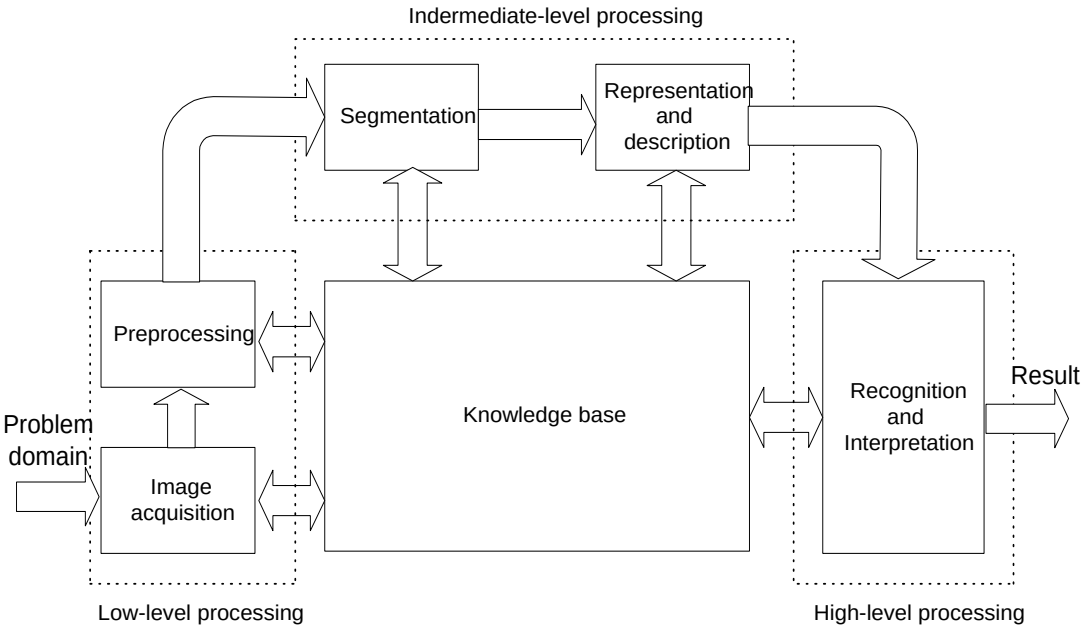
ΚΕΦΑΛΑΙΟ 1: ΕΠΕΞΕΡΓΑΣΙΑ, ΑΝΑΛΥΣΗ ΚΑΙ ΤΜΗΜΑΤΟΠΟΙΗΣΗ ΕΙΚΟΝΑΣ

1.1 ΒΑΣΙΚΑ ΒΗΜΑΤΑ ΣΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Η ψηφιακή επεξεργασία εικόνας (Digital Image Processing) είναι το μέσο εκείνο με το οποίο περνάμε από την περιοχή ενός προβλήματος (problem domain),που φυσικά σχετίζεται με εικόνα, στο τελικό αποτέλεσμα (βλ. σχήμα 1.1). Για παράδειγμα ας θεωρήσουμε το πρόβλημα της αυτόματης ανάγνωσης διευθύνσεων πάνω σε φακέλους. Στην περίπτωση αυτή, η περιοχή του προβλήματος μας είναι οι φάκελοι με τις διευθύνσεις, και ο σκοπός είναι η ανάγνωση των διευθύνσεων που βρίσκονται πάνω στους φακέλους, τον οποίο βέβαια θα πετύχουμε με την βοήθεια της ψηφιακής επεξεργασίας εικόνας. Οπότε, η επιθυμητή έξοδος (αποτέλεσμα) σ' αυτή την περίπτωση είναι μία σειρά από αλφαριθμητικούς χαρακτήρες (δηλαδή οι διευθύνσεις πάνω στους φακέλους).



Σχήμα 1.1 Βασικά βήματα στην επεξεργασία εικόνας.

Στο σχήμα 1.1 βλέπουμε τα βασικά βήματα της επεξεργασίας εικόνας **[1]**. Το πρώτο βήμα είναι η απόκτηση της εικόνας (Image acquisition). Αυτό απαιτεί την ύπαρξη αισθητηρίου εικόνας (imaging sensor) και τη δυνατότητα της ψηφιοποίησης του σήματος που παίρνουμε από το αισθητήριο. Αφού λοιπόν πάρουμε σε ψηφιακή μορφή την εικόνα την οποία θέλουμε, το επόμενο βήμα έχει να κάνει με την προεπεξεργασία της εικόνας (Image Preprocessing). Η βασική λειτουργία της προεπεξεργασίας είναι να βελτιώσει την εικόνα την οποία έχουμε πάρει, χρησιμοποιώντας τεχνικές τόνωσης της εικόνας (Image enhancement), αφαίρεσης του θορύβου κλπ. Τα δύο πρώτα βήματα, όπως φαίνεται και από το σχήμα 1.1, ανήκουν στο χαμηλό επίπεδο επεξεργασίας (Low level processing).

Το επόμενο βήμα, το οποίο και μας αφορά περισσότερο, ασχολείται με την τμηματοποίηση της εικόνας (Image segmentation). Λέγοντας τμηματοποίηση εννοούμε τον χωρισμό της εικόνας στα τμήματα ή αντικείμενα από τα οποία αποτελείται. Γενικά, η τμηματοποίηση είναι ένα από τα δυσκολότερα κομμάτια της ψηφιακής επεξεργασίας εικόνας. Μία καλή διαδικασία τμηματοποίησης συμβάλλει αποφασιστικά στην επίλυση ενός προβλήματος εικόνας. Από την άλλη όμως, ένας προβληματικός αλγόριθμος για segmentation μπορεί σχεδόν πάντα να εγγυηθεί αποτυχία στο τελικό αποτέλεσμα. Αναλυτικότερα για το πρόβλημα του segmentation, τις μεθόδους που έχουν κατά καιρούς προταθεί και τα αποτελέσματα στα οποία αυτές οδηγούν, θα αναφερθούμε στο κεφάλαιο που ακολουθεί.

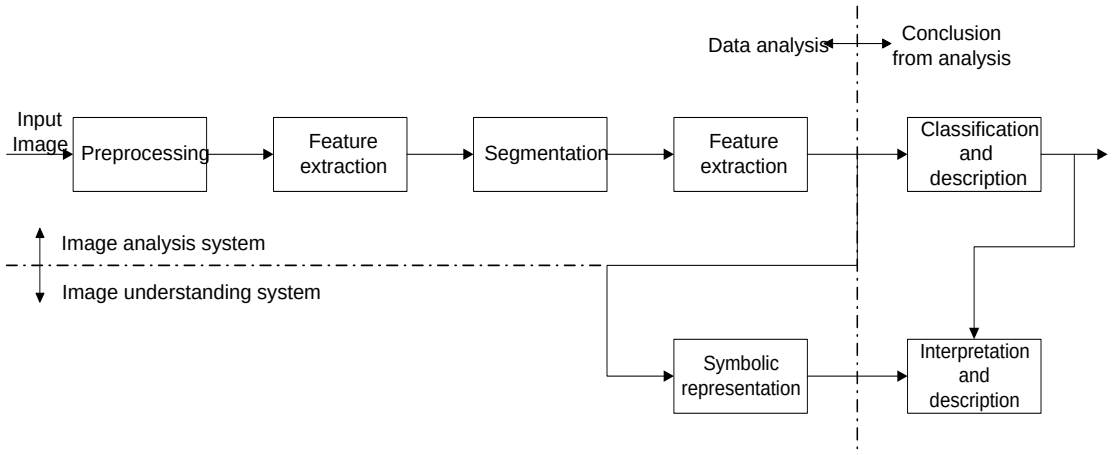
Η έξοδος (το αποτέλεσμα) της διαδικασίας του segmentation είναι συνήθως ακατέργαστα δεδομένα υπό τη μορφή pixels, τα οποία αποτελούν είτε τα σύνορα των περιοχών της εικόνας είτε ολόκληρο το εσωτερικό των περιοχών της εικόνας. Όμως σε κάθε περίπτωση είναι απαραίτητη η μετατροπή των δεδομένων σε μία μορφή κατάλληλη για επεξεργασία από υπολογιστή. Περάσαμε δηλαδή στη φάση της αναπαράστασης και της περιγραφής των αντικειμένων της εικόνας, η οποία έχει να κάνει με τον τρόπο αναπαράστασης των δεδομένων του segmentation (αν δηλαδή θα αναπαριστούν σύνορα περιοχών ή τις ίδιες τις περιοχές) και την περιγραφή τους (σύμφωνα με την οποία θα είμαστε σε θέση να πούμε σε ποια κλάση αντικειμένων ανήκει το συγκεκριμένο). Τα στάδια της τμηματοποίησης, της αναπαράστασης και της περιγραφής ανήκουν, σύμφωνα με το σχήμα, στο ενδιάμεσο επίπεδο επεξεργασίας.

Το τελευταίο βήμα του σχήματος περιλαμβάνει την αναγνώριση και τη μετάφραση. Αναγνώριση είναι η διαδικασία με την οποία αποδίδουμε ένα χαρακτηρισμό (label) σε ένα αντικείμενο σύμφωνα πάντα με την πληροφορία την οποία έχουμε από το προηγούμενο στάδιο. Μετάφραση είναι η διαδικασία με την οποία αποδίδουμε νόημα σε ένα σύνολο από αναγνωρισμένα αντικείμενα. Τα δύο αυτά τελευταία στάδια ανήκουν (βλ. σχήμα) στο υψηλο επίπεδο επεξεργασίας. Τέλος, όπως φαίνεται από το σχήμα, όλα τα προηγούμενα στάδια αλληλεπιδρούν με τη βάδη δεδομένων του συστήματος επεξεργασίας εικόνας με την οποία ανταλλάσουν πληροφορίες,για την γρηγορότερη και ακριβέστερη προσέγγιση του τελικού αποτελέσματος.

1.2 ΤΕΧΝΗΤΗ ΟΡΑΣΗ

Ο απώτερος σκοπός ενός μεγάλου αριθμού εφαρμογών επεξεργασία εικόνας είναι η εξαγωγή σημαντικών χαρακτηριστικών από δεδομένα εικόνας των οποίων η περιγραφή (description), μετάφραση (interpretation) και κατανόηση (understanding) μπορεί να γίνει κατόπιν από μία μηχανή. Για παράδειγμα, ένα σύστημα τεχνητής όρασης(computer vision system) μπορεί να διαχωρίσει τα κομμάτια σε μία γραμμή παραγωγής και να τα ταξινομήσει ανάλογα με τα χαρακτηριστικά τους (όπως το μέγεθος, το σχήμα κλπ). Πιο εξελιγμένα συστήματα τεχνητής όρασης μπορούν να μεταφράσουν τα αποτελέσματα των αναλύσεων, να περιγράψουν τα διάφορα αντικείμενα και να βρουν τις σχέσεις τους μέσα στη σκηνή. Κατά αυτή την έννοια η ανάλυση της εικόνας (Image Analysis) είναι διαφορετική από άλλες λειτουργίες επεξεργασίας εικόνες όπως ανακατασκευή (restoration), τόνωση (enhancement) και κωδικοποίηση (coding). Η Ανάλυση Εικόνας (Image Analysis) βασικά περιλαμβάνει την εξαγωγή χαρακτηριστικών (feature extraction), την τμηματοποίηση (segmentation), και την ταξινόμηση (classification).

Σε ένα σύστημα τεχνητής όρασης **[3]** όπως αυτό του σχήματος 1.2, η εικόνα που δίνουμε σαν είσοδο, πρώτα θα υποστεί μία επεξεργασία η οποία προφανώς περιλαμβάνει ανακατασκευή ή τόνωση έτσι ώστε να αποκτήσουμε μία καλή αναπαράσταση της εικόνας. Κατόπιν, εφαρμόζοντας τεχνικές τμηματοποίησης (segmentation), προσπαθούμε να εξάγουμε κάποια χαρακτηριστικά της εικόνας που συνήθως είναι κάποια περιγράμματα ή αντικείμενα. Έπειτα, η τμηματοποιημένη εικόνα τροφοδοτείται σε ένα ταξινονομητή ενός συστήματος κατανόησης (image understanding system).



Σχήμα 1.2 Ένα σύστημα τεχνητής όρασης.

Με την ταξινόμηση οι περιοχές και τα τμήματα της εικόνας αντιστοιχίζονται σε αντικείμενα στα οποία προσδίδονται και ονόματα (labels). Αυτό δηλαδή που κάνει ένα σύστημα ταξινόμησης, είναι να καθορίζει τις σχέσεις ανάμεσα σε διαφορετικά αντικείμενα σε μία σκηνή με σκοπό να μας δώσει την περιγραφή τους. Στο σχήμα 1.2, μπορούμε να δούμε πια είναι η θέση της τμηματοποίησης της εικόνας (Image Segmentation) σε ένα σύστημα τεχνητής όρασης και ποια η σχέση της με την διαδικασία της ανάλυσης της εικόνας (Image Analysis). Βλέπουμε δηλαδή ότι το κομμάτι του segmentation, το οποίο είναι και αυτό που μας αφορά, αποτελεί μέρος της διαδικασίας της ανάλυσης της εικόνας η οποία με τη σειρά της αποτελεί το πρώτο κομμάτι ενός συτήματος τεχνητής όρασης.

1.3 ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

Η συγκεκριμένη διπλωματική εργασία μελετά τεχνικές τμηματοποίησης εικόνας. Παρουσιάζονται κάποιες τεχνικές τμηματοποίησης οι οποίες είναι παρμένες από κάποιες πρόσφατες αναφορές. Πρώτα, υλοποιώντας τις τεχνικές αυτές διαπιστώσαμε την πολύ καλή τους απόδοση, και κατόπιν, παρεμβαίνοντας στις τεχνικές αυτές τροποποιώντας ή προσθέτοντας κάποια επιπλεόν ιδέα στη λειτουργία των αλγορίθμων, μελετήσαμε και συγκρίναμε τα τελικά αποτελέσματα. Μάλιστα σε κάποιες περιπτώσεις πετύχαμε βελτίωση ως προς το τελικό αποτέλεσμα, καθώς καταφέραμε να προσαρμόσουμε τον

αλγόριθμο στα μέτρα μας και να εντοπίσουμε περιοχές στην εικόνα για τις οποίες πραγματικά ενδιαφερόμασταν.

Ωστόσο, η μεγαλύτερη συμβολή μας στην περιοχή της τμηματοποίησης και γενικότερα της επεξεργασίας εικόνας, μπορεί να θεωρηθεί η προσπάθεια που έγινε για εφαρμογή της τμηματοποίησης στο πεδίο του wavelet, προσπάθεια που σε κάποιες περιπτώσεις έδωσε ικανοποιητικά αποτελέσματα και αν μη τι άλλο έδειξε ότι το συγκεκριμένο πεδίο είναι πολλά υποσχόμενο και στον τομέα της τμηματοποίησης εικόνας.

Στα επόμενα κεφάλαια θα μελετήσουμε κάποιους αλγορίθμους για τμηματοποίηση εικόνας, θα προτείνουμε κάποιους καινούριους και εφαρμόζοντας τους πάνω σε πραγματικές εικόνες θα συγκρίνουμε και θα σχολιάζουμε τα εκάστοτε αποτελέσματα. Στα σημεία εκείνα στα οποία κρίθηκε ότι ήταν απαρίτητο, παρουσιάζουμε κάποια θεωρητικά κομμάτια, τα οποία σκοπό έχουν να διευκολύνουν την ομαλή μετάβαση του αναγνώστη στα πιο εξειδικευμένα κομμάτια που θα ακολουθήσουν. Ας γίνουμε όμως πιο συγκεκριμένοι.

Ήδη στο παρόν κεφάλαιο (Κεφάλαιο 1), προσδιορίσαμε το πρόβλημα του “segmentation” σαν ένα βασικό αλλά δύσκολο βήμα (στάδιο) στην επεξεργασία εικόνας. Στο Κεφάλαιο 2 θα παρουσιάσουμε κάποιες γενικές κατηγορίες segmentation και θα δούμε κάποιες μεθόδους από κάθε κατηγορία. Στο Κεφάλαιο 3 θα παρουσιάσουμε ένα αλγόριθμο για ανίχνευση ακμών σε μία εικόνα, ενώ στο Κεφάλαιο 4 θα παρουσιάσουμε ένα αλγόριθμο για ανίχνευση των βασικών περιοχών σε μία εικόνα (clustering algorithm). Στο Κεφάλαιο 5 θα κάνουμε μία εισαγωγή στο πεδίο του wavelet, ενώ στο Κεφάλαιο 6 θα δούμε την εφαρμογή του clustering αλγόριθμου στο πεδίο του wavelet και θα προτείνουμε μία εναλλακτική μέθοδο για Image Segmentation της οποίας τα αποτελέσματα θα μελετήσουμε και θα σχολιάσουμε. Στο τελευταίο κεφάλαιο (Κεφάλαιο 7), θα παρουσιάσουμε τα συμπεράσματα στα οποία καταλήξαμε μετά το πέρας της εργασίας αυτής και θα προτείνουμε κάποιες ιδέες για μελλοντική εξέλιξη της εργασίας.

ΚΕΦΑΛΑΙΟ 2: ΤΕΧΝΙΚΕΣ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ

Στο κεφάλαιο αυτό θα παρουσιάσουμε μερικές βασικές κατηγορίες μεθόδων για τμηματοποίηση εικόνας **[1]**. Από κάθε κατηγορία θα παρουσιάσουμε τις βασικότερες μεθόδους, θα εξηγήσουμε τον τρόπο λειτουργίας τους, χωρίς ωστόσο να περάσουμε σε πολλές λεπτομέρειες, μιας και οι μέθοδοι αυτοί μας ενδιαφέρουν πιο πολύ για συγκρίσεις.

2.1 ΕΙΣΑΓΩΓΗ

Όπως είδαμε και στο Κεφάλαιο 1, μετά την προεπεξεργασία της εικόνας, το επόμενο βήμα στη διαδικασία της ανάλυσης της εικόνας, είναι η τμηματοποίησή της (segmentation). Η τμηματοποίηση διαιρεί την εικόνα στα κομμάτια (ή αντικείμενα) τα οποία την αποτελούν. Ο βαθμός στον οποίο αυτή η υποδιαίρεση προχωρά, εξατάται από τις απαιτήσεις του εκάστοτε προβλήματος. Γενικά το segmentation πρέπει να σταματά όταν τα αντικείμενα "που ενδιαφέρουν" σε μία εικόνα έχουν απομονωθεί (ανιχνευθεί).

Γενικά, η διαδικασία του segmentation είναι ένα από τα δυσκολότερα βήματα στην επεξεργασία εικόνας. Είναι το βήμα που θα καθορίσει, στο μεγαλύτερο ποσοστό, την τελική επιτυχία ή αποτυχία της ανάλυσης. Γι' αυτό, πρέπει να δίνεται ιδιαίτερη προσοχή, ώστε τα αποτελέσματα του segmentation να είναι τα καλύτερα δυνατά, πράγμα το οποίο δεν εξαρτάται μόνο από τη σχεδίαση ενός καλού αλγορίθμου για segmentation αλλά και από την εκτέλεση των προηγούμενων του segmentation βημάτων και από τις συνθήκες κάτω από τις οποίες θα εφαρμοστεί το segmentation.

Οι αλγόριθμοι για segmentation μονόγχρωμων εικονών χωρίζονται γενικά σε δύο κατηγορίες: η πρώτη βασίζεται στην ιδιότητα της ασυνέχειας των τιμών του gray-level και η δεύτερη στην ιδιότητα της ομοιότητας των τιμών αυτών. Στην πρώτη κατηγορία, η ιδεά είναι να χωρίσουμε την εικόνα βασιζόμενοι σε απότομες αλλαγές του gray-level. Βασικές μέθοδοι που ανήκουν στην κατηγορία αυτή, έχουν να κάνουν με την ανίχνευση απομονωμένων σημείων, την ανίχνευση γραμμών και γενικότερα την ανίχνευση ακμών σε μία εικόνα. Στη δεύτερη κατηγορία η ιδέα είναι να ομαδοποιήσουμε γειτονικά pixels

με παρόμοιο gray-level. Οι βασικές μέθοδοι της κατηγορίας αυτής έχουν να κάνουν με τις τεχνικές thresholding, region growing, και region splitting and meging.

Η ιδέα της τμηματοποίησης της εικόνας με βάση την ασυνέχεια ή την ομοιότητα των τιμών του gray-level (των pixels της εικόνας), βρίσκει εφαρμογή τόσο σε στατικές όσο και σε δυναμικές (time varying) εικόνες. Εμείς όμως, στο κεφάλαιο αυτό (όπως και στα επόμενα), θα ασχοληθούμε μόνο με την περίπτωση των στατικών εικονών.

2.2 ΑΝΙΧΝΕΥΣΗ ΑΣΥΝΕΧΕΙΩΝ

Στην παράγραφο αυτή παρουσιάζουμε κάποιες τεχνικές για ανίχνευση των τριών βασικών τύπων ασυνεχειών σε μία εικόνα: τα σημεία, τις γραμμές και τις ακμές. Στην πράξη ο πιο συνηθισμένος τρόπος να ψάξεις για ασυνέχειες είναι μέσω μίας μάσκας την οποία εφαρμόζεις στην εικόνα. Για παράδειγμα, χρησιμοποιώντας τη μάσκα

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

και εφαρμόζοντας την σε κάθε σημείο της εικόνας (συνήθως κεντράρουμε τη μάσκα στο σημείο που μας ενδιαφέρει), η απόκριση της μάσκας είναι το άθροισμα R:

$$\begin{aligned} R &= w_1 z_1 + w_2 z_2 + \ldots + w_9 z_9 \\ &= \sum_{i=1}^{9} w_i z_i \end{aligned} \qquad (2.1)$$

όπου z_i είναι το gray level του pixel που σχετίζεται με το συντελεστή μάσκας w_i .

2.2.1 Ανίχνευση σημείου.

Η ανίχνευση "απομονωμένων" σημείων (σημείων δηλαδή που είναι διαφορετικά από ένα σταθερό background) είναι αρκετά απλή και συνήθως γίνεται εφαρμόζοντας τη

μάσκα του σχήματος 2.1 στην εικόνα μας, οπότε λέμε ότι ένα σημείο έχει απομονωθεί όταν η απόκριση R της μάσκας στο σημείο αυτό είναι

$$|R| > T$$

όπου T είναι ένα μη αρνητικό κατώφλι το οποίο καθορίζει το είδος των σημείων που θα ανιχνέυσουμε (όσο πιο μικρό είναι το κατώφλι τόσο πιο "απομονωμένο" θα πρέπει να είναι το σημείο για ανιχνευθεί). Βασικά, η ιδέα είναι ότι το gray level ενός "απομονωμένου" σημείου θα είναι διαφορετικό από αυτό των γειτόνων του και αυτό ακριβώς θεωρεί το συγκεκριμένο φίλτρο το οποίο βλέπουμε ότι έχει μεγάλο και θετικό συντελεστή για το κεντρικό σημείο (8) και μικρούς και αρνητικούς συντελεστές για τους γειτονες (-1).

-1	-1	-1
-1	8	-1
-1	-1	-1

Σχήμα 2.1 Η μάσκα που χρησιμοποιούμε για την ανίχνευση "απομονωμένων" σημείων σε μία εικόνα.

2.2.2 Ανίχνευση γραμμών.

Το επόμενο επίπεδο πολυπλοκότητας αφορά την ανίχνευση γραμμών σε μία εικόνα. Ας θεωρήσουμε το σχήμα της επόμενης σελίδας (σχ. 2.2) στο οποίο βλέπουμε τέσσερις διαφορετικές μάσκες. Αν μετακινούσαμε (εφαρμόζαμε) την πρώτη μάσκα μέσα στην

-1	-1	-1
2	2	2
-1	-1	-1

Οριζόντια

-1	-1	2
-1	2	-1
2	-1	-1

$+45^0$

-1	2	-1
-1	2	-1
-1	2	-1

κάθετη

2	-1	-1
-1	2	-1
-1	-1	2

-45^0

Σχήμα 2.2 Μάσκες ανίχνευσης γραμμών διαφορετικών διευθύνσεων.

εικόνα, την υψηλότερη απόκριση θα την παίρναμε όταν η μάσκα περνούσε από γραμμές (με πάχος ένα pixel) με οριζόντια διεύθυνση, καθώς όπως βλέπουμε οι συντελεστές της μάσκας με υψηλή τιμή (2) βρίσκονται σε οριζόντια διάταξη. Ομοίως, συμπεραίνουμε ότι η δεύτερη μάσκα του σχήματος 2.2 θα αποκρινόταν καλύτερα σε γραμμές με διεύθυνση 45^0, η τρίτη μάσκα θα αποκρινόταν καλύτερα σε γραμμές με κάθετη διεύθυνση και η τέταρτη σε γραμμές με διεύθυνση -45^0.

Ας θεωρήσουμε τώρα ότι R_1, R_2, R_3, R_4 είναι οι αποκρίσεις των μασκών του σχήματος 2.2 από αριστερά προς τα δεξιά. Αν εφαρμόσουμε κάθε μάσκα χωριστά σε μία εικόνα και για ένα συγκεκριμένο pixel βρούμε ότι ισχύει $|R_i| > |Rj|$ για όλα τα $i \neq j$ τότε λέμε ότι το σημείο αυτό είναι πιο πιθανό να ανήκει σε γραμμή με διεύθυνση την διεύθυνση της μάσκας i. Υπό αυτή την έννοια δηλαδή θεωρούμε την ανίχνευση των γραμμών σε μία εικόνα.

2.2.3 Ανίχνευση ακμών.

Στην πράξη η ανίχνευση των ακμών σε μία εικόνα είναι αυτή που ενδιαφέρει περισσότερο καθώς σπάνια θα συναντήσουμε απομονωμένα σημεία ή γραμμές σε μία εικόνα. Αντίθετα αυτό που μας απασχολεί περισσότερο είναι η ανίχνευση των ακμών της εικόνας, δηλαδή των "συνόρων" μεταξύ δύο περιοχών. Στο υπόλοιπο της παραγράφου, η υπόθεση η οποία κάνουμε είναι ότι οι περιοχές της εικόνας μας είναι αρκετά ομογενείς, ώστε η ανίχνευση της διαχωριστικής τους γραμμής να μπορεί να καθοριστεί αποκλειστικά και μόνο χρησιμοποιώντας το gray level των περιοχών. Στην πράξη αυτό κάνουμε προκειμένου να ανιχνεύσουμε τις ακμές σε μία εικόνα είναι να εφαρμόσουμε ένα διαφορικό τελεστή ο οποίος είναι σε θέση να ανιχνεύσει τις "διαφορές" ανάμεσα στις περιοχές, δηλαδή τις ακμές τις εικόνας.

Ο διαφορικός τελεστής που ονομάζεται και "gradient", για μία εικόνα f(x,y) ορίζεται ως εξής:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} \qquad (2.2)$$

Είναι γνωστό, ότι το παραπάνω διάνυσμα "δείχνει" προς την κατεύθυνση του μέγιστου ρυθμού αλλαγής της f στο (x,y). Στην ανίχνευση ακμών αυτό που μας ενδιαφέρει είναι το μέτρο (magnitude) του παραπάνω διανύσματος (που συμβολίζεται με ∇f) δηλαδη η ποσότητα:

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = [G_x^2 + G_y^2]^{1/2}. \quad (2.3)$$

Αυτή η ποσότητα ισούται με το μέγιστο ρυθμό αύξησης της f(x,y) ανά μονάδα μήκους στην κατεύθυνση του $\nabla \mathbf{f}$. Συνήθως αυτό που κάνουμε είναι να προσεγγίζουμε την παραπάνω ποσότητα με απόλυτες τιμές (κάτι που διευκολύνει την υλοποίηση) ως εξής:

$$\nabla f \approx |G_x| + |G_y| \quad (2.4)$$

Όσο αφορά την υλοποίηση του παραπάνω τελεστή σε διακριτή μορφή υπάρχουν αρκετοί τρόποι. Για παράδειγμα ας θεωρήσουμε τους τελεστές Sobel που φαίνονται στο σχήμα 2.3 (β,γ). Χρησιμοποιώντας τις μάσκες (τελεστές) του σχήματος αυτού, μπορούμε να υπολογίσουμε τις παραγώγους G_x και G_y (στο σημείο z_5) ως εξής:

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3) \quad (2.5)$$

και

$$G_x = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7) \quad (2.6)$$

όπου, όπως και πριν, z_i είναι τα gray level των pixels (βλ. σχήμα 2.3(α)). Κατόπιν, για τον υπολογισμό του gradient στη θέση z_5, απλά κάνουμε χρήση της εξίσωσης (2.4).

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

(α)

-1	-2	-1
0	0	0
1	2	1

(β)

-1	0	1
-2	0	2
-1	0	1

(γ)

Σχήμα 2.3 (α) Μία 3 x 3 περιοχή εικόνας. (β) Μάσκα που χρησιμοποιείται για τον υπολογισμό του G_x στο κεντρικό σημείο (z_5) της 3 x 3 περιοχής εικόνας. (γ) Μάσκα για τον υπολογισμό του G_y στο ίδιο σημείο.

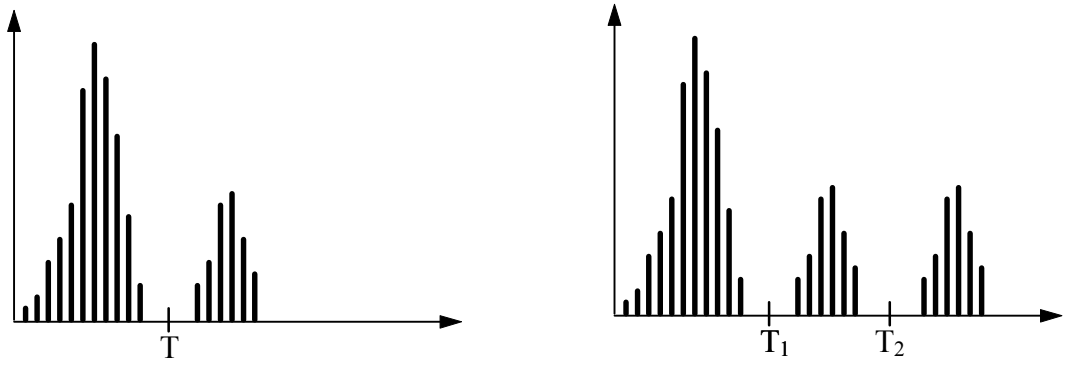
Την παραπάνω διαδικασία την εφαρμόζουμε για κάθε pixel της εικόνας μας και το αποτέλεσμα που παίρνουμε είναι η "gradient" εικόνα της αρχικής (που φυσικά έχει τις ίδιες διαστάσεις) και στην οποία έχουν ανιχνευθεί οι ακμές της εικόνας μας.

Η τεχνική που είδαμε στην παράγραφο αυτή για την ανίχνευση των ακμών σε μία εικόνα, είναι ίσως η απλούστερη που μπορεί να εφαρμοστεί για τον συγκεκριμένο σκοπό. Στηρίζεται στην εφαρμογή (σε κάθε σημείο της εικόνας) ενός φίλτρου του οποίου οι ιδιότητες ευνοούν την ανίχνευση των μεταβολών του gray level σε μία εικόνα. Για να δώσει ικανοποιητικά αποτελέσματα η συγκεκριμένη τεχνική, πρέπει να εφαρμοστεί σε εικόνες με σχετικά ομαλές περιοχές και απαλλαγμένες από θόρυβο. Κι αυτό γιατί, η μη ομαλότητα των περιοχών ή η παρουσία θορύβου, θα δημιουργήσει μεταβολές στην εικόνα μας οι οποίες θα ανιχνευθούν από το φίλτρο, κάνοντας έτσι μη αποτελεσματική τη χρήση του.

Ωστόσο, το πλεονέκτημα της τεχνικής είναι ότι δίνει αποτελέσματα αρκετά γρήγορα, καθώς το μόνο που έχει να γίνει είναι η εφαρμογή του φίλτρου σε κάθε σημείο της εικόνας και ο υπολογισμός της νέας τιμής. Συνοψίζοντας, θα λέγαμε ότι η συγκεκριμένη τεχνική συνήθως ενδείκνυται για την εφαρμογή της σε εικόνες με σκοπό μία πρώτη εκτίμηση των ασυνεχειών (ακμών) της εικόνας, και όχι για την εξαγωγή αποτελεσμάτων για περαιτέρω επεξεργασία.

2.3 THRESHOLDING

Η τεχνική του thresholding είναι από τις πιο σημαντικές προσεγγίσεις στην τμηματοποιήση εικόνας. Η "λογική" της τεχνικής αυτής μπορεί να εξηγηθεί με τη βοήθεια του σχήματος 2.4. Στο σχήμα 2.4(α) βλέπουμε το πιθανό ιστόγραμμα μίας εικόνας με ένα φωτεινό αντικείμενο σε ένα σκοτεινό background. Ένας προφανής τρόπος για να διαχωρίσουμε το αντικείμενο από το background, είναι να θεωρήσουμε ότι ένα σημείο ανήκει στο αντικείμενο αν το gray level f(x,y) του σημείου είναι μεγαλύτερο από ένα κατώφλι (threshold) T, ενώ αντίθετα να θεωρήσουμε ότι ένα σημείο ανήκει στο background αν το gray level f(x,y) του σημείου είναι μικρότερο από το κατώφλι T (βλ. σχήμα 2.4(α)).



(α) (β)

Σχήμα 2.4 Ιστογράμματα εικονών που μπορεί να χωριστούν από (α) ένα threshold και (β) περισσότερα (εδώ 2) thresholds.

Στην περίπτωση του σχήματος 2.4(β), το οποίο είναι το πιθανό ιστόγραμμα μίας εικόνας με δυό φωτεινά αντικείμενα σε ένα σκοτεινό background, βλέπουμε ότι τρεις είναι οι βασικές πειροχες του ιστογράμματος τις οποίες μπορούμε να διαχωρίσουμε χρησιμοποιώντας δυό κατώφλια T_1 και T_2, όπως φαίνεται στο σχήμα. Μπορούμε δηλαδή να θεωρήσουμε ότι ένα σημείο ανήκει στο πρώτο αντικείμενο αν για το gray level f(x,y) του σημείου ισχύει $T_1 < f(x,y) \leq T_2$. Επίσης ότι ανήκει στο δεύτερο αντικείμενο αν ισχύει $f(x,y) > T_2$ και τέλος ότι ανήκει στο background αν ισχύει $f(x,y) \leq T_1$.

Βασισμένο στα προηγούμενα παραδείγματα, το thresholding μπορεί να θεωρηθεί σαν μία λειτουργία η οποία περιλαμβάνει έλεγχο μίας συνάρτησης T του τύπου

$$T = [x, y, p(x,y), f(x,y)] \tag{2.7}$$

όπου f(x,y) είναι το gray level του σημείου (x,y), και p(x,y) δηλώνει κάποια τοπική ιδιότητα του σημείου (για παράδειγμα το μέσο gray level της γειτονίας του (x,y)). Οπότε μία "thresholded" εικόνα θα ορίζεται ως εξής:

$$g(x,y) = \begin{cases} 1 & \text{if } f(x,y) > T \\ 0 & \text{if } f(x,y) \leq T. \end{cases} \tag{2.8}$$

Όταν το T εξαρτάται μόνο από το f(x,y) τότε το threshold λέγεται συνολικό (global). Όταν το T εξαρτάται και από το p(x,y) τότε το threshold λέγεται τοπικό (local). Τέλος αν

το T εξαρτάται και από τις συντεταγμένες (x,y) τότε το threshold λέγεται δυναμικό (dynamic).

Από την παραπάνω ανάλυση, είναι φανερό ότι το πρόβλημα που έχουμε να αντιμετωπίσουμε όταν θα επιχειρήσουμε την εφαρμογή μίας "thresholding" τεχνικής, είναι η επιλογή του "threshold" ή των "thresholds" που θα χρησιμοποιήσουμε. Αν για παράδειγμα η εικόνα μας περιλαμβάνει ομαλές περιοχές, φανερά διαχωρίσιμες μεταξύ τους, τότε μπορούμε να επιλέξουμε τα thresholds με απλή παρατήρηση του ιστογράμματος της εικόνας. Αντίθετα, στην περίπτωση που οι περιοχές δεν είναι εντελώς ομαλές ή στην εικόνα μας υπάρχει θόρυβος, τότε χρειαζόμαστε επιπλεόν πληροφορία για την εικόνα, απώλεια της οποίας θα οδηγήσει σε μη ικανοποιητικά αποτελέσματα.

Συνήθως, thresholding τεχνικές χρησιμοποιούνται είτε σε συνδυασμό με άλλες τεχνικές για την επίτευξη καλύτερων αποτελεσμάτων, ή ως πρώτο βήμα για την απόκτηση ενός αρχικού segmentation, το οποίο στη συνέχεια θα βελτιώσουμε χρησιμοποιώντας κάποια άλλη μέθοδο.

2.4 "REGION ORIENTED" ΤΕΧΝΙΚΕΣ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ

2.4.1 Ανάπτυξη περιοχής (Region Growing) με ενοποίηση των pixels.

Η μέθοδος αυτή, αυτό το οποίο κάνει είναι να ομαδοποιεί γειτονικά pixels ή και μικρές περιοχές (και να δώσει έτσι μεγαλύτερες περιοχές), ξεκινώντας από κάποια αρχικά σημεία (seeds). Με βάση τα σημεία αυτά, η μέθοδος ξεκινά, επισυνάπτοντας στα αρχικά σημεία όλα τα γειτονικά τα οποία έχουν παρόμοιες ιδιότητες (χρώμα, υφή κλπ.). Για να γίνει κατανοητό αυτό ας θεωρήσουμε το σχήμα 2.5.

