin, Zmed, *g ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      // ΣΤΑΔΙΟ Α
      size2 = size ;
      flag = true ;
      while( flag )
      {
        g = new unsigned char [ size2*size2] ;

        for( r = 0; r < size2; r++ )
        {
          for( c = size2-1; c >= 0; c-- )
          {
            cur = im.pel[ j + size2/2 - r][ k + size2/2 - c] ;
            g[ r + c*size2] = cur ;
            if( cur > Zmax )  Zmax = cur ;
            if( cur < Zmin )  Zmin = cur ;
          }
        }

        // Αριθμητική ταξινόμηση των τιμών απόχρωσης από την μικρότερη
        // στην μεγαλύτερη
        flag2 = true ;
        while( flag2 )
        {
          flag2 = false ;
          for( i = 0; i < size2*size2; i++ )
          {
            if( g[ i] > g[ i + 1] )
            {
              flag2   = true ;
              swap    = g[ i] ;
              g[ i]   = g[ i+1] ;
              g[ i+1] = swap ;
        } } }

        Zmed = g[ size2*size2/2] ;  // Το μεσαίο στοιχείο

        A1 = Zmed - Zmin ;
        A1 = Zmed - Zmax ;
```

```
        if( ( A1 > 0 ) && ( A2 < 0 ) )
        {
          // ΣΤΑΔΙΟ Β
          flag = false ;
          B1 = im.pel[ j][ k] - Zmin ;
          B2 = im.pel[ j][ k] - Zmax ;
          if( ( B1 > 0 ) && ( B2 < 0 ) )
            im2.pel[ j][ k] = im.pel[ j][ k] ;
          else
            im2.pel[ j][ k] = Zmed ;
        }

        else
        {
          if( ( (size2 + 2)/2 < j ) && ( (size2 + 2)/2 + j < maxsize )
&&            ( (size2 + 2)/2 < k ) && ( (size2 + 2)/2 + k < maxsize ) )
              size2 = size2 + 2 ;

          else
          {
            flag = false ;
            im2.pel[ j][ k] = Zmed ;
          }
        }

        delete[] g ;
      }
    }
  }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------------
```

MIDPOINT FILTER

ΦΙΛΤΡΟ ΜΕΣΟΥ ΣΗΜΕΙΟΥ

```
//----------- Filter: Midpoint / Φίλτρο Μέσου σημείου -----------
image imfilterMidpoint( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur, max, min ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      max = 0 ;  min = 255 ;
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( cur > max )  max = cur ;
          if( cur < min )  min = cur ;
      } }
      im2.pel[ j][ k] = (max + min) / 2 ;
  } }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//---------------------------------------------------------------------
```

MINIMUM PIXEL VALUE FILTER

ΦΙΛΤΡΟ ΕΛΑΧΙΣΤΟΥ

```
//----------- Filter: Minimum Pixel Value / Φίλτρο Ελαχίστου -----------
image imfilterMinimumPixelValue( image im, int size )
{
  image im2 ;  im2.J = im.J ;  im2.K = im.K ;  im2.pel = NULL ;
  int j, k, r, c ;
  unsigned char cur, min ;

  memalloc( im2 ) ;

  for( j = size/2; j < im.J - size/2; j++ )
  {
    for( k = size/2; k < im.K - size/2; k++ )
    {
      min = 255 ;
      for( r = 0; r < size; r++ )
      {
        for( c = size-1; c >= 0; c-- )
        {
          cur = im.pel[ j + size/2 - r][ k + size/2 - c] ;
          if( cur < min )  min = cur ;
      } }
      im2.pel[ j][ k] = min ;
  } }

  createBlackOutline( im2, size ) ;
  return im2 ;
}
//-------------------------------------------------------------------------
```

image

```
myheader1.h
...
struct image
{
        int J, K, bppel ;
        unsigned char **pel ;
        bool empty;
};
```

mask

```
myheader1.h
...
struct mask
{
        float **v ;
        short rows, cols ;
};
```

memalloc

```
void memalloc( image &im )
{
  int j ;

  if( NULL == im.pel )  // memory has not been free when im.pel != NULL
  {
    im.pel = new unsigned char * [ im.J] ;
    for( j = 0; j < im.J; j++ ) im.pel[ j] = new unsigned char [ im.K] ;
  }
}
//---------------------------------------------------------------------
```

free

```
void free( image &im )
{
  int j ;

  if( NULL != im.pel )  // memory is already free when im.pel = NULL
  {
    for( j = 0; j < im.J; j++ )  delete[] im.pel[ j] ;
    delete[] im.pel ;
  }
  im.pel = NULL ;
}
```

imcorel

```
float ** imcorel( image im, mask m )
{
        float **res ;
        int k, j, r, c, x, y ;

        res = new float *[ im.J];  for( j = 0; j < im.J; j++ )  res[ j]=
new float [ im.K];

        for( j = 0; j < im.J; j++ )  for( k = 0; k< im.K; k++ )
        {
                res [ j][ k] = 0.0 ;
                for( r = 0; r < m.rows; r++ )  for( c = 0; c < m.cols;
c++ )
                {
                        y = j + r - m.rows/2 ;  x = k + c - m.cols/2 ;
                        if( -1 < y && y < im.J && -1 < x && x < im.K )
                                res[ j][ k] += im.pel[ y][ x] * m.v[ r][
c] ;
                }
        }
        return res ;
}
//---------------------------------------------------------------------
```

toimage

```
image toimage( float **m, int R, int C )
{
  image im ;  im.J = R ;  im.K = C ;  im.pel = NULL ;  memalloc( im ) ;
  int r, c ;

  float min = m[ 0][ 0] ;
  for( r = 0; r < R; r++ )  for( c = 0; c < C; c++ )
                              if( m[ r][ c] < min )  min = m[ r][ c] ;

  for( r = 0; r < R; r++ )  for( c = 0; c < C; c++ )  m[ r][ c] -= min ;

  float max = m[ 0][ 0];
  for( r = 0; r < R; r++ )  for( c = 0; c < C; c++ )
                              if( m[ r][ c] > max )  max = m[ r][ c] ;

  for( r = 0; r < R; r++ )  for( c = 0; c < C; c++ )  m[r][c]/=max ;

  for( r = 0; r < R; r++ )  for( c = 0; c < C; c++ )
    im.pel[ r][ c] = (unsigned char) ( 255 * m[ r][ c] );

  return im;
}
//---------------------------------------------------------------------
```

createBlackOutline

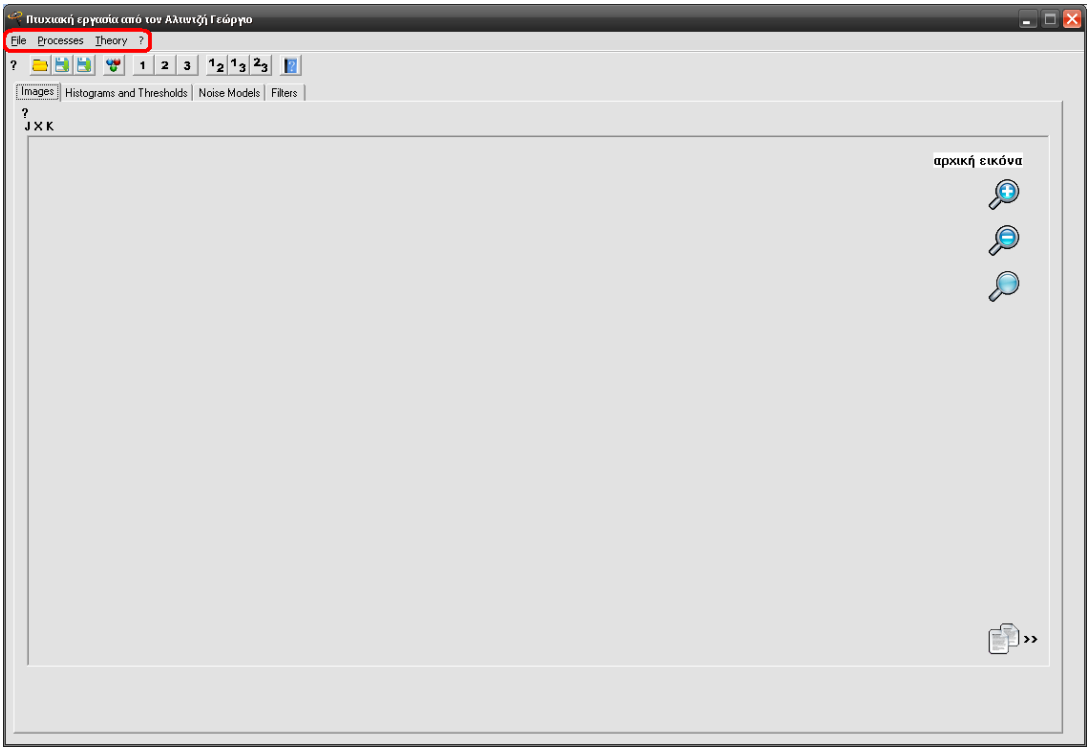
```
//----------- Δημιουργία μαύρου περιγράμματος -----------
/*
 Το παράθυρο των φίλτρων μεγέθους size αφήνει μια περιοχή μεγέθους
size/2 γύρω από την εικόνα μη αποθορυβοποιημένη. Η συνάρτηση αυτή
αναλαμβάνει να χρωματίσει αυτήν την περιοχή με το χρώμα μαύρο ώστε να
μην εμφανίζονται τυχαίες τιμές.
*/
void createBlackOutline( image im, int size )
{
  int j, k;
  for( j = 0; j < size/2; j++ )  for( k = 0; k < im.K; k++ )
                           im.pel[ j][ k] = 0 ;  // πάνω οριζόντια γραμμή

  for( j = 0; j <im.J; j++ )
  {
    for( k = 0; k < size/2; k++ )
      im.pel[ j][ k] = 0 ;  // δεξιά κάθετη γραμμή