Στο σχήμα (2.5(α)) η αρχική εικόνα έχει αναπαρασταθεί με τη μορφή πίνακα και οι τιμές του πίνακα είναι τα gray levels της εικόνας. Στην πρώτη περίπτωση, ξεκινώντας με αρχικά σημεία (seeds) αυτά που φαίνονται υπογραμμισμένα (το 1 και το 7), το αποτέλεσμα είναι ότι παίρνουμε δύο περιοχές, την R_1 (τα σημεία που ανήκουν σ' αυτήν συμβολίζονται με α) και την R_2 (τα σημεία που ανήκουν σ' αυτήν συμβολίζονται με β), (βλ. σχ.2.5(β)).

0	0	5	5	7
1	1	5	8	7
0	**1**	6	**7**	7
2	0	7	6	6
0	1	5	6	5

(α)

α	α	β	β	β
α	α	β	β	β
α	α	β	β	β
α	α	β	β	β
α	α	β	β	β

(β)

α	α	α	α	α
α	α	α	α	α
α	α	α	α	α
α	α	α	α	α
α	α	α	α	α

(γ)

Σχήμα 2.5 (α) Αρχική εικόνα (σε μορφή πίνακα). (β) αποτέλεσμα της τεχνικής χρησιμοποιώντας σαν κριτήριο ομοιότητας μία διαφορά μικρότερη από 3 επίπεδα του γκρι. (γ) όπως και πριν μόνο που τώρα η διαφορά που χρησιμοποιούμε είναι 8 επίπεδα του γκρι.

Το κριτήριο με το οποίο ταξινομούμε το κάθε σημείο σε μία από τις δύο περιοχές, είναι ότι η απόλυτη τιμή της διαφοράς του gray level του σημείου από το gray level του αρχικού σημείου (seed) πρέπει να είναι μικρότερη από ένα κατώφλι T. Στην περίπτωση που το αποτέλεσμα είναι αυτό του σχήματος 2.5(β), το κατώφλι που χρησιμοποιήσαμε ήταν T=3 (επίπεδα του γκρι).

Στην περίπτωση του σχήματος 2.5(γ), το κατώφλι που χρησιμοποιήσαμε ήταν T=8. Εδώ, όπως φαίνεται από το σχήμα, το αποτέλεσμα ήταν να πάρουμε μόνο μία περιοχή. Βλέπουμε δηλαδή, πως επηρεάζει το τελικό αποτέλεσμα η επιλογή του κατωφλίου T,

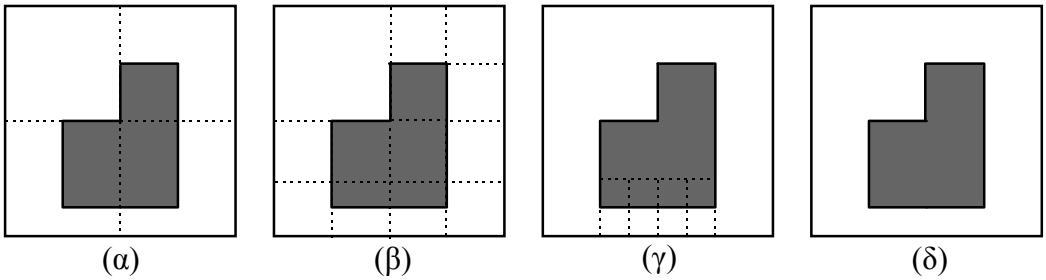
καθώς παρότι ξεκινήσαμε από δυό αρχικά σημεία το αποτέλεσμα ήταν να καταλήξουμε σε μία μόνο περιοχή (λόγω της υψηλής τιμής του Τ).

Είναι φανερό, ότι εφαρμόζοντας την παραπάνω τεχνική, **τρεις είναι οι πρωταρχικές δυσκολίες** που θα συναντήσουμε. Τρία δηλαδή είναι τα προβλήματα που θα αντιμετωπίσουμε, και τα οποία πρέπει να λύσουμε προκειμένου να καταλήξουμε σε ένα αποδεκτό αποτέλεσμα. Το πρώτη έχει να κάνει με την επιλογή των αρχικών σημείων (seeds) που θα χρησιμοποιήσουμε, καθώς θα πρέπει αυτά να είναι κατά το δυνατόν αντιπροσωπευτικά των περιοχών που θέλουμε να διαχωρίσουμε, πράγμα βέβαια το οποίο εξαρτάται από το είδος της εικόνας που έχουμε και συνεπώς από τη φύση του προβλήματος που αντιμετωπίζουμε. Εκ των προτέρων γνώση σ' αυτές τις περιπτώσεις είναι ιδιαίτερα χρήσιμη. Η δεύτερη δυσκολία έχει να κάνει με το κριτήριο το οποίο θα επιλέξουμε προκειμένου να συμπεριλάβουμε το κάθε σημείο σε κάποια από τις περιοχές. Η επιλογή του κριτηρίου στην περίπτωση αυτή, εξαρτάται εκτός από τη φύση του προβλήματος που αντιμετωπίζουμε, και από τον τύπο των δεδομένων εικόνας που έχουμε στη διάθεση μας. Τέλος η τρίτη δυσκολία έχει να κάνει με τον τρόπο με τον οποίο θα σαρώνουμε την εικόνα προκειμένου να ομαδοποιήσουμε τα γειτονικά pixels (όταν λέμε τρόπο εννοούμε την κατεύθυνση προς την οποία θα ξεκινάμε να ομαδοποιούμε, π.χ. από πάνω προς τα κάτω, από δεξιά προς αριστερά κλπ).

2.4.2 Χώρισμα και συνένωση περιοχών (Region Splitting and Merging).

Η τεχνική που παρουσιάσαμε στην προηγούμενη παράγραφο αυτό που κάνει είναι να "φτιάχνει" περιοχές, ξεκινώντας από ένα σύνολο αρχικών σημείων (seeds). Μία εναλλακτική μέθοδος είναι αρχικά να διασπάμε (σύμφωνα πάντα με κάποιο κριτήριο) την εικόνα σε ένα σύνολο από επιμέρους περιοχές, και στη συνέχεια να προσπαθούμε να ενώσουμε κάποιες από αυτές τις περιοχές σύμφωνα πάλι με κάποιο κριτήριο.

Όσον αφορά τη διάσπαση της εικόνας (splitting), μία τεχνική η οποία συχνά χρησιμοποιείται είναι αυτή της διάσπασης της εικόνας σε τεταρτημόρια (σε τέσσερις δηλαδή ίσες περιοχές). Για να γίνει κατανοητό αυτό ας παρατηρήσουμε το σχήμα 2.6.



Σχήμα 2.6 Ένα παράδειγμα της διαδικασίας split-and-merge.

Στην εικόνα 2.6(α) βλέπουμε ένα (σκοτεινό) μαύρο αντικείμενο σε ένα φωτεινό (άσπρο) background. Εφαρμόζοντας την τεχνική της διάσπασης βλέπουμε ότι η αρχική εικόνα διασπάται σε τέσσερα κομμάτια. Η διάσπαση συνεχίζεται για την κάθε υποεικόνα που προκύπτει και κατοπιν για τις υπόλοιπες κ.ο.κ, αν και μόνο αν υπάρχουν διαφορές του gray level σ΄ αυτήν. Πιο απλά, στο συγκεκριμένο παράδειγμα, μία δεδομένη υποεικόνα σε ένα ενδιάμεσο στάδιο θα υποστεί διάσπαση αν περιέχει και τα δύο χρώματα. Σε αντίθετη περίπτωση (αν δηλαδή είναι όλη άσπρη ή όλη μαύρη) δε θα υποστεί διάσπαση.

Τα παραπάνω αφορούν τη στρατηγική με την οποία διασπάμε την εικόνα μας κάθε φορά. Παράλληλα όμως με τη διάσπαση, πραγματοποιούμε και μία συνένωση γειτονικών περιοχών με το ίδιο χρώμα (άσπρο ή μαύρο). Δηλαδή, παράλληλα με τον έλεγχο που κάνουμε κάθε φορά για να δούμε αν μία περιοχή χρειάζεται “σπάσιμο”, ελέγχουμε αν γειτονικές περιοχές μπορούν να ενωθούν σε μία περιοχή με κριτήριο το gray level.

Με βάση τα παραπάνω, ξεκινώντας από την αρχική εικόνα, φθάνουμε στο αποτέλεσμα της εικόνας 2.6(δ) όπου βλέπουμε ότι το σκοτεινό αντικείμενο διαχωρίστηκε από το φωτεινό background. Αν και το συγκεκριμένο παράδειγμα είναι αρκετά απλό, ωστόσο επιδυκνύει τη στρατηγική των μεθόδων split and merge γενικότερα.

Η παραπάνω προσέγγιση είναι μία από τις πολλές που υπάρχουν και ανήκουν στην κατηγορία μεθόδων split and merge. Οι παράμετροι που διαφοροποιούν τις μεθόδους αυτής της κατηγορίας και **οι οποίες θα καθορίσουν την αποτελεσματικότητα της μεθόδου** είναι, πρώτον, ο τρόπος με τον οποίο θα γίνει το spliiting της εικόνας σε περιοχές, δεύτερον, το κριτήριο σύμφωνα με το οποίο θα γίνεται το spliiting και τρίτον, το κριτήριο σύμφωνα με το οποίο θα γίνεται το merging. Ανάλογα πάντα με την εφαρμογή, απαιτείται προσεκτική επιλογή των παραμέτρων που θα χρησιμοποιήσουμε,

ώστε το αποτέλεσμα να είναι το καλύτερο δυνατό. Όσον αφορά την πρώτη από τις παραμέτρους που αναφέραμε παραπάνω, δηλαδή τον τρόπο με τον οποίο θα γίνει το spliiting, αυτό που θα πρέπει να επιλέξουμε είναι ο αριθμός και το είδος (σχήμα) των περιοχών που θα παίρνουμε μετά από κάθε διάσπαση. Στο παράδειγμα που χρησιμοποιήσαμε παραπάνω, ο αριθμός των περιοχών που παίρναμε μετά από κάθε διάσπαση ήταν τέσσερις ενώ το σχήμα των περιοχών ήταν τετράγωνο.

Στο επόμενο κεφάλαιο θα παρουσιάσουμε μία τεχνική η οποία ανήκει στην κατηγορία των μεθόδων split and merge, και θα δούμε ποιά αποτελέσματα έχει η επιλογή ενός συγκεκριμένου κριτηρίου για splitting ενώ θα δούμε επίσης και πως επηρεάζει την απόδοση της μεθόδου η επιλογή του κριτηριού για το merging των περιοχών.

2.5 ΣΥΝΟΨΗ

Στο κεφάλαιο αυτό παρουσιάσαμε κάποιες βασικές τεχνικές για τμηματοποίηση εικόνας. Επισημαίνουμε ότι οι τεχνικές που παρουσιάσαμε ήταν απλά αντιπροσωπευτικές των βασικών κατηγοριών τμηματοποίησης. Ο αριθμός των μεθόδων που έχουν κατά καιρούς προταθεί είναι μεγάλος, αλλά όλες λίγο πολύ κάνουν χρήση των παραπάνω ιδεών, είτε συνδυάζοντας τις είτε διαφοροποιώντας τις ως προς κάποια χαρακτηριστικά. Συνήθως, κάθε προτεινόμενη μέθοδος εφαρμόζεται για μία ορισμένη κατηγορία εικόνων για την οποία και εγγυάται ικανοποιητικά αποτελέσματα. Δεν υπάρχουν τεχνικές τμηματοποίησης που να βρίσκουν εφαρμογή σ’ ολόκληρο το πεδίο των εικονών. Πάντα, ανάλογα με το είδος της εικόνας την οποία θέλουμε να επεξεργαστούμε και ανάλογα με το αποτέλεσμα στο οποίο θέλουμε να φτάσουμε σχεδιάζουμε την τεχνική μας έτσι ώστε να επιτυγχάνει (κατά το δυνατόν) τους παραπάνω στόχους.

Έχοντας όλα τα παραπάνω υπόψη μας, υπενθυμίζουμε ότι σκοπός του κεφαλαίου αυτού ήταν η εισαγωγή στην έννοια της τμηματοποίησης της εικόνας και η κατανόηση από τον αναγνώστη των βασικών συστατικών (αλλά και προβλημάτων) κάθε τεχνικής, ώστε να γίνει κατανοητή η παρουσίαση **δύο βασικών προσεγγίσεων** στην τμηματοποίηση εικόνας **(ανίχνευση ακμών και clustering)** που θα ακολουθήσουν στα επόμενα κεφάλαια και κατόπιν συνεκτιμώντας τα αποτελέσματα των δύο προσεγγίσεων να εξαχθούν κάποια χρήσιμα συμπεράσματα.

ΚΕΦΑΛΑΙΟ 3: ΕΝΑΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ ΑΝΙΧΝΕΥΣΗ ΑΚΜΩΝ

Στο κεφάλαιο αυτό θα παρουσιάσουμε ένα αλγόριθμο για ανίχνευση των ακμών σε μία εικόνα. Ο αλγόριθμος, που είναι δανεισμένος από paper **[12]**, δεν υλοποιήθηκε ακριβώς όπως παρουσιάστηκε στο paper, πρώτον γιατί η πολυπλοκότητα της προτεινόμενης υλοποίησης ήταν αρκετά μεγάλη ώστε να κάνει προβληματική τη χρησιμοποίηση του και δεύτερο για λόγους σύγκρισης του αρχικού αλγορίθμου με τη δικιά μας υλοποίηση. Άλλωστε, ο βασικός λόγος της υλοποίησης και παρουσίας του αλγορίθμου στο κεφάλαιο αυτό είναι η σύγκριση των αποτελεσμάτων που δίνει με τα αποτελέσματα κάποιων αλγορίθμων clustering που θα παρουσιάσουμε σε επόμενα κεφάλαια.

3.1 ΕΙΣΑΓΩΓΗ

Η απόδοση των κλασσικών μεθόδων segmentation βασισμένη την τεχνική Split-and-Merge επηρεάζεται σοβαρά από την "άκαμπτη" διαδικασία του Split-and-Merge η οποία δεν φαίνεται να λαμβάνει υπόψη της τη σημασιολογία της εικόνας (Image Semantics). Η τεχνική που παρουσιάζουμε εδώ προτείνει ενα επαρκή αλγόριθμο για τμηματοποίηση εικόνας ο οποίος στηρίζεται στην ελαχιστοποίηση μιας συνάρτησης σφάλματος η οποία ορίζεται στην επόμενη παράγραφο. Σκόπος είναι η ανίχνευση των τμημάτων (Segments) της εικόνας αλλά και των ακμών της (Edge Detection).

Ο αλγόριθμος(τουλάχιστον με τον τρόπο που περιγράφεται στο paper) φαίνεται να δίνει ικανοποιητικά αποτελέσματα αφού προσαρμόζεται στα σημασιολογικά χαρακτηριστικά της εικόνας και συνεπώς βελτιώνει την απόδοση των ήδη υπάρχοντων μεθόδων. Επίσης, με κατάλληλη υλοποίηση, ο χρόνος εκτέλεσης του αλγορίθμου μπορεί να γίνει αρκετά μικρός (όπως θα δούμε σε επόμενη παράγραφο) παρόλο που όπως θα φανεί και από την ανάλυση που θα ακολουθήσει ο αριθμός των υπολογισμών που απαιτούνται για να αλοκληρωθεί ο αλγόριθμος είναι πραγματικά τεράστιος. Ωστόσο εκμεταλλευόμενοι την υπάρχουσα υπολογιστική ισχύ και εφαρμόζοντας ειδικές τεχνικές

βελτιστοποίησης, μπορούμε να πετύχουμε αρκετά χαμηλούς χρόνους εκτέλεσης του προγράμματος.

Στις επόμενες παραγράφους θα παρουσιάσουμε αναλυτικά τον προτεινόμενο αλγόριθμο (χωρίς ωστόσο να καταφύγουμε σε εξαντλητικές λεπτομέρειες) και θα εστιάσουμε την προσοχή μας στο σημείο εκείνο στο οποίο διαφοροποιηθήκαμε από την προτεινόμενη υλοποίηση. Προκαταβολικά αναφέρουμε ότι η διαφοροποίηση μας από την υλοποίηση που προτείνεται στο paper είναι το κριτήριο σύμφωνα με το οποίο κάνουμε το merging των περιοχών, το οποίο θα δούμε τελικά πως επηρεάζει το τελικό αποτέλεσμα.

3.2 Ο ΑΛΓΟΡΙΘΜΟΣ

Ο αλγόριθμος που θα παρουσιάσουμε, ξεκινώντας από την αρχική εικόνα εκτελεί πρώτα το "σπάσιμο" (χώρισμα, splitting) της εικόνας σε μικρότερες περιοχές και στη συνέχεια εκτελεί τη συνένωση(merging) ορισμένων περιοχών. Τόσο το "splitting" όσο και το "merging" των περιοχών γίνεται σύμφωνα με κάποια κριτήρια τα οποία θα παρουσιάσουμε στις αμέσως επόμενες παραγράφους.

3.2.1 Η διαδικασία του splitting.

Μία εικόνα μπορεί να παρουσιαστεί σαν μία συνάρτηση δύο μεταβλητών g(x,y) που παίρνει ακέραιες τιμές και που ορίζεται σε ένα N_x x N_y πλέγμα *G*. Γενικά η ελαχίστων τετραγώνων σε K κομμάτια προσέγγιση της συνάρτησης g(x,y) απαιτεί την ελαχιστοποίηση του συνολικού μέσου σφάλματος:

$$\sum_{i=1}^{K} E(G_i) = \sum_{i=1}^{K} \sum_{(x,y)\in G_i} [g(x,y)-\mu(G_i)]^2$$
$$G_i \neq \varnothing,\ \cup_{1\le i\le K} G_i = G,\ G_i \cap_{i\neq j} G_j = \varnothing \qquad (3.1)$$

όπου G_i είναι ένα Segment της εικόνας και $\mu(G_i)= \sum_{G_i} g(x,y)/|G_i|$ είναι η μέση τιμή του intensity του Segment G_i και $E(G_i)$ είναι το τετραγωνικό λάθος στην προσέγγιση της g(x,y) στο Segment G_i από την σταθερά $\mu(G_i)$.

Αυτό βέβαια είναι ένα εξαιρετικά δύσκολο πρόβλημα λόγω του τεράστιας ελευθερίας που υπάρχει όσων αφορά το χωρισμό της εικόνας σε κομμάτια. Γι' αυτό επιλέγουμε το χωρισμό της εικόνας σε δύο κομμάτια κάθε φορά, δηλαδή διαλέγουμε

K=2. Όμως και πάλι ο αριθμός των δυνατών χωρισμών της εικόνας σε δύο κομμάτια είναι εξαιρετικά μεγάλος οπότε από όλους τους δυνατούς χωρισμούς, επιλέγουμε το σπάσιμο της εικόνας από οριζόντιες, κάθετες και διαγώνιες (45^0, 135^0) γραμμές. Ανάμεσα σε όλα αυτά τα "σπασίματα" της εικόνας σε κομμάτια, επιλέγουμε εκείνο που ελαχιστοποιεί το άθροισμα $E(G_1)+E(G_2)$ το οποίο ονομάζεται και *optimal four way cut.*

Με πιο απλά λόγια, αυτό το οποίο κάνουμε ξεκινώντας από την αρχική εικόνα, είναι να βρούμε ποιος από όλους τους δυνατούς τρόπους με τους οποίους μπορεί να χωριστεί η αρχική εικόνα **σε δύο κομμάτια**(χρησιμοποιώντας **μόνο** οριζοντιες, κάθετες ή διαγώνιες γραμμές με κλίση 45^0 ή 135^0) είναι ο καλύτερος δυνατός, από την άποψη βέβαια της ελαχιστοποίησης του αθροίσματος (3.1). Στην ουσία, με τον παραπάνω τρόπο προσπαθούμε να χωρίσουμε την εικόνα μας σε δύο κατά το δυνατόν ομοιόμορφα κομμάτια. Αυτό σημαίνει, πρακτικά, ελαχιστοποίηση του αθροίσματος (3.1) για K=2 τμήματα.

Αυτό το σπάσιμο λοιπόν χρησιμοποιούμε και χωρίζουμε την εικόνα σε δύο κατά το δυνατόν ομοιόμορφα κομμάτια (όπως άλλωστε μας εξασφαλίζει και το κριτήτιο των ελάχιστων τετραγώνων). Κατόπιν εφαρμόζουμε την ίδια ακριβώς διαδικασία για κάθε μία από τις επιμέρους εικόνες, βρίσκουμε δηλαδή το βέλτιστο σπάσιμο για την κάθε υποεικόνα και ομοίως προχωράμε και για τις υποεικόνες που παράγονται. Βέβαια να πούμε εδώ ότι για προχωρήσουμε στο σπάσιμο μιας υποεικόνας G_i σε κομμάτια, θα πρέπει το $E(G_i)$ της εικόνας να είναι μεγαλύτερο από κάποιο **κατώφλι ε** γεγονός το οποίο εξασφαλίζει την ανομοιομορφία της υποεικόνας και μας οδηγεί σε περαιτέρω σπάσιμο. Η διαδικασία αυτή συνεχίζεται έως ότου καμία από τις υποεικόνες που πάράγονται να μην χρειάζεται περαιτέρω σπάσιμο (πράγμα που σημαίνει είτε ότι $E(G_i)<\varepsilon$ είτε ότι το G_i έχει γίνει ένα pixel).

Η παραπάνω διαδικασία είναι αναδρομική και είναι γνωστή ως Recursive- Optimal-Four-way-Split (ROFS). Η αλγοριθμική της μορφή φαίνεται παρακάτω.

procedure ROFS(G)

begin

if E(G)<ε then return (G);

else begin

partition G into G_1 and G_2 by minimizing

$$\sum_{1\le i\le 2}\sum_{(x,y)\in G_i}[g(x,y)-\mu(G_i)]^2$$

over all possible 45-j degree cuts, j=0,1,2,3.

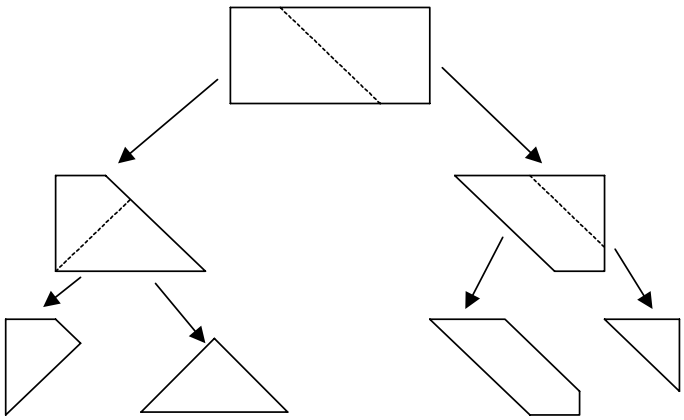
ROFS(G_1);

ROFS(G_2);

end;

end.

Είναι φανερό, ότι ο αριθμός των πολυγώνων από τα οποία θα αποτελείται τελικά η εικόνα θα εξαρτάται από το **κατώφλι ε** το οποίο θα επιλέξουμε, καθώς όσο μικρότερο είναι αυτό τόσο μεγαλύτερη ομοιομορφία θα απαιτείται και συνεπώς τόσο περισσότερο θα πρέπει να σπάμε την κάθε υποεικόνα προκειμένου να πετύχουμε την ομοιομορφία αυτή. Επίσης, όπως φαίνεται και από το σχήμα 3.1, τα σχήματα που προκύπτουν από το σπάσιμο της κάθε υποεικόνας σε νέες υποεικόνες, δεν είναι απλά τετράγωνα ή ορθογώνια όπως συνέβαινε σε άλλες περιπτώσεις splitting, αλλά μπορεί να είναι τρίγωνα, πεντάγωνα, εξάγωνα κ.λ π.



Σχήμα 3.1 Πολύγωνα τα οποία προκύπτουν από τη διαδικασία του splitting.

Αποδεικνύεται ότι τα πολύγωνα που προκύπτουν με τη διαδικασία που αναφέραμε, είναι κυρτά n-γωνα όπου $3\le n\le 8$ και επίσης οι γωνίες που σχηματίζουν οι ακμές αυτών

των πολυγώνων θα είναι $45j^0$ μοίρες όπου j=0,1,2,3. Λεπτομέρειες για την περιγραφή και τη διαχείριση των κανονικών πολυγώνων που προκύπτουν, υπάρχουν στην αναφορά **[13],** και λόγους πληρότητας τις παραθέτουμε και στο παράρτημα στο τέλος του κεφαλαίου.

Ακολουθούν τώρα κάποιες εικόνες που πήραμε εφραμόζοντας τον αλγόριθμο ROFS σε δύο αρχικές εικόνες. Η μία είναι μία εικόνα από μία ανθρώπινη κύστη (με διαστάσεις 112x84), (βλ. σχ. 3.2(α)) και η άλλη είναι η γνωστή Lena (βλ. σχ. 3.3(α)). Στο σχήμα 3.2(β) φαίνεται το αποτέλεσμα της εφαρμογής του αλγορίθμου ROFS στην αρχική εικόνα (κύστη) χρησιμοποιώντας τιμή 20000 για το **κατώφλι ε.** Ο αριθμός των πολυγώνων που πήραμε στην περίπτωση αυτή ήταν 243. Στο σχήμα 3.2(γ) φαίνεται το αποτέλεσμα χρησιμοποιώντας όμως τιμή 2000 για το **κατώφλι ε.** Στην περίπτωση αυτή, ο αριθμός των πολυγώνων που πήραμε ήταν αρκετά μεγαλύτερος (1007), όπως άλλωστε αναμενότανε, καθώς το **κατώφλι ε** ήταν αρκετά μικρότερο.

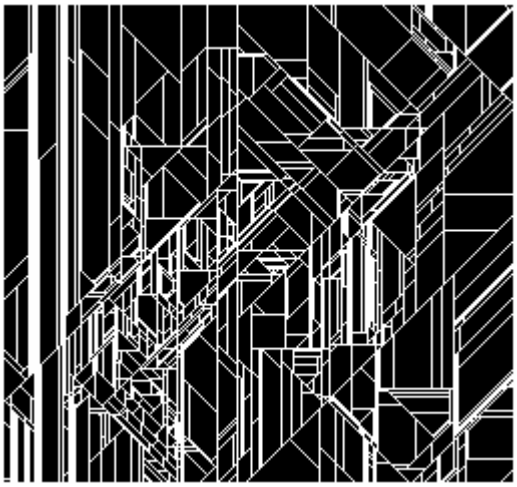
Στο σχήμα 3.3(β) φαίνεται το αποτέλεσμα της εφαρμογής του αλγορίθμου ROFS στην Lena (256x240), χρησιμοποιώντας τιμή 40000 για το **κατώφλι ε.** Ο αριθμός των πολυγώνων που προέκυψαν ήταν 930. Τέλος, στο σχήμα 3.3(γ) φαίνεται το αποτέλεσμα χρησιμοποιώντας τιμή 4000 για το **κατώφλι ε.** Ο αριθμός των πολυγώνων αυξήθηκε και έγινε 4521 στην περίπτωση αυτή.

Από τα σχήματα παρατηρούμε ότι μικραίνοντας την τιμή του **κατωφλιού ε** αυξάνεται ο αριθμός των πολυγώνων που μας δίνει η διαδικασία του splitting πράγμα που σημαίνει καλύτερη (πιο ακριβής) προσέγγιση των ακμών της εικόνας και ανίχνευση περισσοτέρων λεπτομερειών.

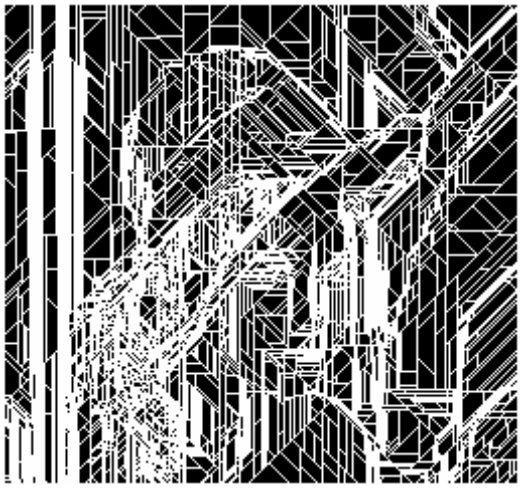
(α) (β) (γ)

Σχήμα 3.2 Αποτελέσματα (β),(γ) της εφαρμογής του αλγορίθμου ROFS στην αρχική εικόνα της κύστης (α).

(α)



(β)



(γ)

Σχήμα 3.3 Αποτελέσματα (β),(γ) της εφαρμογής του αλγορίθμου ROFS στην αρχική εικόνα Lena (α).

3.2.2 Η διαδικασία του merging.

Ο αλγόριθμος που παρουσιάσαμε παραπάνω αυτό που κάνει είναι να προσεγγίζει τις ακμές τις εικόνας με διαδοχικές οριζόντιες, κάθετες ή διαγώνιες γραμμές. Βέβαια όπως φαίνεται και από τις παραπάνω εικόνες το segmentaion στο οποίο οδηγεί η εφαρμογή του αλγορίθμου στις εικόνες μας δεν είναι έγκυρο. Για μεγαλύτερη ακρίβεια στον προσδιορισμό των ακμών και των τμημάτων πρέπει να εκτελέσουμε ένα merging κατά το οποίο θα ενώνουμε γειτονικές περιοχές σύμφωνα με κάποιο κριτήριο.

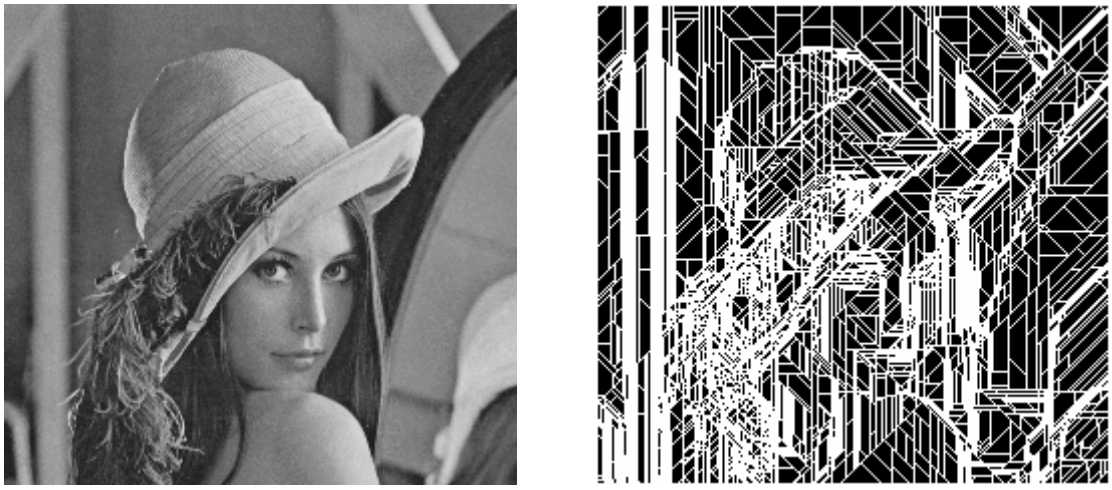
Το κριτήριο με το οποίο γίνεται το merging, σύμφωνα με την αναφορά **[12]**, είναι ότι το merging δύο γειτονικών περιοχών γίνεται αν και έφοσον μετά το merging η νέα περιοχή δεν υπερβαίνει κάποιο όριο λάθους (error threshold). Επίσης το merging που προτείνεται στην αναφορά δεν είναι εξαντλητικό, δηλαδή από όλα τα δυνατά mergings που μπορούν να γίνουν κάθε φορά, γίνεται μόνο αυτό που προκαλεί το μικρότερο λάθος. Επειδή κρίθηκε ότι η προτεινόμενη μέθοδος για το merging των περιοχών είχε μεγάλη πολυπλοκότητα, πράγμα που θα την έκανε προβληματική στη χρήση της, γι' αυτό υλοποιήσαμε το merging με άλλο τρόπο και σύμφωνα με διαφορετικό κριτήριο. Αυτό που κάναμε δηλαδή, ήταν να κάνουμε εξαντλητικό merging μεν, αλλά με κριτήριο την μέση τιμή των περιοχών που κάνουμε merge. Δηλαδή απλά ενώναμε δυό περιοχές κάθε φορά που η μέση τιμή της μιας απείχε από τη μέση τιμή της άλλης λιγότερο από μία τιμή -την οποία από δω και στο εξής θα ονομάζουμε **παράμετρο merging**- την οποία εμείς καθορίζουμε και η οποία περνάει σαν παράμετρος στο πρόγραμμα μας (επίσης σαν παράμετρο στο πρόγραμμα περνάμε και το κατώφλι ε).

Το κόστος που είχε βέβαια μια τέτοια υλοποίηση είναι φανερό στο αποτέλεσμα που παίρνουμε, το οποίο δεν είναι απόλυτα ικανοποιητικό, αφού εξαιτίας του κριτηρίου που χρησιμοποιούμε χάνουμε κάποιες ακμές (αφού γίνονται merge κάποιες περιοχές που δε θα έπρεπε να γίνονται).

Στα σχήματα 3.4 και 3.5 φαίνονται οι εικόνες που είδαμε στα σχήματα 3.2(γ) και 3.3(γ) αντίστοιχα, αλλά τώρα έχει γίνει και η φάση του merging. Και στις δύο περιπτώσεις η **παράμετρος merging** που χρησιμοποιήσαμε έχει τιμή 14 (πράγμα που σημαίνει ότι το merging δύο γειτονικών περιοχών θα γίνεται μόνο αν η διαφορά των μέσων τιμών (του gray level) των περιοχών είναι μικρότερη ή ίση του 14). Είναι φανερό ότι όσο μεγαλώνει η τιμή της **παραμέτρου merging** τόσο μεγαλώνει και ο αριθμός των περιοχών (πολυγώνων) που γίνονται merge, πράγμα που σημαίνει μείωση των ακμών που ανιχνεύει ο αλγόριθμος. Δοκιμάζοντας 3 ή 4 διαφορετικές τιμές της **παραμέτρου merging,** μπορούμε εύκολα να βρούμε την τιμή εκείνη για την οποία παίρνουμε το καλύτερο αποτέλεσμα. Σε επόμενη παράγραφο παραθέτουμε κάποιους ενδεικτικούς χρόνους εκτέλεσης του προγράμματος για την κάθε εικόνα αλλά και κάποια σχόλια γύρω από την απόδοση του συνολικού αλγορίθμου.

(α) (β) (γ)

Σχήμα 3.4 (α) Αρχική εικόνα. (β) Η διαδικασία του splitting. (γ) Η διαδικασία του merging.



(α) (β)

(γ)

Σχήμα 3.5 (α) Αρχική εικόνα. (β) Η διαδικασία του splitting. (γ) Η διαδικασία του merging.

3.3 ΠΕΙΡΑΜΑΤΙΚΑ ΑΠΟΤΕΛΕΣΜΑΤΑ ΚΑΙ ΑΠΟΔΟΣΗ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Παραθέτουμε τώρα τους χρόνους εκτέλεσης του προγράμματος μας για τις παραπάνω δύο εικόνες. Υπενθυμίζουμε ότι οι διαστάσεις της πρώτης εικόνας (κύστη) είναι 112x84 ενώ της δεύτερης (Lena) είναι 256x240. Όλοι οι χρόνοι που δίνονται είναι σε seconds. Επίσης, το μηχάνημα που χρησιμοποιήθηκε για να παρθούν οι μετρήσεις ήταν ένα SUN station μοντέλο Ultra 1 με λειτουργικό Unix.

	ROFS (1007πολύγωνα)	merging
Εικόνα 1 (κύστη)	0.3	0.25

	ROFS (4521 πολύγωνα)	merging
Εικόνα 2 (lena)	1.9	5

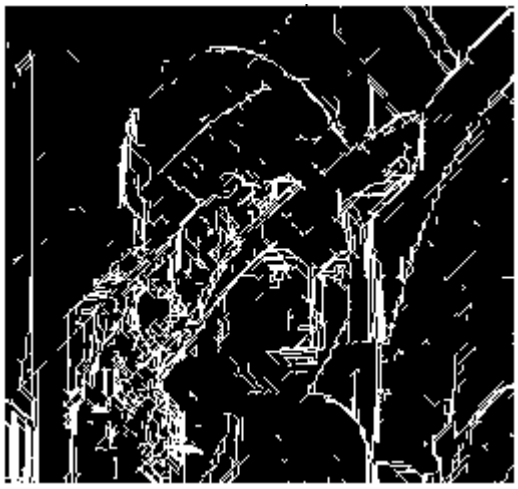
Ο χρόνος εκτέλεσης του αλγορίθμου ROFS είναι γραμμικά ανάλογος της κάθε διάστασης της εικόνας πράγμα το οποίο είναι επιθυμητό. Δηλαδή, με τον τρόπο που έγινε η υλοποίηση του αλγορίθμου, πετύχαμε ώστε ο χρόνος εκτέλεσης του αλγορίθμου ROFS να είναι ανάλογος του γινομένου των διαστάσεων της εικόνας (δηλαδή του εμβαδού της εικόνας). Αν δηλαδή N_x είναι το μήκος της εικόνας και N_y το πλάτος της τότε η πολυπλοκότητα του αλγορίθμου με τον τρόπο που υλοποιήθηκε είναι $O(N_xN_y)$. Αυτό επαληθεύται και από τα δεδομένα του παραπάνω πίνακα, καθώς παρατηρούμε ότι ο χρονος εκτέλεσης του αλγορίθμου ROFS για την δεύτερη εικόνα είναι περίπου 6.5 φορές μεγαλύτερος από αυτόν της πρώτης εικόνας, όσος είναι δηλαδή και ο λόγος των εμβαδών των δύο εικονών.

Όσον αφορά τώρα το στάδιο του merging, ο χρόνος εκτέλεσης του είναι ανάλογος με το τετράγωνο του αριθμού των πολυγώνων που μας έδωσε το στάδιο του αλγορίθμου ROFS. Αν δηλαδή N είναι ο αριθμός των πολυγώνων που προέκυψε από την εκτέλεση του αλγορίθμου ROFS, τότε η πολυπλοκότητα του merging είναι O(N). Αυτό επαληθεύεται και πάλι από τα δεδομένα του πίνακα, όπου παρατηρούμε ότι ο χρόνος του merging για την δέυτερη εικόνα είναι περίπου 20 φορές μεγαλύτερος από αυτόν της πρώτης εικόνας, όσο άλλωστε είναι και το τετράγωνο του λόγου των αριθμών των πολυγώνων για την κάθε περίπτωση ($(4521/1007)^2 \approx 20$). Και βέβαια, ο αριθμός των

πολυγώνων που θα πάρουμε (για ένα δεδομένο **κατώφλι ε**) εξαρτάται εκτός από τις διαστάσεις της εικόνας και από το περιεχόμενο της.

Τα συμπεράσματα που βγαίνουν από τους χρόνους εκτέλεσης του αλγορίθμου αλλά και από την παρατήρηση των αποτελεσμάτων που δίνει, είναι πρώτον ότι η ταχύτητα εκτέλεσης του αλγορίθμου είναι αρκετά ικανοποιητική, αν αναλογιστούμε τουλάχιστο ότι ο αριθμός των υπολογισμών που πρέπει να γίνουν είναι πραγματικά τεράστιος, και δεύτερον ότι τα αποτελέσματα που δίνει δεν είναι αρκετά ικανοποιητικά, σε σχέση τουλάχιστον με αυτά που δίνει ο αλγόριθμος υλοποιούμενος με τις προδιαγραφές της αναφοράς. Βλέπουμε δηλαδή πόσο καθοριστικό είναι το κριτήριο με το οποίο κάνουμε το merging των περιοχών, καθώς αυτό είναι που εξασφαλίζει την ανίχνευση όλων των ακμών της εικόνας αλλά και τη συνέχεια αυτών, πράγμα το οποίο δε συμβαίνει στην περίπτωση μας καθώς όπως μπορούμε να δούμε οι ακμές της εικόνας που ανιχνεύονται δεν είναι συνεχείς αλλά σε κάποια σημεία "σπάνε".

Στις προηγούμενες παραγράφους είδαμε κάποια αποτελέσματα από την εφαρμογή του αλγορίθμου σε εικόνες χωρίς την παρουσία θορύβου. Εμείς μελετήσαμε την απόδοση του αλγορίθμου και παρουσία θορύβου και παραθέτουμε κάποια αποτελέσματα. Στο σχήμα 3.6 βλέπουμε το αποτέλεσμα της εφαρμογής του αλγορίθμου στην Lena στην οποία όμως προσθέσαμε White Gaussian Noise με τυπική απόκλιση σ=8. Το κατώφλι ε που χρησιμοποιήσαμε είχε τιμή 10000 ενώ για την **παράμετρο merging** χρησιμοποιήσαμε τιμή 18. Το αποτέλεσμα μπορεί να χαρακτηριστεί ικανοποιητικό μόνο αν συγκριθεί με αυτό του σχήματος 3.5 καθώς βλέπουμε ότι οι βασικές ακμές της εικόνας έχουν ανιχνευθεί σχεδόν όπως ανιχνεύθηκαν και στην περίπτωση που δεν είχαμε θόρυβο. Ο αλγόριθμος αυτός δηλαδή, παρουσιάζει "καλή σχετικά" απόδοση παρουσία θορύβου πράγμα το οποίο οφείλεται τόσο στο κριτήριο με το οποίο κάνουμε το splitting της εικόνας όσο και στο κριτήριο με το οποίο κάνουμε το merging των περιοχών. Ωστόσο αυτό δε σημαίνει ότι η απόδοση του αλγορίθμου παραμένει καλή ακόμα και αν αυξηθεί πολύ η ποσότητα του θορύβου. Σε περιπτώσεις που η τυπική απόκλιση του θορύβου που προσθέτουμε στην εικόνα είναι 16 ή και μεγαλύτερη, ο αλγόριθμος αδυνατεί να δώσει ικανοποιητικά αποτελέσματα καθώς ο θόρυβος αλλοιώνει τις πραγματικές ακμές της εικόνας ενώ παράλληλα ενισχύει τις αντιθέσεις στα υπόλοιπα σημεία της εικόνας, με αποτέλεσμα αρκετές ακμές που ανιχνεύονται να είναι διακεκομμένες.

Σχήμα 3.6 (α) Αρχική εικόνα με white gaussian noise (σ=8). (β) Αποτέλεσμα της εφαρμογής του αλγορίθμου ανίχνευσης ακμών.

3.4 ΣΥΝΟΨΗ

Στις προηγούμενες παραγράφους παρουσιάσαμε ένα αλγόριθμο για ανίχνευση των ακμών σε μία εικόνα. Όπως αναφέραμε και στην αρχή, το πλεονέκτημα του αλγορίθμου αυτού έναντι των κλασσικών split-and-merge, είναι ότι προσαρμόζεται καλύτερα στα σημασιολογικά χαρακτηριστικα της εικόνας και προσεγγίζει καλύτερα τις ακμές της εικόνας χρησιμοποιώντας και διαγώνιες γραμμές. Η χρήση των διαγώνιων γραμμών το μόνο κόστος που έχει είναι στον υπολογιστικό χρόνο ενώ από την άλλη συμβάλλει και στη μείωση του αριθμού των παραγόμενων πολυγώνων, αλλά και των πολυγώνων που γίνονται merge.

Όσων αφορά το **κατώφλι ε** και την **παράμετρο merging**, οι τιμές που θα δώσουμε εξαρτώνται (όπως ήδη έχουμε αναφέρει) από τις διαστάσεις τις εικόνας, από το περιεχόμενο της και από το αποτέλεσμα στο οποίο θελουμε να φτάσουμε. Για το κατώφλι ε ενδεικτικές τιμές ξεκινάνε από 1000 και φτάνουν έως 40000,50000 ή και παραπάνω ανάλογα πάντα με την εικόνα και το τι ανάλυση (σε πολύγωνα) ζητάμε. Για την παράμετρο merging ενδεικτικές τιμές ξεκινάνε από 5 ή 6 και φτάνουν σε 20 ή και παραπάνω ανάλογα με το ποσοστό του merging που θέλουμε να πετύχουμε. Με λίγες δοκιμές μπορούμε εύκολα να βρούμε τον συνδυασμό **κατωφλιού ε** και και **παραμέτρου merging** που δίνει το καλύτερο αποτέλεσμα.

Υπενθυμίζουμε για άλλη μία φορά ότι η μοναδική ίσως ουσιαστική απόκλιση από τον αλγόριθμο του paper, ήταν στο κριτήριο σύμφωνα με το οποίο έγινε το merging (το οποίο είδαμε τελικά πόσο καθοριστικό ήταν) και στο οποίο οφείλονται τα όποια μη αποδεκτά κομμάτια στο τελικό segmentation. Όπως όμως αναφέραμε και στην αρχή του κεφαλαίου, ο λόγος για τον οποίο υλοποιήσαμε το συγκεκριμένο αλγόριθμο, είναι γιατί θα μας χρειαστεί σε επόμενο κεφάλαιο, όταν θα συγκρίνουμε τα αποτελέσματα του με αυτά που δίνουν κάποιοι αλγόριθμοι για clustering (βλ. Κεφάλαια 4,6) της εικόνας. Θα συγκρίνουμε δηλαδή την απόδοση των αλγορίθμων ανίχνευσης ακμών με αυτή των αλγορίθμων clustering, με κριτήριο τόσο την "χρησιμότητα" της τελικής εικόνας όσο και την αποτελεσματικότητα τους σε περιβάλλον θορύβου.

ΠΑΡΑΡΤΗΜΑ

Στην παράγραφο αυτή θα αναφέρουμε λεπτομέρειες για τη γεωμετρία που χρησιμοποιούμε προκειμένου να διαχειριστούμε τα πολύγωνα που προκύπτουν από το χώρισμα της εικόνας με οριζόντιες, κάθετες και διαγώνιες γραμμές.

Όπως αναφέραμε σε προηγούμενη παράγραφο, τα πολύγωνα που προκύπτουν με τη διαδικασία του splitting **[14]** είναι κανονικά κυρτά n-γωνα όπου $3 \le n \le 8$. Είναι φανερό ότι το πιο πολύπλοκο πολύγωνο που μπορεί να προκύψει είναι ένα κυρτό οκτάγωνο. Έστω $V=\{(\upsilon_{i,x}, \upsilon_{i,y})\}$, $1 \le i \le 3$, $3 \le n \le 8$, το σύνολο των κορυφών του κανονικού πολυγώνου. Από τις κορυφές αυτές ορίζουμε τώρα τις top-left (TL), bottom-right (BR), left-bottom (LB) και right-top (RT) κορυφές ως εξής:

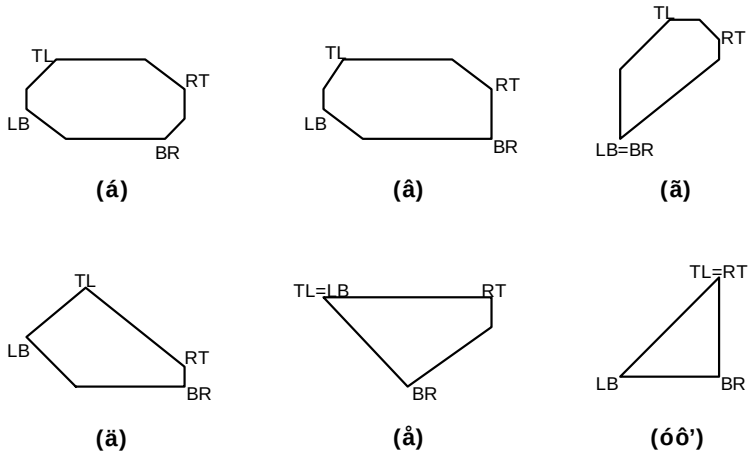
$$TL_y = \max_{\mathbf{v} \in V}\{\tilde{o}_{i,y}\}, \qquad TL_x = \min_{\tilde{o}_{i,y}=TL_y}\{\tilde{o}_{i,x}\},$$

$$BR_y = \min_{\mathbf{v} \in V}\{\tilde{o}_{i,y}\}, \qquad BR_x = \max_{\tilde{o}_{i,y}=BR_y}\{\tilde{o}_{i,x}\},$$

$$LB_x = \min_{\mathbf{v} \in V}\{\tilde{o}_{i,x}\}, \qquad LB_y = \min_{\tilde{o}_{i,x}=LB_x}\{\tilde{o}_{i,y}\},$$

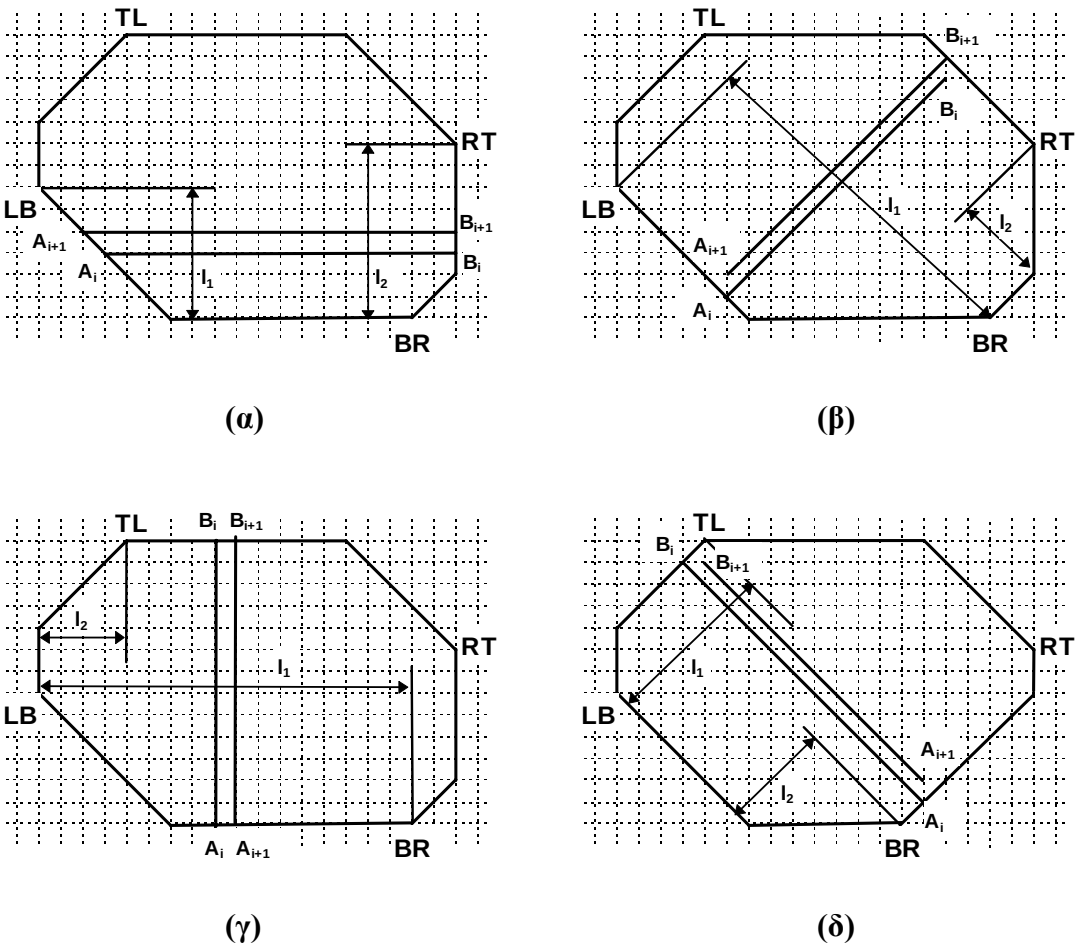
$$RT_x = \max_{\mathbf{v} \in V}\{\tilde{o}_{i,x}\}, \qquad RT_y = \max_{\tilde{o}_{i,x}=RT_x}\{\tilde{o}_{i,y}\},$$

Οι κορυφές αυτές ονομάζονται και κορυφές "κλειδιά" του πολυγώνου και φαίνονται στο σχήμα 3.7. Από τη στιγμή που ο αριθμός των κορυφών των πολυγώνων που προκύπτουν κυμαίνεται από 3 έως 8 είναι πιθανό κάποιες κορυφές "κλειδιά" να ταυτίζονται, όπως άλλωστε φαίνεται και από το σχήμα (περιπτώσεις γ,ε,στ').



Σχήμα 3.7 Κανονικά πολύγωνα και οι κορυφές "κλειδιά" τους.

Αποδεικνύεται ότι τα πολύγωνα που προκύπτουν με τη διαδικασία του splitting καθορίζονται μοναδικά από τις κορυφές "κλειδιά" τους. Δηλαδή γνωρίζοντας μόνο τις τέσσερις κορυφές "κλειδιά" ενός κανονικού πολυγώνου μπορούμε να βρούμε και τις υπόλοιπες κορυφές του (αν υπάρχουν, αν δηλαδή έχει παραπάνω από τέσσερις κορυφές το πολύγωνο). Οπότε αυτό που πρέπει να κάνουμε είναι να υπολογίζουμε τις κορυφές "κλειδιά" των πολυγώνων που προκύπτουν μετά από κάθε χώρισμα. Στο σχήμα 3.8 φαίνονται η οριζόντια, η κατακόρυφη και η διαγώνια (45^0,135^0) γραμμή σάρωσης της εικόνας (πολυγώνου). Επίσης φαίνονται και κάποιες μεταβλητές (l_1 , l_2 , A_i , A_{i+1} , B_i , B_{i+1}) που θα εξηγήσουμε παρακάτω τι συμβολίζουν. Ο προσδιορισμός των σημειών τομής της γραμμής σάρωσης (καθώς αυτή κινείται μέσα στο πολύγωνο) με τις ακμές του πολυγώνου περιγράφεται στο **[15]**.



Σχήμα 3.8 Χώρισμα κανονικού πολυγώνου από οριζόντιες (α), κάθετες (γ) και διαγώνιες γραμμές (β,δ).

Ας δούμε τώρα πως υπολογίζουμε τις κορυφές "κλειδιά" των πολυγώνων G_1 και G_2 που προκύπτουν με το χώρισμα του αρχικού πολυγώνου G από τη γραμμή σάρωσης A_iB_i. Στα σχήματα, με $A_{i+1}B_{i+1}$ συμβολίζουμε τη θέση της γραμμής σάρωσης την αμέσως επόμενη χρονική στιγμή.

Ξεκινώντας από την περίπτωση της **οριζόντιας σάρωσης (τομής)** (βλ. σχ. 3.8(α)) ορίζουμε $l_1 = LB_y - BR_y$ και $l_2 = RT_y - BR_y$. Ακολουθεί τώρα ο αλγόριθμος σύμφωνα με τον οποίο υπολογίζουμε τις κορυφές "κλειδιά" των πολυγώνων που προκύπτουν από το χώρισμα του αρχικού χρησιμοποιώντας οριζόντια γραμμή χωρισμού. Χρησιμοποιούμε "δείκτες" 1 και 2 για να αναφερθούμε στις κορυφές "κλειδιά" των πολυγώνων που παράγονται.

$BR_1 = BR$; $TL_1 = A_i$; $TL_2 = TL$; $BR_2 = B_{i+1}$;

IF $i<l_1$ THEN BEGIN $LB_1 = A_i$; $LB_2 = LB$ END

ELSE BEGIN $LB_1 = LB$; $LB_2 = A_{i+1}$ END;

IF $i<l_2$ THEN BEGIN $RT_1 = B_i$; $RT_2 = RT$ END

ELSE BEGIN $RT_1 = RT$; $RT_2 = B_{i+1}$ END.

Ο αλγόριθμος για τον υπολογισμό των κορυφών "κλειδιά" στην περίπτωση της **κάθετης σάρωσης (τομής)** (βλ. σχ. 3.8(γ)) είναι συμμετρικός με αυτόν της οριζόντιας περίπτωσης. Απλά αντικαθιστούμε το σημείο αναφοράς BR με το LB, οπότε τα l_1 και l_2 ορίζονται τώρα $l_1 = BR_x - LB_x$ και $l_2 = TL_x - LB_x$.

Στην περίπτωση τώρα της **διαγώνιας σάρωσης (τομής)** (βλ. σχ. 3.8(β)), και συγεκριμένα με γωνία $\mathbf{45^0}$, η λογική με την οποία καθορίζουμε τις κορυφές "κλειδιά" των πολυγώνων G_1 και G_2 που προκύπτουν είναι ανάλογη με πριν. Απλά παρατηρούμε το σχήμα και εύκολα υπολογίζουμε τα σημεία αυτά. Η μόνη διαφορά όμως είναι ότι η γεωμετρία που χρησιμοποιούμε αυτή τη φορά είναι λίγο πιο πολύπλοκη καθώς τα ακραία σημεία των τομών A_iB_i και $A_{i+1}B_{i+1}$ δεν πέφτουν απαραίτητα πάνω στα όρια του πολυγώνου G. Ο αριθμός των διαγώνιων θέσεων ξεκινώντας από το BR και φθάνοντας ως το TL είναι $\|TL,BR\|_1$ (η L_1 απόσταση των pixels BR,TL). Ορίζουμε τώρα $l_1 =$

$\|LB,BR\|_1$ να είναι ο αριθμός των βημάτων (θέσεων) για να φτάσουμε με διαγώνιες γραμμές από το BR στο LB. Επίσης, ορίζουμε $l_2 = \|RT,BR\|_1 - 2(RT_x - BR_x)$ να είναι ο αριθμός των βημάτων για να φτάσουμε με διαγώνιες γραμμές από το BR στο RT αντίστοιχα.

Βασισμένοι τώρα στους παραπάνω ορισμούς των l_1 και l_2 μπορούμε να δώσουμε τον αλγόριθμο για τον υπολογισμό των κορυφών "κλειδιά" των πολυγώνων που προκύπτουν για μία δεδομένη διαγώνια (45^0) τομή A_iB_i. Ο αλγόριθμος αυτός φαίνεται παρακάτω.

$BR_1 = BR$; $TL_1=B_i$; $TL_2 = TL$; $BR_2 = A_{i+1}$;

IF $i<l_1$ THEN BEGIN

$LB_2 = LB$;

IF A_i is on the boundary of G THEN $LB_1 = A_i$

ELSE BEGIN $LB_{1,x} = A_{i,x}$; $LB_{1,y} = A_{i,y} - 1$ END

END

ELSE BEGIN $LB_1 = LB$; $LB_2 = A_{i+1}$ END;

IF $i<l_2$ THEN BEGIN $RT_1 = B_i$; $RT_2 = RT$ END

ELSE BEGIN

$RT_1 = RT$;

IF B_{i+1} is on the boundary of G THEN $RT_2 = B_{i+1}$

ELSE BEGIN $RT_{2,x} = B_{i+1,x}$; $RT_{2,y} = B_{i+1,y} + 1$ END

END.

Ο αλγόριθμος για τον υπολογισμό των κορυφών "κλειδιά" στην περίπτωση της **διαγώνιας σάρωσης (τομής) με γωνία 135^0** (βλ. σχ. 3.8(δ)) είναι συμμετρικός με αυτόν της διαγώνιας περίπτωσης με γωνία 45^0 που παρουσιάσαμε παραπάνω, με μόνη διαφορά ότι το σημείο LB θα είναι τώρα το σημείο αναφοράς (αντί του BR) και τα l_1 και l_2 θα ορίζονται τώρα ως $l_1 = \|TL,LB\|_1$ και $l_2 = \|BR,LB\|_1 - 2(LB_y - BR_y)$.

ΚΕΦΑΛΑΙΟ 4: ΕΝΑΣ ΠΡΟΣΑΡΜΟΖΟΜΕΝΟΣ ΑΛΓΟΡΙΘΜΟΣ ΓΙΑ CLUSTERING ΕΙΚΟΝΑΣ

Στο κεφάλαιο αυτό θα παρουσιάσουμε μία τεχνική για Image Segmentation και πιο συγκεκριμένα για Image Clustering. Η συγκεκριμένη τεχνική **[16]** δίνει αρκετά καλά αποτελέσματα για μία μεγάλη γκάμα εικονών. Αρκετές δοκιμές έγιναν με διάφορες τιμές των παραμέτρων της εικόνας, ενώ έγιναν και κάποιες τροποποιήσεις στον αλγόριθμο τις οποίες και θα παρουσιάσουμε αναλυτικότερα παρακάτω. Σκοπός ήταν πάντα η βελτίωση του τελικού αποτελέσματος, πράγμα που σε μερικές περιπτώσεις επιτεύχθηκε. Το κεφάλαιο αυτό δεν είναι ανεξάρτητο από τα υπόλοιπα. Είναι μεν αυτοδύναμο αλλά εξυπηρετεί έναν σκοπό. Ο ρόλος του είναι η παρουσίαση ενός συγκεκριμένου αλγορίθμου τον οποίο θα χρησιμοποιήσουμε στο Κεφάλαιο 6 που, σε συνδυασμό με τη θεωρία από το πεδίο των wavelets (Κεφάλαιο 5) θα παρουσιάσουμε μία εναλλακτική μέθοδο για Image Segmentation (Clustering) και θα μελετήσουμε την απόδοσή της.

4.1 ΕΙΣΑΓΩΓΗ

Θα παρουσιάσουμε μία τεχνική για Image Segmentation grayscale εικόνων (τυπικά 256 επιπέδων του γκρι) σε περιοχές με σταθερό η ελαφρά μεταβαλλόμενο intensity. Η "τμηματοποιημένη" εικόνα θα αποτελείται από λίγα επίπεδα του γκρι (συνήθως 2-4) κάθε ένα από τα οποία θα αναπαριστά και μία περιοχή. Θα αποτελεί στην ουσία ένα σκίτσο ή καρικατούρα της αρχικής εικόνας το οποίο θα διατηρεί τα σημαντικότερα χαρακτηριστικά της ενώ δε θα διατηρεί τις όποιες λεπτομέρειες τις εικόνας (το βαθμό στον οποίο θα διατηρούνται ή δε θα διατηρούνται οι λεπτομέρειες θα είμαστε σε θέση να τον καθορίζουμε, μεταβάλλοντας την τιμή μιας παραμέτρου, όπως θα δούμε παρακάτω). Συνεπώς θα μπορεί να χρησιμοποιηθεί σαν ένα πρώτο στάδιο ενός συστήματος για αναγνώριση εικονών (Image Recognition System). Τα πλεονεκτήματα της "καρικατούρας" που παίρνουμε είναι αφενός μεν εύκολη εκτύπωση σε οποιδήποτε μέσο με πολύ λίγα επίπεδα και αφετέρου κωδικοποίηση και συμπίεση της εικόνας σε μεγάλο βαθμό αφού το μόνο που έχουμε να κωδικοποιήσουμε είναι οι μεταβολές ανάμεσα σε πολύ λίγα επίπεδα του γκρι.

Ο αλγόριθμος που παρουσιάζουμε κατανέμει τα pixels της εικόνας σε clusters, βασιζόμενος τόσο στο επίπεδο του γκρι που έχουν, όσο και στη σχετική τους θέση. Η υπόθεση είναι ότι οι εικόνες περιέχουν αντικείμενα με ομαλές (λείες) επιφάνεις και όχι υφή (texture).

Ένας γνωστος αλγόριθμος για clustering είναι ο K-means αλγόριθμος(βλ. Παράρτημα). Όμως όταν εφαρμόζεται στην τμηματοποίηση (segmentation) της εικόνας, παρουσιάζει δύο προβλήματα: πρώτον δεν χρησιμοποιεί καθόλου χωρικούς περιορισμούς και δεύτερον υποθέτει ότι κάθε cluster χαρακτηρίζεται από σταθερό intensity. Η τεχνική που παρουσιάζουμε σ' αυτό το κεφάλαιο μπορεί να θεωρηθεί σαν μία γενίκευση του K-means αλγορίθμου ως προς δύο χαρακτηριστικά: είναι προσαρμοζόμενη και λαμβάνει υπόψη της χωρικούς περιορισμούς.

Όπως και ο K-means αλγόριθμος έτσι και αυτός που παρουσιάζουμε εδώ είναι ένας επαναληπτικός (iterative) αλγόριθμος. Κάθε περιοχή χαρακτηρίζεται από μία αργά μεταβαλλόμενη συνάρτηση έντασης (intensity). Δεδομένης αυτής της συνάρτησης ορίζουμε την a posteriori συνάρτηση πυκνότητας πιθανότητας (probability density function ή pdf για συντομία) για την κατανομή των περιοχών με δεδομένη την αρχική παρατηρούμενη εικόνα. Η pdf έχει δύο παράγοντες, ο ένας περιορίζει το intensity της περιοχής να είναι κοντά στα δεδομένα και ο άλλος εξασφαλίζει χωρική συνέχεια.

Ο αλγόριθμος εναλλάσεται ανάμεσα στη μεγιστοποίηση της a posteriori pdf και στον υπολογισμό των συναρτήσεων έντασης (intensity functions). Αρχικά οι συναρτήσεις έντασης είναι σταθερές σε κάθε περιοχή και ίσες με τα κέντρα των clusters που έχουν προκύψει με τον K-means αλγόριθμο. Καθώς ο αλγόριθμος προχωράει οι εντάσεις (intensities) ενημερώνονται, παίρνοντας το μέσο όρο πάνω σε ένα κινούμενο παράθυρο του οποίου το μέγεθος σταδιακά μικραίνει. Έτσι ο αλγόριθμος ξεκινά με κάποιους συνολικούς (global) υπολογισμούς και προοδευτικά προσαρμόζεται στα τοπικά χαρακτηριστικά της κάθε περιοχής.

Ο αλγόριθμος δίνει πολύ καλά αποτελέσματα, σαφώς ανώτερα από αυτά του K-means, αλλά και από άλλες τεχνικές που έχουν κατά καιρούς προταθεί. Εφαρμόστηκε με επιτυχία σε ένα μεγάλο αριθμό εικονών, διαφόρων ειδών και προελεύσεων. Τα αποτελέσματα ήταν σε κάθε περίπτωση αρκετά ικανοποιητικά.

Το μοντέλο για την κλάση των εικονών που θεωρούμε παρουσιάζεται στην παράγραφο 4.2. Ο αλγόριθμος παρουσιάζεται στην παράγραφο 4.3. Στην παράγραφο 4.4 παρουσιάζουμε τα αποτελέσματα της εφαρμογής του αλγορίθμου πάνω σε διάφορες εικόνες. Στην παράγραφο 4.5 μελετάμε μία ιεραρχική υλοποίηση του αλγορίθμου και την συγκρίνουμε με την αντίστοιχη απλή. Στην παράγραφο 4.6 παρουσιάζουμε τα συμπεράσματα και κάποια σχόλια για τον αλγόριθμο που υλοποιήσαμε. Στο τέλος του κεφαλαίου ακολουθεί ένα παράρτημα, στο οποίο παρουσιάζουμε την γενική λειτουργία του αλγορίθμου K-means και επεξηγούμε πως ακριβώς εφαρμόστηκε ο αλγόριθμος αυτός στην περίπτωση μας (δηλαδή τι τροποποιήσεις έγιναν στον αλγόριθμο προκειμένου αυτός να μπορέσει να εφαρμοστεί σε gray-level εικόνες).

4.2 ΤΟ ΜΟΝΤΕΛΟ

Αυτό που κάνουμε είναι να μοντελοποιήσουμε τις gray-scale εικόνες μας σαν μία συλλογή από περιοχές με ομοιόμορφη ή ελαφρά μεταβαλλόμενη ένταση (intensity). Οι μοναδικές απότομες μεταβολές στο gray level συμβαίνουν στα σύνορα των περιοχών. Έστω y η θεωρούμενη gray-scale εικόνα. Η ένταση ενός pixel στη θέση s θα συμβολίζεται με y_s και θα παίρνει τιμές από 0 μέχρι 255. Μια τμηματοποίηση (segmentation) της εικόνας σε περιοχές θα συμβολίζεται με x όπου $x_s = i$ θα σημαίνει ότι το pixel στη θέση s ανήκει στην περιοχή i. Ο αριθμός των διαφορετικών περιοχών είναι K.

Σ' αυτήν την παράγραφο αναπτύσσουμε το μοντέλο για την a posteriori probability density function $p(x|y)$. Από το θεώρημα του Bayes έχουμε

$$p(x|y) \propto p(y|x)p(x) \tag{4.1}$$

όπου $p(x)$ είναι η a priori πυκνότητα (density) της διαδικασίας των περιοχών (region process) και $p(y|x)$ είναι η πυκνότητα υπό συνθήκη της παρατηρούμενης εικόνας δεδομένης της κατανομής των περιοχών.

Η διαδικασία των περιοχών μοντελοποιείται με ένα τυχαίο πεδίο Markov (Markov random field MRF). Δηλαδή αν N_s είναι μία γειτονιά ενός pixel στη θέση s τότε:

$$p(x_s|x_q, \text{all } q \neq s) = p(x_s|x_q, q \in N_s). \tag{4.2}$$

Με απλά λόγια, αυτό που εκφράζει η παραπάνω εξίσωση είναι ότι η περιοχή (cluster) στην οποία θα ανήκει το pixel στη θέση s, θα εξαρτάται σε κάποιο βαθμό μόνο από τα pixels της γειτονίας N_s του pixel s και όχι από τα υπόλοιπα pixels της εικόνας.

Θεωρούμε εικόνες που ορίζονται πάνω στο Καρτεσιανό πλέγμα και τη γειτονιά N_s ενός pixel s να αποτελείται από τα 8 κοντινότερα pixels. Σύμφωνα με γνωστό θεώρημα η πυκνότητα p(x) θα δίνεται από την πυκνότητα του Gibbs και θα έχει την ακόλουθη μορφή

$$p(x_s)=\frac{1}{Z}\exp\left\{-\sum_{C} V_c(x_s)\right\}. \tag{4.3}$$

Εδώ Z είναι μία σταθερά κανονικοποίησης(normalizing constant) και το άθροισμα γίνεται πάνω σε όλες τις "κλίκες" C. Μία "κλίκα" είναι ένα σύνολο από σημεία που είναι γειτονικά το ένα στο άλλο. Τα δυναμικά των κλικών (V_c) εξαρτώνται μόνο από τα pixels που ανήκουν στη κλίκα C.

Το μοντέλο μας υποθέτει ότι τα μόνα μη μηδενικά δυναμικά είναι αυτά που αντιστοιχούν στις κλίκες ενός ή δύο σημείων όπως φαίνεται στο σχήμα 4.1. Τα δυναμικά δύο σημείων ορίζονται ως εξής:

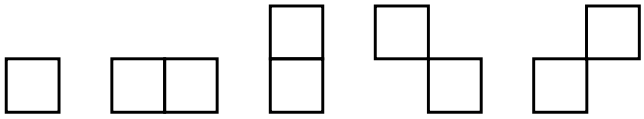
$$V_C(x_s)=\begin{cases}-\beta, & \text{if } x_s = x_q \quad \text{and } s,q \in C\\ +\beta, & \text{if } x_s \neq x_q \quad \text{and } s,q \in C.\end{cases} \tag{4.4}$$

Η παράμετρος β είναι θετική έτσι ώστε δύο γειτονικά pixels να είναι πιο πιθανό να ανήκουν στην ίδια κλάση (cluster) παρά σε διαφορετικές κλάσεις. Αυξάνοντας την τιμή του β έχει σαν αποτέλεσμα την αύξηση του μεγέθους των περιοχών και της εξομάλυνση (smoothing) των συνόρων. Ουσιαστικά δηλαδή, αυξάνοντας την τιμή β ο αλγόριθμος προχωράει σε μεγαλύτερη ομαδοποίηση των γειτονικών pixels πράγμα το οποίο συντελεί στην εξομάλυνση των περιοχών και στην εξάλειψη των μικρών περιοχών (λεπτομερειών) της εικόνας. Τα δυναμικά κλίκας ενός σημείου ορίζονται ως εξής:

$$V_c(x_s)=\alpha^i, \quad \text{if } x_s = i \text{ and } s \in C, \text{ all } i. \tag{4.5}$$

Όσο μικρότερη η τιμή του α^i, τόσο πιο πιθανό είναι το pixel στη θέση s να ανήκει στο cluster i. Σημειώνουμε ότι ο εκθέτης i δεν εκφράζει δύναμη του α αλλά απλά είναι ένας δείκτης. Αυτό που στην ουσία εκφράζουν οι παράμετροι α^i είναι η a priori γνώση μας της σχετικής πιθανότητας ύπαρξης των διαφορετικών περιοχών. Αν δηλαδή ξέραμε από πριν

ότι, από τις δύο πιθανές περιοχές μίας εικόνας η μία είχε πιθανότητα εμφάνισης 0.6 και η άλλη είχε 0.4, αυτή η πληροφορία θα καθόριζε και τις τιμές του α^i. Και βέβαια η τιμή του α^i για την περιοχή με πιθανότητα εμφάνισης 0.6 θα ήταν μικρότερη από την τιμή του α^i για την περιοχή με πιθανότητα εμφάνισης 0.4. Εμείς όμως θα θεωρούμε από δω και στο εξής ότι όλες οι περιοχές έχουν την ίδια πιθανότητα εμφάνισης και γι' αυτό θα θέσουμε α^i =0 για όλα τα i.



Σχήμα 4.1 Κλίκες ενός και δύο σημείων για την πυκνότητα Gibbs.

Η πυκνότητα υπό συνθήκη p(y|x) μοντελοποιείται σαν μία white Gaussian process με μέση τιμή (mean) μ^i_s και διασπορά (variance) σ^2. Σημειώνουμε και πάλι ότι ο εκθέτης i δεν συμβολίζει κάποια δύναμη αλλά είναι απλά ένας ακόμα δείκτης. Κάθε περιοχή i χαρακτηρίζεται από μία διαφορετική μ^i_s η οποία είναι μια **αργά** μεταβαλλόμενη συνάρτηση του s (μιας και έχουμε θεωρήσει ότι οι περιοχές μας θα είναι ομαλές). Έτσι η ένταση κάθε περιοχής μοντελοποιείται σαν ένα σήμα μ^i_s συν λευκό Gaussian θόρυβο με διασπορά (variance) σ^2. Η a posteriori probability density function p(x|y) έχει συνεπώς την ακόλουθη μορφή:

$$p(x|y) \propto \exp\left\{-\sum_s \frac{1}{2\sigma^2}[y_s - \mu_s^{x_s}]^2 - \sum_C V_C(x_s)\right\}. \qquad (4.6)$$

Αυτό το οποίο εκφράζει η (6) είναι το γεγονός ότι για να βρούμε την κατανομή των περιοχών x (δηλαδή το segmentation της εικόνας) με δεδομένη την αρχική εικόνα y, πρέπει να μεγιστοποιήσουμε το άθροισμα που βρίσκεται μέσα στις αγκύλες στο δεξί μέλος της (6). Αυτό όμως θα γίνει στην επόμενη παράγραφο.

Παρατηρούμε από την (6) ότι η pdf p(x|y) έχει δύο συνιστώσες. Η μία (πρώτο κομμάτι στο δεξί μέλος της αναλογίας) περιορίζει την ένταση της περιοχής να είναι κοντά στα δεδομένα. Η άλλη (δεύτερο κομμάτι στο δεξί μέλος της αναλογίας) επιβάλλει την απαραίτητη χωρική συνέχεια.

Παρατηρούμε επίσης, ότι αν οι συναρτήσεις έντασης μ^i_s δεν εξαρτώνταν από τη θέση s και αν θέταμε την παράμετρο β=0 τότε έχουμε την περίπτωση του K-means αλγόριθμου. Ο συγκεκριμένος αλγόριθμος δηλαδή είναι μία επέκταση του K-means σε δύο κατευθύνσεις, όπως άλλωστε είχαμε αναφέρει και στην προηγούμενη παράγραφο.

Αν τώρα υποθέσουμε ότι η παράμετρος β και η διασπορά θορύβου (noise variance) είναι γνωστές τότε πρέπει να υπολογίσουμε την κατανομή των περιοχών και τις συναρτήσεις έντασης μ^i_s. Αυτό θα γίνει στην επόμενη παράγραφο.