    for( k = im.K - size/2; k < im.K; k++ )
      im.pel[ j][ k] = 0 ;  // αριστερή κάθετη γραμμή
  }
  for( j = im.J - size/2; j < im.J; j++ )  for( k = 0; k < im.K; k++ )
                           im.pel[ j][ k] = 0 ;  // κάτω οριζόντια γραμμή
}
//---------------------------------------------------------------------
```

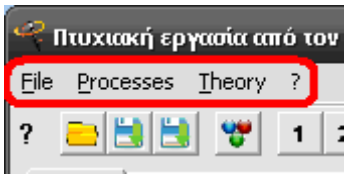
ΠΑΡΑΡΤΗΜΑ 2: Εγχειρίδιο χρήσης προγράμματος

MainMenu

Κεντρικό μενού

Το κεντρικό μενού είναι το μενού που περιέχει τις εντολές του προγράμματος για ενέργειες πάνω στις εικόνες που υπόκεινται επεξεργασία.





Το μενού **File** προσφέρει τις ακόλουθες επιλογές:

- Open… (Άνοιγμα εικόνας)
- Save noised image (Αποθήκευση θορυβοποιημένης εικόνας)

- Save denoised image (Αποθήκευση αποθορυβοποιημένης εικόνας)
- Exit (Έξοδος)

Το μενού **Processes** προσφέρει τις ακόλουθες επιλογές:

- Histogram (Ιστόγραμμα)
    - Source Image... (Αρχική εικόνα)
    - Noised Image... (Θορυβοποιημένη εικόνα)
    - Denoised Image... (Αποθορυβοποιημένη εικόνα)
    - All Images... (Όλες τις παραπάνω εικόνες)
- Histogram Equalization (Εξισορρόπηση ιστογράμματος)
    - Source Image... (Αρχική εικόνα)
    - Noised Image... (Θορυβοποιημένη εικόνα)
    - Denoised Image... (Αποθορυβοποιημένη εικόνα)
- Thresholding (Κατωφλίωση)
    - Variance (Otsu) (Διακύμανσης)
    - Entropy (Kapur) (Εντροπίας)
    - Arbitrary
- Multithresholding (Πολυκατωφλίωση)
- Negative (Αρνητικό χρώμα)
- Add noise (Προσθήκη θορύβου)
    - Exponential (Εκθετικός θόρυβος)
    - Uniform (Ομοιόμορφος θόρυβος)
    - Bipolar (Διπολικός θόρυβος)
    - Impulse (Salt and Pepper noise / Κρουστικός θόρυβος (αλατοπίπερου))
    - Gaussian (Θόρυβος Gauss (Κανονικός θόρυβος))
    - Rayleigh (Θόρυβος Rayleigh)
    - Erlang (Θόρυβος Erlang (Gamma))
- Remove noise (Αφαίρεση θορύβου)
    - Alpha-Trimmed Mean (Φίλτρο Περικεκομμένου μέσου)
    - Arithmetic Mean (Φίλτρο Αριθμητικού μέσου)
    - Contra-Harmonic Mean (Φίλτρο Αντι-Αρμονικού μέσου)
    - Gaussian (Φίλτρο Gauss)

- Geometric Mean (Φίλτρο Γεωμετρικού μέσου)
- Harmonic Mean (Φίλτρο Αρμονικού μέσου)
- Local Adaptive (Προσαρμοζόμενο φίλτρο διαμέσου)
- Maximum Pixel Value (Φίλτρο Μεγίστου)
- Median (Φίλτρο Ενδιάμεσης τιμής)
- Median Adaptive (Προσαρμοζόμενο φίλτρο μείωσης τοπικού θορύβου)
- Midpoint (Φίλτρο Μέσου σημείου)
- Minimum Pixel Value (Φίλτρο Ελαχίστου)
- Various Weights (Φίλτρο Διάφορων Βαρών)

• Edge Detection (Αναγνώριση ακμών)
- First Derivative (Πρώτης παραγώγου)
  - Horizontal (Οριζόντια κατεύθυνση)
  - Vertical (Κάθετη κατεύθυνση)
  - Both (Και στις δύο κατευθύνσεις)
- Second Derivative (Δεύτερης παραγώγου)
  - Laplacian Application (Εφαρμογή Laplace)
  - Laplacian Threshold (Εφαρμογή κατωφλίου Laplace)
  - LoG (Laplace on Gaussian)

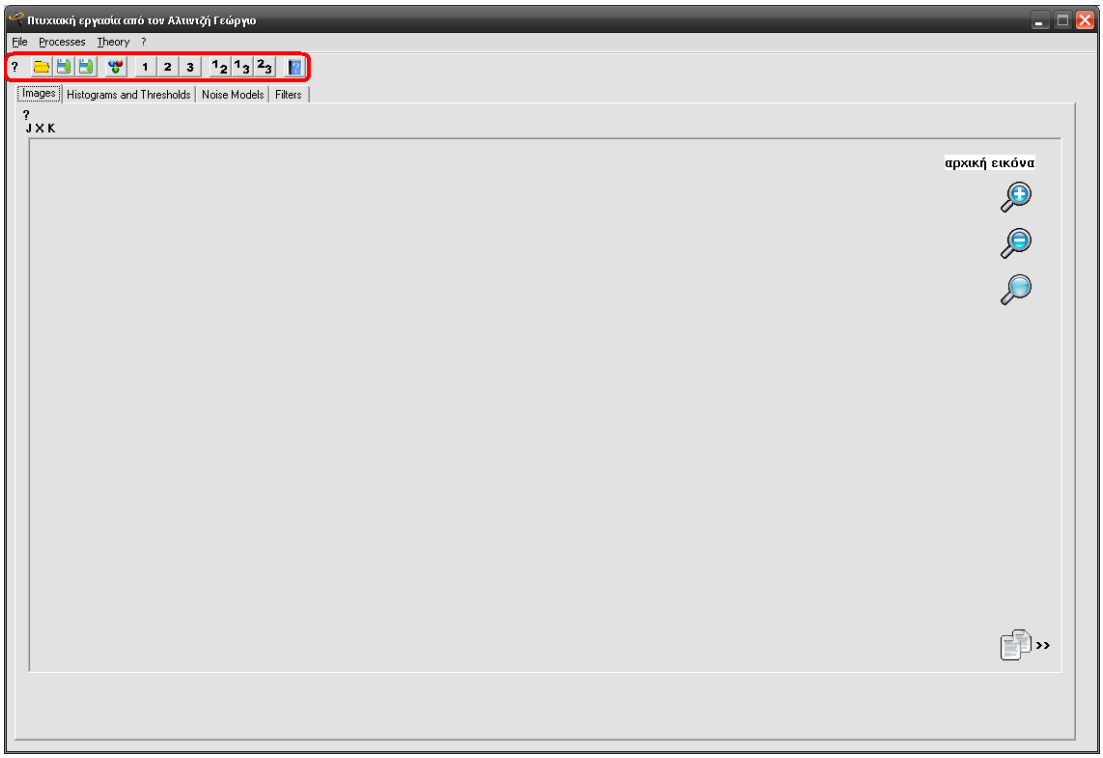
Η επιλογή **Theory** οδηγεί στην φόρμα θεωρίας του προγράμματος.

Η επιλογή **?** οδηγεί στις πληροφορίες σχετικά με το κεντρικό μενού.

ToolBar

Μενού εργαλείων

Το μενού εργαλείων είναι το μενού που περιέχει τις εντολές του προγράμματος του μενού File του κεντρικού μενού και εντολές για τις εικόνες που εμφανίζονται στην περιοχή εικόνων.



Τα παραπάνω κουμπιά είναι τα ακόλουθα:

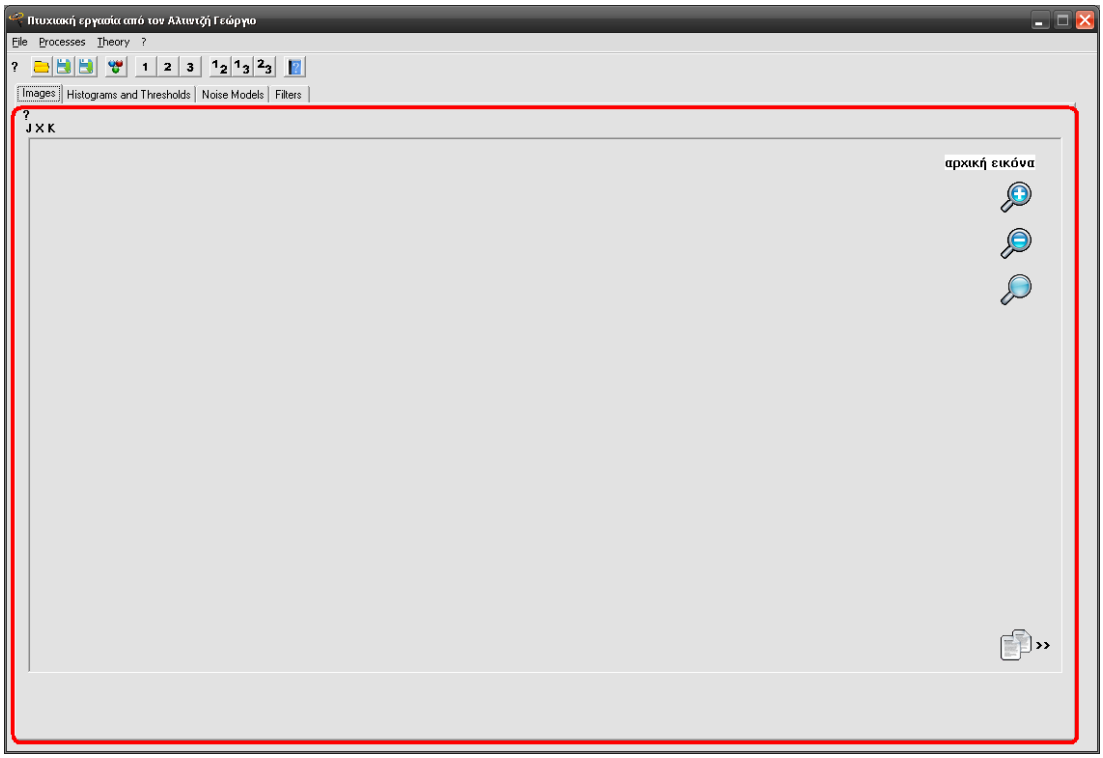
- Το κουμπί **?** οδηγεί στις πληροφορίες σχετικά με το μενού εργαλείων.
- Το κουμπί του **φακέλου** ανοίγει μια εικόνα.
- Το κουμπί της **1ης δισκέτας** αποθηκεύει την θορυβοποιημένη εικόνα.

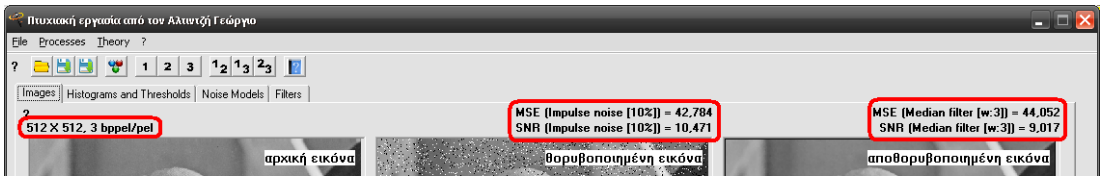
- Το κουμπί της **2ης δισκέτας** αποθηκεύει την αποθορυβοποιημένη εικόνα.
- Το κουμπί με τους **3 κύκλους** εμφανίζει όλες τις εικόνες (αρχική, θορυβοποιημένη και απόθορυβοποιημένη) εφόσον αυτές υπάρχουν.
- Το κουμπί **1** εμφανίζει την αρχική εικόνα.
- Το κουμπί **2** εμφανίζει την θορυβοποιημένη εικόνα.
- Το κουμπί **3** εμφανίζει την αποθορυβοποιημένη εικόνα.
- Το κουμπί **1 2** εμφανίζει την αρχική και την θορυβοποιημένη εικόνα.
- Το κουμπί **1 3** εμφανίζει την αρχική και την αποθορυβοποιημένη εικόνα.
- Το κουμπί **2 3** εμφανίζει την θορυβοποιημένη και την αποθορυβοποιημένη εικόνα.
- Το κουμπί με το **βιβλίο** οδηγεί στην φόρμα θεωρίας του προγράμματος.

Images TabSheet

Περιοχή εικόνων

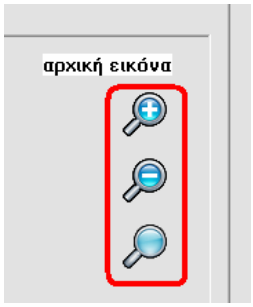
Η περιοχή εικόνων είναι εκείνη η περιοχή του προγράμματος στην οποία παρουσιάζονται οι εικόνες του είτε πρόκειται για την αρχική είτε για την θορυβοποιημένη είτε για την αποθορυβοποιημένη εικόνα.



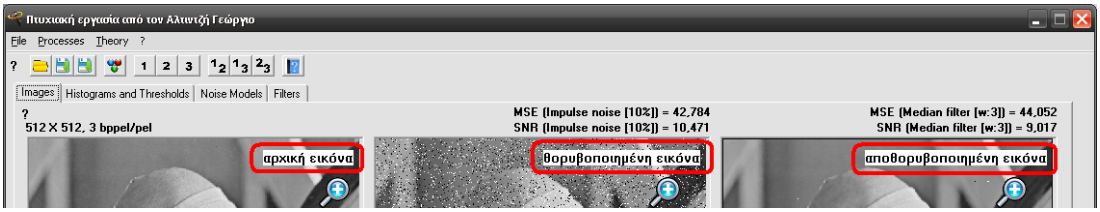