Μία παρατήρηση που πρέπει να κάνουμε πριν περάσουμε στην ανάλυση του αλγορίθμου, είναι ότι το τυχαίο πεδίο Gibbs είναι ένα καλό μοντέλο για τη διαδικασία των περιοχών αλλά μόνο αν έχουμε και ένα καλό μοντέλο για την πιθανότητα υπό συνθήκη. Δηλαδή το πεδίο Gibbs από μόνο του δεν είναι πολύ χρήσιμο. Δίνει όμως καλά αποτελέσματα αν συνδυαστεί σωστά με το μοντέλο που αναφέραμε παραπάνω.

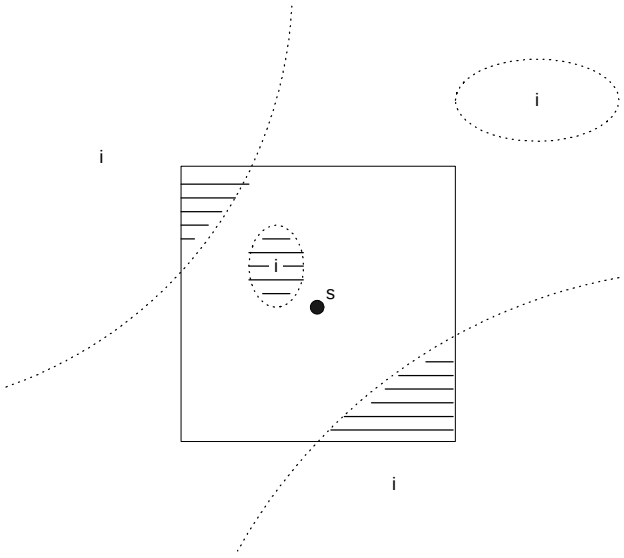
4.3 Ο ΑΛΓΟΡΙΘΜΟΣ

Στην παράγραφο αυτή θα δείξουμε τον αλγόριθμο για τον υπολογισμό των συναρτήσεων έντασης μ^i_s και της κατανομής των περιοχών x. Οι συναρτήσεις έντασης ορίζονται πάνω στο ίδιο πλέγμα όπως άλλωστε και η αρχική εικόνα μας αλλά και η κατανομή των περιοχών x. Όπως προαναφέραμε ο αλγόριθμος μας είναι επαναληπτικός. Εναλλάσεται ανάμεσα στον υπολογισμό των συναρτήσεων έντασης μ^i_s και τον υπολογισμό της κατανομής των περιοχών x (δηλαδή το segmentation της εικόνας).

Πρώτα ας θεωρήσουμε το πρόβλημα του υπολογισμού των συναρτήσεων έντασης μ^i_s. Με τη δεδομένη κατανομή των περιοχών x, υπολογίζουμε την ένταση μ^i_s σε κάθε pixel s ως εξής: παίρνουμε το μέσο όρο των επιπέδων του γκρι όλων των pixels που ανήκουν στην περιοχή (cluster) i και βρίσκονται μέσα σε ένα παράθυρο πλάτους W το οποίο έχει κέντρο το σημείο s, όπως φαίνεται στο σχήμα 4.2. Όσον αφορά το σχήμα αυτό, διευκρινίζουμε, ότι οι περιοχές (clusters) που συμβολίζουμε με i στο σχήμα είναι διαφορετικές μεταξύ τους (δηλαδή, για κάθε περιοχή το i έχει διαφορετική τιμή), απλά χρησιμοποιούμε το ίδιο σύμβολο για λόγους γενικότητας. Όταν ο αριθμός των pixels τύπου i μέσα στο παράθυρο με κέντρο s είναι πολύ μικρός τότε ο υπολογισμός μ^i_s δεν είναι αξιόπιστος. Στα σημεία αυτά η ένταση μ^i_s δεν ορίζεται και όταν συμβαίνει αυτό το x_s δεν μπορεί να θεωρηθεί ότι ανήκει στο επίπεδο(cluster) i. Γι' αυτό πρέπει να

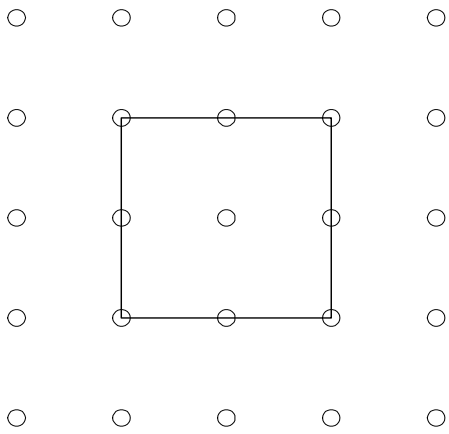
καθορίσουμε τον ελάχιστο αριθμό από pixels T_{min} τα οποία είναι απαραίτητα για τον υπολογισμό των μ^i_s. Είναι φανερό ότι περιοχές με έκταση μικρότερη από το κατώφλι T_{min} ίσως εξαφανιστούν. Μία λογική τιμή για το κατώφλι T_{min} είναι η T_{min} =W, δηλαδή να θέσουμε το T_{min} ίσο με το πλάτος του παραθύρου W. Αυτή η τιμή για το κατώφλι T_{min} εγγυάται ότι μακρίες περιοχές με φάρδος ένα pixel θα διατηρούνται.

Είναι φανερό ότι πρέπει να κάνουμε υπολογισμούς για όλους τους τύπους των περιοχών i και για όλα τα pixels s. Και είναι επίσης φανερό ότι οι υπολογισμοί που χρειάζονται για κάτι τέτοιο είναι υπερβολικά πολλοί, ειδικά για μεγάλα μεγέθη παραθύρων. Αντί λοιπόν να κάνουμε εξαντλητικά όλους τους παραπάνω υπολογισμούς, υπολογίζουμε τα μεγέθη μ^i_s μόνο πάνω σε ένα πλέγμα σημείων. Για τα ενδιάμεσα σημεία χρησιμοποιούμε μία τεχνική που ονομάζεται **bilinear interpolation** και υπολογίζουμε τις ενδιάμεσες τιμές. Η απόσταση των σημείων του πλέγματος διαλέγουμε ώστε να ισούται με το μισό του πλάτους του παραθύρου έτσι ώστε να έχουμε 50% επικάλυψη. Στο σχήμα 4.3 φαίνεται ένα παράθυρο και οι θέσεις των κέντρων των γειτονικών παραθύρων. Στο σχήμα, τα μικρά κυκλάκια συμβολίζουν τις δυνατές θέσεις των κεντρών των διαφόρων παραθύρων.



Σχήμα 4.2 Υπολογισμός των μ^i_s.

Σημειώνουμε εδώ ότι επειδή οι συναρτήσεις έντασης μ^i_s είναι ομαλές(smooth), οι τιμές που υπολογίζουμε με το bilinear interpolation είναι καλές προσεγγίσεις των τιμών που θα παίρναμε με ακριβείς (εξαντλητικούς) υπολογισμούς. Επίσης παρατηρούμε ότι όσο μεγαλύτερο είναι το μέγεθος του παραθύρου τόσο πιο smooth θα είναι οι συναρτήσεις μ^i_s ενώ παράλληλα θα μεγαλώνει και οι απόσταση των κέντρων των παραθύρων. Από την άλλη όμως, λόγω της εξάρτησης της παραπάνω απόστασης από το μέγεθος του παραθύρου W, συμπεραίνουμε ότι η ποσότητα των υπολογισμών είναι ανεξάρτητη από το μέγεθος του παραθύρου.



Σχήμα 4.3 Πλέγμα υπολογισμού των συναρτήσεων μ^i_s και θέσεις των παραθύρων.

Ας θεωρήσουμε τώρα το πρόβλημα του υπολογισμού της κατανομής των περιοχών. Με δεδομένες τις συναρτήσεις των εντάσεων μ^i_s, αυτό που θέλουμε είναι να μεγιστοποιήσουμε την a posteriori probability density (6) για να βρούμε το βέλτιστο segmentation x. Όμως το να βρεις το συνολικό μέγιστο αυτής της συνάρτησης απαιτεί πάρα πολλούς υπολογισμούς. Μία τεχνική που μπορεί να χρησιμοποιηθεί για αυτό είναι το simulated annealing **[17]**. Ωστόσο αντί να εκτελέσουμε την παραπάνω διαδικασία, αυτό που κάνουμε είναι να μεγιστοποιήσουμε την πιθανότητα υπό συνθήκη σε κάθε σημείο x_s δεδομένης της εικόνας y και του παρόντος segmentation σ' όλα τα υπόλοιπα σημεία. Δηλαδή για κάθε x_s μεγιστοποιούμε την ακόλουθη συνάρτηση:

$$
\begin{aligned}
&p(x_s | y, x_q,\ \text{all}\, q \neq s) \\
&\quad = p(x_s | y_s, x_q,\ q \in N_s) \\
&\quad \propto \exp\left\{ -\frac{1}{2\sigma^2}[y_s - \mu_s^{x_s}]^2 - \sum_{x_s \in C} V_C(x_s) \right\}.
\end{aligned}
\tag{4.7}
$$

Ουσιαστικά, για ένα συγκεκριμένο pixel, η (7) μεγιστοποιείται για την περιοχή εκείνη για την οποία, πρώτον η μέση τιμή της μ^i_s είναι αρκετά κοντά στην τιμή του pixel και δεύτερον τα γειτονικά pixel (όσο το δυνατό περισσότερα) να ανήκουν στην ίδια περιοχή.

Η μεγιστοποίηση γίνεται για κάθε σημείο της εικόνας και ο κύκλος επαναλαμβάνεται μέχρι να επιτευχθεί σύγκλιση. Η παραπάνω διαδικασία συγκλίνει σε ένα τοπικό μέγιστο. Η διαδικασία τερματίζει όταν ο αριθμός των pixels που αλλάζουν σ' ένα κύκλο είναι μικρότερος από ένα κατώφλι (συνήθως M/10 για μία MxM εικόνα). Η σύγκλιση είναι αρκετά γρήγορη, συνήθως σε λιγότερους από πέντε κύκλους σύμφωνα με τις παρατηρήσεις του Besag **[18],[19]**.

Ας μελετήσουμε τώρα τον συνολικό αλγόριθμο. Αρχικά παίρνουμε ένα αρχικό segmentation εφαρμόζοντας τον K-means αλγόριθμο στην αρχική εικόνα. Ξεκινώντας από αυτό το segmentation ο αλγόριθμος εναλλάσεται ανάμεσα στον υπολογισμό των συναρτήσεων έντασης μ^i_s και τον υπολογισμό της κατανομής των περιοχών x. Ορίζουμε μία "επανάληψη" να αποτελείται από μία ενημέρωση (υπολογισμό) των μ^i_s και μία ενημέρωση (υπολογισμό) του x. Το μέγεθος του παραθύρου για τον υπολογισμό των μ^i_s διατηρείται σταθερό έως ότου η διαδικασία συγκλίνει (συνήθως σε λιγότερες από δέκα επαναλήψεις, αν και ο αριθμός αυτός εξαρτάται από το περιεχόμενο της εικόνας). Το κριτήριο σύγκλισης είναι ότι η τελευταία επανάληψη συγκλίνει σε ένα κύκλο. Κατόπιν η όλη διαδικασία επαναλαμβάνεται με ένα καινούριο μέγεθος παραθύρου (μικρότερο). Η ικανότητα προσαρμογής του αλγορίθμου έγγειται στο γεγονός ότι το μέγεθος του παραθύρου μεταβάλλεται. Αρχικά το μέγεθος του παραθύρου είναι ίσο με τις διαστάσεις τις εικόνας. Καθώς ο αλγόριθμος προχωράει το μέγεθος του παραθύρου σταδιακά μικραίνει. Ο λόγος για τον οποίο γίνεται αυτό είναι ότι στα πρώτα βήματα του αλγορίθμου, το segmentation είναι "χοντροκομμένο" και ένα μεγάλο παράθυρο είναι απαραίτητο για τον υπολογισμό των συναρτήσεων έντασης. Καθώς ο αλγόριθμος συνεχίζει το segmentation γίνεται καλύτερο, και μικρότερα παράθυρα δίνουν πιο αξιόπιστα αποτελέσματα. Έτσι ο αλγόριθμος ξεκινώντας από συνολικούς (global)

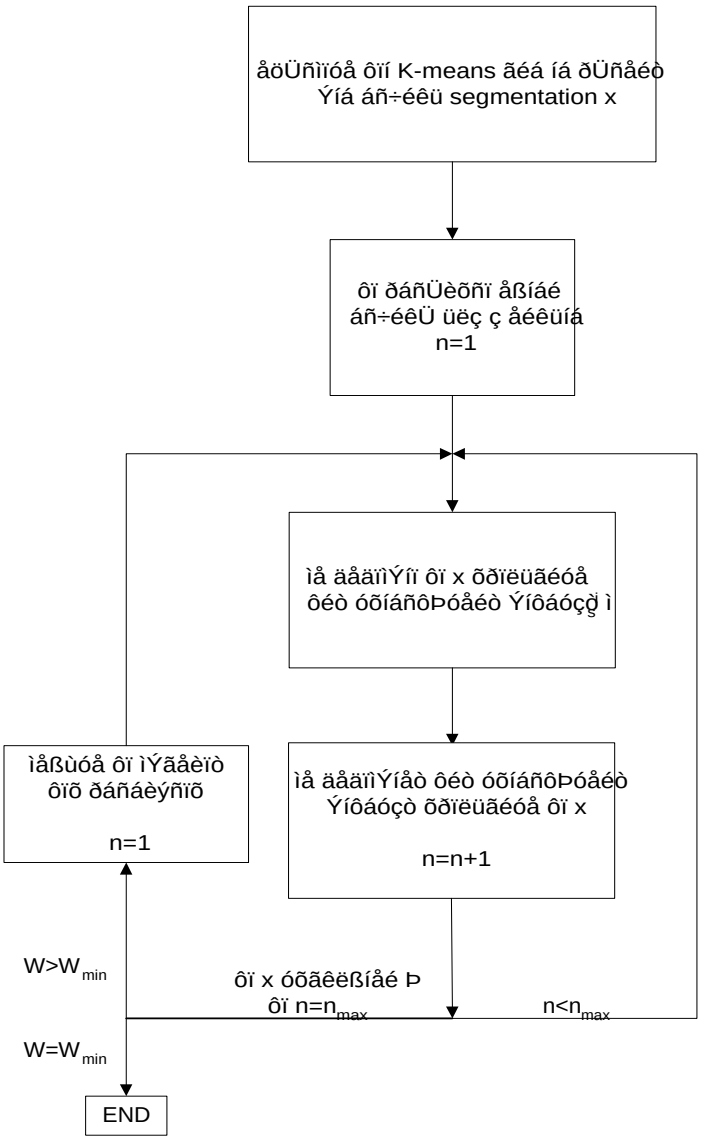
υπολογισμούς, σιγά-σιγά προσαρμόζεται στα τοπικά χαρακτηριστικά της κάθε περιοχής. Ο αλγόριθμος τερματίζει όταν φτάσει στο ελάχιστο επιτρεπτό μέγεθος παραθύρου (το οποίο εμείς καθορίζουμε). Στην πράξη, αυτό που εφαρμόζουμε είναι μείωση του μεγέθους του παραθύρου κατά ένα παράγοντα δύο, έως να φτάσουμε σε ένα ελάχιστο μέγεθος παραθύρου W=7 pixels (η κάθε του διάσταση) , οπότε και ο αλγόριθμος τερματίζει. Στο σχήμα 4.4 φαίνεται το λεπτομερές διάγραμμα ροής (flowchart) του αλγορίθμου.

Αν το μέγεθος του παραθύρου διατηρούνταν σταθερό (δηλαδή αν δεν ήταν adaptive ο αλγόριθμός μας) τότε τα αποτελέσματα που θα παίρναμε, θα ήταν κάτι ενδιάμεσο ανάμεσα στον K-means αλγόριθμο και τον αλγόριθμό μας. Η "δύναμη" και η αποτελασματικότητα της τεχνικής που παρουσιάσαμε, έγγειται στην σταδιακή μεταβολή του μεγέθους του παραθύρου, το οποίο επιτρέπει στις συναρτήσεις έντασης να προσαρμοστούν στα τοπικά χαρακτηριστικά της εικόνας.

Όσων αφορά τη διασπορά του θορύβου, για την οποία είχαμε συζητήσει παραπάνω, υποθέτουμε ότι μπορεί να υπολογιστεί ανεξάρτητα από τον αλγόριθμο. Από την εξίσωση (6) παρατηρούμε ότι αύξηση του σ^2 ισοδυναμεί με αύξηση του β. Στην υλοποίηση θεωρήσαμε σταθερή τιμή του β για όλες τις επαναλήψεις και ίση με 0.5. Ωστόσο θα μπορούσε αρχικά να επιλέγαμε μία μικρή τιμή του β για τα πρώτα στάδια του αλγορίθμου, και στην συνέχεια (καθώς μίκραινε το μέγεθος του παραθύρου) να την αυξάναμε για να παίρναμε πιο ομαλά (smooth) περιγράμματα (βλ. παράγραφο 4.6). Συνεπώς η μόνη παράμετρος που μένει να καθορίσουμε είναι η διασπορά του θορύβου, η οποία είναι και η πιο σημαντική, καθώς είναι αυτή που θα καθορίσει την ποσότητα των λεπτομερειών που θα ανιχνευτούν από τον αλγόριθμο.

Τέλος, πρέπει να καθορίσουμε ακόμα τον αριθμό των διαφορετικών περιοχών K που θα ανιχνεύσει ο αλγόριθμος. Για K=4 ο αλγόριθμος δουλεύει αρκετά καλά για μία μεγάλη γκάμα εικονών. Ωστόσο η επιλογή του K δεν μπορεί να θεωρηθεί καθοριστική για την πορεία του αλγορίθμου, καθώς η προσαρμοστικότητα του, επιτρέπει σε μία περιοχή να έχει διαφορετικές εντάσεις σε διαφορετικά κομμάτια της εικόνας. Αυτό δεν ισχύει στην περίπτωση του αλγορίθμου K-means και άλλων παρόμοιων τεχνικών, στις οποίες η επιλογή των περιοχών είναι καθοριστική για την εξέλιξη του αλγορίθμου και συνεπώς την εγκυρότητα των αποτελεσμάτων.

Ο αριθμός των υπολογισμών για το τελικό segmentation εξαρτάται από τις διαστάσεις της εικόνας, από τον αριθμό των διαφορετικών παραθύρων που θα χρησιμοποιηθούν, από τον αριθμό των περιοχών, και φυσικά από το περιέχομενο της εικόνας (το οποίο και καθορίζει την ταχύτητα σύγκλισης των εκάστοτε διαδικασιών). Όπως αναφέραμε και προηγουμένως, ο αριθμός των υπολογισμών για τις συναρτήσεις έντασης δεν εξαρτάται από το μέγεθος του παραθύρου.



Σχήμα 4.4 Το διάγραμμα ροής του αλγορίθμου.

4.4 ΠΑΡΑΔΕΙΓΜΑΤΑ

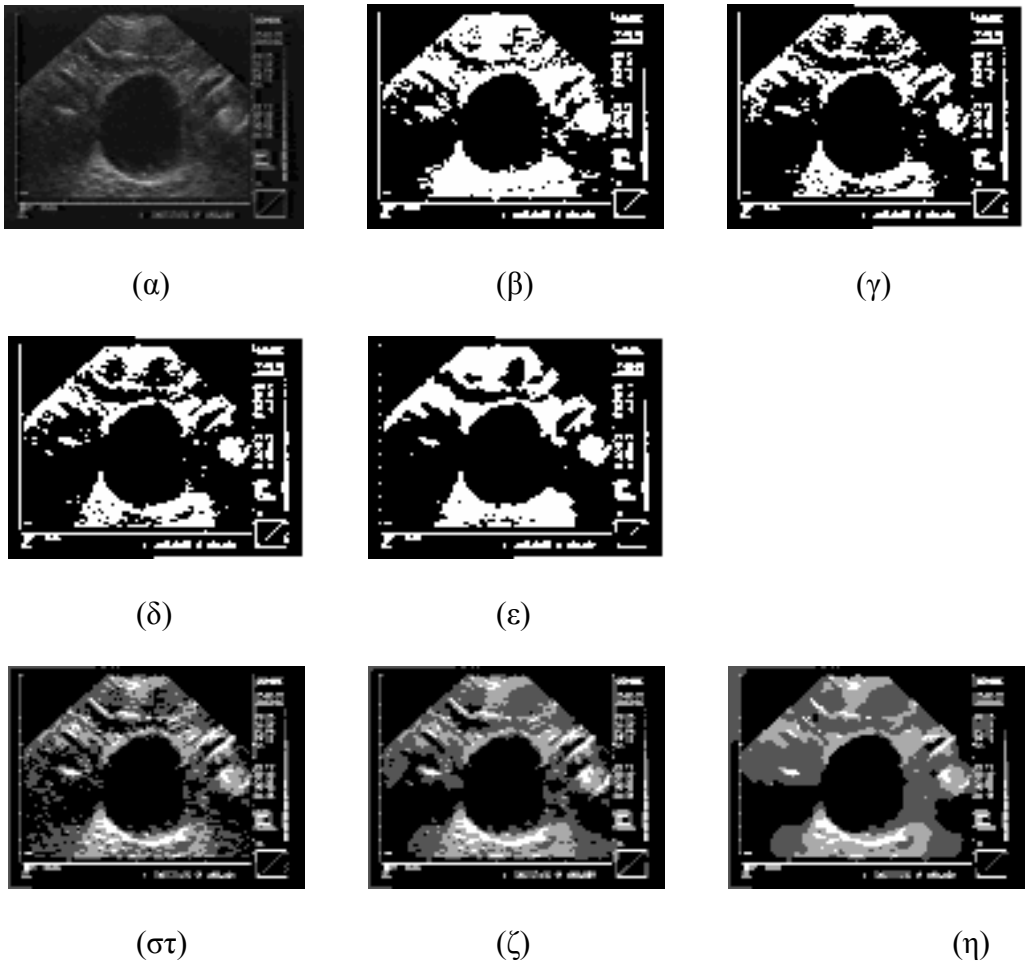
Στην παράγραφο αυτή εξετάζουμε την απόδοση του αλγορίθμου μέσω της εφαρμογής του σε κάποιες εικόνες. Επίσης συγκρίνουμε τον αλγόριθμο με τον K-means και αποδεικνύουμε την ανωτερότητά του πρώτου. Ανάλογα με την τιμή που δίνουμε στην παράμετρο σ (noise variance) και τον αριθμό των clusters που επιλέγουμε, παίρνουμε διάφορες εικόνες που είτε διατηρούν πολλές από τις λεπτομέρειες της εικόνας, είτε λίγες, είτε καθόλου, οδηγούν πάντα σε σαφείς αναπαραστάσεις της αρχικής εικόνας με 2-4 επίπεδα του γκρι.

Στο σχήμα 4.5(α) φαίνεται μία εικόνα η οποία αναπαριστά μία ανθρώπινη κύστη. Η εικόνα είναι παρμένη με υπερηχοτομογράφο, οι διαστάσεις της είναι 112x84 και χρησιμοποιεί 8 bits για την κωδικοποίηση του κάθε pixel, δίνοντας έτσι μέγιστο αριθμό χρωμάτων 255. Στο σχήμα 4.5(β) φάινεται το αποτέλεσμα του K-means αλγόριθμου στην εικόνα, με K=2 επίπεδα. Στο σχήμα 4.5(γ) βλέπουμε το αποτέλεσμα της εφαρμογής του αλγορίθμου που παρουσιάσαμε για K=2 επίπεδα και υποθετική τιμή της παραμέτρου σ 0.71. Επίσης στα σχήματα 4.5(δ) και 4.5(ε) βλέπουμε τις εικόνες που πήραμε για σ=4 και σ=8 αντίστοιχα. Τέλος στα σχήματα 4.5(στ),(ζ),(η) βλέπουμε τις εικόνες για K=4 επίπεδα και σ=.71, σ=4, και σ=8 αντίστοιχα.

Από τα σχήματα 4.5(β) και 4.5(γ) φαίνεται ήδη η αποτελεσματικότητα του αλγορίθμου που παρουσιάσαμε και η οποία είναι σαφώς ανώτερη από αυτή του K-means. Ο adaptive αλγόριθμος, όπως φαίνεται, έχει ήδη εντοπίσει κάποιες περιοχές (μαύρες) στην εικόνα τις οποίες δεν έχει καταφέρει να εντοπίσει ο K-means. Αυτό βέβαια ισχύει για πολλή μικρή τιμή της παραμέτρου σ (σ=0.71). **Καθώς αυξάνουμε την τιμή της παραμέτρου σ**, οι λεπτομέρειες τις εικόνας αρχίζουν να εκλείπουν, οι ακμές αρχίζουν να εξομαλύνονται αλλά τα βασικά χαρακτηριστικά της εικόνας εξακολουθούν να διατηρούνται (βλ. σχήμα 4.5(δ),(ε)). Η παράμετρος σ λοιπόν είναι αυτή που καθορίζει την ποσότητα των λεπτομερειών που θα ανιχνευθούν από τον αλγόριθμο.

Στα σχήματα 4.5(στ),(ζ),(η) βλέπουμε το αποτέλεσμα της εφαρμογής του αλγορίθμου στην αρχική εικόνα με K=4 επίπεδα τώρα. Όπως φαίνεται από τα σχήματα, η αναπαράσταση της εικόνας είναι τώρα ρεαλιστικότερη μιας και χρησιμοποιούνται περισσότερα επίπεδα του γκρι. Η διαφορά στο αποτέλεσμα έχει μοναδικό τίμημα το χρόνο εκτέλεσης ο οποίος είναι τώρα σχεδόν διπλάσιος. Αναλυτικότερα για τους χρόνους

εκτέλεσης τόσο του K-means όσο και του adaptive αλγορίθμου, ακολουθεί ένας πίνακας (πίνακας Α) με τους χρόνους για κάθε περίπτωση. Οι δοκιμές έγιναν σε ένα sparc Ultra 1 SUN-station σε πλατφόρμα UNIX. Οι χρόνοι που φαίνονται στον πίνακα είναι όλοι σε δευτερόλεπτα. Από τον πίνακα παρατηρούμε τη διαφορά ανάμεσα στο χρόνο εκτέλεσης του K-means και του adaptive αλγόριθμου για κάθε περίπτωση. Ωστόσο, πρέπει να πούμε εδώ, ότι δεν έγινε καμμία προσπάθεια για μείωση του χρόνου εκτέλεσης του αλγορίθμου, χρησιμοποιώντας κάποιο είδος βελτιστοποίησης.



Σχήμα 4.5 (α) Αρχική εικόνα. (β) K-means (K=2 επίπεδα). (γ) Adaptive clustering (K=2, σ=0.71). (δ) K=2, σ=4. (ε) K=2, σ=8. (στ) K=4, σ=0.71. (ζ) K=4, σ=4. (η) K=4, σ=8.

ΠΙΝΑΚΑΣ Α

	K-means(K=2)	Adaptive(K=2,σ=8)	Adaptive(K=4,σ=8)
χρόνος εκτέλεσης	0.2	0.8	2.2

Μία άλλη εικόνα, η οποία χρησιμοποιήθηκε για να τεστάρουμε τον adaptive αλγόριθμο, φαίνεται στο σχήμα 4.6(α). Η εικόνα (η γνωστή Lena) έχει διαστάσεις 256x240 και η κλίμακα του γκρι που χρησιμοποιεί είναι από 0-255. Το αποτέλεσμα του K-means αλγόριθμου (για K=2 επίπεδα) φαίνεται στο σχήμα 4.6(β) ενώ το αποτέλεσμα του adaptive αλγόριθμου (για K=2, σ=8) φαίνεται στο σχήμα 4.6(γ).

(α) (β)

(γ)

Σχήμα 4.6 (α) Αρχική εικόνα. (β) K-means (K=2 επίπεδα). (γ) Adaptive clustering (K=2, σ=8).

Από τα σχήματα 4.6(β),(γ) παρατηρούμε τη διαφορά της απόδοσης του αλγορίθμου K-means από τον adaptive αλγόριθμο. Είναι φανερό, ότι ο adaptive αλγόριθμος προσαρμόζεται πολύ καλά στα χαρακτηριστικά της εικόνας, με αποτέλεσμα να εντοπίζει τα βασικότερα απ' αυτά και να εξαφανίζει τα λιγότερο σημαντικά (αυτό βέβαια καθορίζεται από την τιμή της παραμέτρου σ). Από το σχήμα 4.6(β) φαίνεται η αδυναμία του K-means αλγορίθμου να εντοπίσει τη σκία που δημιουργεί η μύτη της κοπέλας, πολλά από τα φτερά που βρίσκονται στο καπέλο και την περιοχή που βρίσκεται στο κάτω δεξιά κομμάτι της εικόνας.

Όσών αφορά την εφαρμογή του K-means αλγορίθμου, αυτό που πρέπει να επισημάνουμε είναι ότι η απόδοση του εξαρτάται σε πολύ μεγάλο βαθμό από τη εκλογή των αρχικών κέντρων των clusters αλλά και από τον αριθμό τους. Το αποτέλεσμα του σχήματος 4.6(β) προέκυψε μετά από αρκετές δοκιμές, καθώς τόσο η θέση των κέντρων στην εικόνα όσο και το gray-level του κάθε κέντρου είναι παράγοντες καθοριστικής σημασίας για το αποτέλεσμα που θα πάρουμε. Παρά τις δοκιμές που έγιναν, το αποτέλεσμα εξακολουθεί να μην είναι αρκετά ικανοποιητικό, πράγμα που φανερώνει την αδυναμία του αλγορίθμου K-means να δώσει μία ρεαλιστική αναπαράσταση της αρχικής εικόνας χρησιμοποιώντας μόνο δύο επίπεδα του γκρι. Από την άλλη, η αναπαράσταση που δίνει ο adaptive αλγόριθμος είναι αρκετά ικανοποιητική, παρόλο που χρησιμοποιούνται μόνο δύο επίπεδα του γκρι. Επιπλέον ο καθορισμός των αρχικών κέντρων των clusters δεν είναι πλέον καθοριστικής σημασίας, καθώς, ο adaptive αλγόριθμος, με τις συνεχείς επαναλήψεις κατορθώνει να προσαρμόζεται στα χαρακτηριστικά της εικόνας, ακόμα κι όταν το αρχικό segmentation (αυτό που έχει προκύψει από τον K-means αλγόριθμο) δεν είναι αρκετά καλό.

Το τίμημα που πληρώσαμε για τα καλύτερα αποτελέσματα του adaptive αλγόριθμου, δεν θα μπορούσε να ήταν άλλο από υπολογιστικό. Στον Πίνακα Β φαίνονται οι χρόνοι εκτέλεσης τόσο του K-means όσο και του adaptive αλγορίθμου. Και πάλι παρατηρούμε την μεγάλη διαφορά ανάμεσα στους δύο χρόνους. Υπενθυμίζουμε όμως,ότι δεν έγινε καμμία προσπάθεια για μείωση του χρόνου εκτέλεσης του adaptive αλγορίθμου. Επίσης, μπορεί να αποδειχθεί, ότι ο υπολογισμός των συναρτήσεων έντασης μ^i_s και κάθε κύκλος μεγιστοποίησης της πιθανότητας $p(x_s|y_s,x_q)$, μπορεί να γίνει παράλληλα. Έτσι, αν και ο αλγόριθμος είναι σχετικά αργός σε σειριακές μηχανές, μπορεί να γίνει αρκετά γρηγορότερος αν εκτελεστεί από μία παράλληλη μηχανή.

ΠΙΝΑΚΑΣ Β

	K-means αλγόριθμος	Adaptive αλγόριθμος
χρόνος εκτέλεσης	1.5	14

4.5 ΙΕΡΑΡΧΙΚΗ ΥΛΟΠΟΙΗΣΗ

Στην παράγραφο αυτή παρουσιάζουμε μία ιεραρχική υλοποίηση του αλγόριθμου που παρουσιάσαμε στην παράγραφο ΙΙΙ. Σκοπός αυτής της υλοποίησης είναι τόσο η βελτίωση της απόδοσης του αλγορίθμου όσο και η μείωση του χρόνου εκτέλεσης. Η διαδικασία που ακολούθουμε για την υλοποίηση αυτή περιγράφεται αναλυτικά παρακάτω.

Αρχικά δημιουργούμε μία πυραμίδα από εικόνες διαφορετικών αναλύσεων (resolution) η κάθε μία, ως εξής: ξεκινώντας από την υψηλότερης ανάλυσης εικόνα, παίρνουμε την επόμενη εικόνα φιλτράροντας την αρχική με ένα βαθυπερατό (low pass) φίλτρο και κατόπιν περνώντας την από ένα φίλτρο αποδεκατισμού (decimation filter) το οποίο αυτό που κάνει είναι να δώσει μία εικόνα της οποίας οι διαστάσεις είναι οι μισές των αρχικών. Αν για παράδειγμα η εικόνα μας έχει διαστάσεις NxN τότε το φίλτρο αποδεκατισμού (συγκεκριμένα ο τελεστής αποδεκατισμού) D θα έχει διαστάσεις (N/2 x N) και η μορφή του θα είναι η ακόλουθη:

$$D = \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 & 0 \\ 0 & 0 & 1 & . & . & . & 0 & 0 \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ 0 & 0 & 0 & . & . & . & 1 & 0 \end{bmatrix}$$

Το μοντέλο για τις εικόνες που περιγράψαμε στην παράγραφο 4.2, μπορεί να γραφεί ως εξής:

$$y_s = I_s + n_s \tag{4.8}$$

όπου I_s είναι το σήμα της εικόνας μας και n_s είναι ο λευκός Gaussian θόρυβος με την τυπική απόκλιση σ. Το συνολικό σήμα, στην αμέσως επόμενη ανάλυση της πυραμίδας, θα έχει την μορφή:

$$y'_s = I'_s + n'_s \tag{4.9}$$

όπου $I_s^{'}$ θα είναι το φιλτραρισμένο και αποδεκατισμένο σήμα I_s, και $n_s^{'}$ είναι το αποδεκατισμένο n_s , το οποίο είναι "σχεδόν" λευκός Gaussian θόρυβος με τυπική απόκλιση $\sigma^{'} = \sigma/2$. Έτσι το μοντέλο εξακολουθεί να ισχύει και για τις χαμηλότερες αναλύσεις, με μόνη διαφορά ότι η τυπική απόκλιση σ, αλλάζει κατά ένα παράγοντα 2 κάθε φορά.

Όσων αφορά τώρα την παράμετρο β, για την οποία είχαμε συζητήσει σε προηγούμενη παράγραφο, αυτή είναι λογικό να θωρήσουμε ότι δεν αλλάζει από ανάλυση σε ανάλυση. Αν και έχουν προταθεί διάφορες προσεγγίσεις της παραμέτρου β, ανάλογα με το επίπεδο της ανάλυσης (resolution level) στο οποίο βρισκόμαστε, εμείς θα τη θεωρούμε σταθερή κατά της διάρκεια της μελέτης μας (άλλωστε και στην υλοποίηση η παράμετρος αυτή σταθερή θεωρήθηκε). Έτσι, αφού επίλεξουμε τελικά τις παραμέτρους β και σ, το μόνο που μας μένει είναι να καθορίσουμε είναι το κατώφλι T_{min}. Στο υψηλότερο επίπεδο ανάλυσης, είναι λογικό να θεωρήσουμε $T_{min} = W$. Από κει και πέρα, καθώς προχωράμε σε χαμηλότερα επίπεδα ανάλυσης, τα σημαντικά χαρακτηριστικά της εικόνας γίνονται μικρότερα και συνεπώς θα πρέπει να μειώσουμε το κατώφλι T_{min}. Έτσι μειώνουμε το κατώφλι T_{min} κατά ένα παράγοντα 2 από το ένα επίπεδο ανάλυσης στο άλλο.

Η ιεραρχική υλοποίηση του αλγορίθμου περιγράφεται στο σχήμα 4.7. Περιγραφικά, ο ιεραρχικός αλγόριθμος προχωράει ως εξής: ξεκινά από την χαμηλότερη ανάλυση και εφαρμόζει τον αλγόριθμο που περιγράψαμε στην παράγραφο ΙΙΙ. Όταν φτάσει στο μικρότερου μεγέθους παράθυρο, τότε "πηδάει" στο επόμενο (μεγάλυτερο) επίπεδο ανάλυσης και χρησιμοποιώντας το segmentation που πήρε (από το προηγούμενο επίπεδο), διευρυμένο κατά ένα παράγοντα 2, εφαρμόζει τον ίδιο αλγόριθμο στην εικόνα τώρα της μεγαλύτερης ανάλυσης. Έτσι τώρα έχουμε ένα καλό σημείο για ξεκίνημα και δεν χρειάζεται να χρησιμοποιήσουμε όλη τη σειρά των παραθύρων για το νέο επίπεδο ανάλυσης. Στο νέο επίπεδο ανάλυσης ξεκινάμε από το διπλάσιο μέγεθος παραθύρου από

αυτό της προηγούμενης ανάλυσης. Με όμοιο τρόπο συνεχίζουμε και στις επόμενες αναλύσεις μέχρι να φτάσουμε στην υψηλότερη.