Στο πάνω μέρος της περιοχής εικόνων εμφανίζονται οι πληροφορίες σχετικά με την ανάλυση της υπό επεξεργασίας εικόνας και τα μεγέθη του μέσου τετραγωνικού σφάλματος MSE και του λόγου σήματος προς θόρυβο SNR στην θορυβοποιημένη και στην αποθορυβοποιημένη εικόνα. Στην θορυβοποιημένη εικόνα, μέσα σε παρένθεση δίπλα στα προαναφερθέντα μεγέθη, εμφανίζονται οι

πληροφορίες του θορύβου που προστέθηκε στην εικόνα (όνομα, επιλογές). Στην αποθορυβοποιημένη εικόνα, μέσα σε παρένθεση δίπλα στα προαναφερθέντα μεγέθη, ομοίως, εμφανίζονται οι πληροφορίες του φίλτρου που αποκαθιστά την θορυβοποιημένη εικόνα (όνομα, μέγεθος παραθύρου φίλτρου, επιλογές).

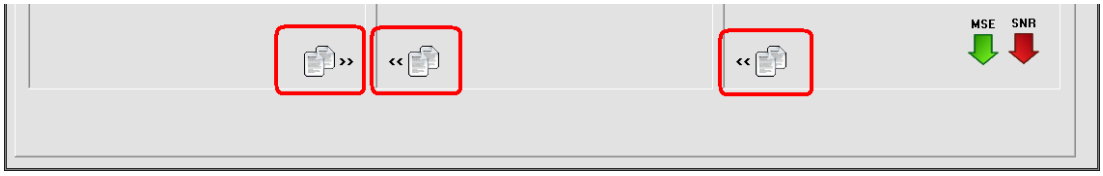


Το μενού εργαλείων παρέχει σημαντικό μέρος από τα κουμπιά που είναι απαραίτητα στην περιοχή εικόνων. Ωστόσο υπάρχουν και άλλα κουμπιά που πρέπει να περιγραφούν. Τα κουμπιά των μεγεθυντικών φακών που εμφανίζονται σε όλες τις εικόνες δίνουν -με τη σειρά εμφάνισής τους, από επάνω προς τα κάτω- την δυνατότητα μεγέθυνσης της εικόνας 2 φορές/πάτημα, τη δυνατότητα σμίκρυνσης της εικόνας 2 φορές/πάτημα και τη δυνατότητα επαναφοράς της εικόνας στο κανονικό της μέγεθος.



Σε κάθε εικόνα υπάρχει πάνω της μία ετικέτα μαύρων χαρακτήρων άσπρου φόντου που αποτελεί σύνδεσμο στην εμφάνιση ολόκληρης της εικόνας, μόνη της, στο δικό της παράθυρο. Πιο συγκεκριμένα:

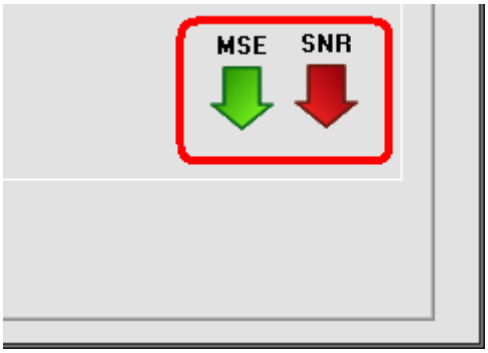
- της αρχικής στο παράθυρο που οδηγεί το κουμπί **1** του μενού εργαλείων,
- της θορυβοποιημένης στο παράθυρο που οδηγεί το κουμπί **2** του μενού εργαλείων και
- της αποθορυβοποιημένης στο παράθυρο που οδηγεί το κουμπί **3** του μενού εργαλείων.



Τα παραπάνω κουμπιά αποτελούν κουμπιά αντιγραφής μιας εικόνας προς την κατεύθυνση που δείχνουν τα διπλά βελάκια **>>** ή **<<** δίπλα σε κάθε εικονίδιο. Πιο συγκεκριμένα, από αριστερά προς τα δεξιά, υπάρχουν τα ακόλουθα κουμπιά:

- το κουμπί αντιγραφής της αρχικής εικόνας στην θορυβοποιημένη.
- της αρχικής στο παράθυρο που οδηγεί το κουμπί **1** του μενού εργαλείων,
- της θορυβοποιημένης στο παράθυρο που οδηγεί το κουμπί **2** του μενού εργαλείων και
- της αποθορυβοποιημένης στο παράθυρο που οδηγεί το κουμπί **3** του μενού εργαλείων.