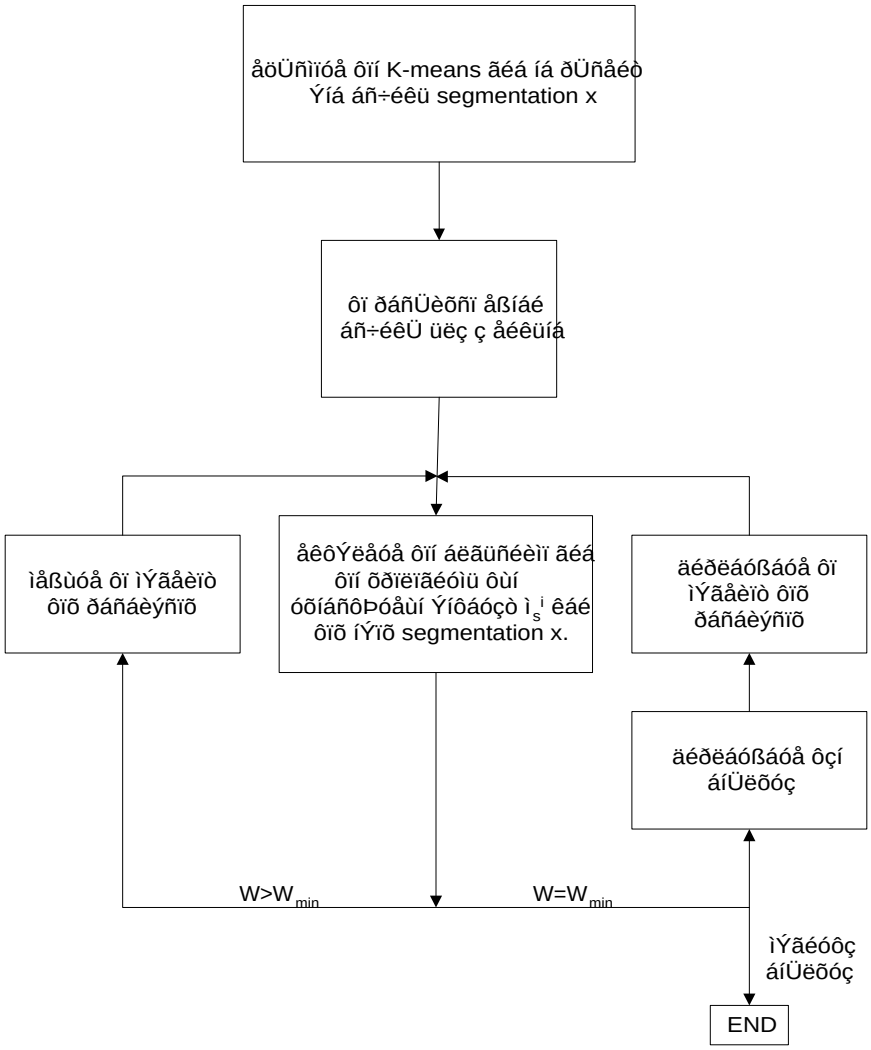
Η παραπάνω ιεραρχική μέθοδος υλοποίησης του αλγορίθμου μειώνει σημαντικά τον αριθμό των υπολογισμών που απαιτούνται για την ολοκλήρωση του αλγορίθμου, καθώς οι περισσότερες επαναλήψεις συμβαίνουν στο χαμηλότερο επίπεδο ανάλυσης (εκεί δηλαδή όπου η εικόνα έχει τις μικρότερες διαστάσεις). Η βελτίωση είναι όντως σημαντική, όπως άλλωστε φαίνεται και από το παραδείγμα που ακολουθεί.

Στο σχήμα 4.8(α) βλέπουμε πάλι τη γνωστή μας Lena ενώ το αποτέλεσμα της εφαρμογής του adaptive αλγόριθμου (για K=2) της παραγράφου 4.3 φαίνεται στο σχήμα 4.8(β). Οι εικόνες 4.8(α) και 4.8(β) είναι στην ουσία οι 4.7(α) και 4.7(β) της προηγούμενης παραγράφου, απλά για λόγους σύγκρισης τις επαναλαμβάνουμε εδώ. Στο σχήμα 4.8(γ) φαίνεται το αποτέλεσμα της εφαρμογής του ιεραρχικού αλγορίθμου (πάλι για K=2). Όπως φαίνεται από τα σχήματα 4.8(β),(γ), τόσο ο adaptive αλγόριθμος όσο και ο ιεραρχικός, δίνουν πολύ καλά αποτελέσματα. Όμως, όπως φαίνεται και από τη σύγκριση των δύο εικονών, ο ιεραρχικός αλγόριθμος προσαρμόζεται ακόμα καλύτερα στα χαρακτηριστικά της εικόνας, όπως μπορεί να φανεί από τη διαφορά στα χείλη της κοπέλας.

Όσων αφορά τώρα τη μείωση του χρόνου εκτέλεσης του ιεραρχικού αλγορίθμου, στον Πίνακα Γ φαίνονται οι χρόνοι εκτέλεσης τόσο του adaptive αλγορίθμου όσο και του ιεραρχικού. Όπως παρατηρούμε η μείωση είναι αρκετά σημαντική και μάλιστα χωρίς καμμία αρνητική επίδραση στο τελικό αποτέλεσμα, αντιθέτως το segmentation είναι ακόμα καλύτερο.

ΠΙΝΑΚΑΣ Γ

	Adaptive αλγόριθμος	Ιεραρχικός αλγόριθμος
χρόνος εκτέλεσης	14	4.6



Σχήμα 4.7 Το διάγραμμα ροής του ιεραρχικού αλγορίθμου.

(α)

(β)

(γ)

Σχήμα 4.8 (α) Αρχική εικόνα. (β) Adaptive clustering (K=2). (γ) Ιεραρχικός αλγόριθμος (K=2).

4.6 ΤΡΟΠΟΠΟΙΗΣΕΙΣ ΤΟΥ ΑΛΓΟΡΙΘΜΟΥ

Στις προηγούμενες παραγράφους, παρουσιάσαμε αναλυτικά έναν adaptive αλγόριθμο για Image Segmentation, σε εικόνες με αντικείμενα που έχουν σχετικά ομαλές (smooth) επιφάνειες. Παρουσιάσαμε επίσης μία ιεραρχική υλοποίηση του αλγορίθμου αυτού η οποία μας οδήγησε σε καλύτερα αποτελέσματα, και μάλιστα απαιτώντας σημαντικά λιγότερο υπολογιστικό χρόνο. Θα συνοψίσουμε τώρα τις όποιες αλλαγές κάναμε στον αλγόριθμο και τις αποκλίσεις μας από την προτεινόμενη υλοποίηση του, και θα δείξουμε πως επηρέασε η κάθε αλλαγή στο τελικό αποτέλεσμα.

Όπως είδαμε και προηγουμένως, ο υπολογισμός των συναρτήσεων έντασης γίνεται πάνω σε ένα πλέγμα, στο οποίο ορίζουμε οι αποστάσεις των κέντρων των διαδοχικών παραθύρων να είναι ίσες με το μισό του πλάτους του παραθύρου, ενώ για τις εναπομείναντες τιμές χρησιμοποιούμε bilinear interpolation. Αυτό, όπως αναφέραμε, γίνεται καθαρά για λόγους υπολογιστικού κόστους, καθώς, ο χρόνος που θα απαιτούνταν για ένα εξαντλητικό υπολογισμό των συναρτήσεων έντασης είναι πραγματικά τεράστιος. Εμείς όμως, υλοποιήσαμε αυτήν την εξαντλητική μέθοδο υπολογισμού, και συγκρίναμε το αποτέλεσμα με αυτό του προτεινόμενου αλγορίθμου (που χρησιμοποιεί το πλέγμα). Στο σχήμα 4.9(α) φαίνεται πάλι η γνωστή μας Lena, στο σχήμα 4.9(β) φαίνεται το αποτέλεσμα του adaptive αλγορίθμου (που χρησιμοποιεί πλέμα) και στο σχήμα 4.9(γ) βλέπουμε το αποτέλεσμα της εξαντλητικής μεθόδου. Στο σχήμα 4.9(δ), φαίνεται ένα αποτέλεσμα το οποίο πήραμε υλοποιώντας τον adaptive αλγόριθμο χρησιμοποιώντας μεν πλέγμα, αλλά αυτή τη φορά πιο πυκνό σε σχέση με αυτό που αναφέραμε παραπάνω. Στην περίπτωση αυτή τα κέντρα των διαδοχικών παραθύρων απέχουν μεταξύ τους απόσταση δύο pixel. Για τα ενδιάμεσα σημεία, αντί για bilinear interpolation αυτό που κάναμε ήταν να τους δώσουμε την τιμή που υπολογίσαμε για το πιο κοντινό σ' αυτά pixel.

Συγκρίνοντας τα σχήματα 4.9(β) και 4.9(γ), βλέπουμε ότι αν εξαιρέσει κανείς την σκιά πάνω στο καπέλο της κοπέλας, την οποία εντόπισε καλύτερα ο exhaustive αλγόριθμος, κατά τα άλλα οι δύο εικόνες είναι σχεδόν όμοιες. Παρατηρούμε δηλαδή, ότι η διαφορά στην απόδοση των δύο μεθόδων είναι μικρή, και ρίχνοντας μία ματιά και στους χρόνους εκτέλεσης για την κάθε περίπτωση (Πίνακας Δ) συμπεραίνουμε ότι γενικά δεν αξίζει τον κόπο μία εξαντλητική υλοποίηση. Από την άλλη, αυτό που ίσως θα άξιζε τον κόπο, είναι η υλοποίηση με το πυκνό πλέγμα, η οποία από άποψη αποτελέσματος είναι η καλύτερη καθώς εντοπίζει το κάτω χείλος της κοπέλας (βλ. σχήμα 4.9(δ)), ενώ

από άποψη υπολογιστικού χρόνου είναι κάτι μεταξύ των χρόνων που αναφέρθηκαν για την εξαντλητική υλοποίηση και την υλοποίηση με το αραιό πλέγμα. Οι χρόνοι εκτέλεσης για κάθε περίπτωση συνοψίζονται στον Πίνακα Δ.

Το τελικό συμπέρασμα όσων αφορά αυτό το κομμάτι της υλοποίησης, είναι ότι η υλοποίηση με αραιό πλέγμα είναι αυτή που συνδυάζει το καλό αποτέλεσμα με τον χαμηλό χρόνο εκτέλεσης, ενώ υπάρχουν και οι άλλες δύο λύσεις για κάτι καλύτερο, αλλά με κατά πολύ μεγαλύτερο υπολογιστικό κόστος.

(α) (β)

(γ) (δ)

Σχήμα 4.9 (α) Αρχική εικόνα. (β) Adaptive clustering (με αραιό πλέγμα). (γ) Adaptive clustering (exhaustive). (δ) Adaptive clustering (με πυκνό πλέγμα).

ΠΙΝΑΚΑΣ Δ

	Adaptive clustering (αραιό πλέγμα)	Adaptive clustering (exhaustive)	Adaptive clustering (πυκνό πλέγμα)
χρόνος εκτέλεσης	14	202	51

Εκτός από τις αλλαγές που δοκιμάσαμε παραπάνω, μελετήσαμε επίσης την επίδραση που είχε στο τελικό segmentation η μεταβολή κάποιων άλλων παραμέτρων του αλγορίθμου. Για παράδειγμα, υπενθυμίζουμε ότι n_{max} (βλ. σχήμα 4.4) είναι ο μέγιστος αριθμός των επαναλήψεων που επιτρέπουμε στον αλγόριθμο για ένα δεδομένο μέγεθος παραθύρου. Συνήθως n_{max}=10 είναι αρκετό, υπήρξαν όμως και περιπτώσεις στις οποίες σε 10 επαναλήψεις ο αλγόριθμος δεν είχε συγκλίνει, και έτσι αναγκαστήκαμε να περάσουμε στο επόμενο μέγεθος παραθύρου χωρίς να έχουμε σύγκλιση στο προηγούμενο επίπεδο. Ωστόσο, αυτό δεν φάνηκε να επηρεάζει το τελικό αποτέλεσμα, καθώς ήδη από τις δέκα πρώτες επαναλήψεις ο αλγόριθμος είχε ήδη "σχηματίσει άποψη" για τη μορφή του segmentation στο επίπεδο αυτό. Έτσι η τιμή 10 για το n_{max} θεωρήθηκε καλή για κάθε περίπτωση. Πληροφοριακά αναφέρουμε ότι στις περισσότερες περιπτώσεις η σύγκλιση είχε επιτευχθεί μέσα στις πρώτες 6-7 επαναλήψεις (ο αριθμός ποικίλει ανάλογα με το μέγεθος και το περιεχόμενο της εικόνας, κυρίως με το περιεχόμενο).

Κάτι ανάλογο με την παράμετρο n_{max} συμβαίνει και με την παράμετρο T_{min} , η οποία φαίνεται να μην έχει μεγάλη επίδραση τουλάχιστον στο βασικά χαρακτηριστικά της εικόνας. Εν πάσει περιπτώση, μία τιμή για το T_{min} κόντα στο μέγεθος του πλάτους του παραθύρου (όπως έχουμε ήδη αναφέρει σε προηγούμενη παράγραφο), αποτελεί μία καλή λύση. Πρέπει να τονίσουμε εδώ, ότι όταν λέμε ότι η μεταβολή μιας παραμέτρου δεν έχει μεγάλη επίδραση στο τελικό αποτέλεσμα, δε σημαίνει ότι όσο και να αλλάξουμε την παράμετρο το αποτέλεσμα θα παραμείνει το ίδιο. Εννοείται ότι αυτό ισχύει για μικρές μεταβολές της παραμέτρου, ενώ αν πρόκειται για ακραίες μεταβολές, τότε είναι πολύ φυσικό το αποτέλεσμα που θα πάρουμε να είναι κάθε άλλο παρά αποδεκτό.

Κάτι άλλο το οποίο εφαρμόσαμε, ήταν να μεταβάλλουμε την παράμετρο β (η οποία όπως έχουμε πει, καθορίζει το δυναμικό κλίκας δύο σημείων), την οποία είχαμε πει ότι θα θεωρούμε σταθερή και ίση με 0.5 για όλες τις επαναλήψεις. Αυτό που κάναμε, ήταν να διατηρούμε σταθερή την τιμή του β για όσο διάστημα παρέμενε σταθερό το μέγεθος του

παραθύρου και να την αυξάναμε κάθε φορά που μειωνόταν το μέγεθος του παραθύρου έτσι ώστε να πετυχαίνουμε πιο ομαλά (smooth) περιγράμματα. Μία αύξηση από 0.1 έως 0.2 κάθε φορά βρέθηκε ότι είναι καλή για τις περισσότερες περιπτώσεις αν και για μεγάλες εικόνες (που σημαίνει μεγαλύτερο αριθμό διαφορετικών παραθύρων) καλό είναι το βήμα αύξησης σταδιακά να μικραίνει καθώς υπάρχει ο κίνδυνος να εκλείψουν πολλές από τις λεπτομέρειες της εικόνας. Το αποτέλεσμα της υλοποίησης με μεταβλητό β ήταν κάτι ανάλογο με αυτό στην περίπτωση που το β παρέμενε σταθερό, με μόνη διαφορά ότι, τα περιγράμματα έγιναν πιο ομαλά ενώ μεμονωμένες περιοχές στην εικόνα εξαφανίστηκαν.

Όσων αφορά το δυναμικό κλίκας ενός σημείου, δηλαδή την παράμετρο α^i την οποία είχαμε θέσει ίση με μηδέν για κάθε i, δοκιμάσαμε να δώσουμε στα α^i τιμές διάφορες του μηδενός, ανάλογα με τη δικιά μας κρίση και κρίνοντας βέβαια από το γεγονός ότι η παράμετρος α^i η θα πρέπει να είναι μικρή αν η κυρίαρχη περιοχή είναι η i και μεγάλη σε κάθε άλλη περίπτωση. Στο σχήμα 4.10 φαίνεται το αποτέλεσμα για K=2, σ=16 και α^i=0.38 για τη μαύρη περιοχή την οποία θεωρούμε και ως κυρίαρχη, και α^i=0.62 για την άσπρη περιοχή. Από το σχήμα 4.10(γ), φαίνεται αμέσως, ότι κάποιες σκούρες περιοχές στο πάνω κομμάτι της κύστης έχουν ανιχνευθεί παρά την υψηλή τιμή του σ, κάτι το οποίο δεν έχει γίνει στην περίπτωση του σχήματος 4.10(β) στην οποία θεωρούμε ισοπίθανες περιοχές.

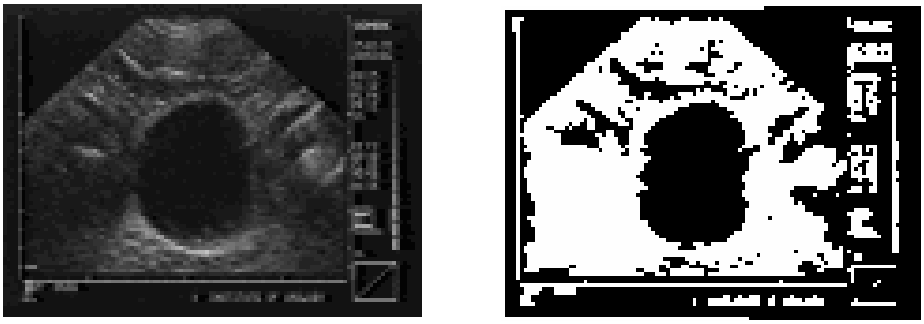
(α) (β) (γ)

Σχήμα 4.10 (α) Αρχική εικόνα. (β) α^i σταθερό(=0). (γ) διαφορετικό για την κάθε περιοχή(α^i=0.38 για την σκούρα περιοχή και α^i=0.62 για την ανοιχτή).

Στο σημείο αυτό, και μιας και μιλάμε για την παράμετρο α^i και πως αυτή επηρεάζει το segmentation, θα δείξουμε πως μπορούμε επιλέγοντας συγκεκριμένες τιμές για την

παράμετρο αυτή, να φέρουμε τον αλγόριθμο στα μέτρα μας και να πετύχουμε τον εντοπισμό περιοχών για τις οποίες πραγματικά ενδιαφερόμαστε.

Πιο συγκεκριμένα, αυτό που παρατηρούμε από τα προηγούμενα σχήματα για το segmentation που δίνει ο αλγόριθμος στην περίπτωση της εικόνας με την κύστη είναι, ότι ναι μεν προσαρμόζεται στα χαρακτηριστικά της εικόνας αλλά λόγω της φύσης της συγκεκριμένης εικόνας αυτό που πραγματικά μας ενδιαφέρει είναι η περιοχή της κύστης (στο κέντρο της εικόνας) την οποία και επιθυμούμε να διαχωρίσουμε από τις υπόλοιπες περιοχές. Στο σχήμα 4.11 βλέπουμε το segmentation που δίνει ο αλγόριθμος επιλέγοντες τιμές α^1=4 για τη μαύρη περιοχή και α^2=-4 για την άσπρη περιοχή. Οι επιλογή των τιμών αυτών στην ουσία δίνει προτεραιότητα στην άσπρη περιοχή την οποία θεωρεί και πιο πιθανή. Υπενθυμίζουμε ότι όσο πιο μικρή είναι η τιμή της παραμέτρου α^i για μία περιοχή i, τόσο πιο μεγάλη είναι η πιθανότητα για ένα pixel να ανήκει στην περιοχή i. Όπως φαίνεται από το σχήμα το αποτέλεσμα είναι ο επιτυχής εντοπισμός και διαχωρισμός της κύστης από τις γύρω σκοτεινές περιοχές.



(α) (β)

Σχήμα 4.11 Ανίχνευση της περιοχής της κύστης (β) στην αρχική εικόνα του υπερηχοτομογράφου (α).

Στην περίπτωση της Lena, ρίχνοντας μία ματιά στο σχήμα 4.9 και κυρίως στο 4.9(β) που αφορά την υλοποίηση με αραιό πλέγμα, παρατηρούμε ότι το καπέλο δεν διαχωρίζεται πλήρως από την περιοχή πάνω απ' αυτό αλλά σε κάποιο σημείο σμίγει με την περιοχή αυτή. Αυτό καταφέραμε να το βελτιώσουμε επεμβαίνοντας σε δύο σημεία του αλγορίθμου. Πρώτον θεωρήσαμε ότι δύο περιοχές της εικόνας δεν είναι ισοπίθανες

και συγκεκριμένα δώσαμε τιμή α^1=0.38 για τη μαύρη περιοχή και α^2=0.62 για την άσπρη. Δεύτερον αλλάξαμε τις αρχικές τιμές των gray levels των κέντρων των clusters που δίναμε αρχικά στον K-means. Συγκεκριμένα οι τιμές που δώσαμε ήταν οι 0 και 255. Το αποτέλεσμα φαίνεται στο σχήμα 4.12(β) στο οποίο και παρατηρούμε ότι η άσπρη περιοχή πάνω από το καπέλο της κοπέλας έχει πλεόν σχεδόν εξαλειφθεί και το καπέλο έχει πλήρως διαχωριστεί από την περιοχή με την οποία έως πριν ήταν ενωμένο. Επιπλεόν παρατηρούμε ότι και το κάτω χείλος της κοπέλας έχει επίσης ανιχνευθεί.

(α) (β)

Σχήμα 4.12 (α) Αρχική εικόνα. (β) Segmentation στο οποίο φαίνεται η εξάλειψη της άσπρης περοχής πάνω από το καπέλο της κοπέλας.

Όλες οι αλλαγές που αναφέρθηκαν παραπάνω, έγιναν στον adaptive αλγόριθμο που παρουσιάσαμε στην παράγραφο 4.3. Όσον αφορά τον ιεραρχικό αλγόριθμο που παρουσιάσαμε στη παράγραφο 4.5, αυτό στο οποίο επέμβαμε ήταν το είδος του βαθυπερατού φίλτρου το οποίο χρησιμοποιήσαμε για το πέρασμα από το ένα επίπεδο ανάλυσης στο επόμενο. Στα προηγούμενα παραδείγματα, το βαθυπερατό φίλτρο το οποίο χρησιμοποιήσαμε ήταν ένα (3x3) averaging φίλτρο. Στο κεφάλαιο 6, θα δούμε πως επηρεάζει η επιλογή του βαθυπερατού φίλτρου το τελικό αποτέλεσμα, και θα προτείνουμε ένα συγκεκριμένο φίλτρο, δανεισμένο από το πεδίο των wavelets, το οποίο φαίνεται να δίνει αρκετά ικανοποιητικά αποτελέσματα.

Στην παράγραφο αυτή, είδαμε πόσα διαφορετικά πράγματα μπορούμε να "πειράξουμε" στον adaptive αλγόριθμο, και να επηρεάσαμε έτσι την απόδοση του προς την κατεύθυνση την οποία εμείς θέλουμε. Σε κάθε περίπτωση τα αποτελέσματα είναι αρκετά ικανοποιητικά, σαφώς καλύτερα από αυτά του K-means αλγορίθμου, με μοναδικό κόστος την επιπλεόν πολυπλοκότητα της υλοποίησης (η οποία όμως παραμένει σε λογικά επίπεδα) και τον υπολογιστικό χρόνο, ο οποίος είναι βέβαια μεγαλύτερος, αλλά έχει προοπτικές μείωσης χρησιμοποιώντας βελτιστοποιήσεις, και επιπλεόν τη δυνατότητα (όπως έχουμε ήδη αναφέρει) της εκτέλεσης του αλγορίθμου από παράλληλη μηχανή. Όσων αφορά τώρα την χρησιμότητα των τελικών εικονών, αυτές μπορούν να χρησιμοποιηθούν για αναγνώριση (προσώπων κ.λ.π) και επιπλεόν μπορούν να κωδικοποιηθούν επαρκέστατα, καθώς το μόνο που έχει να γίνει είναι να κωδικοποιηθούν οι μεταβολές ανάμεσα σε πολύ λίγα επίπεδα(συνήθως 2-4) του γκρι.

ΠΑΡΑΡΤΗΜΑ

Ο αλγόριθμος K-means ανήκει στην κατηγορία των αλγορίθμων εκείνων που χρησιμοποιούνται για ταξινόμηση προτύπων (pattern classification). Βασίζεται στην ελαχιστοποίηση ενός δείκτη απόδοσης, ο οποίος ορίζεται σαν το άθροισμα των τετραγώνων των αποστάσεων όλων των σημείων ενός cluster από το κέντρο του cluster. Αναλυτικά τώρα ο αλγόριθμος αποτελείται από τα ακόλουθα βήματα:

Βήμα 1. Διάλεξε K αρχικά κέντρα των clusters, έστω $\mathbf{z_1(1), z_2(1),. . ., z_K(1).}$ Αυτά είναι αυθαίρετα και συνήθως επίλέγονται σαν τα πρώτα K δείγματα του δεδομένου συνόλου σημείων.

Βήμα 2. Στο κ-επαναληπτικό βήμα ταξινόμησε τα σημεία **{x}** στα K clusters, χρησιμοποιώντας τη σχέση,

$$\mathbf{x} \in \mathbf{S_j}(k) \quad \text{if} \quad \|\mathbf{x} - \mathbf{z_j}(k)\| < \|\mathbf{x} - \mathbf{z_i}(k)\| \qquad (\alpha.1)$$

για όλα τα i=1,2,. . .,K, $i \neq j$, όπου $S_j(k)$ συμβολίζει το σύνολο των σημείων των οποίων το κέντρο(του cluster) είναι το $z_j(k)$.

Βήμα 3. Από τα αποτελέσματα του βήματος 2 υπολόγισε τα νέα κέντρα $z_j(k+1)$ clusters, για j=1,2,. . .,K, έτσι ώστε το άθροισμα των τετραγώνων των αποστάσεων όλων σήμείων στο $S_j(k)$ από το κέντρο του cluster να ελαχιστοποιείται. Με άλλα λόγια το καινούριο κέντρο του cluster j υπολογίζεται έτσι ώστε ο ακόλουθος δείκτης απόδοσης

$$J_j = \sum_{x \in S_j(k)} \|x - z_j(k+1)\|^2, \qquad j = 1,2,...,K \qquad (\alpha.2)$$

να ελαχιστοποιείται. Το κέντρο $z_j(k+1)$ το οποίο ελαχιστοποιεί αυτό το δείκτη απόδοσης είναι απλά το κέντρο των σημείων του $S_j(k)$, το οποίο δίνεται από την ακόλουθη σχέση:

$$z_j(k+1) = \frac{1}{N} \sum_{x \in S_j(k)} x, \qquad j = 1,2,...,K \qquad (\alpha.3)$$

Βήμα 4. Αν $z_j(k) = z_j(k+1)$ για j=1,2,. . .,K, ο αλγόριθμος έχει συγκλίνει και η διαδικασία τερματίζει. Διαφορετικά πήγαινε ξανά στο βήμα 2.

Η συμπεριφορά του αλγόριθμου K-means επηρεάζεται από τον αριθμό των κέντρων των clusters που καθορίζεται, την επιλογή των αρχικών κέντρων των clusters και φυσικά τις γεωμετρικές ιδιότητες των δεδομένων. Αν και δεν υπάρχει γενική απόδειξη για την

σύγκλιση του αλγορίθμου, αναμένεται ότι θα δώσει καλά αποτελέσματα σε περιπτώσεις όπου τα δεδομένα μας επιδεικνύουν τάσεις ομαδοποίησης και είναι σχετικά μακριά μεταξύ τους (οι ομάδες). Πάντως πρακτικά αυτό που γίνεται είναι η δοκιμή διαφόρων τιμών του K και διαφόρων αρχικών κέντρων (cluster centers), έτσι ώστε να πάρουμε τα καλύτερα δυνατά αποτελέσματα.

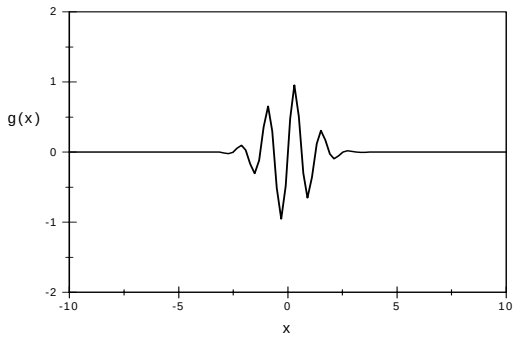
Όσων αφορά τώρα τον τρόπο με τον οποίο ο αλγόριθμος K-means εφαρμόστηκε στην περίπτωσή μας, αυτό που κάναμε ήταν να προσθέσουμε στα διανύσματα x και z άλλη μία συντεταγμένη έτσι ώστε αυτά από δισδιάστατα που ήταν να γίνουν πλέον τρισδιάστατα. Η τρίτη συντεταγμένη που προσθέσαμε ήταν φυσικά το gray-level του σημείου. Έτσι το κάθε δείγμα **x** χαρακτηρίζεται πλέον, εκτός από τις x και y συντεταγμένες του στο επίπεδο, και από το gray-level του σημείου το οποίο αντιπροσωπεύει. Έτσι ο αλγόριθμος εφαρμόζεται ακριβώς με τον τρόπο με τον οποίο περιγράφτηκε παραπάνω, με μόνη διαφορά ότι οι πράξεις που αντιπροσωπεύουν οι σχέσεις (α.1), (α.2) και (α.3) είναι τώρα διαφορετικές λόγω της επιπλέον διάτασης που προστέθηκε στα δείγματα μας προκειμένου να συμπεριληφθεί και το gray-level του δείγματος στους υπολογισμούς μας και να καταστεί έτσι δυνατή η εφαρμογή του αλγορίθμου K-means σε grayscale εικόνες.

ΚΕΦΑΛΑΙΟ 5: ΕΙΣΑΓΩΓΗ ΣΤΗ ΘΕΩΡΙΑ ΤΩΝ WAVELETS

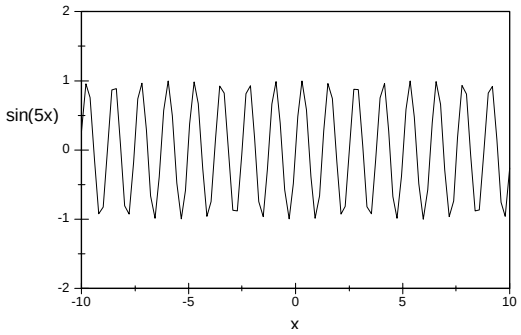
Στο κεφάλαιο αυτό παρουσιάζουμε κάποια βασικά κομμάτια της θεωρίας των Wavelets **[6]**. Σκοπός του κεφαλαίου αυτού δεν είναι να μία εξαντλητική ανάπτυξη και παρουσίαση της θεωρίας αυτής, αλλά μία απλή και κατανοητή εισαγωγή στα κομμάτια εκείνα της θεωρίας, τα οποία είναι απαραίτητα για την κατανόηση της εφαρμογής των Wavelets στην τμηματοποίηση εικόνας, που θα ακολουθήσει στο επόμενο κεφάλαιο.

5.1 ΕΙΣΑΓΩΓΗ

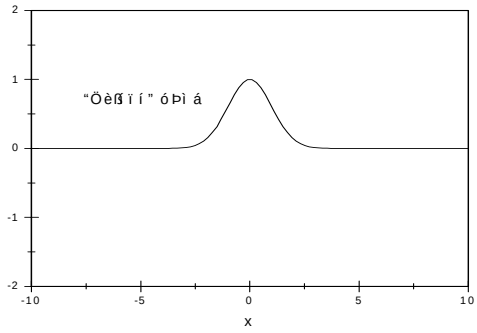
Η θεωρία των Wavelets, είναι τα μαθηματικά που σχετίζονται με την κατασκευή ενός μοντέλου για ένα σήμα, σύστημα ή διαδικασία χρησιμοποιώντας ένα σύνολο από "ειδικά" σήματα. Τα "ειδικά" αυτά σήματα είναι απλά μικρά κύματα (little waves) ή wavelets. Αυτά πρέπει να είναι κυματοειδή (waves) και να έχουν πλάτη τα οποία γρήγορα να πηγαίνουν στο μηδέν τόσο προς τη θετική όσο και προς την αρνητική κατεύθυνση. Στο σχήμα 5.1 φαίνεται ένα παράδειγμα ενός wavelet (στην ουσία πρόκειται για το κλασσικό σήμα wavelet το οποίο είναι γνωστό και σαν Morlet mother wavelet προς τιμή αυτού που το εφηύρε). Αυτό το σήμα στην ουσία κατασκευάζεται από δύο σήματα. Το πρώτο εξασφαλίζει την κυματοειδή μορφή του σήματος και είναι ένα ημίτονο (βλ. σχήμα 5.2). Το δεύτερο εξασφαλίζει την συνθήκη γρήγορης εξασθένισης εκτελώντας μία λειτουργία παραθύρου (βλ. σχήμα 5.3). Το γινόμενο των σημάτων των σχημάτων 5.2 και 5.3 μας δίνει το σήμα wavelet του σχήματος 5.1.



Σχήμα 5.1 Morlet Mother Wavelet.



Σχήμα 5.2 Κυματοειδές σήμα (ημίτονο).

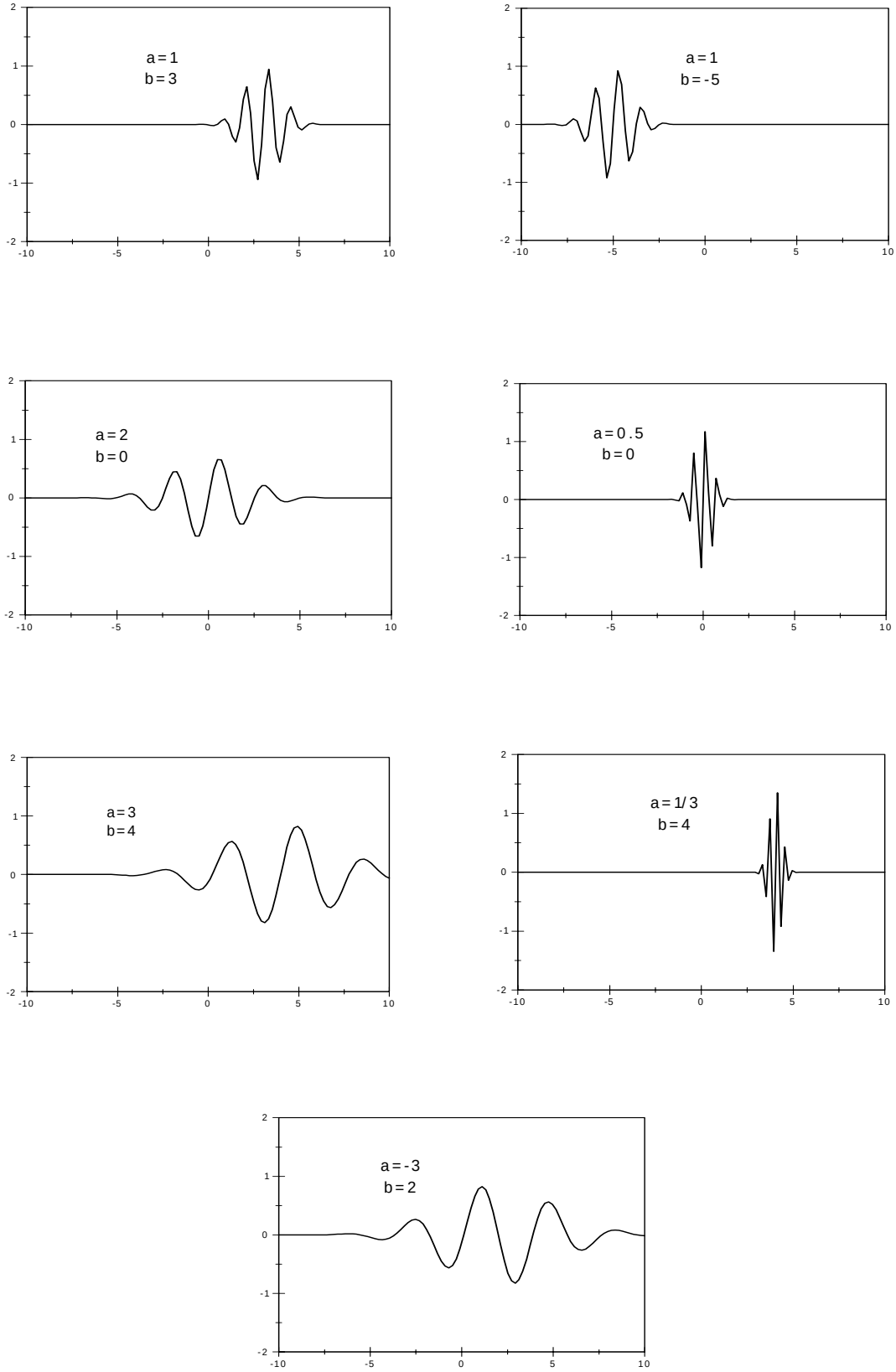


Σχήμα 5.3 Φθίνον σήμα.

Με βάση τώρα το σήμα mother wavelet του σχήματος 1, μπορούμε να κατασκεύασουμε ένα σύνολο από wavelet σήματα, απλά μετατοπίζοντας (shift) ή κλιμακώνοντας (scale) το αρχικό σήμα mother wavelet. Έτσι αν το αρχικό σήμα είναι το g(x) τότε το μετατοπισμένο η κλιμακώμενο σήμα θα έχει τη μορφή

$$\frac{1}{\sqrt{a}} g\left(\frac{t-b}{a}\right)$$

όπου ο όρος $\frac{1}{\sqrt{a}}$ είναι μία σταθερά κανονικοποίησης που χρησιμοποιείται για να διατηρεί σταθερή την ενέργεια των νεών σημάτων και ίση με αυτήν της αρχικής mother wavelet. Στο σχήμα 5.4 φαίνονται μερικά στοιχεία ενός τέτοιου συνόλου.



Σχήμα 5.4 Μετατοπισμένες και κλιμακώμενες Mother Wavelets.

5.2 Ο ΜΕΤΑΣΧΗΜΑΤΙΣΜΟΣ WAVELET

5.2.1 Συνεχής μετασχηματισμός

Στην παράγραφο αυτή θα ορίσουμε τόσο τον συνεχή όσο και το διακριτό μετασχηματισμό wavelet. Θα ορίσουμε επίσης και τον αντίστροφο wavelet μετασχηματισμό ο οποίος είναι που στην ουσία συμπληρώνει τον ευθύ και μας επιτρέπει την επιστροφή μας από το πεδίο του wavelet πίσω στο πεδίο από το οποίο είχαμε ξεκινήσει. Ξεκινώντας λοιπόν, ορίζουμε τον μετασχηματισμό wavelet (wavelet transform) μιας συνάρτησης, f, ως προς μία αποδεκτή mother wavelet, g, ως εξής:

$$\begin{aligned} W_g f(a,b) &= |a|^{-\frac{1}{2}} \int f(x)\, g^*\left(\frac{x-b}{a}\right) dx \\ &= \left\langle f, \frac{1}{\sqrt{|a|}} g\left(\frac{x-b}{a}\right) \right\rangle = < f, g_{a,b} > \end{aligned} \tag{5.1}$$

όπου το "*" στο g υποδηλώνει το μιγαδικό συζυγή του g, και το $< \cdot , \cdot >$ συμβολίζει το εσωτερικό γινόμενο. Η $g_{a,b}$ είναι μία μία έκδοση της mother wavelet, η οποία έχει υποστεί κλιμάκωση κατά ένα παράγοντα a (scale parameter) και μετατόπιση κατά b (translation parameter). Ο μετασχηματισμός wavelet δηλαδή, αυτό που κάνει είναι να απεικονίσει ένα σήμα πεπερασμένης ενέργειας από το πεδίο του χρόνου ή του χώρου, σε ένα δισδιάστατο πεδίο μετατόπισης-κλιμάκωσης (scale-translation domain) ή αλλιώς πεδίο wavelet (wavelet domain).

Κατά ανάλογο τρόπο, ορίζουμε και τον αντίστροφο μετασχηματισμό wavelet (inverse wavelet transform). Στην περίπτωση αυτή, αν $W_g f(a,b)$ είναι ο μετασχηματισμός wavelet μιας συνάρτησης f ως προς μία mother wavelet g, τότε, ισχύει:

$$f(x) = \frac{1}{c_g} \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} W_g f(a,b) \frac{1}{\sqrt{|a|}} g\left(\frac{x-b}{a}\right) \frac{db\, da}{a^2} \tag{5.2}$$

Με τον αντίστροφο μετασχηματισμό wavelet έχουμε την απεικόνιση

$$W_g^{-1} : H(a,b) \mapsto p(x) \tag{5.3}$$

μίας επιφάνειας H(a,b) του δισδιάστατου χώρου κλιμάκωσης-μεταφοράς, στο μονοσδιάστατο πεδίο του χώρου ή του χρόνου.

Ο ευθής μετασχηματισμός wavelet είναι μοναδικός (δηλαδή για κάθε σήμα υπάρχει μόνο ένας μετασχηματισμός wavelet) σε αντίθεση με τον αντίστροφο μετασχηματισμό κατά τον οποίο είναι δυνατό διαφορετικές αναπαραστάσεις στο πεδίο του wavelet, να οδηγούν στο ίδιο σήμα στο πεδίο του χρόνου ή του χώρου. Πάντως, η διαδικασία της εφαρμογής του ευθύ μετασχηματισμού σε μία συνάρτηση, ακολουθούμενη από την εφαρμογή του αντίστροφου μετασχηματισμού, οδηγεί στην αρχική συνάρτηση. Δηλαδή ενώ με τον ευθή μετασχηματισμό περνάμε στο πεδίο του wavelet, ο αντίστροφος μετασχηματισμός είναι αυτός που μας οδηγεί πίσω στο αρχικό πεδίο και την ανακατασκεύη του αρχικού σήματος.

Ο ευθής μετασχηματισμός παίζει το ρόλο ενός φίλτρου ανάλυσης (analysis filter). Σπάει δηλαδή το σήμα μας σε "κομμάτια", και αυτά τα κομμάτια είναι που εξετάζουμε αντί για την αρχική συνάρτηση. Ο αντίστροφος μετασχηματισμός είναι στην ουσία ένα φίλτρο ανακατασκεύης (reconstruction filter) ή σύνθεσης (synthesis filter). Το φίλτρο σύνθεσης είναι αυτό που βάζει ξανά τα κομμάτια πίσω στις αρχικές τους θέσεις και οδηγεί έτσι στην ανακατασκεύη του αρχικού σήματος.

5.2.2 Διακριτός μετασχηματισμός

Κατά αναλογία με το συνεχή μετασχηματισμό, μία σειρά από σημεία (αριθμούς) μπορεί επίσης να αναπαρασταθεί με ένα μετασχηματισμό wavelet και αυτός ο μετασχηματισμός θα ονομάζεται τώρα "σειρά wavelet διακριτού χρόνου" (discrete time wavelet series DTWS). Ο DTWS είναι ανάλογος με τη fourier σειρά διακριτού χρόνου, καθώς, και οι δύο μετασχηματισμοί είναι διακριτοί και ως προς την ανεξάρτητη μεταβλητή και ως προς τη μεταβλητή(ες) μετασχηματισμού. Ο DTWS ορίζεται τώρα ως προς μία διακριτή mother wavelet, h(k), ως εξής:

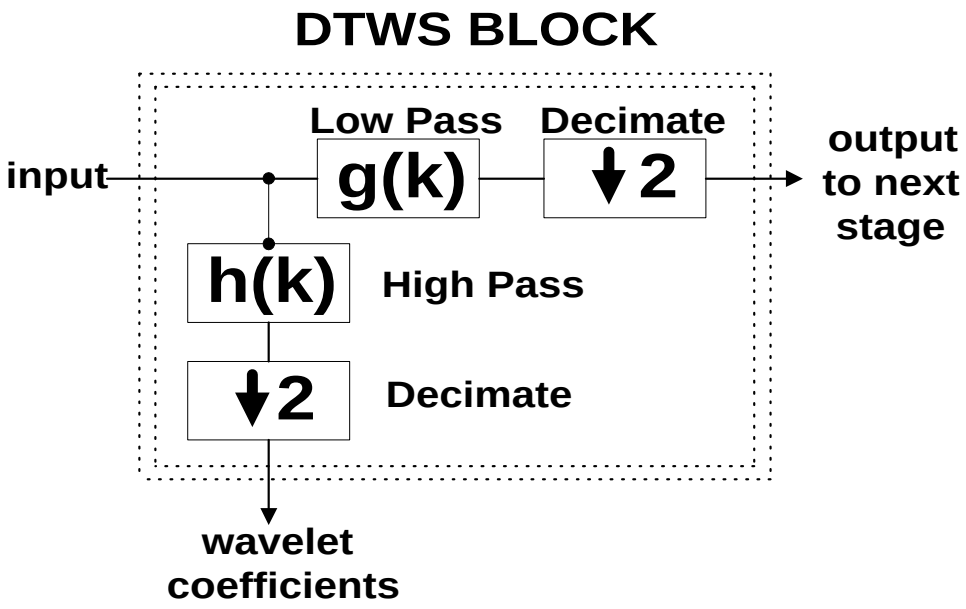
$$W_h f(m,n) = \frac{1}{\sqrt{a_0^m}} \sum_{k=-\infty}^{\infty} f(k)\, h\left(\frac{k - nb_0 a_0^m}{a_0^m}\right) = a_0^{\frac{-m}{2}} \sum_{k=-\infty}^{\infty} f(k)\, h(a_0^{-m}k - nb_0) = < f, h_{m,n} > \qquad (5.4)$$

Κατά αναλογία με τον ευθή διακριτό μετασχηματισμό, ορίζεται και ο ανίστροφος διακριτός μετασχηματισμός wavelet (IDTWS), ο οποίος μας γυρίζει πίσω στο πεδίο του χώρου ή του χρόνου.

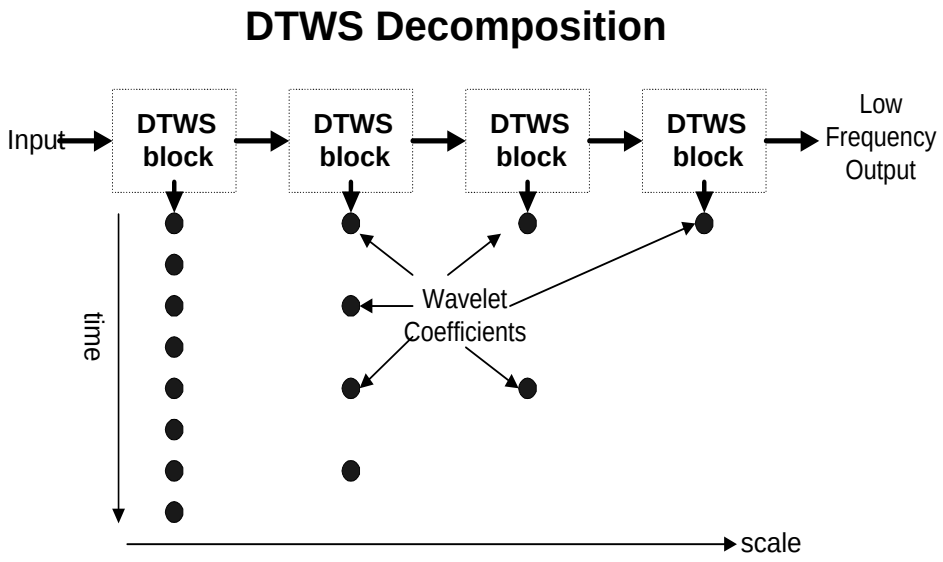
Έως τώρα απλά ορίσαμε τους μετασχηματισμούς wavelet (ευθείς, αντίστροφους, συνεχείς και διακριτούς) οι οποίοι έχουν κάποιες ομοιότητες και αναλογίες με τους αντίστοιχους μετασχηματισμούς Fourier αλλά και κάποιες ιδιαιτερότητες, που αφορούν κυρίως την επιλογή της mother wavelet συνάρτησης και που προς το παρόν δε θα μας απασχολήσουν. Στην επόμενη παράγραφο θα δείξουμε μία συγκεκριμένη δομή του διακριτού μετασχηματισμού, το Discrete Time Wavelet Series Block, το οποίο και θα χρησιμοποιήσουμε για να παρουσιάσουμε τον πολλαπλής ανάλυσης μετασχηματισμό wavelet (multiresolution wavelet transform). Στο κεφάλαιο 6, αυτό που θα μας απασχολήσει θα είναι μία παραλλαγή του multiresolution wavelet transform και η εφαρμογή του τόσο στην ανακατασκεύη όσο και στην τμηματοποίηση της εικόνας.

5.2.3 Το discrete time wavelet series block (DTWS block)

Ο DTWS είναι διακριτός τόσο στο πεδίο του χρόνου όσο και στο πεδίο της κλιμάκωσης-μεταφοράς (wavelet domain). Από τη στιγμή που η διαστολή του χρόνου κατά ένα παράγοντα 2 μπορεί πολύ εύκολα να υλοποιηθεί, απλά με το να δειγματοληπτούμε κάθε δεύτερο δείγμα ένα διακριτό σήμα (η διαδικάσία αυτή λέγεται και decimation by a factor of 2), ο μετασχηματισμός που θα παρουσιάσουμε εδώ θεωρεί κλμάκωση (scaling) κατά ένα παράγοντα 2. Η δομή του DTWS block φαίνεται στο σχήμα 5.5. Τα φίλτρα που που χρησιμοποιούνται συμβολίζονται με g(k) το low pass και με h(k) το high pass, όμως είναι και τα δύο πολύ στενά συνδεδεμένα με την mother wavelet function. Συνήθως το high pass φίλτρο θεωρείται να είναι η mother wavelet (με πιθανή αλλαγή στη σειρά των συντελεστών) και οι εξόδοι της high pass διαδικασίας είναι συνεπώς οι συντελεστές wavelet. Δηλαδή, το high pass φίλτρο συνελίσει την κρουστική του απόκριση (mother wavelet) με το εισερχόμενο σήμα για να δημιουργήσει την έξοδο (τη σειρά των συντελεστών wavelet). Αυτή η δομή μπορεί να χρησιμοποιηθεί για να παράγουμε τον πολλαπλής ανάλυσης (multiresolution) wavelet μετασχηματισμό. Αν παραθέσουμε πολλά DTWS blocks σε σειρά τότε παίρνουμε τη διάταξη του σχήματος 5.6.



Σχήμα 5.5 Discrete time wavelet series block.



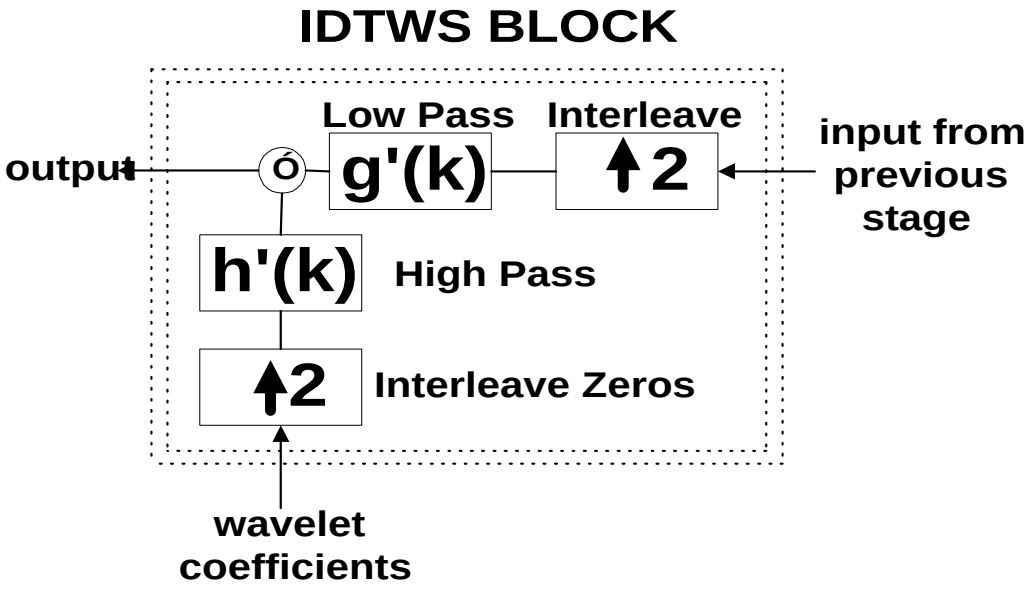
Σχήμα 5.6 Η διαδικασία της αποσύνθεσης του αρχικού σήματος.

Όπως φαίνεται από το σχήμα, το αρχικό σήμα τροφοδοτεί την πρώτη από αριστερά διάταξη και η έξοδος είναι οι συντελεστές wavelet και ένα χαμηλής συχνότητας σήμα το οποίο τροφοδοτεί τη δεύτερη διάταξη κ.ο.κ. Η διάταξη αυτή οδηγεί σε μία αποσύνθεση του σήματος εισόδου, το οποίο πλέον αναπαρίσταται με τους συντελεστές wavelet (όλων

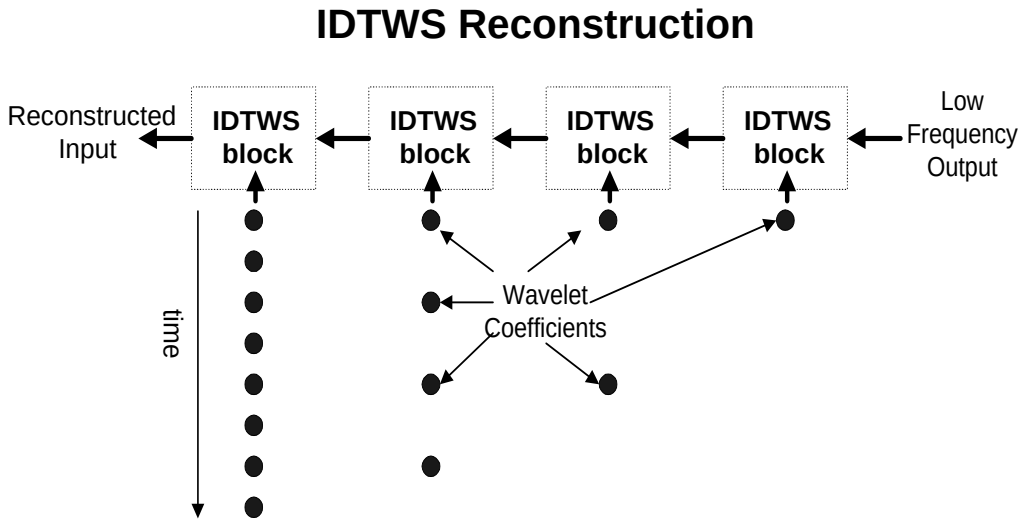
των επιπέδων) και την τελική έξοδο χαμηλής συχνότητας (low frequency output). Η πυραμιδοειδής δομή του σχήματος 5.6, οφείλεται στο ότι όλο και λιγότεροι συντελεστές wavelet εξάγονται από κάθε στάδιο καθώς προχωράει προς τα δεξιά η διαδικασία, έως τον τελευταίο συντελεστή που εξάγεται από το τελευταίο στάδιο (από την κορυφή της πυραμίδας).

Όσον αφορά την αντίστροφη διαδικασία, η οποία θα ανακατασκευάζει το αρχικό σήμα, χρησιμοποιούμε πολλά IDTWS blocks σε σειρά. Το IDTWS block και η διαδικασία της ανακατασκεύης πλέον, φαίνονται στα σχήματα 5.7 και 5.8 αντίστοιχα.

Όπως φαίνεται από το σχήμα 5.7, το IDTWS block έχει την ίδια ακριβώς δομή με το DTWS block, μόνο που λειτουργεί ακριβώς αντίθετα. Το IDTWS παίρνει σαν είσοδο την σειρά χαμηλής συχνότητας (low frequency sequence) και τους συντελεστές wavelet (που είναι στην ουσία η high pass σειρά) και ανακατασκεύαζει το αρχικό σήμα διακριτού χρόνου. Από το σχήμα παρατηρούμε ότι η διαδικάσία του decimation έχει τώρα αντικατασταθεί από μία διαδικασία παρεμβολής μηδενικών (ένα μηδενικό κάθε ένα δείγμα). Επίσης, τα low και high pass φίλτρα που χρησιμοποιούμε εδώ δεν είναι τα ίδια με αυτά του DTWS (βλ. σχήμα 5.5). Στο επόμενο κεφάλαιο θα αναφέρουμε περισσότερα για τα φίλτρα που χρησιμοποιούμε τόσο για το DTWS όσο και για το IDTWS και θα δούμε πως σχετίζονται αυτά τα φίλτρα μεταξύ τους.



Σχήμα 5.7 Inverse discrete time wavelet series block.



Σχήμα 5.8 Η διαδικασία της ανακατασκεύης του αρχικού σήματος.

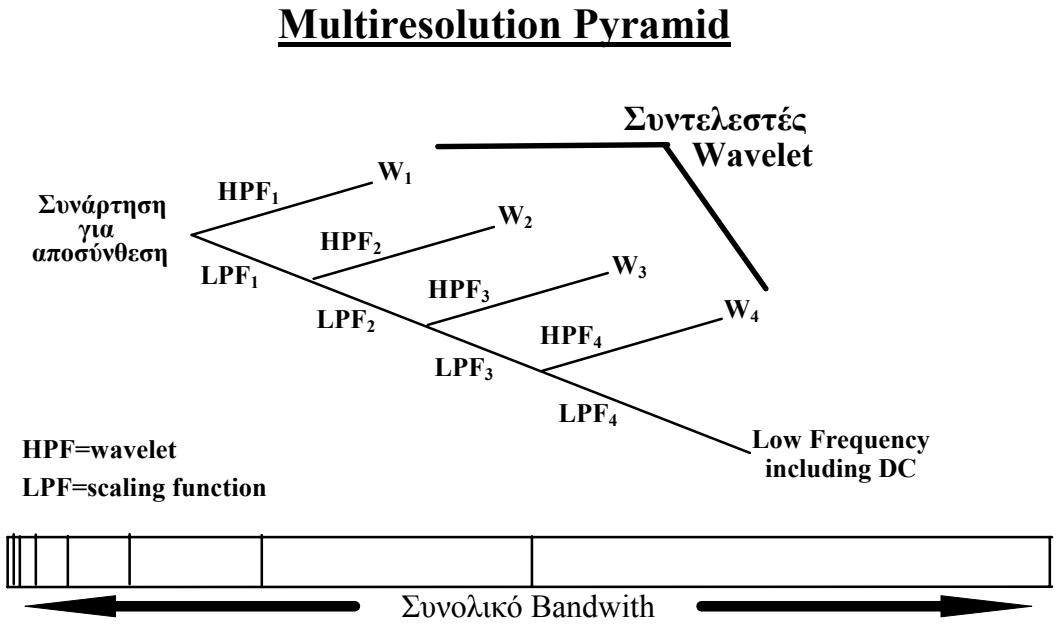
5.2.4 Ο πολλαπλής ανάλυσης μετασχηματισμός wavelet (multiresolution wavelet trasform).

Από τους διάφορους μετασχηματισμούς wavelet που υπάρχουν, ο πολλαπλής ανάλυσης είναι ο πιο γενικός και αυτός που μας αφορά καθώς έχει εφαρμογές στο πεδίο της εικόνας. Η λειτουργία του μοιάζει με αυτή του σχήματος 5.6. Ο πολλαπλής ανάλυσης μετασχηματισμός ενσωματώνει μία πυραμιδοειδή δομή στη λειτουργία του. Η δομή αυτή απαιτεί την επαναληπτική εφαρμογή των ίδιων low-pass και high-pass φίλτρων με τη διαφορά ότι αυτά κλιμακώνονται επί δύο σε κάθε επίπεδο ανάλυσης.

Η κύρια εφαρμογή του multiresolution wavelet transform έχει να κάνει με τη δημιουργία φίλτρων μισής μπάντας (half band filters) τα οποία διαιρούν το φάσμα του σήματος σε ένα κομμάτι υψηλής συχνότητας (high frequency band) και σε ένα κομμάτι χαμηλής συχνότητας (low frequency band). Αυτά τα φίλτρα αρχικά δρουν πάνω στο συνολικό bandwith του σήματος και το χωρίζουν σε low frequenvy band και high frequency band, και στη συνέχεια σταδιακά μειώνουν το bandwith του σήματος καθώς προχωράμε σε μεγαλύτερες τιμές κλιμάκωσης (μικρά bandwith αντιστοιχούν σε μεγάλες τιμές της παραμέτρου κλιμάκωσης). Η όλη διαδικασία, στο πεδίο της συχνότητας αυτή τη φορά, φαίνετα στο σχήμα 5.9. Η μπάντα υψηλής συχνότητας που παίρνουμε στην περίπτωση της υψηλότερης ανάλυσης είναι στην ουσία οι συντελεστές wavelet τους οποίους και διατηρούμε ως έχουν, ενώ την μπάντα χαμηλής συχνότητας την

αποδεκατίζουμε (decimate) κατά ένα παράγοντα 2 (στην ουσία "πετάμε" κάθε δεύτερο δείγμα). Στη συνέχεια τη μπάντα αυτή τη διασπάμε (με τη βοήθεια των φίλτρων βέβαια) σε μία χαμηλής συχνότητας μπάντα και μία υψηλής συχνότητας μπάντα. Αυτή η διαδικασία της αποδεκάτισης και του χωρισμού της μπάντας επαναλαμβάνεται και μας δίνει τελικά μία οκταδική (octave) αναπαράσταση μπάντας του αρχικού σήματος, όπως άλλωστε φαίνεται και από το σχήμα 5.9.

Οι συντελεστές wavelet για μία συγκεκριμένη ανάλυση (για ένα συγκεκριμένο βαθμό κλιμάκωσης) είναι τα δείγματα εξόδου του αντίστοιχου high-pass φίλτρου. Κάθε δείγμα αντιστοιχεί και σε ένα διαφορετικό συντελεστή μεταφοράς, στον συγκεκριμένο βαθμό κλιμάκωσης. Οι συντελεστές wavelet (δηλαδή οι έξοδοι των high-pass φίλτρων), αναπαριστούν τα χαρακτηριστικά του σήματος και την ενέργεια του σε κάθε είπεδο ανάλυσης (scaling). Η έξοδος του τελευταίου low-pass φίλτρου αποτελεί την d.c συνιστώσα του σήματος και συμβολίζει μία θολή και χοντροκομμένη έκδοση του αρχικού σήματος.



Σχήμα 5.9 Η δομή του πολλαπλής ανάλυσης μετασχηματισμού wavelet.

5.3 ΣΥΝΟΨΗ

Στις προηγούμενες παραγράφους παρουσιάσαμε κάποια στοιχεία από τη θεωρία των wavelets και είδαμε κάποιους βασικούς μετασχηματισμούς στο πεδίο αυτό. Από τους μετασχηματισμούς αυτούς, μεγάλο ενδιαφέρον παρουσιάζει ο πολλαπλής ανάλυσης wavelet μετασχηματισμός καθώς ένα από τα πεδία στα οποία εφαρμόζεται είναι και αυτό των εικόνων. Τη γενική μορφή του μετασχηματισμού αυτού είδαμε στην παράγραφο 5.2.4. Ωστόσο, η μορφή που παρουσιάσαμε στην παράγραφο αυτή αφορά διακριτά σήματα μίας διάστασης και όχι δύο διαστάσεων όπως μπορούν να θεωρηθούν οι εικόνες. Αν και ένα διακριτό σήμα δύο διαστασεων (όπως είναι μία εικόνα) μπορούμε να το αναγάγουμε σε σήμα μίας διάστασης και να το μελετήσουμε κανονικά όπως μελετάμε τα σήματα μίας διάστασης (οπότε θα μπορούμε να του εφαρμόσουμε τον μετασχηματισμό της 5.2.4), στο επόμενο κεφάλαιο θα δείξουμε πως μπορεί να εφαρμοστεί ο πολλαπλής ανάλυσης μετασχηματισμός σε μία εικόνα (διακριτό σήμα δύο διαστάσεων) και πως με τη βοήθεια του αντίστροφου μετασχηματισμού μπορούμε επακριβώς να ανακτήσουμε την αρχική εικόνα. Τέλος, οι ιδιότητες του μετασχηματισμού αυτού θα συνδυαστούν με την αποτελεσματικότητα του αλγορίθμου που παρουσιάσαμε στο Κεφάλαιο 4 για Image Segmentation και θα εξεταστεί η απόδοση της νέας μεθόδου.

ΚΕΦΑΛΑΙΟ 6: ΕΦΑΡΜΟΓΗ ΤΩΝ WAVELETS ΣΤΗΝ ΕΠΕΞΕΡΓΑΣΙΑ ΕΙΚΟΝΑΣ

Στο κεφάλαιο αυτό, όπως άλλωστε είχαμε αναφέρει και σε προηγούμενα κεφάλαια, θα μελετήσουμε την εφαρμογή του πεδίου των wavelets στην επεξεργασία εικόνας και πιο συγκεκριμένα στην τμηματοποίηση της εικόνας (Image Segmentation). Το κεφάλαιο χωρίζεται σε τέσσερα κομμάτια. Στο πρώτο, γίνεται μία αναφορά στην ανακατασκευή της εικόνας με χρήση του πεδίου των wavelets και παρουσιάζεται τόσο ο ευθύς όσο και ο αντίστροφος μετασχηματισμός που χρησιμοποιήθηκε. Επίσης, παρουσιάζουμε και δύο συγκεκριμένα wavelet φίλτρα τα οποία θα χρησιμοποιήσουμε στα επόμενα δύο κομμάτια. Στο δεύτερο, δοκιμάζουμε τη εφαρμογή των φίλτρων wavelet (που αναφέραμε παραπάνω) στον ιεραρχικό αλγόριθμο, που παρουσιάσαμε στο Κεφάλαιο 4, και συγκρίνουμε το αποτέλεσμα με αυτό της εφαρμογής άλλων φίλτρων. Στο τρίτο, εφαρμόζουμε τον adaptive αλγόριθμο του Κεφαλαίου 4 στο πεδίο των wavelets, και μετά γυρνώντας πίσω στο αρχικό πεδίο μελετάμε το τελικό αποτέλεσμα. Στο τελευταίο κομμάτι του κεφαλαίου παρουσιάζουμε τα τελικά συμπεράσματα από την όλη δουλειά που έγινε και κάποιες προτάσεις για μελλοντική εξέλιξη της εργασίας.

6.1 ΑΝΑΚΑΤΑΣΚΕΥΗ ΕΙΚΟΝΑΣ ΣΤΟ ΠΕΔΙΟ ΤΩΝ WAVELETS

6.1.1 Ανακατασκεύη εικόνας.

Το πεδίο των wavelets βρίσκει μεγάλη εφαρμογή στην ανακατασκευή της εικόνας (Image Restoration). Οι ιδιότητες των wavelets φαίνεται να δίνουν πολύ καλά αποτελέσματα όταν εφαρμόζονται στην ανακατασκεύη εικόνων αρκετά αλλοιωμένων από θόρυβο ή άλλους παράγοντες. Για μία αναλυτική παρουσίαση μεθόδων που έχουν προταθεί για ανακατασκεύη εικόνας με χρήση των wavelets βλ. **[16],[18]**. Εμείς, από το πεδίο της ανακατασκευής εικόνας με τη χρήση των wavelets, θα δανειστούμε μόνο το θεωρητικό υπόβαθρο το οποίο είναι απαραίτητο για τη μελέτη μας (κυρίως το μετασχηματισμό wavelet για τη μεταφορά της εικόνας μας στο πεδίο του wavelet, καθώς και τον αντίστροφο μετασχηματισμό wavelet για την μεταφορά μας πίσω στο αρχικό

πεδίο) καθώς επίσης και κάποια συγκεκριμένα φίλτρα wavelet τα οποία και θα χρησιμοποιήσουμε παρακάτω. Χρησιμοποιώντας τα παραπάνω στοιχεία, η προσπάθεια μας θα επικεντρωθεί στην εφαρμογή της τμηματοποίησης στο πεδίο του wavelet ή αντίστροφα στην εφαρμογή των wavelets στην τμηματοποίηση της εικόνας.

6.1.2 Τα low-pass και high-pass wavelet φίλτρα.

Στην παράγραφο αυτή παρουσιάζουμε ένα σύνολο από wavelet συντελεστές το οποίο αποτελεί τη βάση των φίλτρων (low-pass, high-pass) αποσύνθεσης που θα χρησιμοποιηθούν για τη μεταφορά της εικόνας στο πεδίο των wavelets **[16],[7]**. Το σύνολο αυτό των συντελεστών παράγεται από μία least asymmetric scaling function **[7]** η οποία σχεδιάζεται χρησιμοποιώντας τους ακόλουθους συντελεστές:

$$\begin{bmatrix} h_0 \\ h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \end{bmatrix} = \sqrt{2} \begin{bmatrix} -0.0802861503271 \\ -0.024308596967 \\ 0.362806341592 \\ 0.550576616156 \\ 0.229036357075 \\ -0.0644368523121 \\ -0.0115565483406 \\ 0.0381688330633 \end{bmatrix} \tag{6.1}$$

Ο συντελεστής κανονικοποίησης $\sqrt{2}$ χρησιμοποιείται για να διατηρήσει σταθερή την ενέργεια του σήματος μετά την επιστροφή από το πεδίο του wavelet και να μην έχουμε έτσι αλλοίωση του τελικού αποτελέσματος. Έτσι, χρησιμοποιώντας αυτούς τους συντελεστές, ορίζουμε τα low-pass και high-pass φίλτρα να έχουν τους ακόλουθους συντελεστές:

$$q_n = h_{-n} , \tag{6.2}$$

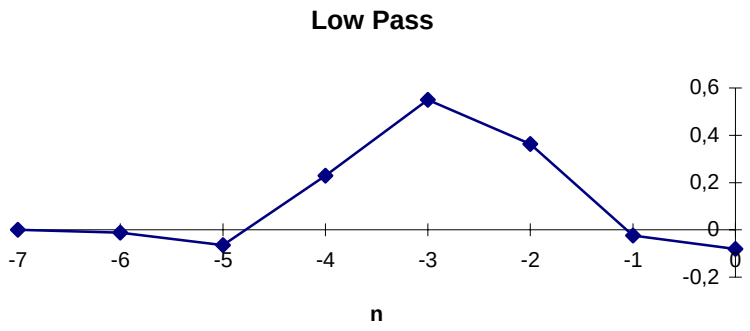
και

$$r_n = (-1)^n h_{n+1} , \tag{6.3}$$

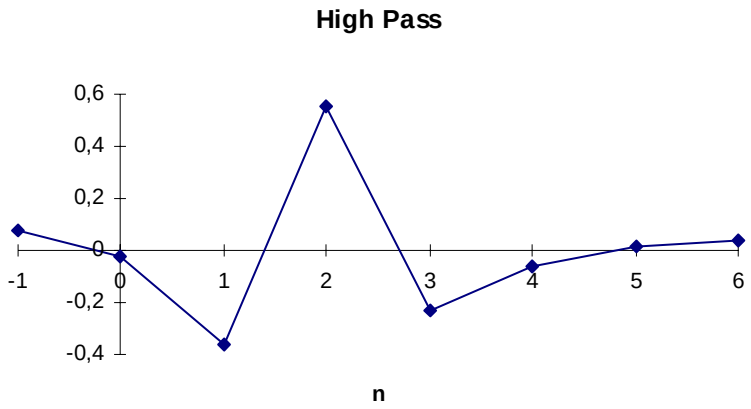
για το low-pass και το high-pass φίλτρο αντίστοιχα. Τα φίλτρα αυτά, των οποίων η μορφή φαίνεται στο σχήμα 6.1, οδηγούν σε καλό φιλτράρισμα της εικόνας καθώς έχουν

απότομες μεταβάσεις και διατηρούν έτσι τα χαρακτηριστικά της εικόνας και την οξύτητα των ακμών.

Παρατηρώντας το σχήμα 6.1, διαπιστώνουμε το λόγο για τον οποίο τα φίλτρα που παρουσιάσαμε ανήκουν στο πεδίο του wavelet. Όπως φαίνεται καθαρά από το σχήμα, η μορφή της κρουστικής απόκρισης του κάθε φίλτρου αντιπροσωπεύει στην ουσία ένα wavelet σήμα. (βλ. Κεφάλαιο 5, σχ. 5.1). Επίσης παρατηρούμε τη διαφορά ανάμεσα στην κρουστική απόκριση του low-pass φίλτρου και αυτή του high-pass φίλτρου. Είναι προφανές, ότι η προσθήκη του παράγοντα $(-1)^n$ στην (6.3), έχει σκοπό την γρηγορότερη εναλλαγή των συντελεστών του high-pass φίλτρου, προσδίδοντας του έτσι ιδιότητες υψιπερατού φίλτρου. Διαισθητικά, καταλαβαίνουμε ότι η απότομες μεταβολές του high-pass φίλτρου είναι αυτές που ανιχνεύουν τις ακμές της εικόνας, σε αντίθεση με το low-pass φίλτρο, οι αργές μεταβολές του οποίου έχουν σαν σκοπό την ολοκλήρωση των τιμών στην περιοχή του φίλτρου και την εξασθένιση(όχι όμως δραματική) των απότομων μεταβολών της εικόνας (ακμές). Αναφερόμενοι στο Κεφάλαιο 5, αν θεωρήσουμε ότι το high-pass φίλτρο (βλ. σχ. 6.1(β)) είναι η mother wavelet τότε το low-pass φίλτρο (βλ. σχ. 6.1(α)) είναι μία "scaling" έκδοση της mother wavelet(βλ. παρ. 5.2.3). Είναι δηλαδή σαν να έχουμε πιάσει από τα άκρα το σχήμα 6.1(β) και να το έχουμε τραβήξει προς τα έξω για να απλώσει. Αν και αυτός δεν είναι ακριβώς ο τρόπος με τον οποίο προκύπτει το low-pass φίλτρο, τουλάχιστο μας βοηθαέι διαισθητικά να καταλάβουμε την έννοια της κλιμάκωσης (scaling) της mother wavelet (high-pass φίλτρο ή "wavelet function" όπως αποκαλείται στην βιβλιογραφία του wavelet) για την παραγωγή του low-pass φίλτρου (που αποκαλείται και "scaling function").



(α)



(β)

Σχήμα 6.1 Η κρουστική απόκριση του low-pass φίλτρου **(α)** και του high-pass φίλτρου **(β)**.

6.1.3 Μετασχηματισμός εικόνας στο πεδίο του wavelet.

Στην παράγραφο αυτή θα παρουσιάσουμε τον μετασχηματισμό [16] τον οποίο χρησιμοποιήσαμε για να μεταφέρουμε την εικόνα μας στο πεδίο του wavelet και στη συνέχεια θα δείξουμε και τον αντίστροφο μετασχηματισμό, με τον οποίο, από το πεδίο του wavelet γυρίζουμε πίσω στο αρχικό πεδίο και ανακτούμε την εικόνα μας. Από δω και στο εξής, όπου αναφέρουμε για ευθύ και αντίστροφο μετασχηματισμό wavelet θα εννοούμε τους μετασχηματισμούς που θα παρουσιάσουμε αμέσως παρακάτω.

Στο προηγούμενο κεφάλαιο (βλ. παρ. 5.2.3, 5.2.4, και σχ. 5.5-5.9), παρουσιάσαμε τόσο την αποσύνθεση ενός μονοδιάστατου σήματος χρησιμοποιώντας DTWS blocks, όσο και την ανακατασκεύη (σύνθεση) του σήματος χρησιμοποιώντας IDTWS blocks. Αυτή άλλωστε η διαδικασία ήταν και μία μορφή μετασχματισμού πολλαπλής ανάλυσης μονοδιάστατου σήματος. Στην περίπτωση τώρα της εικόνας, που θεωρείται σήμα δύο διαστάσεων, ένας τρόπος είναι να μετατρέψουμε το δισδιάστατο σήμα(πίνακα) σε μονοδιάστατο (διάνυσμα), απλά παρατάσοντας τις γραμμές του πίνακα τη μία μετά την άλλη. Αυτό λέγεται και λεξικογραφική αναπαράσταση (lexicographic ordering) του πίνακα. Με μαθηματικούς όρους, αν θεωρήσουμε ότι η εικόνα μας(πίνακας) είναι η

$$X = \{x(m,n)\}$$

με διαστάσεις MxN, τότε η λεχικογραφική αναπαράσταση γραμμής ορίζεται ως εξής:

$$x=[x(1,1)x(1,2)...x(1,N)x(2,1)...x(2,N)...x(M,1)...x(M,N)]^T \qquad (6.4)$$

Με τον τρόπο αυτό μετατρέπουμε ένα πίνακα (εικόνα) σε διάνυσμα και μπορούμε στη συνέχεια να εφαρμόσουμε όλους τους μετασχηματισμούς που ορίσαμε για τα μονοδιάστατα σήματα. Εμείς ωστόσο δε χρησιμοποίησαμε αυτό τον τρόπο αναπαράστασης της εικόνας αλλά εφαρμόσαμε τον πολλαπλής ανάλυσης μετασχηματισμό (multiresolution wavelet transform) κατευθείαν στην εικόνα μας με τον τρόπο που θα δείξουμε παρακάτω **[16]**. Η όλη διαδικασία ήταν αρκετά απλή και έχει ως εξής:

- Στην αρχική εικόνα εφάρμοσε το low-pass wavelet φίλτρο κατά μήκος τον γραμμών της εικόνας. Στην εικόνα που προέκυψε εφάρμοσε το low-pass wavelet φίλτρο κατά μήκος των στηλών της εικόνας. Κατόπιν αποδεκάτισε (decimate) την εικόνα κατά ένα παράγοντα 2 σε κάθε διάσταση της, ώστε να προκύψει μία εικόνα με διαστάσεις (M/2)x(N/2) (αν θεωρήσουμε ότι η αρχική ήταν διαστάσεων MxN). Η εικόνα αυτή, που σε έκταση είναι το 1/4 της αρχικής, αποτελεί το LP-LP (low-pass, low-pass) κομμάτι της εικόνας στο πεδίο του wavelet και τοποθετείται στο πάνω αριστερά κομμάτι της τελικής εικόνας (βλ. σχ. 6.3).