Σημειώνεται ότι δεν είναι αναγκαία η χρήση των κουμπιών αντιγραφής για την αντιγραφή του περιεχομένου μιας εικόνας σε μια άλλη καθώς υπάρχει η δυνατότητα «Drag 'N' Drop». Η λειτουργία «Drag 'N' Drop» δίνει τη δυνατότητα αντιγραφής μιας εικόνας από μια άλλη κάνοντας αριστερό κλικ στην εικόνα προέλευσης, σέρνοντάς την πάνω από την εικόνα προορισμού (ή την περιοχή προορισμού που ανήκει η εικόνα προορισμού) και αφήνοντας ελεύθερο το πατημένο αριστερό κουμπί του ποντικιού.



Οι ενδείξεις με τα βελάκια σημαίνουν την μείωση ή την αύξηση των τιμών του μέσου τετραγωνικού σφάλματος MSE και του λόγου σήματος προς θόρυβο SNR της αποθορυβοποιημένης εικόνας σε σύγκριση με τα αντίστοιχα μεγέθη στην θορυβοποιημένη. Με πράσινο βελάκι που έχει κατεύθυνση προς τα κάτω συμβολίζεται η μείωση του MSE ενώ με κόκκινο και κατεύθυνση προς τα πάνω η αύξησή του. Με κόκκινο βελάκι που έχει κατεύθυνση προς τα κάτω συμβολίζεται η μείωση του SNR ενώ με πράσινο και κατεύθυνση προς τα πάνω η αύξησή του. Γενικά το πράσινο χρώμα χρησιμοποιείται για να δηλώσει μείωση του

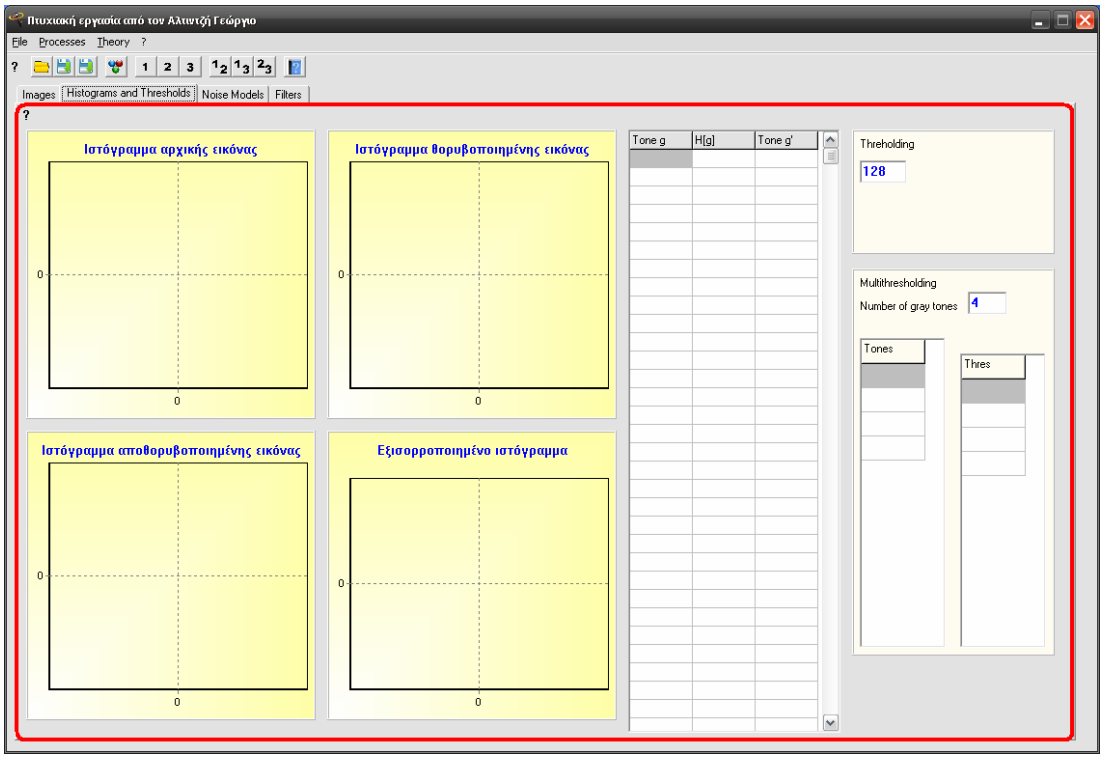
θορύβου ενώ το κόκκινο την αύξησή του. Από την άλλη πλευρά, κατεύθυνση προς τα πάνω σημαίνει την αύξηση ενός μεγέθους ενώ κατεύθυνση προς τα κάτω σημαίνει τη μείωσή του.

Τέλος, το κουμπί **?** οδηγεί στις πληροφορίες σχετικά με την περιοχή εικόνων.

Histograms and Thresholds TabSheet

Περιοχή ιστογραμμάτων και κατωφλίων

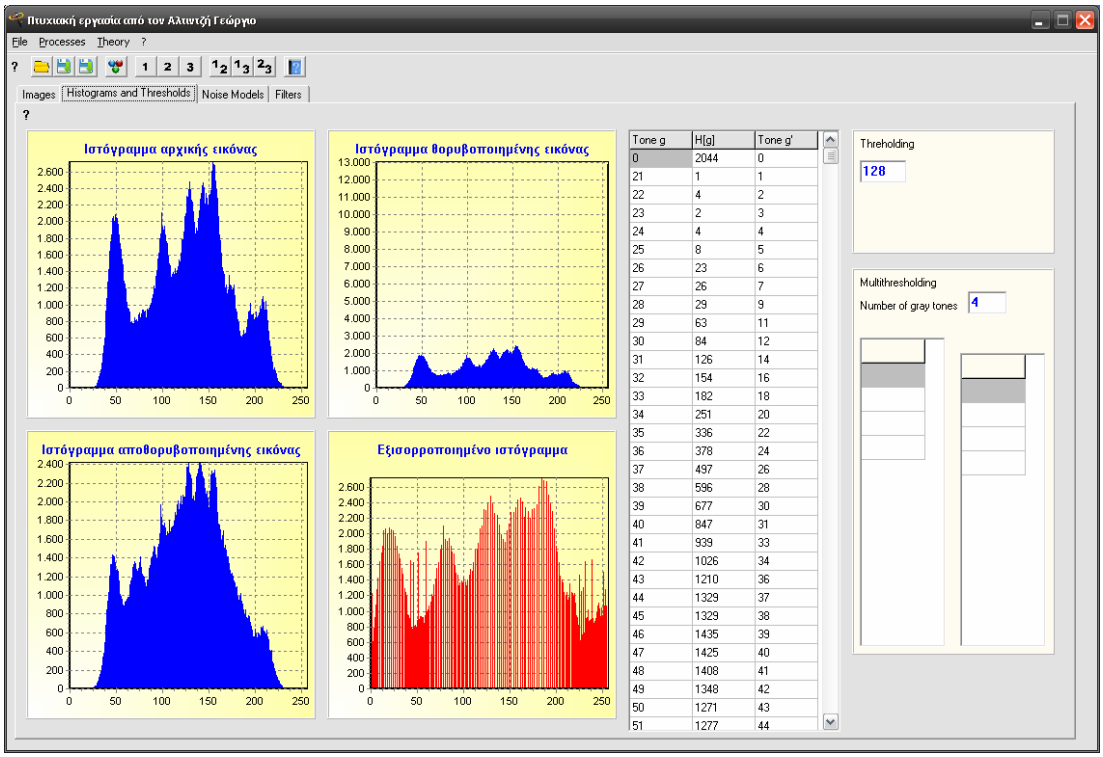
Η περιοχή ιστογραμμάτων και κατωφλίων είναι εκείνη η περιοχή του προγράμματος στην οποία παρουσιάζονται αφενός τα ιστογράμματα της αρχικής της θορυβοποιημένης και της αποθορυβοποιημένης εικόνας και αφετέρου τα κατώφλια στην μεθοδο της κατωφλίωσης και της πολυκατωφλίωσης. Εμφανίζει και το εξισορροποιημένο ιστόγραμμα σε μία από τις τρεις εικόνες που δέχεται η περιοχή εικόνων.



Παρατίθεται από ένα παράδειγμα από την χρήση των παραπάνω μερών για μια εικόνα.

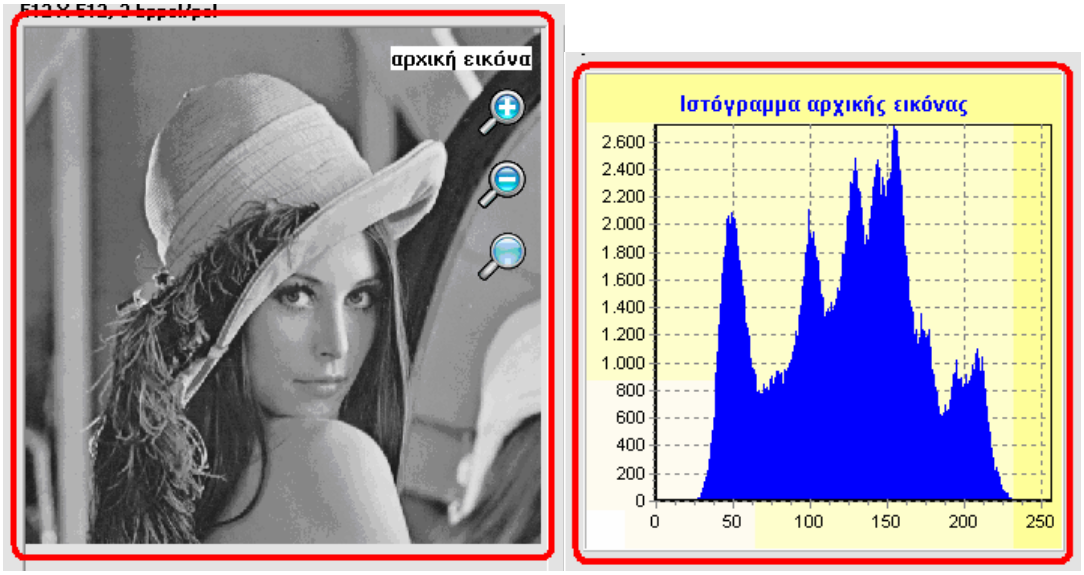
Παράδειγμα ιστογραμμάτων:

θορυβοποίηση και αποθορυβοποίηση της αρχικής ειικόνας

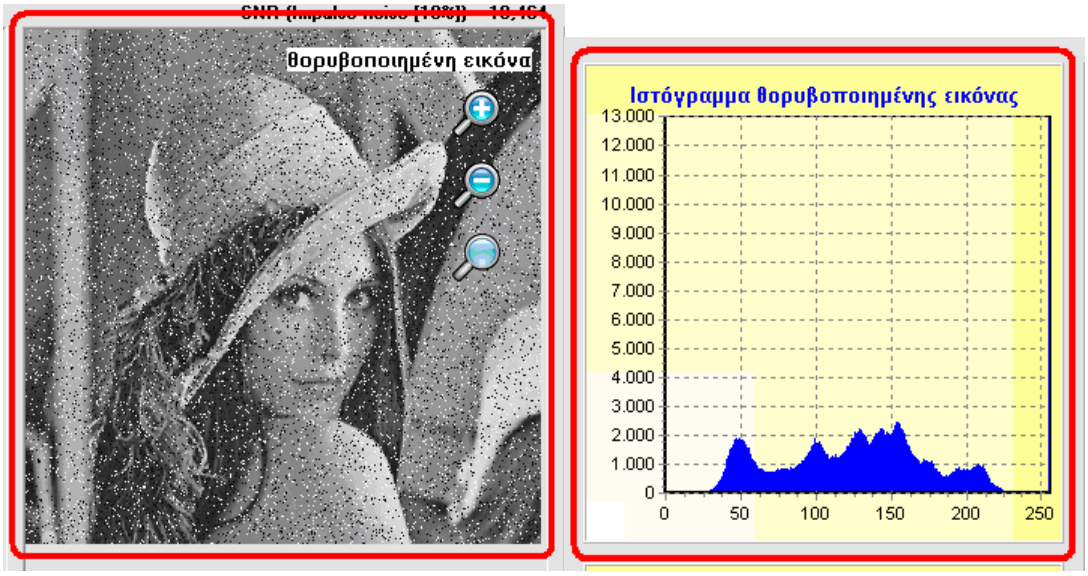


εξαγωγή ιστογράμματος για όλες τις εικόνες και εξισορρόπηση ιστογράμματος για την αρχική

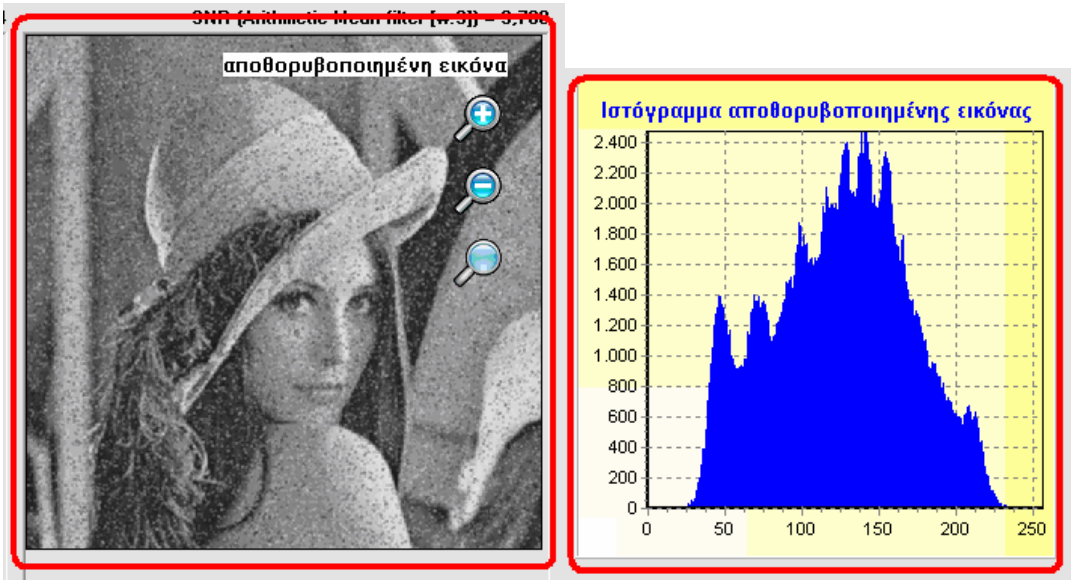
η αρχική εικόνα μετά την εξισορρόπηση ιστογράμματος



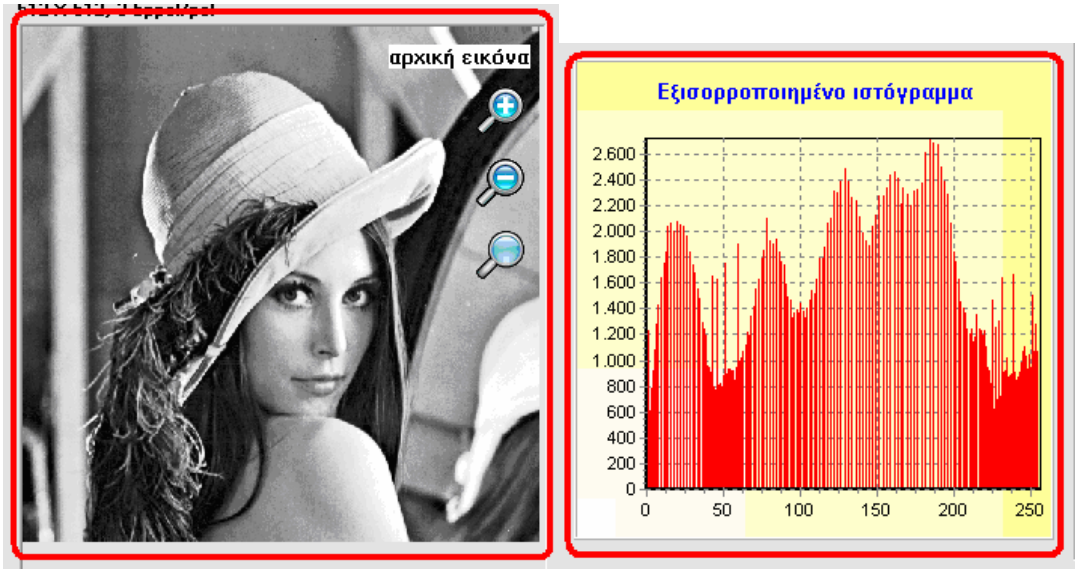
η αρχική εικόνα με το ιστόγραμμά της



η θορυβοποιημένη εικόνα με το ιστόγραμμά της



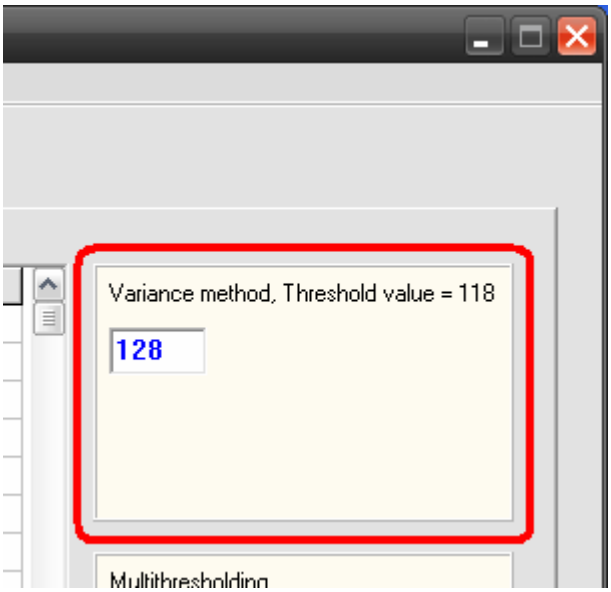
η αποθορυβοποιημένη εικόνα με το ιστόγραμμά της



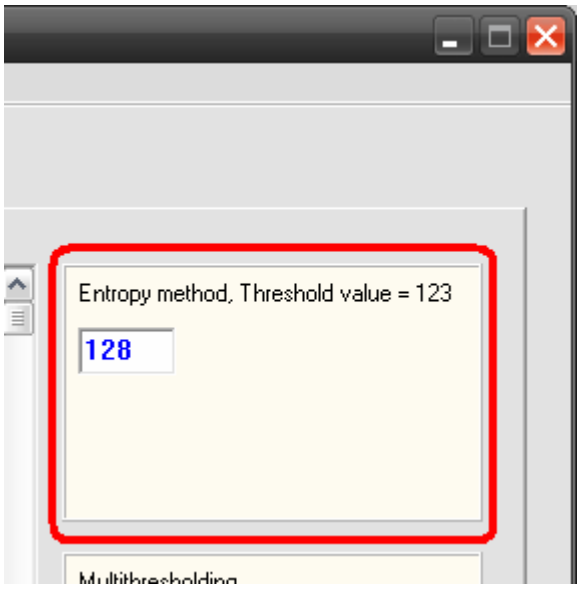
η εξισορροποιημένη αρχική εικόνα με το εξισορροποιημένο ιστόγραμμά της

Παραδείγματα κατωφλίωσης:

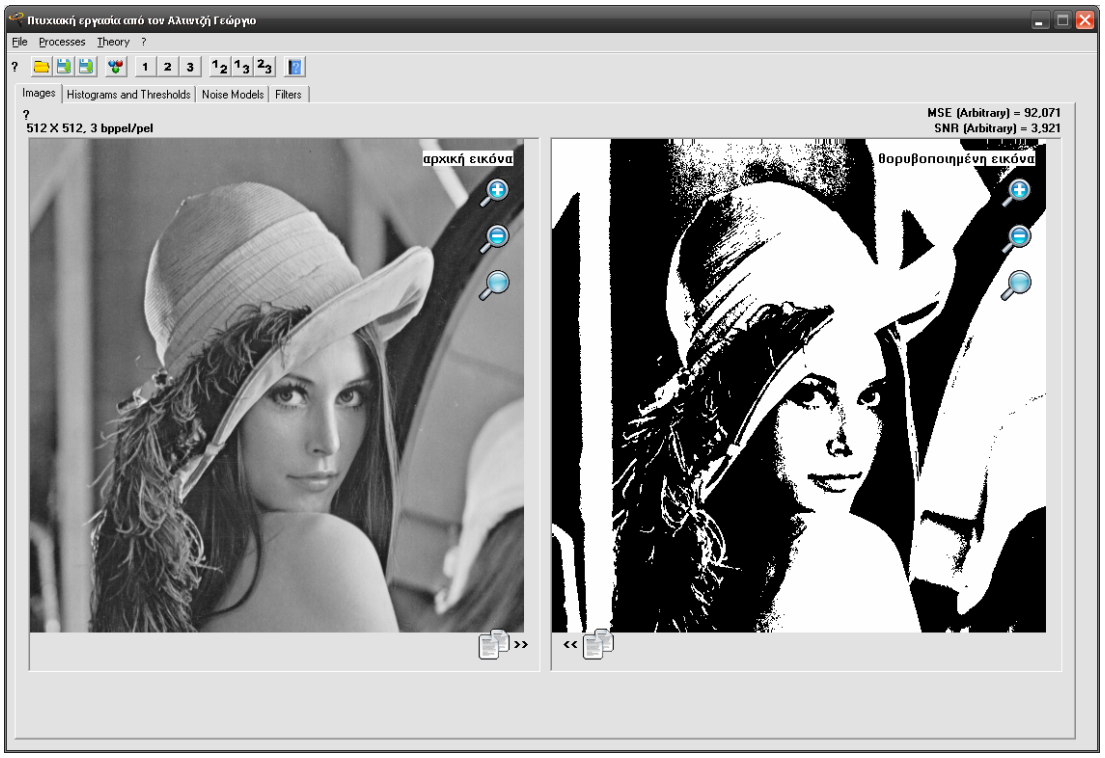
Μέθοδος διακύμανσης (Otsu) με κατώφλι 118.

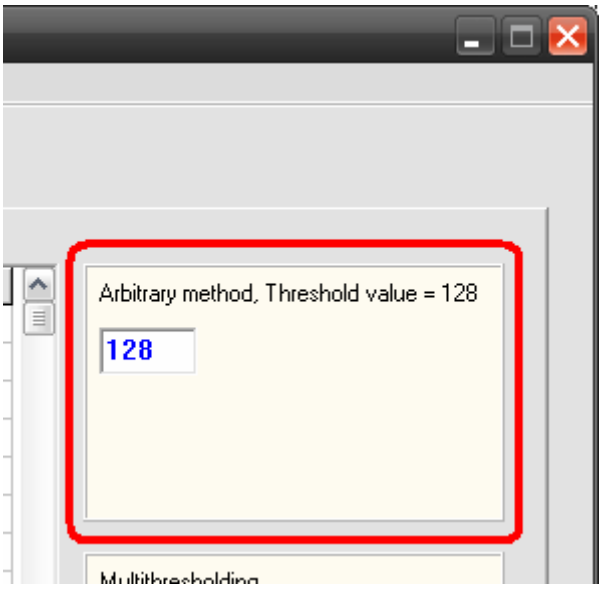


Μέθοδος εντροπίας (Kapur) με κατώφλι 123.

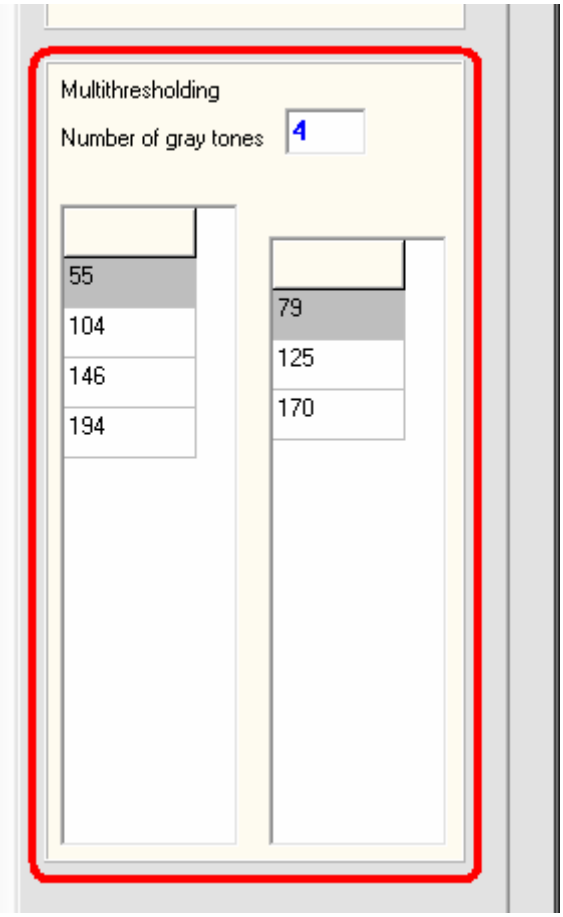


Μέθοδος Arbitrary με κατώφλι ορισμένο από τον χρήστη (π.χ. 128).





Μέθοδος πολυκατωφλίωσης με πλήθος κατωφλίων ορισμένο από τον χρήστη (π.χ. 4).

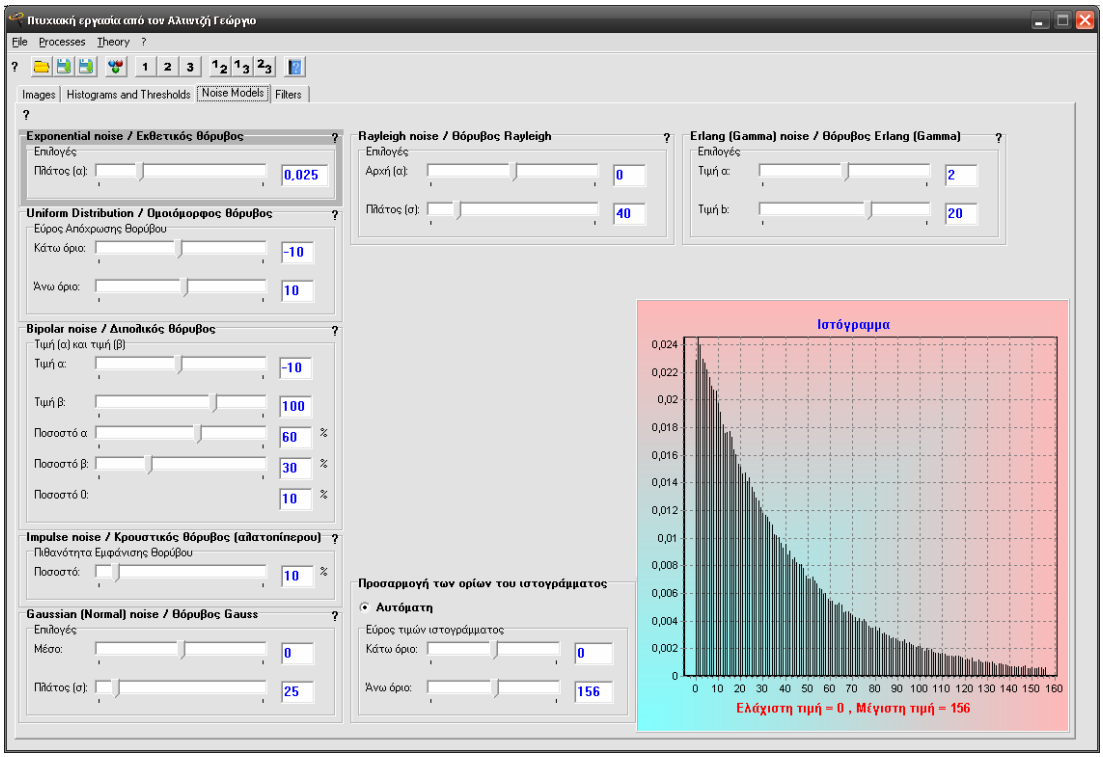


Τέλος, το κουμπί **?** οδηγεί στις πληροφορίες σχετικά με την περιοχή ιστογραμμάτων και κατωφλίων.

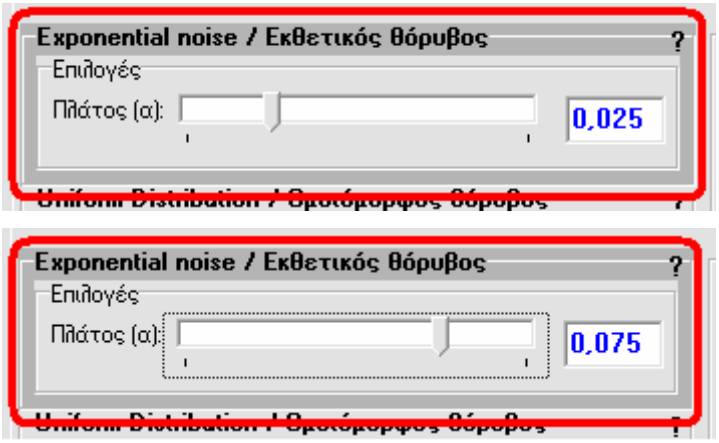
Noise Models TabSheet

Περιοχή μοντέλων θορύβου

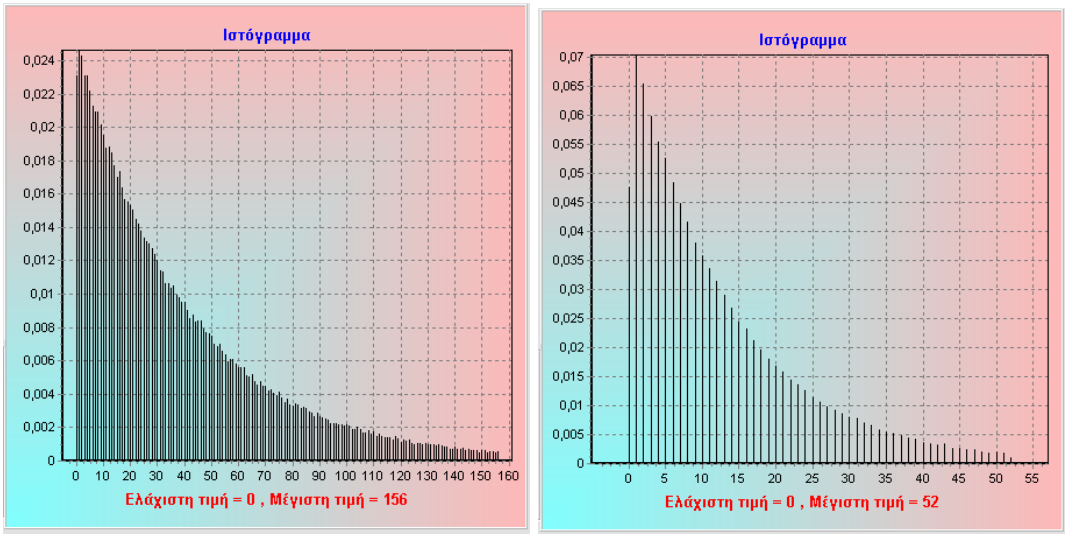
Η περιοχή μοντέλων θορύβου είναι εκείνη η περιοχή του προγράμματος στην οποία παρουσιάζονται τα μοντέλα των θορύβων και οι διαθέσιμες επιλογές ανά θόρυβο.



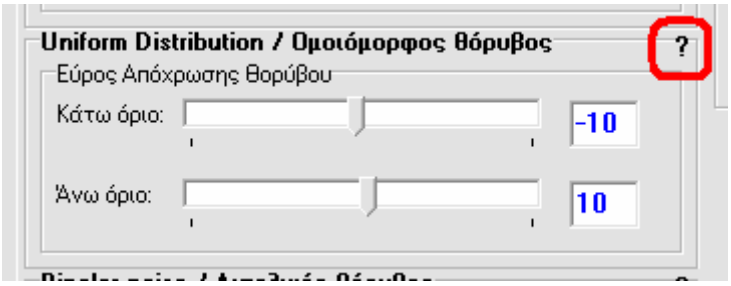
Κάθε μεταβολή στις τιμές των θορύβων έχει αντίκτυπο στο γράφημα που αποτυπώνει το μοντέλο του θορύβου. Για παράδειγμα, η μεταβολή του πλάτους του εκθετικού θορύβου από 0,025 σε 0,075:



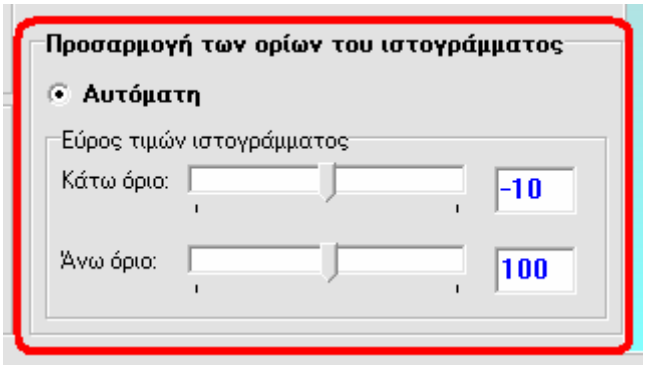
οδηγεί στην ακόλουθη μεταβολή γραφημάτων:



Αυτές είναι και οι τιμές που θα χρησιμοποιηθούν για την εφαρμογή του θορύβου πάνω στην εικόνα.



Στο πάνω αριστερό μέρος του πλαισίου επιλογών κάθε θορύβου υπάρχει ένα κουμπί **?** που οδηγεί στη θεωρία του και στον κώδικα της υλοποίησής του.



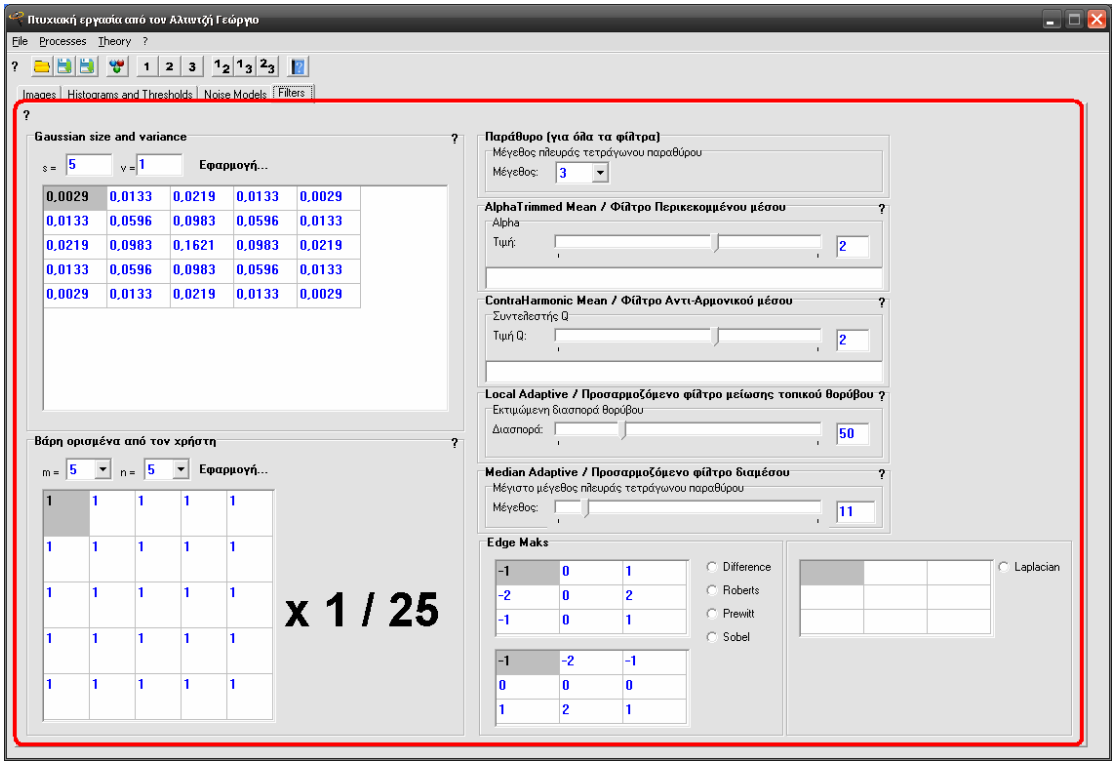
Κάτω αριστερά από το γράφημα του μοντέλου του θορύβου υπάρχει ένα μενού επιλογών σε ένα πλαίσιο όπου μπορεί να ορισθεί το εύρος τιμών του ιστογράμματος. Για να γίνει αυτό θα πρέπει να αποεπιλεχθεί με διπλό αριστερό κλικ η επιλογή **Αυτόματη** και να τεθούν οι επιθυμητές τιμές. Με μονό αριστερό κλικ μπορεί να τεθεί ξανά σε λειτουργία η αυτόματη οριοθέτηση του εύρους τιμών του γραφήματος.

Τέλος, το κουμπί **?** πάνω αριστερά οδηγεί στις πληροφορίες σχετικά με την περιοχή μοντέλων θορύβου.

Filters TabSheet

Περιοχή φίλτρων