- Στην αρχική εικόνα εφάρμοσε το high-pass wavelet φίλτρο κατά μήκος τον γραμμών της εικόνας. Στην εικόνα που προέκυψε εφάρμοσε το low-pass wavelet φίλτρο κατά μήκος των στηλών της εικόνας. Κατόπιν αποδεκάτισε (decimate) την εικόνα κατά ένα παράγοντα 2 σε κάθε διάσταση της, ώστε να προκύψει πάλι μία εικόνα με διαστάσεις (M/2)x(N/2) Η εικόνα αυτή, που σε έκταση είναι πάλι το 1/4 της αρχικής, αποτελεί το HP-LP (high-pass, low-pass) κομμάτι της εικόνας στο πεδίο του wavelet και τοποθετείται στο πάνω δεξιά κομμάτι της τελικής εικόνας(βλ. σχ. 6.3).

- Στην αρχική εικόνα εφάρμοσε το low-pass wavelet φίλτρο κατά μήκος τον γραμμών της εικόνας. Στην εικόνα που προέκυψε εφάρμοσε το high-pass wavelet φίλτρο κατά μήκος των στηλών της εικόνας. Κατόπιν αποδεκάτισε (decimate) την εικόνα κατά ένα παράγοντα 2 σε κάθε διάσταση της, ώστε να προκύψει πάλι μία εικόνα με διαστάσεις (M/2)x(N/2) Η εικόνα αυτή, που σε έκταση είναι πάλι το 1/4 της αρχικής, αποτελεί το

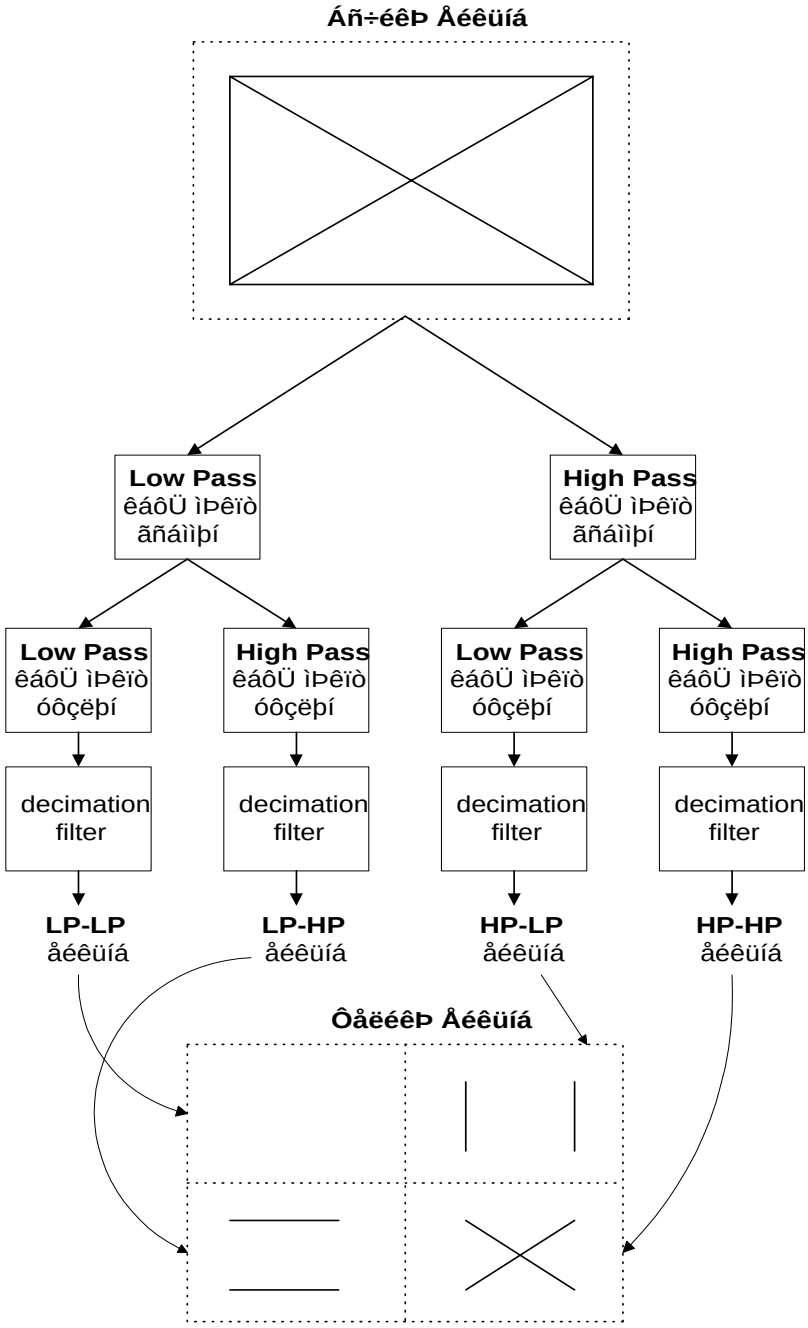
LP-HP (low-pass, high-pass) κομμάτι της εικόνας στο πεδίο του wavelet και τοποθετείται στο κάτω αριστερά κομμάτι της τελικής εικόνας (βλ. σχ. 6.3).

- Στην αρχική εικόνα εφάρμοσε το high-pass wavelet φίλτρο κατά μήκος τον γραμμών της εικόνας. Στην εικόνα που προέκυψε εφάρμοσε το high-pass wavelet φίλτρο κατά μήκος των στηλών της εικόνας. Κατόπιν αποδεκάτισε (decimate) την εικόνα κατά ένα παράγοντα 2 σε κάθε διάσταση της, ώστε να προκύψει πάλι μία εικόνα με διαστάσεις (M/2)x(N/2). Η εικόνα αυτή, που σε έκταση είναι πάλι το 1/4 της αρχικής, αποτελεί το HP-HP (high-pass, high-pass) κομμάτι της εικόνας στο πεδίο του wavelet και τοποθετείται στο κάτω δεξιά κομμάτι της τελικής εικόνας (βλ. σχ. 6.3).

Η παραπάνω διαδικασία αποτελεί το μετασχηματισμό wavelet ενός επιπέδου και απεικονίζεται στο σχήμα 6.2, όπου σαν αρχική εικόνα χρησιμοποιούμε ένα απλό σχήμα που αποτελείται από οριζόντιες, κάθετες και διαγώνιες γραμμές. Όπως φαίνεται από το σχήμα, η LP-LP εικόνα δεν περιλαμβάνει τίποτα καθώς δεν υπάρχει low-pass πληροφορία στο σχήμα. Η LP-HP εικόνα περιλαμβάνει τις οριζόντιες γραμμές της εικόνας αφού αυτές είναι που ανιχνεύει ο συνδυασμός του lowpass-high pass φίλτρου (low pass στις γραμμές, high pass στις στήλες). Ομοίως η HP-LP εικόνα περιλαμβάνει τις κάθετες γραμμές ενώ η HP-HP εικόνα περιλαμβάνει τις διαγώνιες γραμμές. Βλέπουμε δηλαδή ότι κάθε υποεικόνα περιλαμβάνει συγκεκριμένη πληροφορία από την αρχική εικόνα. Η low pass πληροφορία περιλαμβάνεται στην πρώτη (πάνω αριστερά) υποεικόνα, η οποία στην ουσία είναι μία "θολωμένη" έκδοση της αρχικής εικόνας, πράγμα το οποίο θα φανεί καλύτερα σε επόμενο παράδειγμα(βλ. σχ. 6.3). Όσων αφορά τη high pass πληροφορία (ακμές) της εικόνας, αυτή περιλαμβάνεται στις υπόλοιπες τρεις υποεικόνες, κάθε μία από τις οποίες ανιχνεύει είτε τις οριζόντιες, είτε τις κάθετες είτε τις διαγώνιες ακμές της εικόνας. Μετά την παραγωγή των τεσσάρων εικονών, μπορούμε να συνεχίσουμε παίρνοντας την low pass (πάνω αριστερά) εικόνα και εφαρμόζοντας την ίδια ακριβώς διαδικασία από την οποία θα προκύψουν πάλι τέσσερις υποεικόνες της low pass υποεικόνας. Η διαδικασία αυτή μπορεί να συνεχιστεί για να περάσουμε στο τρίτο επίπεδο ανάλυσης κ.ο.κ.

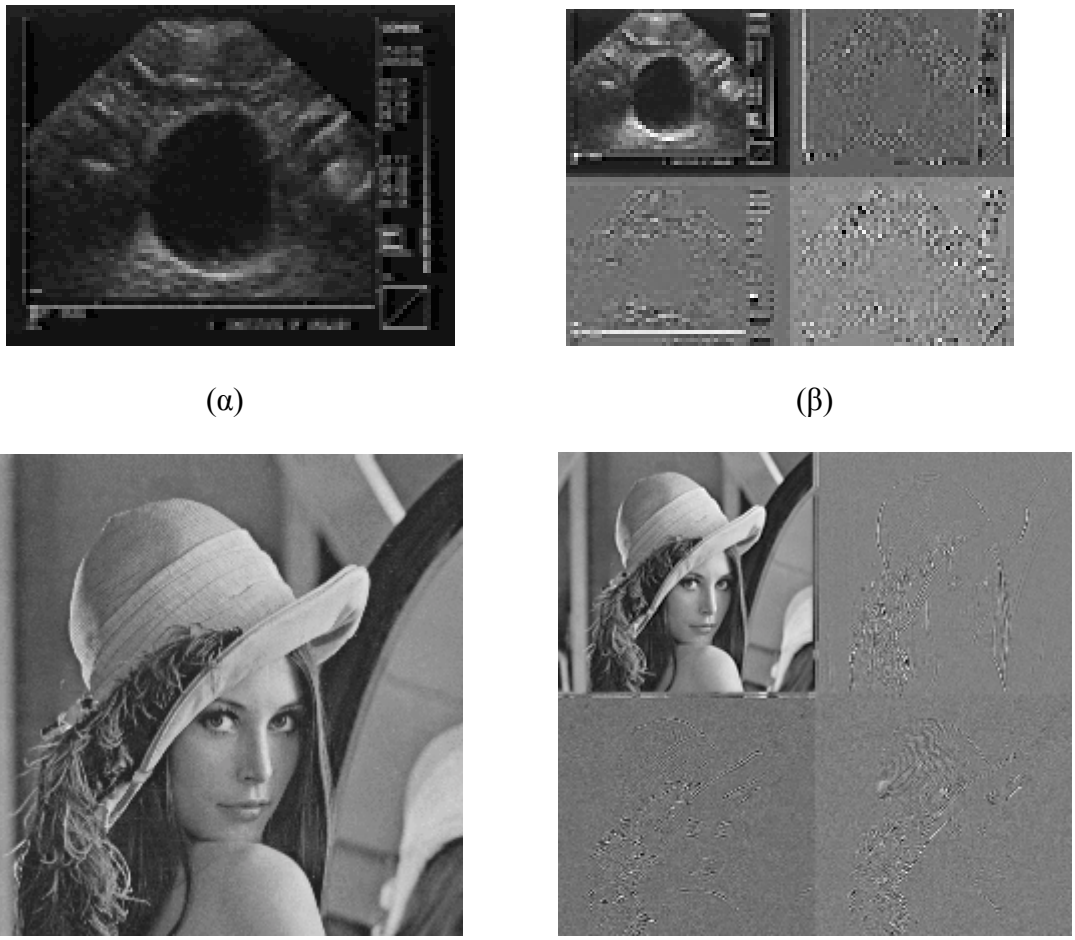
Τα low pass και high pass φίλτρα τα οποία χρησιμοποιήσαμε για το μετασχηματισμό είναι αυτά που παρουσιάσαμε στην προηγούμενη παράγραφο. Τα φίλτρα αυτά έχουν την ιδιότητα ότι είναι διαχωρίσιμα (separable) πράγμα το οποίο απαιτείται από το

μετασχηματισμό wavelet που παρουσιάσαμε, στον οποίο όπως είδαμε εφαρμόσαμε τα φίλτρα χωριστά στην οριζόντια και χωριστά στην κατακόρυφη διεύθυνση. Η απαίτηση για διαχωρίσιμα φίλτρα εξασφαλίζει την επιστροφή μας από το πεδίο του wavelet πίσω στο αρχικό πεδίο και την ανάκτηση της αρχικής εικόνας με τη βοήθεια του αντίστροφου μετασχηματισμού που θα παρουσιάσουμε στην επόμενη παράγραφο.



Σχήμα 6.2 Μετασχηματισμός (ενός επιπέδου) εικόνας στο πεδίο του wavelet.

Όσον αφορά τον τρόπο εφαρμογής των φίλτρων στην εικόνα, αυτό που γίνεται είναι η συνέλιξη της κρουστικής απόκρισης του φίλτρου με την κάθε γραμμή (ή στήλη) της εικόνας χωριστά. Επειδή τα φίλτρα που παρουσιάσαμε έχουν οκτώ συντελεστές, είναι προφανές ότι κάποιο πρόβλημα θα παρουσιάζεται στον υπολογισμό των επτά πιο ακραίων τιμών (pixels) της εικόνας σε κάθε πλευρά καθώς θα πρέπει να συμπεριλαμβάνονται και σημεία τα οποία βρίσκονται έξω από τα όρια της εικόνας. Στην περίπτωση αυτή ο μετασχηματισμός wavelet απαιτεί την περιοδική επέκταση της εικόνας στο επίπεδο, έτσι ώστε στα σημεία στα οποία έως πρότινος δεν είχαμε τιμές (καθώς βρίσκονταν εκτός ορίων) τώρα να είναι πλήρως καθορισμένα. Επίσης να πούμε ότι όσον αφορά τη αποδεκάτιση (decimation), αυτή υλοποιείται απλά "πετώντας" κάθε δεύτερο δείγμα (pixel), πρώτα προς την x και μετά προς y κατεύθυνση (η σειρά δεν έχει σημασία).



(α) (β)

(γ) (δ)

Σχήμα 6.3 (α),(γ) Αρχικές εικόνες. (β),(δ) Εικόνες στο πεδίο wavelet.

6.1.4 Ο αντίστροφος μετασχηματισμός.

Θα δείξουμε τώρα τον αντίστροφο μετασχηματισμό wavelet ο οποίος μας επιτρέπει την μεταφορά μας πίσω στο αρχικό πεδίο και την ανάκτηση της αρχικής εικόνας. Τα φίλτρα τα οποία θα χρησιμοποιήσουμε για τον αντίστροφο μετασχηματισμό δεν είναι τα ίδια με αυτά που χρησιμοποιήσαμε στον ευθύ μετασχηματισμό wavelet. Σχετίζονται όμως μ' αυτά και μπορούν να προκύψουν κατευθείαν αφού το μόνο που έχουμε να κάνουμε είναι να πάρουμε τα συμμετρικά τους ως προς τον άξονα των y **[14]**. Δηλαδή τα ζητούμενα φίλτρα είναι τώρα τα q_{-n} και r_{-n}. Επίσης, αντί για αποδεκάτιση (decimation) τώρα θα έχουμε επέκταση (expansion) των υποεικόνων. Η διαδικασία μοιάζει με αυτή της προηγούμενης παραγράφου και έχει ως εξής:

- Από την τελική εικόνα του σχήματος 6.2, δηλαδή από την εικόνα στο πεδίο του wavelet, παίρνουμε την πάνω αριστερή υποεικόνα και διπλασιάζουμε την κάθε διάστασή της παρεμβάλοντας μηδενικά (ένα μηδενικό μετά από κάθε pixel για κάθε γραμμή και μετά για κάθε στήλη). Έτσι προκύπτει μία εικόνα με διαστάσεις ίδιες με αυτές της αρχικής μας εικόνας. Στη συνέχεια εφαρμόζουμε το low pass φίλτρο (που αναφέραμε παραπάνω ποιο θα είναι) κατά μήκος των γραμμών και το low pass φίλτρο κατά μήκος των στηλών. Την εικόνα που προκύπτει θα ονομάζουμε προς το παρόν τελική εικόνα 1.

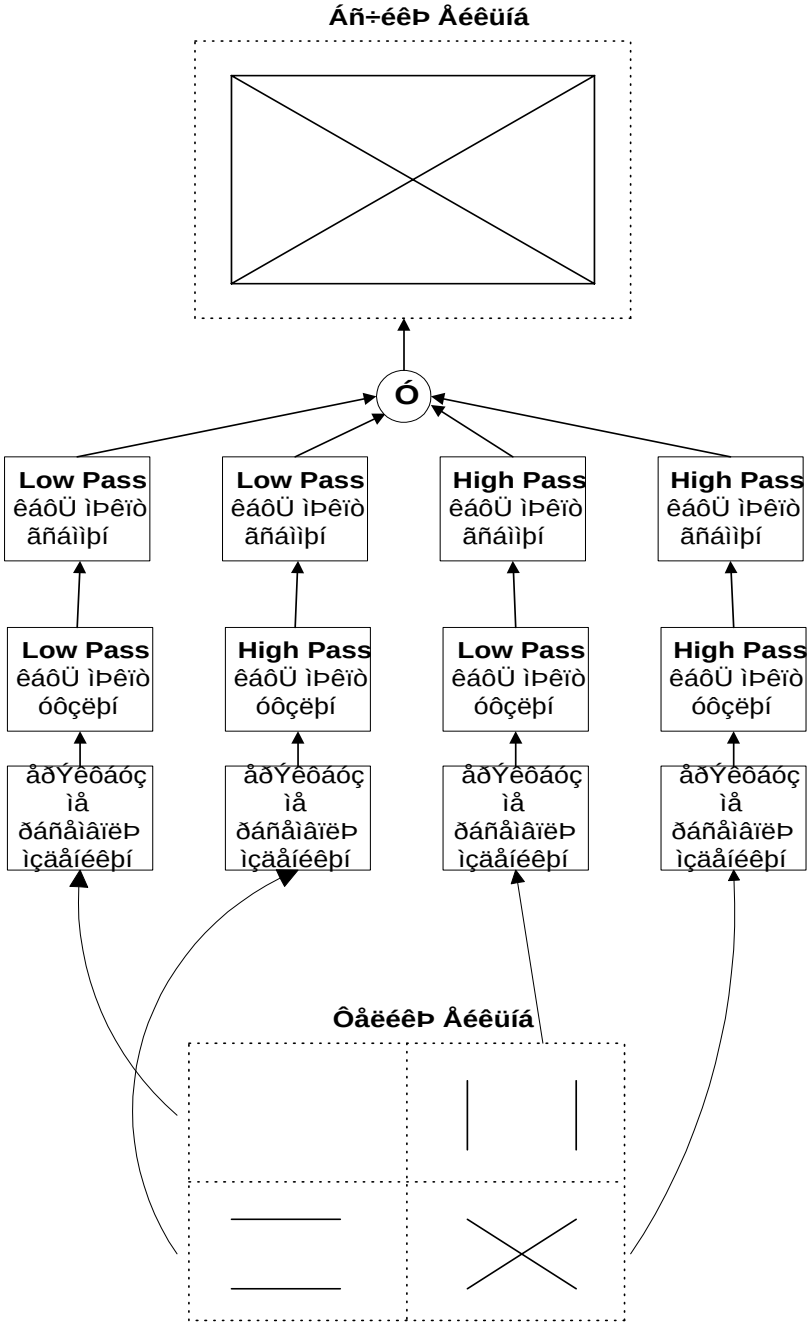
- Από την τελική εικόνα του σχήματος 6.2, παίρνουμε την πάνω δεξιά υποεικόνα και την επεκτείνουμε με μηδενικά όπως αναφέραμε παραπάνω. Έτσι προκύπτει πάλι μία εικόνα με διαστάσεις ίδιες με αυτές της αρχικής μας εικόνας. Στη συνέχεια εφαρμόζουμε το high pass φίλτρο (που αναφέραμε παραπάνω ποιο θα είναι) κατά μήκος των γραμμών και το low pass φίλτρο κατά μήκος των στηλών. Την εικόνα που προκύπτει ονομάζουμε προσωρινά τελική εικόνα 2.

- Από την τελική εικόνα του σχήματος 6.2, παίρνουμε την κάτω αριστερά υποεικόνα και την επεκτείνουμε με μηδενικά. Έτσι προκύπτει πάλι μία εικόνα με διαστάσεις ίδιες με αυτές της αρχικής μας εικόνας. Στη συνέχεια εφαρμόζουμε το low pass φίλτρο κατά μήκος των γραμμών και το high pass φίλτρο κατά μήκος των στηλών. Την εικόνα που προκύπτει ονομάζουμε προσωρινά τελική εικόνα 3.

- Από την τελική εικόνα του σχήματος 6.2, παίρνουμε την κάτω δεξιά υποεικόνα και την επεκτείνουμε με μηδενικά. Έτσι προκύπτει πάλι μία εικόνα με διαστάσεις ίδιες με

αυτές της αρχικής μας εικόνας. Στη συνέχεια εφαρμόζουμε το high pass φίλτρο κατά μήκος των γραμμών και το high pass φίλτρο κατά μήκος των στηλών. Την εικόνα που προκύπτει ονομάζουμε προσωρινά τελική εικόνα 4.

- Αθροίζουμε τις τελικές εικόνες 1-4 και το αποτέλεσμα που παίρνουμε είναι η αρχική εικόνα χωρίς καμμία αλλοίωση.



Σχήμα 6.4 Αντίστροφος μετασχηματισμός wavelet (ενός επιπέδου).

Για την εφαρμογή των φίλτρων θεωρούμε και πάλι περιοδική επέκταση της εικόνας στο επίπεδο. Η όλη διαδικασία απεικονίζεται στο σχήμα 6.4 όπου και πάλι δείχνουμε τον αντίστροφο μετασχηματισμό ενός επιπέδου μόνο (για λόγους απλότητας). Ωστόσο, η γενίκευση του μετασχηματισμού ώστε να περιλαμβάνει παραπάνω από δύο επίπεδα ανάλυσης είναι αρκετά απλή. Εμάς όμως δε θα μας απασχολήσει, καθώς οι μετασχηματισμοί που θα χρησιμοποιήσουμε από δω και στο εξής θα είναι πάντα ενός επιπέδου.

Βλέπουμε λοιπόν πως με τη βοήθεια του αντίστροφου μετασχηματισμού wavelet μπορούμε να ανακτήσουμε επακριβώς την αρχική μας εικόνα. Υπολογίσαμε ότι το μέσο σφάλμα ανάμεσα στην αρχική εικόνα και την εικόνα που ανακτήσαμε με τον αντίστροφο μετασχηματισμό, ήταν της τάξης του 10^{-6} για όλες τις εικόνες που δοκιμάσαμε (και αυτό οφειλόμενο στο trancating κάποιων δεκαδικών ψηφίων από τον υπολογιστή). Αναφερόμενοι στο σχήμα 6.3, βλέπουμε ότι ο ευθής μετασχηματισμός χρησιμοποιώντας σαν αρχική εικόνα την 6.3(α) μας δίνει σαν αποτέλεσμα την εικόνα 6.3(β), ενώ χρησιμοποιώντας σαν αρχική εικόνα την 6.3(γ) μας δίνει σαν αποτέλεσμα την εικόνα 6.3(δ). Αντιθέτως, ο αντίστροφος μετασχηματισμός wavelet που παρουσιάσαμε, παίρνοντας σαν είσοδο την εικόνα 6.3(β) μας δίνει πίσω την εικόνα 6.3(α), ενώ παίρνοντας σαν είσοδο τη εικόνα 6.3(δ) μας δίνει πίσω την εικόνα 6.3(γ).

Τέλος, υπενθυμίζουμε τις βασικές προυποθέσεις που πρέπει να ισχύουν για την επιτυχή ανάκτηση της αρχικής εικόνας οι οποίες είναι, πρώτον η ιδιότητα της διαχωρισιμότητας των φίλτρων, δεύτερο η συμμετρικότητα ανάμεσα στα φίλτρα του ευθύ μετασχηματισμού και αυτά του αντίστροφου μετασχηματισμού και τρίτον η θεώρηση της περιοδικής επέκτασης των εικονών κατά τη διάρκεια της εφαρμογής των φίλτρων.

6.2 ΕΦΑΡΜΟΓΗ ΤΩΝ ΦΙΛΤΡΩΝ WAVELET ΣΤΟΝ ΙΕΡΑΡΧΙΚΟ ΑΛΓΟΡΙΘΜΟ

6.2.1 Απόδοση ιεραρχικού αλγορίθμου χωρίς τη χρήση φίλτρων.

Στο Κεφάλαιο 4 (παρ. 4.5) παρουσιάσαμε τον ιεραρχικό αλγόριθμο για Image Segmentation. Το βαθυπερατό φίλτρο που χρησιμοποιήσαμε στην υλοποίηση του αλγορίθμου ήταν ένα (3x3) averaging φίλτρο. Είχε δηλαδή τη μορφή:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Τα αποτέλεσμα του ιεραρχικού αλγορίθμου με χρήση του παραπάνω φίλτρου φαίνεται στο σχήμα 6.5(γ). Επίσης στο σχήμα 6.5(β) φαίνεται το αποτέλεσμα του ιεραρχικού αλγορίθμου χωρίς τη χρήση βαθυπερατου φίλτρου.

(α)

(β)

(γ)

Σχήμα 6.5 Αποτελέσματα ιεραρχικού αλγορίθμου χωρίς τη χρήση βαθυπερατού φίλτρου (β) και με τη χρήση φίλτρου (γ).

Από τη σύγκριση των σχημάτων 6.5(β) και 6.5(γ) διαπιστώνουμε πως επηρεάζει η παρουσία του βαθυπερατού φίλτρου στη μεταγωγή μας από το ένα επίπεδο ανάλυσης στο άλλο και τελικά στην απόδοδη του αλγορίθμου. Παρατηρούμε δηλαδή, ότι η χρήση βαθυπερατού φίλτρου, αντίθετα με ότι θα περιμέναμε, δε βελτιώνει την απόδοδη της μεθόδου, και αυτό φαίνεται από την σκιασμένη περιοχή πάνω από το καπέλο της κοπέλας την οποία εντοπίζει καλύτερα ο αλγόριθμος χωρίς την χρήση φίλτρου. Αυτό, μπορεί να δικαιολογηθεί μόνο από το γεγονός ότι το βαθυπερατό φίλτρο που χρησιμοποιήσαμε δεν έχει κάποια ιδιαίτερα χαρακτηριστικά και το μόνο που κάνει είναι να παίρνει τον μέσο όρο του pixel στο οποίο εφαρμόζεται και των 8 γειτονικών. Είναι δηλαδή μία απλή περίπτωση φίλρου που προφανώς δεν ενδύκνειται για την περίπτωση μας, καθώς η συγκεκριμένη μέθοδος θα απαιτούσε χρήση φίλτρου που ναι μεν θα εξομάλυνε τις αντιθέσεις τις εικόνας και θα βοηθούσε στη ομαλή μας μεταγωγή από το ένα επίπεδο ανάλυσης στο επόμενο, αλλά που θα είχε και την επιπλέον ιδιότητα της "απότομης συχνότητας αποκοπής" η οποία θα διατηρούσε την οξύτητα των ακμών και στις χαμηλότερες αναλύσεις. Μία τέτοια περίπτωση φίλτρου παρουσιάζεται στη επόμενη παράγραφο.

6.2.2 Απόδοση ιεραρχικού αλγορίθμου με χρήση των φίλτρων wavelet.

Στην παράγραφο αυτή εξετάζουμε την απόδοση του ιεραρχικού αλγορίθμου όταν εφαρμόζουμε τα φίλτρα wavelet για τη μεταγωγή μας από το ένα επίπεδο ανάλυσης στο επόμενο. Συγκεκριμένα εφαρμόζουμε τα φίλτρα της παραγράφου 6.1.2. Τα αποτελέσματα της μεθόδου φαίνονται στο σχήμα 6.6(β). Στο σχήμα 6.6(γ) επαναλαμβάνουμε, για λόγους σύγκρισης, το αποτέλεσμα της μεθόδου χωρίς τη χρήση φίλτρου και στο σχήμα 6.6(δ) το αποτέλεσμα της μεθόδου με χρήση του φίλτρου της προηγούμενης παραγράφου.

Από το σχήμα παρατηρούμε τα αποτελέσματα της εφαρμογής του φίλτρου wavelet στην απόδοδη του αλγορίθμου. Όπως φαίνεται από τη σύγκριση των σχημάτων, η περίπτωση του σχήματος 6.6(β) που αφορά την εφαρμογή των φίλτρων wavelet, φαίνεται να δίνει τα καλύτερα αποτελέσματα, πράγμα το οποίο διαπιστώνουμε παρατηρώντας τη σκιά στο καπέλο της κοπέλας, η οποία έχει πλήρως ανιχνευθεί, και το λευκό κομμάτι πάνω από το καπέλο το οποίο τείνει να εξαλειφθεί. Όπως αναφέραμε και σε προηγούμενη

παράγραφο, τα φίλτρα wavelet τα οποία χρησιμοποιήσαμε, έχουν την ιδιότητα της απότομης αποκοπής των χαμηλών συχνοτήτων από τις ψηλές. Έτσι, από τη μία εξομαλύνουν το αποτέλεσμα του decimation filtering (για τη μετάβαση από το ένα επίπεδο ανάλυσης στο επόμενο) και από την άλλη διατηρούν την οξύτητα των ακμών στις μικρότερες αναλύσεις, διατηρώντας έτσι, όσο είναι δυνατό, την σημαντική πληροφορία της εικόνας σε όλα τα επίπεδα των αναλύσεων.

(α) (β)

(γ) (δ)

Σχήμα 6.6 Αποτελέσματα ιεραρχικού αλγορίθμου με χρήση φίλτρου wavelet (β), χωρίς χρήση βαθυπερατού φίλτρου (γ) και με τη χρήση φίλτρου (δ).

6.3 ΤΜΗΜΑΤΟΠΟΙΗΣΗ ΕΙΚΟΝΑΣ ΣΤΟ ΠΕΔΙΟ ΤΟΥ WAVELET

6.3.1 Εφαρμογή του adaptive αλγορίθμου στο πρώτο επίπεδο wavelet.

Θα δοκιμάσουμε τώρα την εφαρμογή του adaptive αλγορίθμου (που παρουσιάσαμε στο Κεφάλαιο 4), στο πεδίο του wavelet. Η διαδικασία η οποία εφαρμόζουμε και την οποία θα ονομάζουμε **wavelet clustering** από δω και πέρα, έχει ως εξής:

- Αρχικά, μεταφορά της αρχικής εικόνας στο πεδίο του wavelet (στο πρώτο επίπεδο). Αν για παράδειγμα θεωρήσουμε σαν αρχική εικόνα αυτή του σχήματος 6.3(γ), η νεά εικόνα η οποία θα πάρουμε θα είναι αυτή του σχήματος 6.3(δ).
- Στην πάνω αριστερά υποεικόνα του σχήματος (δηλαδή την LP-LP υποεικόνα), εφάρμοσε τον adaptive αλγόριθμο.
- Στη εικόνα που προέκυψε από το προηγούμενο βήμα, εφάρμοσε αντίστροφο μετασχηματισμό wavelet.
- Στο αποτέλεσμα εφάρμοσε ένα thresholding έτσι ώστε η τελική εικόνα να είναι δύο μόνο επιπέδων του γκρι.

Όπως είπαμε και παραπάνω ο μετασχηματισμός wavelet που χρησιμοποιήσαμε ήταν ενός επιπέδου. Κατόπιν εφαρμόσαμε τον adaptive αλγόριθμο (K=2) στην LP-LP (low pass,low pass) εικόνα οπότε και μας έδωσε μία εικόνα δύο επιπέδων (άσπρο-μαύρο) και στη συνέχεια χρησιμοποιήσαμε τον αντίστροφο μετασχηματισμό wavelet ακολουθούμενο από μία διαδικασία απλού thresholding η οποία είναι απαραίτητη έτσι ώστε η τελική εικόνα να είναι δύο μόνο επιπέδων. Το thresholding το εφαρμόζουμε γιατί η εικόνα που παίρνουμε με τον αντίστροφο wavelet μετασχηματισμό δεν είναι δύο μόνο επιπέδων αλλά περισσοτέρων. Ωστόσο τα περισσότερα από αυτά τα επίπεδα "μαζεύονται" γύρω από δυό τιμές που απέχουν αρκετά μεταξύ τους έτσι ώστε με ένα απλό thresholding (με κατώφλι γύρω στο 120) να μπορούμε να μετατρέψουμε την εικόνα μας σε απρόμαυρη χωρίς να χάσουμε πληροφορία.

Η παραπάνω διαδικασία εφαρμόστηκε στη γνωστή μας Lena. Το αποτέλεσμα φαίνεται στο σχήμα 6.7(β). Τα πλεονεκτήματα-μειονεκτήματα της μεθόδου συζητούνται στην επόμενη παράγραφο, όπου και γίνεται μία σύγκριση με τα αποτελέσματα προηγούμενων μεθόδων.

(α) (β)

Σχήμα 6.7 (α) Αρχική εικόνα. (β) Segmentation στο wavelet επίπεδο.

6.3.2 Πλεονεκτήματα-μειονεκτήματα της μεθόδου wavelet clustering.

Στο σχήμα 6.8(α) φαίνεται το αποτέλεσμα του segmentation στο επίπεδο του wavelet ενώ στο σχήμα 6.8(β) βλέπουμε το αποτέλεσμα του ιεραρχικού αλγορίθμου που παρουσιάσαμε στο Κεφάλαιο 4. Επίσης στο σχήμα 6.8(γ) βλέπουμε το αποτέλεσμα του wavelet clustering χρησιμοποιώντας το τέχνασμα που αναφέραμε στο Κεφάλαιο 4 ενώ στο σχήμα 6.8(δ) φαίνεται το αντίστοιχο αποτέλεσμα για τον ιεραρχικό αλγόριθμο. Σε όλες τις περιπτώσεις θεωρήσαμε K=2 και σ=8.

(α) (β)

(γ)

(δ)

Σχήμα 6.8 (α) Segmentation στο wavelet επίπεδο. (β) Αποτέλεσμα του ιεραρχικού αλγορίθμου. (γ),(δ) Αντίστοιχα των (α),(β) με χρήση όμως της παραμέτρου α.

Από το σχήμα, παρατηρούμε καταρχήν, ότι το segmentation στο πεδίο του wavelet έδωσε ένα καλό αποτέλεσμα όσον αφορά τουλάχιστον την ανίχνευση των βασικών χαρακτηριστικών της εικόνας. Βλέπουμε δηλαδή (βλ. σχ. 6.8(α)), ότι η σκιά στο καπέλο της κοπέλας ανιχνεύθηκε πλήρως, ενώ και η λευκή περιοχή πάνω από το καπέλο της κοπέλας τείνει να εξαφανιστεί. Αντίθετα, ο ιεραρχικός αλγόριθμος (βλ. σχ. 6.8(β)), απέτυχε να ανιχνεύσει πλήρως τη σκιά στο καπέλο ενώ από την άλλη δεν ξεχώρισε και την λευκή περιοχή από το καπέλο (βλέπουμε ότι σμίγουν). Επίσης παρατηρώντας τα σχήματα 6.8(γ),(δ), βλέπουμε τα αποτελέσματα των δυό αλγορίθμων στην περίπτωση που χρησιμοποιήσουμε το τέχνασμα που παρουσιάσαμε στο Κεφάλαιο 4 (όπου χρησιμοποιούμε την "παράμετρο apriori γνώσης" α). Παρατηρούμε ότι και στις δύο περιπτώσεις τα αποτελέσματα είναι αρκετά ικανοποιητικά, καθώς και η λευκή περιοχή πάνω από το καπέλο τείνει να εξαλειφθεί.

Στο σχήμα 6.9 παρουσιάζουμε την εικόνα της κύστης και τα αποτελέσματα τόσο του wavelet clustering (6.9(β)), όσο και του ιεραρχικού αλγορίθμου (6.9(γ)). Παρατηρούμε, ότι και σ' αυτή την περίπτωση τα αποτελέσματα του wavelet clustering (αν εξαιρέσουμε ότι είναι λίγο πιο χοντροκομμένα) είναι ικανοποιητικά, καθώς εντοπίζεται αρκετά καλά η περιοχή της κύστης στην εικόνα, πράγμα που είναι ο απωτερός μας σκοπός σ' αυτού του είδους τις εικόνες.

Σχήμα 6.9 (α) Αρχική εικόνα της κύστης. (β) Αποτέλεσμα του wavelet clustering. (γ) Αποτέλεσμα του ιεραρχικού αλγορίθμου.