Η περιοχή φίλτρων είναι εκείνη η περιοχή του προγράμματος στην οποία παρουσιάζονται κυρίως οι επιλογές για τα φίλτρα της εφαρμογής. Επίσης περιέχει τα απαραίτητα μενού για την εισαγωγή τιμών για τις μεθόδους εύρεσης ακμών στις εικόνες.



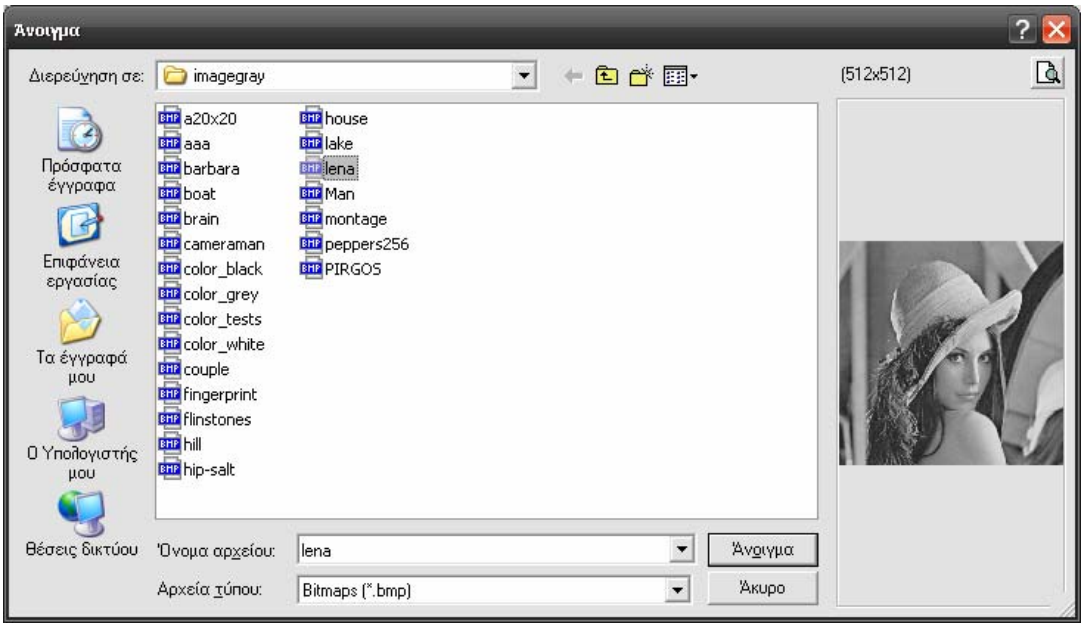
Από όλα τα φίλτρα μόνο το φίλτρο του περικεκομμένου μέσου, το φίλτρο του αντι-αρμονικού μέσου, το προσαρμοζόμενο φίλτρο μείωσης τοπικού θορύβου και το προσαρμοζόμενο φίλτρο διαμέσου διαθέτουν παραμέτρους προς επεξεργασία. Από την άλλη σχεδόν για κάθε φίλτρο μπορεί να ρυθμιστεί το τετράγωνο παράθυρο της εφαρμογής του με πλευρά που ξεκινάει από το 3. Εξαίρεση αποτελούν το φίλτρο Gauss και το «φίλτρο» διαφόρων βαρών που εφαρμόζονται με μάσκες παραμετροποιήσιμες σε αυτήν την περιοχή.

Όσον αφορά τις ακμές δεν μπορούν να αναλυθούν σε αυτό το εγχειρίδιο καθώς δεν αποτελούν μέρος της πτυχιακής εργασίας.

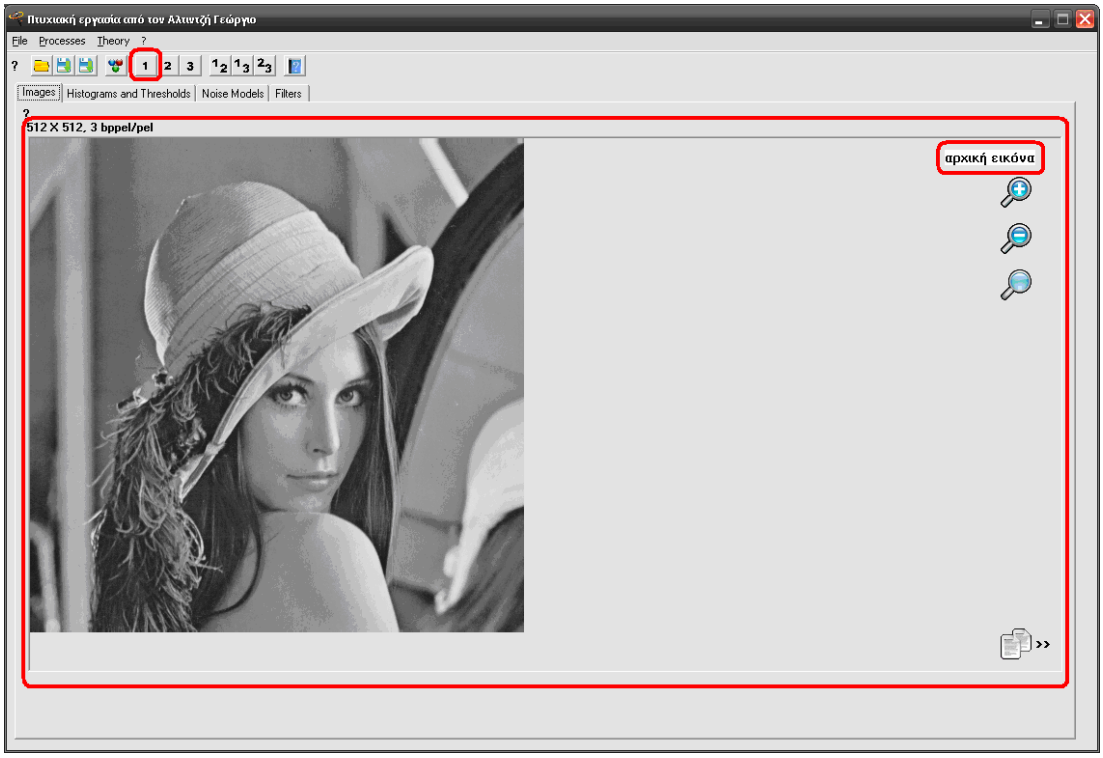
Τέλος, το κουμπί **?** πάνω αριστερά οδηγεί στις πληροφορίες σχετικά με την περιοχή φίλτρων.

Άνοιγμα εικόνας

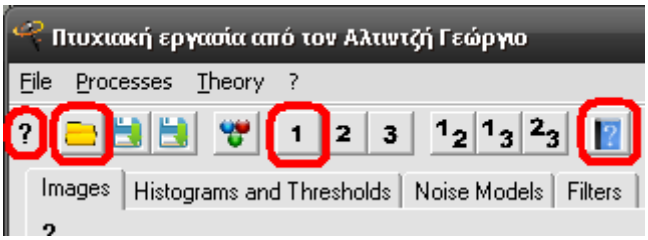
Το πρόγραμμα της πτυχιακής εργασίας μπορεί να επεξεργαστεί ασπρόμαυρες εικόνες αποχρώσεων του γκρι τύπου bitmap. Οπότε κατά τη διαδικασία του ανοίγματος αυτός είναι και ο μόνος τύπος δεδομένων που μπορεί να επιλεχθεί.



Αφού φορτωθεί μια εικόνα στη εφαρμογή, το πρόγραμμα θα την εμφανίσει στην περιοχή εικόνων, στο μέρος της αρχικής εικόνας.

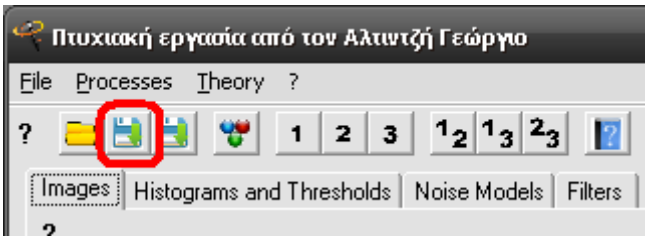


Σε αυτό το σημείο οποιαδήποτε εικόνα υπήρχε στο μέρος της θορυβοποιημένης και στο μέρος της αποθορυβοποιημένης θα διαγραφεί. Αυτό σημαίνει ότι από το μενού εργαλείων, από όλα τα κουμπιά, από αριστερά προς τα δεξιά, μόνο το κουμπί **?**, το κουμπί του **φακέλου**, το κουμπί **1** και το κουμπί με το **βιβλίο** θα λειτουργούν.

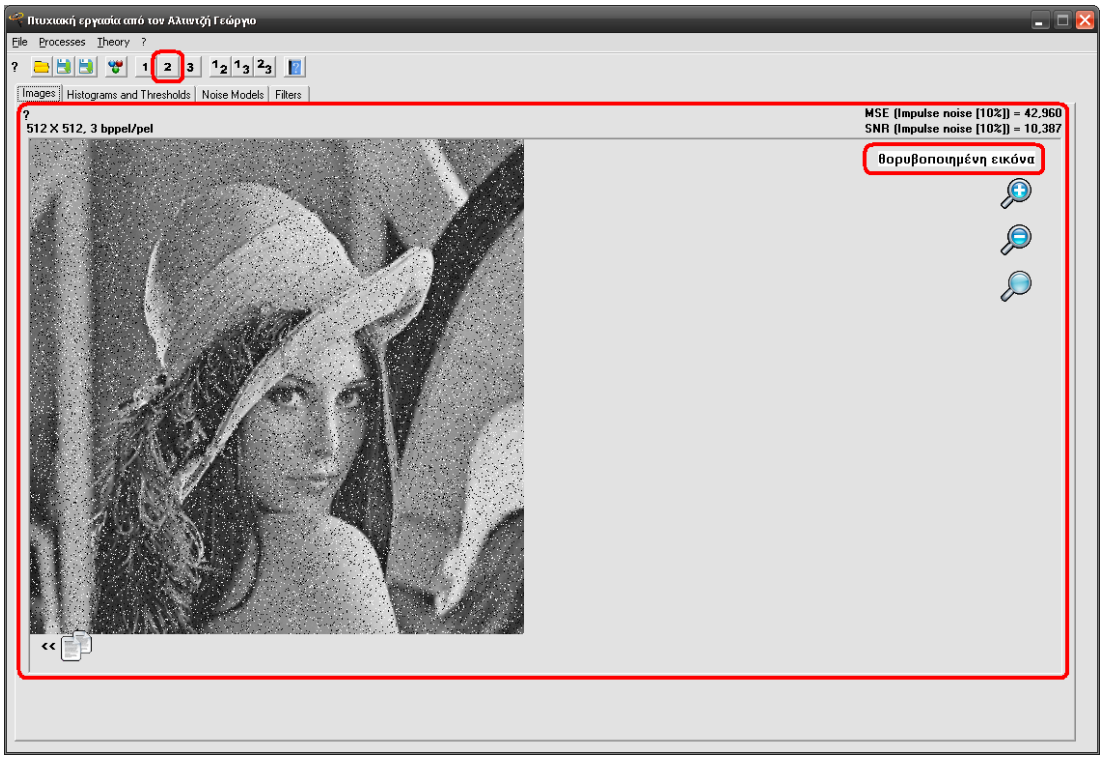


Αποθήκευση θορυβοποιημένης εικόνας

Αποθήκευση θορυβοποιημένης εικόνας ονομάζεται η αποθήκευση της εικόνας που βρίσκεται στο μέρος της θορυβοποιημένης εικόνας στην περιοχή εικόνων.

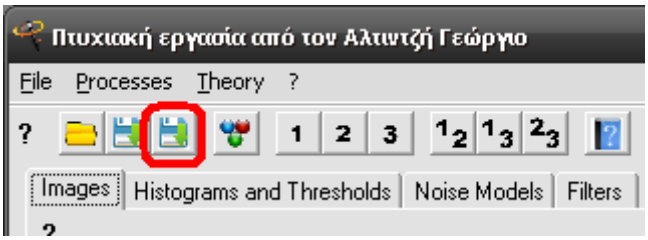


Ως θορυβοποιημένη εικόνα αναφέρεται εκείνη η εικόνα που εμφανίζεται ολόκληρη και μόνης της όταν πατηθεί το κουμπί **2** από το μενού εργαλείων. Οφείλει το όνομά της στο ότι αποτελεί, συνήθως, το αποτέλεσμα από την προσθήκη θορύβου στην αρχική εικόνα.

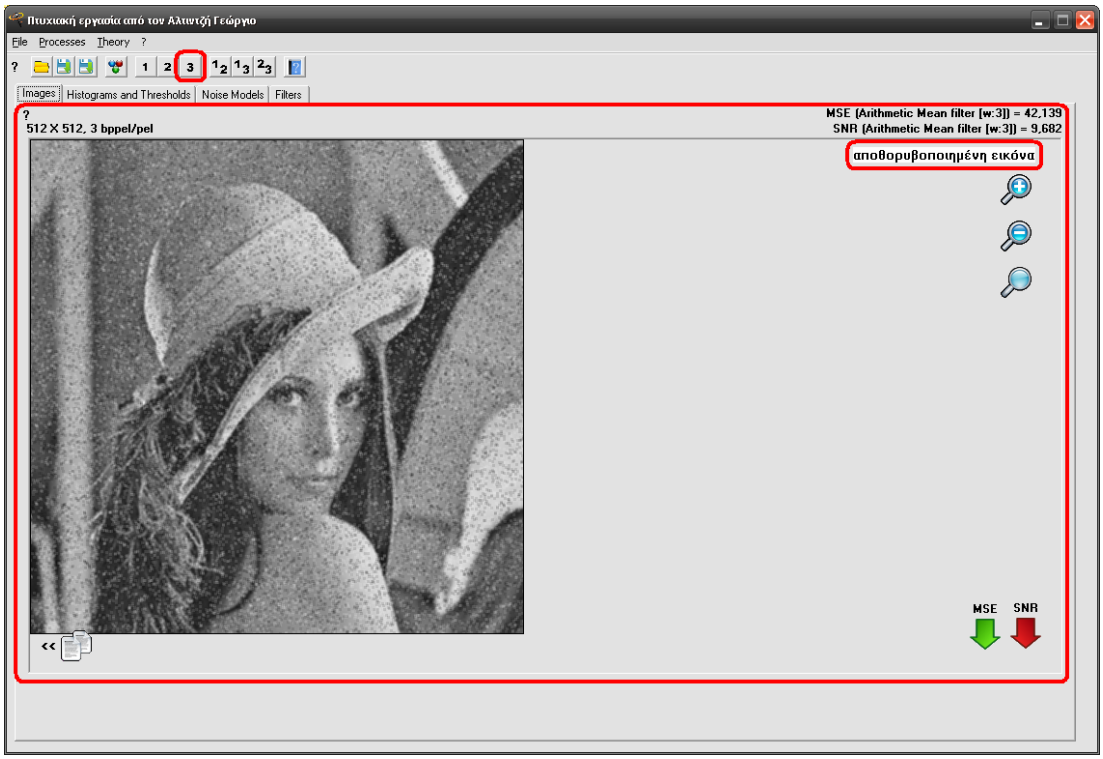


Αποθήκευση αποθορυβοποιημένης εικόνας

Αποθήκευση αποθορυβοποιημένης εικόνας ονομάζεται η αποθήκευση της εικόνας που βρίσκεται στο μέρος της αποθορυβοποιημένης εικόνας στην περιοχή εικόνων.

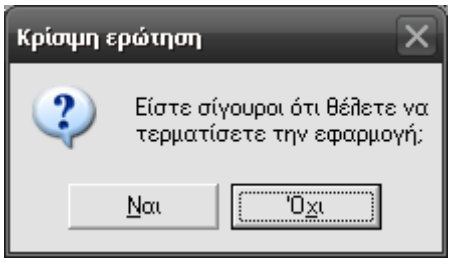


Ως αποθορυβοποιημένη εικόνα αναφέρεται εκείνη η εικόνα που εμφανίζεται ολόκληρη και μόνης της όταν πατηθεί το κουμπί **3** από το μενού εργαλείων. Οφείλει το όνομά της στο ότι αποτελεί, συνήθως, το αποτέλεσμα από την μείωση του θορύβου στην θορυβοποιημένη εικόνα.

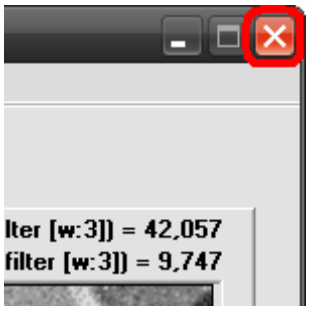


Έξοδος

Έξοδος ονομάζεται ο τερματισμός της εφαρμογής και πραγματοποιείται από το μενού **File** του κεντρικού μενού. Πριν αυτή ολοκληρωθεί υπάρχει μια ερώτηση ασφαλείας που ουσιαστικά υπέχει θέση προειδοποίησης για την επικύρωση του τερματισμού.

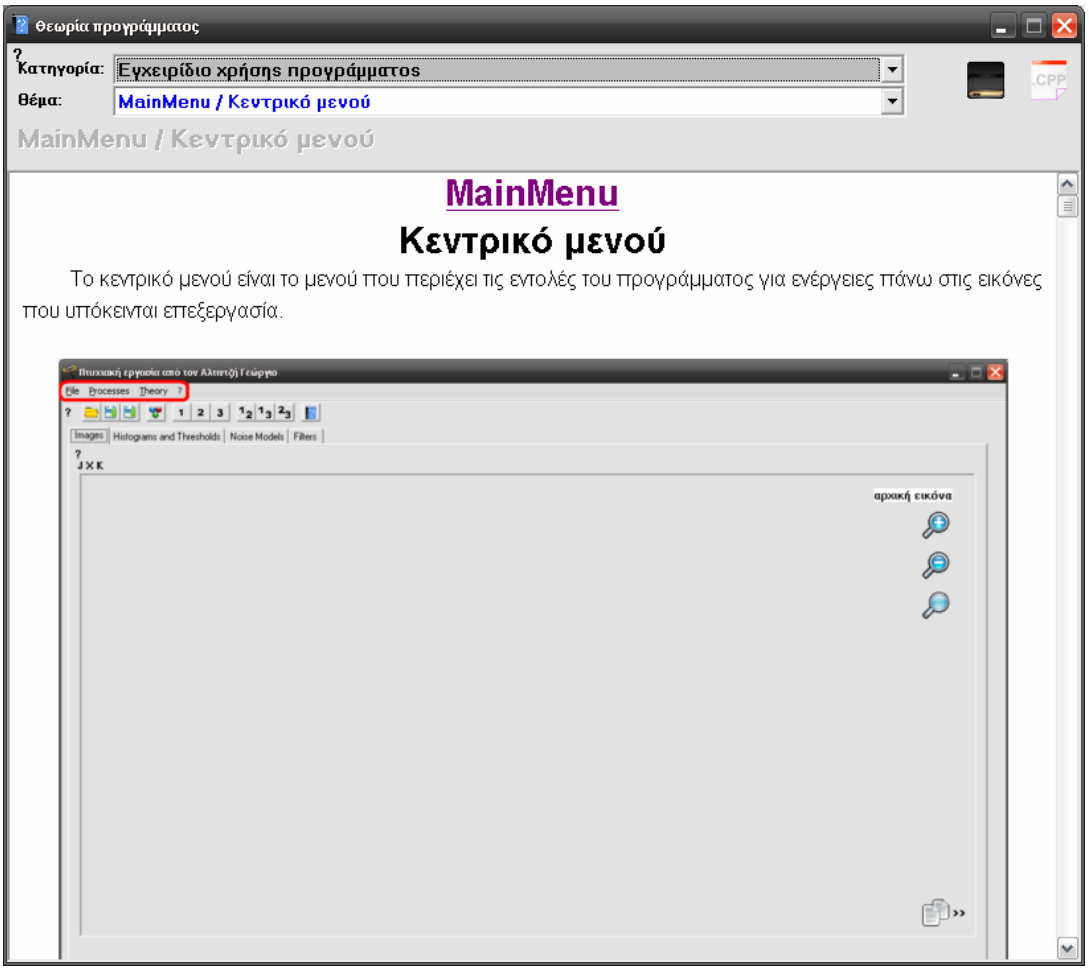


Έξοδος από το πρόγραμμα μπορεί να επιτευχθεί και πιο απλά, χωρίς την εμφάνιση της παραπάνω ερώτησης, εφόσον πατηθεί το πάνω δεξιά σύμβολο εξόδου, το **X**.



Φόρμα θεωρίας προγράμματος

Ως φόρμα θεωρίας του προγράμματος ορίζεται εκείνη η περιοχή της εφαρμογής που περιέχει το σύνολο της θεωρίας και το ουσιαστικό μέρος του πηγαίου κώδικα της πτυχιακής εργασίας.



Φόρμα θεωρίας του προγράμματος είναι εκείνη η περιοχή στην οποία οδηγούν τα κουμπιά **?** και περιλαμβάνει τις ακόλουθες κατηγορίες:

- Εγχειρίδιο χρήσης προγράμματος
- Θόρυβοι
- Φίλτρα

Κάθε κατηγορία εμφανίζει μια λίστα θεμάτων από κάτω της. Για να μεταβεί κάποιος σε ένα άλλο θέμα αρκεί να διαλέξει ένα από τα υπάρχοντα. Μόλις το διαλέξει το κείμενο της θεωρίας θα κινηθεί προς την επιθυμητή κατεύθυνση.

Σημειώνεται ότι αυτή η περιοχή διαπραγματεύεται οτιδήποτε αποτελεί θέμα της παρούσας πτυχιακής εργασίας και μπορεί να αναλυθεί είτε σε θεωρητικό κείμενο είτε και σε πηγαίο κώδικα.

Όσον αφορά την κατηγορία των θορύβων και την κατηγορία των φίλτρων, η μετάβαση από την θεωρία στον πηγαίο κώδικα και αντίστροφα γίνεται με το κουμπί του **βιβλίου** και το κουμπί του **πηγαίου κώδικα** ανάλογα με την επιθυμία του χρήστη ανεξάρτητα με το σημείο περιήγησής του και σύμφωνα πάντα με το τελευταίο θέμα που έχει επιλέξει.

Τέλος, το κουμπί **?** πάνω αριστερά οδηγεί στις πληροφορίες σχετικά με την φόρμα θεωρίας του προγράμματος.