Μοναδικό μειονέκτημα το οποίο παρουσιάζει η μέθοδος (την οποία από δω και στο εξής θα αποκαλούμε **wavelet clustering**) είναι τα σχετικά πιο χοντροκομμένα αποτελέσματα τα οποία δίνει, πράγμα το οποίο διαπιστώνεται αν παρατηρήσουμε, για παράδειγμα, τα φτερά στο καπέλο της κοπέλας και στις δύο περιπτώσεις. Ωστόσο αυτό δεν αποτελεί ιδιαίτερα σοβαρό πρόβλημα καθώς, στις περισσότερες περιπτώσεις, αυτό που πραγματικά μας ενδιαφέρει είναι η ανίχνευση των επικρατέστερων περιοχών της εικόνας, στόχος στον οποίο φαίνεται να πετυχαίνει η μέθοδος που παρουσιάσαμε. Από την άλλη ένα μεγάλο πλεονέκτημα της μεθόδου είναι ο χρόνος εκτέλεσης, ο οποίος μειώνεται σημαντικά. Ρίχνοντας μία ματιά στον Πίνακα Α (στον οποίο βλέπουμε τους **συνολικούς χρόνους εκτέλεσης των προγραμμάτων**) βλέπουμε ότι ο χρόνος εκτέλεσης του αλγορίθμου στο πεδίο του wavelet είναι λιγότερος από το μισό του χρόνου εκτέλεσης του ιεραρχικού αλγορίθμου. Και στις δύο περιπτώσεις η εικόνα που χρησιμοποιήθηκε (Lena) ήταν διαστάσεων 512x480 και οι δοκιμές έγιναν στο SUN Ultra 1. Και πάλι, κανένα είδος βελτιστοποίησης δεν εφαρμόστηκε στο πρόγραμμά μας.

ΠΙΝΑΚΑΣ Α

	Wavelet clustering	Ιεραρχικός αλγόριθμος
χρόνος εκτέλεσης	19	41

Ένα άλλο πλεονέκτημα του αλγορίθμου, είναι η καλή απόδοση που παρουσιάζει παρουσία θορύβου. Στο σχήμα 6.10(α) βλέπουμε τη γνωστή μας Lena, στην οποία όμως έχουμε προσθέσει λευκό Gaussian θόρυβο με τυπική απόκλιση σ=32 (το πρόγραμμα που χρησιμοποιήσαμε για το σκοπό αυτό ήταν το Adobe PhotoShop 3.0). Η ποσότητα του

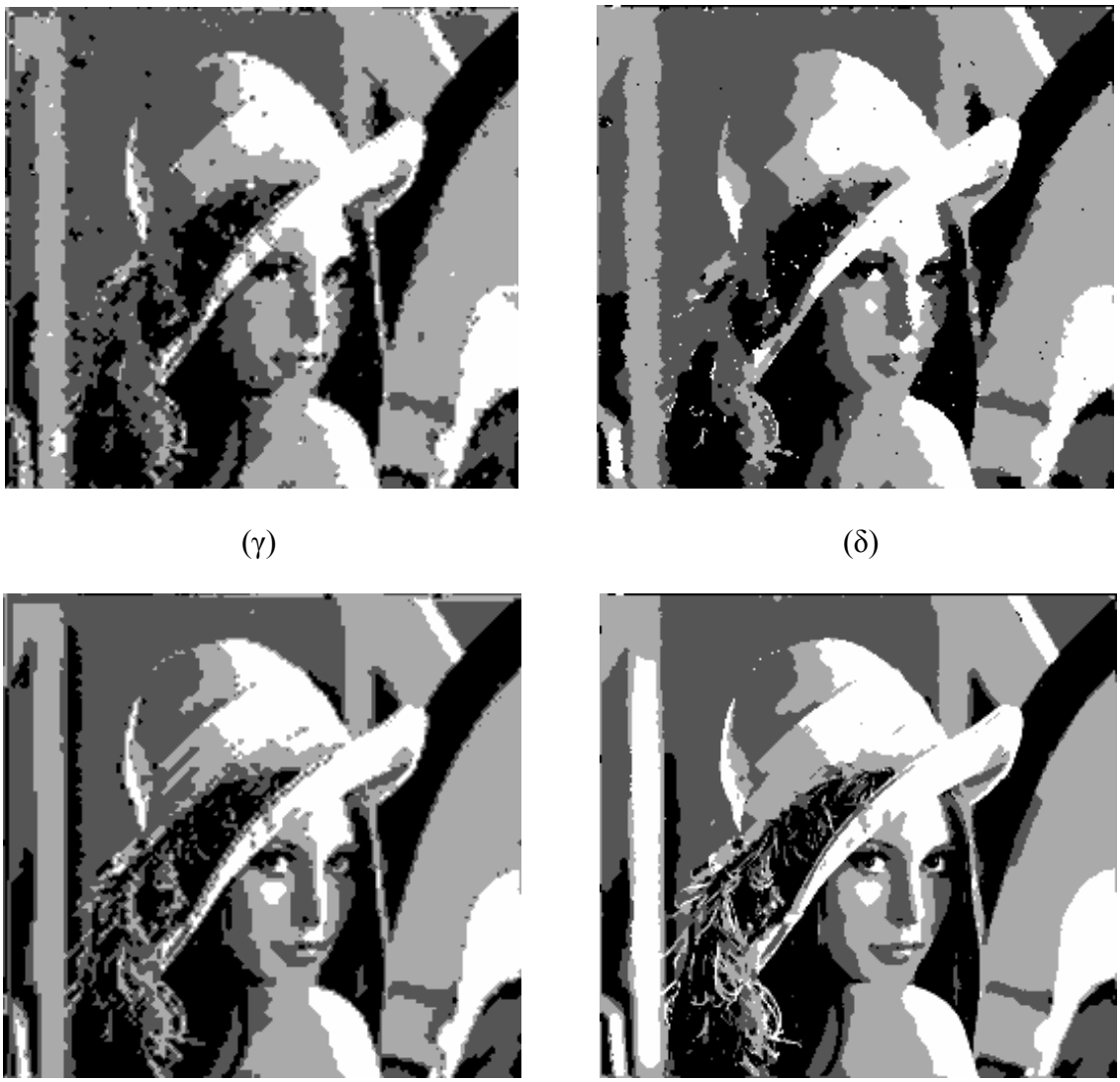
θορύβου θεωρείται αρκετά μεγάλη, πράγμα που φαίνεται και από το σχήμα αφού η αλλοίωση της εικόνας είναι σημαντική.

Για την αντιμετώπιση του συγκεκριμένου προβλήματος εφαρμόσαμε τον αλγόριθμο wavelet clustering όπως ακριβώς τον περιγράψαμε στην παράγραφο 6.3.1 με μία όμως διαφορά. Αυτό που κάναμε επιπλεόν, ήταν να εφαρμόσαμε κάποιο **thresholding** στις HL, LH, HH υποεικόνες έτσι ώστε να παραμείνουν οι ισχυρές ακμές (οι οποίες προφαμώς αντιπροσωπεύουν τις πραγματικές ακμές της εικόνας) και να εξαλειφθούν οι υπόλοιπες ακμές οι οποίες δημιουργήθηκαν εξαιτίας του θορύβου. Προσπαθήσαμε με τον τρόπο αυτό να μειώσουμε την παρουσία του θορύβου και συνεπώς να περιορίσουμε την επίδρασή του στο τελικό αποτέλεσμα.

Στο σχήμα 6.10(β) βλέπουμε το αποτέλεσμα της εφαρμογής του wavelet clustering ενώ στο σχήμα 6.10(γ) βλέπουμε την τελική εικόνα μετά το thresholding που εφαρμόσαμε έτσι ώστε να έχουμε μόνο τέσσερα χρώματα. Στο σχήμα 6.10(δ) βλέπουμε το αποτέλεσμα της εφαρμογής του ιεραρχικού αλγορίθμου. Επίσης στο σχήμα 6.10(ε) βλέπουμε το αποτέλεσμα του wavelet clustering στην εικόνα χωρίς όμως την παρουσία θορύβου, ενώ το αντίστοιχο αποτέλεσμα για τον ιεραρχικό αλγόριθμο φαίνεται στο σχήμα 6.10(στ΄). Επειδή η ποσότητα του θορύβου είναι μεγάλη, χρησιμοποιήσαμε K=4 σε όλες τις περιπτώσεις για καλύτερα αποτελέσματα. Επίσης χρησιμοποιήσαμε σ=24 (και στις δύο περιπτώσεις) καθώς αυτή η τιμή βρέθηκε να δίνει τα καλύτερα αποτελέσματα.

(α)

(β)



(γ) (δ)

(ε) (στ΄)

Σχήμα 6.10 (α) Αρχική εικόνα με λευκό Gassian θόρυβο (σ=32). (β) Αποτέλεσμα του wavelet clustering πριν το thresholding. (γ) Αποτέλεσμα του wavelet clustering μετά την εφαρμογή του thresholding. (δ) Αποτέλεσμα ιεραρχικού αλγορίθμου. (ε) Αποτέλεσμα του wavelet clustering στην αρχική εικόνα χωρίς θόρυβο. (στ΄) Αποτέλεσμα του ιεραρχικού αλγορίθμου στην αρχική εικόνα χωρίς θόρυβο.

Αυτό που παρατηρούμε από το σχήμα αρχικά, είναι ότι παρά την μεγάλη αλλοίωση της εικόνας από το θόρυβο, και στις δύο περιπτώσεις το αποτέλεσμα είναι ικανοποιητικό, καθώς και πάλι τα βασικά χαρακτηριστικά της εικόνας ανιχνεύονται και η τελική εικόνα εξακολουθεί να είναι αναγνωρίσιμη. Αντίθετα, το αποτέλεσμα που έδωσε για αυτή την εικόνα ο K-means για παράδειγμα, ήταν κάτι το εντελώς ακαθόριστο (γι΄αυτό και δεν το

παραθέτουμε εδώ), πράγμα που φανερώνει την αδυναμία του K-means να εντοπίσει τα χαρακτηριστικά της εικόνας παρουσία θορύβου.

Συγκρίνοντας τα αποτελέσματα των σχημάτων 6.10(γ) και 6.10(δ), παρατηρούμε ότι η μέθοδος του wavelet clustering (παρόλο που δίνει ένα σχετικά πιο χοντροκομμένο αποτέλεσμα) εξακολουθεί να ανιχνεύει τα βασικά χαρακτηριστικά της εικόνας και μάλιστα ανιχνεύει κάπως καλύτερα ορισμένα απ' αυτά, και αυτό φαίνεται από τη σκία πάνω στο καπέλο αλλά και από το "βλέμμα" της κοπέλας το οποίο είναι πιο ρεαλιστικό στην περίπτωση αυτή. Δίνει δηλαδή μια σχετικά καλή αναπαράσταση της αρχικής εικόνας, και την καθιστά αναγνωρίσιμη παρά την έντονη παρουσία θορύβου στην αρχική εικόνα και παρά το γεγονός ότι ο αριθμός των επιπέδων του γκρι που χρησιμοποιούνται είναι αρκετά μικρός (K=4).

Κάτι άλλο το οποίο είναι επίσης πολύ σημαντικό και το οποίο ίσως να μας αφορά περισσότερο και από την εγκυρότητα των αποτελεσμάτων, είναι η ανθεκτικότητα (robustness) του αλγορίθμου στο θόρυβο. Αυτό που μας ενδιαφέρει σε έναν αλγόριθμο είναι, τα αποτελέσματα που δίνει παρουσία θορύβου να είναι κατά το δυνατόν πιο κοντά σ' αυτά που δίνει χωρίς την παρουσία του θορύβου. Γι' αυτό το λόγο, στο σχήμα 6.10, παραθέσαμε και τα αποτελέσματα του κάθε αλγορίθμου στην αρχική εικόνα, χωρίς την παρουσία θορύβου. Όσον αφορά το wavelet clustering, συγκρίνοντας το σχήμα 6.10(γ) με το σχήμα 6.10(ε) βλέπουμε ότι το αποτέλεσμα είναι αλλοιωνένο ως προς κάποιες λεπτομέρειες της εικόνας (οι οποίες ίσως θα μπορούσαν να βελτιωθούν με χρήση κάποιου median φίλτρου) ενώ από την άλλη οι βασικές περιοχές ανιχνεύονται όπως και πριν. Όσον αφορά τον ιεραρχικό αλγόριθμο, συγκρίνοντας τα σχήματα 6.10(δ) και 6.10(στ΄) βλέπουμε ότι η αλλοίωση η οποία έχει επέλθει τώρα είναι μεγαλύτερη (φαίνεται από τα μάτια, το στόμα και το καπέλο της κοπέλας).

Συμπεραίνουμε δηλαδή ότι η μέθοδος wavelet clustering φαίνεται να είναι αρκετά ανθεκτική (robust) στο θόρυβο και μάλιστα δείχνει να είναι πιο "robust" και από ένα ήδη πολύ καλό αλγόριθμο για clustering εικόνας (εννούμε τον ιεραρχικό). Η ιδιότητα του "robustness" μιας μεθόδου είναι πολύ σημαντική, καθώς αυτό σημαίνει ότι αν καταφέρουμε να βελτιώσουμε κι άλλο την απόδοση της μεθόδου αυτό σημαίνει βελτίωση της μεθόδου και παρουσία θορύβου.

Η αρχική εικόνα έχει διαστάσεις 256x240 ενώ για τις δοκιμές χρησιμοποιήθηκε και πάλι το SUN Ultra 1. Ο χρόνος εκτέλεσης σε κάθε περίπτωση φαίνεται στον Πίνακα Γ, από όπου και πάλι διαπιστώνουμε το πλεονέκτημα της μεθόδου του wavelet clustering.

ΠΙΝΑΚΑΣ Γ

	Wavelet clustering	Ιεραρχικός αλγόριθμος
χρόνος εκτέλεσης	4	8.2

Συμπερασματικά, θα λέγαμε ότι η μέθοδος που παρουσιάσαμε είναι μία εναλλακτική λύση για Image Segmentation, η οποία συνδυάζοντας την αποτελεσματικότητα του adaptive αλγορίθμου (του Κεφαλαίου 4) και τις ιδιότητες των wavelets, δίνει ένα αποδεκτό αποτέλεσμα (segmentation) το οποίο είναι σε μερικές περιπτώσεις καλύτερο από αυτό άλλων μεθόδων και μάλιστα απαιτώντας σημαντικά λιγότερο χρόνο εκτέλεσης. Όσων αφορά το τελευταίο, δηλαδή το μειωμένο χρόνο εκτέλεσης, ο λόγος είναι προφανής και έγγειται στο γεγονός ότι ο αλγόριθμος δεν εφαρμόζεται πλέον στην αρχική εικόνα αλλά στην LP-LP υποεικόνα της, η οποία έχει τις μισές διαστάσεις της αρχικής εικόνας. Αυτή μάλιστα είναι και η αιτία που τα αποτελέσματα που δίνει η μέθοδος είναι πιο "χοντροκομμένα" από αυτά που δίνει ο ιεραρχικός αλγόριθμος. Το γεγόνος δηλαδή ότι ο δουλεύουμε σε χαμηλότερο επίπεδο ανάλυσης, έχει το κόστος της απώλειας κάποιων λεπτομερειών (όχι όμως σημαντικών) της εικόνας. Η χρήση βέβαια του low pass φίλτρου, περιορίζει κατά το δυνατό το πρόβλημα αυτό, σε καμμία περίπτωση όμως δε μπορεί να το εξαλείψει, καθώς από ένα σημείο και μετά, η απώλεια των pixels (που προκύπτει με το στάδιο του decimation) καθιστά "σχεδόν" αδύνατη την ανάκτηση της χαμένης πληροφορίας. Λέμε "σχεδόν", γιατί υπάρχει και ο αντίστροφος μετασχηματισμός wavelet, ο οποίος συλλέγει την πληροφορία και από τις άλλες τρεις υποεικόνες (οι οποίες κρατούν πληροφορία για τις high pass συνιστώσες της εικόνας, δηλαδή τις ακμές) και προσπαθεί να κατασκευάσει μία όσο το δυνατόν ρεαλιστικότερη αναπαράσταση της αρχικής εικόνας, πράγμα που έως ένα βαθμό το καταφέρνει.

ΚΕΦΑΛΑΙΟ 7: ΣΥΜΠΕΡΑΣΜΑΤΑ-ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

7.1 ΣΥΝΟΨΗ

Στα προηγούμενα κεφάλαια μελετήσαμε κάποιους αλγορίθμους για τμηματοποίηση εικόνας, προτείναμε κάποιους καινούριους και είδαμε την εφαρμογή τους πάνω σε πραγματικές εικόνες. Ξεκινήσαμε αρχικά (Κεφάλαιο 1), προσδιορίζοντας το πρόβλημα του "segmentation" σαν ένα βασικό αλλά δύσκολο βήμα (στάδιο) στην επεξεργασία εικόνας. Κατόπιν (Κεφάλαιο 2), παρουσιάσαμε κάποιες γενικές κατηγορίες segmentation και είδαμε κάποιες μεθόδους από κάθε κατηγορία. Στο Κεφάλαιο 3 παρουσιάσαμε ένα αλγόριθμο για ανίχνευση ακμών σε μία εικόνα, ενώ στο Κεφάλαιο 4 παρουσιάσαμε ένα αλγόριθμο για ανίχνευση των βασικών περιοχών σε μία εικόνα (clustering algorithm). Στο Κεφάλαιο 5 κάναμε μία εισαγωγή στο πεδίο του wavelet, ενώ στο Κεφάλαιο 6 επιχειρήσαμε να εφαρμόσουμε τον clustering αλγόριθμο στο πεδίο του wavelet, προτείνοντας μία εναλλακτική μέθοδο για Image Segmentation της οποίας τα αποτελέσματα μελετήσαμε και σχολιάσαμε. Στο παρόν Κεφάλαιο θα παρουσιάσουμε τα συμπεράσματα στα οποία καταλήξαμε μετά το πέρας της εργασίας. Επίσης θα ακολουθήσουν και κάποιες ιδέες για μελλοντική εξέλιξη της εργασίας.

7.2 ΤΕΛΙΚΑ ΣΥΜΠΕΡΑΣΜΑΤΑ

Ξεκινώντας από την παρουσίαση του αλγορίθμου ανίχνευσης ακμών του Κεφαλαίου 3, μία πρώτη παρατήρηση είναι η αλλοίωση της απόδοσης της μεθόδου, οφειλόμενη βέβαια στο κριτήριο με το οποίο έγινε το merging των περιοχών. Όπως παρατηρήσαμε από τα αποτελέσματα της εφαρμογής του αλγορίθμου σε εικόνες, ενώ οι ακμές που ανιχνεύονται είναι έγκυρες, ωστόσο σε κάποια σημεία οι τελικές ακμές δεν έχουν πλήρως διατηρηθεί, γεγονός που υποδεικνύει ότι σε εκείνα τα σημεία έγινε merging περιοχών το οποίο δε θα πρεπε να είχε γίνει.

Στο Κεφάλαιο 4 παρουσιάσαμε έναν clustering αλγόριθμο ο οποίος, αυτό που στην ουσία κάνει, είναι να μετασχηματίσει μία αρχική grayscale εικόνα σε μία "καρικατούρα" της αρχικής χρησιμοποιώντας 2-4 επίπεδα του γκρι. Υλοποιήσαμε τον αλγόριθμο με αρκετούς εναλλακτικούς τρόπους, πειραματιστήκαμε με διάφορες τιμές των παραμέτρων του αλγορίθμου, προσθέσαμε κάποιες καινούριες παραμέτρους, μελετήσαμε τα αποτελέσματα που παίρναμε κάθε φορά και το συμπέρασμα στο οποίο καταλήξαμε ήταν ότι μπορούμε, αξιοποιώντας σωστά τον αλγόριθμο, να τον προσαρμόσουμε στις εκάστοτε ανάγκες μας ώστε να μας αποδώσει τα μέγιστα όσον αφορά την ποιότητα του τελικού αποτελέσματος. Εκτός δηλαδή από την υλοποίηση μίας ιδέας (ενός αλγορίθμου) στον υπολογιστή, αυτό που είναι εξίσου σημαντικό είναι μία εκτεταμένη έρευνα των δυνατοτήτων του αλγορίθμου, και αυτό θα γίνει μόνο εξαντλώντας κάθε δυνατή περίπτωση για τις τιμές που μπορούν να πάρουν οι παράμετροι του αλγορίθμου. Δεν αρκεί δηλαδή να κατασκευάσεις κάτι, αυτό που είναι εξίσου σημαντικό είναι να μάθεις να το χρησιμοποιείς σωστά και να το φέρνεις κάθε φορά στα μέτρα του εκάστοτε προβλήματος ώστε να αποδίδει το καλύτερο δυνατό αποτέλεσμα. Στο σημείο αυτό, πρέπει να παρατηρήσουμε ότι η αυτοματοποίηση μιας τέτοιας διαδικασίας είναι προφανώς ένα πολύ σημαντικό κομμάτι για έρευνα.

Ο συνδυασμός του πεδίου wavelet με τον αλγόριθμο που αναφέραμε παραπάνω, μπορεί να μην έδωσε κάποιο αποτέλεσμα καλύτερο από τα ήδη υπάρχοντα (αν και σε κάποιες περιπτώσεις συνέβη κι αυτό), ωστόσο, αυτό που έδωσε ήταν κάποια καλά σημάδια ότι με σωστή αξιοποίηση όλης της πληροφορίας που κρύβει το πεδίο του wavelet μπορούμε να σχεδιάσουμε νέους αλγορίθμους για Image Segmentation, οι οποίοι να είναι πιο αποτελεσματικοί από τους ήδη υπάρχοντες (ιδιαίτερα σε περιβάλλον θορύβου) αλλά και πολύ πιο "οικονομικοί" σε υπολογιστικό χρόνο.

Συγκρίνοντας τα αποτελέσματα τα οποία δίνει η εφαρμογή ενός αλγορίθμου ακμών (και συγκεκριμένα αυτού που παρουσιάσαμε στο Κεφάλαιο 3) με τα αντίστοιχα αποτελέσματα ενός clustering αλγορίθμου (και συγκεκριμένα αυτών που παρουσιάσαμε στα Κεφάλαια 4,6), καταλήγουμε σε κάποια χρήσιμα συμπεράσματα όσον αφορά την απόδοση των δυό διαφορετικών προσεγγίσεων για τμηματοποίηση εικόνας. Πρώτα απ' όλα παρατηρούμε ότι ένας clustering αλγόριθμος οδηγεί σε εικόνες οι οποίες παραμένουν "αναγνωρίσιμες" παρά το ότι διαθέτουν ένα πολύ μικρό αριθμό επιπέδων του γκρι (2-4), και συνεπώς οι εικόνες αυτές μπορούν να χρησιμοποιηθούν από συσκευές αναγνώρισης

(σε ένα σύστημα αναγνώρισης εικόνας). Αντίθετα, ένας αλγόριθμος ανίχνευσης ακμών το μόνο που διατηρεί είναι η "high pass" πληροφορία της εικόνας (ακμές), με αποτέλεσμα η τελική εικόνα να μην είναι πάντα αναγνωρίσιμη, εκτός κι αν ξέρουμε από πριν περί τίνος πρόκειται. Ωστόσο, όσον αφορά το θέμα της κωδικοποίησης και συμπίεσης των εικόνων, και οι μεν και οι δε μπορούν να κωδικοποηθούν πετυχαίνοντας πολύ υψηλούς βαθμούς συμπίεσης.

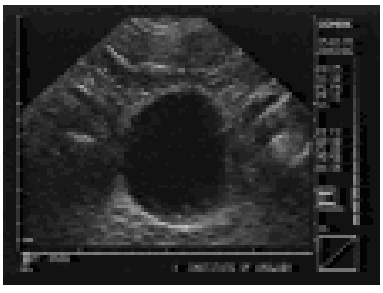
Κάτι άλλο το οποίο είναι επίσης σημαντικό, είναι το γεγονός ότι η απόδοση του clustering αλγορίθμου που παρουσιάσαμε είναι σαφώς ανώτερη αυτής του αλγορίθμου ανίχνευσης ακμών σε περιβάλλον θορύβου. Όπως είδαμε και από παραδείγματα στο Κεφάλαιο 6, ακόμα και με noise variance σ=32, η τελική εικόνα εξακολουθεί να διατηρεί τα βασικά της χαρακτηριστικά και να παραμένει αναγνωρίσιμη. Αντίθετα, τα αποτελέσματα του αλγορίθμου ανίχνευσης ακμών σε περιβάλλον ισχυρού θορύβου δεν είναι καθόλου ικανοποιητικά, καθώς η έντονη παρουσία του θορύβου δημιουργεί επιπλέον ακμές κάνοντας έτσι σχεδόν αδύνατη την ανίχνευση των πραγματικών ακμών (και μόνο) της εικόνας.

7.3 ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ ΚΑΙ ΠΡΟΟΠΤΙΚΕΣ ΤΗΣ ΕΡΓΑΣΙΑΣ

Πριν αναφέρουμε κάποιες ιδέες για μελλοντική επέκταση της εργασίας ή για τις προοπτικές που αυτή έχει, θεωρούμε σκόπιμο να σημειώσουμε ότι κάποιες από τις ιδέες προτάσεις που αναφέρουμε παρακάτω, για την εξέλιξη και βελτίωση της παρούσας εργασίας, δοκιμάστηκαν αλλά με όχι ιδιαίτερη επιτυχία. Και αυτό γιατί ενώ σε ορισμένες περιπτώσεις εφαρμόζοντας μία βελτίωση τα αποτελέσματα ήταν ικανοποιητικά (καλύτερα από πριν), σε άλλες πάλι ήταν πολύ χειρότερα. Παραβλέποντας το γεγονός ότι καμία από τις υπάρχουσες μεθόδους δε μπορεί να δώσει καλά αποτελέσματα για την πληθώρα των εικόνων, και ότι μπορεί μία συγκεκριμένη εικόνα να "μηδενίσει" την αποτελεσματικότητα μίας μεθόδου, ο σημαντικότερος λόγος για τον οποίο οι δοκιμές (επεκτάσεις) που επιχειρήθηκαν δεν είχαν συνολική επιτυχία, ήταν ότι κάτι τέτοιο (όπως αποδείχτηκε στην πράξη) θα απαιτούσε μία πιο ριζική, ολοκληρωμένη αλλά και χρονοβόρα έρευνα, πράγμα που όμως ξεφεύγει από τους σκοπούς της συγκεκριμένης διπλωματικής εργασίας. Ας γίνουμε όμως πιο συγκεκριμένοι.

Μία προσπάθεια η οποία έγινε και η οποία αξίζει να συνεχιστεί, τόσο για ερευνητικούς λόγους όσο και για την προοπτική βελτίωσης των υπαρχουσών μεθόδων, αφορά στο segmentation των υπόλοιπων τριών εικόνων του πεδίου του wavelet. Εμείς, όπως αναφέραμε, εφαρμόσαμε segmentation μόνο στην LP-LP υποεικόνα του πεδίου wavelet, καθώς αυτή είναι που εξακολουθεί να διατηρεί τη σημαντικότερη πληροφορία της εικόνας. Η προσπάθεια για εφαρμογή segmentation και στις υπόλοιπες τρεις υποεικόνες, απέδωσε μόνο σε ορισμένες περιπτώσεις, και σαν αποτέλεσμα είχε την ανίχνευση κάποιων επιπλέον ακμών της εικόνας. Σε κάποιες άλλες όμως εικόνες το αποτέλεσμα ήταν κακό. Το πρόβλημα εντοπίστηκε αρχικά στην αδυναμία του adaptive αλγορίθμου να δώσει ένα καλό segmentation των τριών εικόνων (καθώς αυτές περιείχαν κάποιες ακμές όχι διαίτερα έντονες και με πολύ λίγα σημεία). Το πρόβλημα αυτό με τη σειρά του "τροφοδοτούσε" τον αντίστροφο μετασχηματισμό wavelet με συνέπεια τα αποτελέσματα τα οποία έδινε να μην είναι ικανοποιητικά. Στις περιπτώσεις που το segmentation των τρίων εικόνων ήταν αποδεκτό, το τελικό αποτέλεσμα ήταν καλύτερο μόνο στην περίπτωση της ανίχνευσης κάποιων μεμονωμένων ακμών.

Για να γίνει κατανοητό αυτό ας ρίξουμε μία ματιά στο σχήμα 7.1. Στο σχήμα 7.1(α) φαίνεται η εικόνα μίας κύστης την οποία έχουμε χρησιμοποιήσει και σε προηγούμενα παραδείγματα. Στο σχήμα 7.1(β) βλέπουμε το αποτέλεσμα της εφαρμογής του ιεραρχικού αλγορίθμου. Στο σχήμα 7.1(γ) βλέπουμε το αποτέλεσμα της εφαρμογής του wavelet clustering, ενώ στο σχήμα 7.1(δ) μπορούμε να δούμε το αποτέλεσμα της τροποποίησης της μεθόδου με τον τρόπο που προαναφέραμε. Παρατηρούμε (βλ. σχ. 7.1(δ)), ότι τώρα ανιχνεύονται οι κατακόρυφες γραμμές στα αριστερά και δεξιά όρια της εικόνας, αλλά και η οριζόντια γραμμή στο κάτω μέρος, πράγμα στο οποίο έχουν αποτύχει οι μέθοδοι των σχημάτων 7.1(β) και 7.1(γ).



(α)

(β)

(γ)

(δ)

Σχήμα 7.1 (α) Αρχική εικόνα. (β) Αποτέλεσμα του ιεραρχικού αλγορίθμου. (γ) Αποτέλεσμα του wavelet clustering. (δ) Αποτέλεσμα του wavelet clustering με την τροποποίηση την οποία προτείναμε (segmentation σε όλες τις υποεικόνες).

Εκτός από την προσπάθεια για segmentation και στις υπόλοιπες τρεις υποεικόνες, κάτι άλλο το οποίο δοκιμάστηκε, ήταν να γίνει το segmentation στην LP-LP υποεικόνα όπως γινόταν αρχικά, λαμβάνοντας όμως τώρα υπόψη μας και την πληροφορία που μας έδιναν οι υπόλοιπες τρεις υποεικόνες. Για παράδειγμα, αυτό το οποίο έγινε ήταν η εφαρμογή του adaptive αλγόριθμου να γίνεται όπως και πριν στην low pass υποεικόνα, μόνο που, σε περίπτωση που στα γειτονικά σημεία του pixel το οποίο εξέταζε ο αλγόριθμος φαίνονταν ακμές (τις οποίες αντιλαμβανόταν ο αλγόριθμος παρατηρώντας τις HL, LH, HH υποεικόνες), ο αλγόριθμος το λάμβανε υπόψη του και φρόντιζε να μην κατατάξει το pixel σε καμία από τις περιοχές τις οποίες εκπροσωπούσαν τα pixels των ακμών. Στην περίπτωση αυτή τα αποτελέσματα δεν ήταν διαφορετικά από τα προηγούμενα. Ούτε καλύτερα, ούτε χειρότερα. Δεν είδαμε δηλαδή κάποια βελτίωση χρησιμοποιώντας το τέχνασμα αυτό αλλά και ούτε κάποια επιδείνωση των αποτελεσμάτων. Ο πιο πιθανός λόγος για τον οποίο δεν παρατηρήσαμε κάποια βελτίωση, είναι ότι ο adaptive αλγόριθμος δίνει ούτως ή άλλως καλά αποτελέσματα και διαχωρίζει ούτως ή άλλως τις διαφορετικές περιοχές σε μία εικόνα, έτσι ώστε η επιπλέον πληροφορία που του παρέχουν οι άλλες τρεις υποεικόνες (με τον τρόπο τουλάχιστον που χρησιμοποιήθηκε) να μην του είναι ιδιαίτερα χρήσιμη.

Ωστόσο, το γεγονός ότι οι προηγούμενες δύο προσπάθειες μας δεν είχαν επιτυχία, μας οδηγεί σε μία νέα κατεύθυνση για περαιτέρω έρευνα. Είδαμε ότι η εφαρμογή του adaptive αλγορίθμου ανεξάρτητα στην κάθε υποεικόνα δεν απέδωσε, ούτε όμως η εφαρμογή του adaptive αλγορίθμου στην LL υποεικόνα με απλό έλεγχο στις άλλες τρεις

υποεικόνες δεν έδωσε καλύτερα αποτελέσματα. Ίσως λοιπόν, μία εφαρμογή του adaptive αλγορίθμου και στις τέσσερις υποεικόνες **ταυτόχρονα**, να έδινε καλύτερα αποτελέσματα. Στην περίπτωση αυτή, θα χρειαζόταν να ορίσουμε "κλίκες" (βλ. Κεφάλαιο 4, παρ. 4.2) τριών διαστάσεων και όχι δύο όπως αυτές του Κεφαλαίου 4. Φυσικά η τρίτη διάσταση θα είχε να κάνει αποκλειστικά με την πληροφορία (των ακμών προφανώς) από τις άλλες τρεις υποεικόνες.

Μία άλλη βελτίωση η οποία θα μπορούσε να εφαρμοστεί, θα ήταν η εφαρμογή του ιεραρχικού αλγορίθμου (εννοούμε τον αλγόριθμο που παρουσιάσαμε στην παράγραφο 4.5) στο πεδίο του wavelet. Αντί δηλαδή να εφαρμόζουμε τον adaptive αλγόριθμο στην υποεικόνα (ή στις υποεικόνες) για το segmentation, θα μπορούσαμε να εφαρμόσουμε τον ιεραρχικό αλγόριθμο, ο οποίος, όπως είδαμε, και καλύτερα αποτελέσματα φαίνεται να δίνει και το χρόνο εκτέλεσης μειώνει σημαντικά. Προς αυτήν την κατεύθυνση, δηλαδή τη μείωση του χρόνου εκτέλεσης, θα μπορούσαμε να εφαρμόσουμε τεχνικές βελτιστοποίησης στα προγράμματά μας παρεμβαίνοντας τόσο στην υλοποίηση του adaptive (ή του ιεραρχικού) αλγορίθμου όσο και στην υλοποίηση του ευθύ και αντίστροφου μετασχηματισμού wavelet. Ενώ δηλαδή η υλοποίηση που έχουμε σχεδιάσει (όσων αφορά την εφαρμογή των φίλτρων wavelet) εκτελεί τη συνέλιξη του κάθε φίλτρου με την εικόνα, εναλλακτικά θα μπορούσε να υλοποιηθεί με τη βοήθεια του μετασχηματισμού Fourier ο οποίος για μεγάλες εικόνες προφανώς θα δούλευε αρκετά γρηγορότερα.

Ενδιαφέρον επίσης θα παρουσίαζε η εφαρμογή του μετασχηματισμού wavelet σε παραπάνω από ένα επίπεδα. Όπως φάνηκε από την ανάλυση που προηγήθηκε, ο μετασχηματισμός wavelet που χρησιμοποιήσαμε ήταν πάντα ενός επιπέδου. Εφαρμόζαμε δηλαδή σε κάθε περίπτωση το μετασχηματισμό μόνο μία φορά, και κατόπιν επεξεργαζόμασταν τις υποεικόνες που προέκυπταν. Ωστόσο αυτό θα μπορούσε να συνεχιστεί (να εφαρμόσουμε δηλαδή ξανά μετασχηματισμό wavelet) και να περάσουμε έτσι σε ακόμα χαμηλότερα επίπεδα ανάλυσης. Θα ήταν ενδιαφέρον να δούμε πως θα επηρέαζε τις διαδικασίες που θα εφαρμόζαμε στα επίπεδα αυτά η χαμηλή τους ανάλυση. Και επίσης ενδιαφέρον θα είχε να δούμε, με ποιο τρόπο θα μπορούσαμε εφαρμόζοντας segmentation στο χαμηλότερο δυνατό επίπεδο ανάλυσης, να επιστρέψουμε κατόπιν στην υψηλή ανάλυση και στις πραγματικές διαστάσεις της εικόνας μας και το αποτέλεσμα να μοιάζει σαν να είχαμε εφαρμόσει segmentation στην αρχική εικόνα. Κάτι τέτοιο βέβαια

θα επέφερε νέα μεγάλη μείωση του χρόνου εκτέλεσης της μεθόδου. Όμως, πραγματοποίηση όλων αυτών που αναφέρθηκαν παραπάνω, απαιτεί την πλήρη κατανόηση της λειτουργίας του μετασχηματισμού στο πρώτο επίπεδο και την ολοκλήρωση της προσπάθειας που ξεκινήσαμε (για segmentation και των υπολοίπων εικόνων).

Όσον αφορά μεθόδους με τις οποίες θα μπορούσαμε να μειώσουμε την επίδραση του θορύβου στο τελικό αποτέλεσμα, αυτό το οποίο μπορεί να δοκιμαστεί είναι ένας συνδυασμός μιας τεχνικής thresholding με τον μετασχηματισμό Hough **[1]**, ώστε ναι μεν να διατηρούνται οι ακμές στις HL, LH, HH υποεικόνες αλλά και ο θόρυβος να μειώνεται δραστικά στις υποεικόνες αυτές και συνεπώς η επίδρασή του στην τελική εικόνα (υπενθυμίζουμε ότι εμείς χρησιμοποιήσαμε απλό thresholding για την εξάλειψη του θορύβου).

Τέλος, ενδιαφέρον θα παρουσίαζε και η μελέτη της συμπεριφοράς των φίλτρων wavelet (όταν αυτά εφαρμόζονται σε εικόνες), και η προοπτική του σχεδιασμού νέων φίλτρων με ακόμα καλύτερες ιδιότητες για την επίτευξη ακόμα καλύτερων αποτελεσμάτων.

ΚΕΦΑΛΑΙΟ 8: ΟΔΗΓΟΣ ΧΡΗΣΗΣ ΥΛΟΠΟΙΗΜΕΝΩΝ ΠΡΟΓΡΑΜΜΑΤΩΝ

8.1 ΥΛΟΠΟΙΗΣΗ

Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση των αλγορίθμων που παρουσιάσαμε στα προηγούμενα κεφάλαια ήταν η C++. Η υλοποίηση ξεκίνησε στο PC και στην BORLAND C++ αλλά επειδή οι απαιτήσεις σε μνήμη κάποιες φορές έγιναν πολύ μεγάλες (πάνω από το όριο του 1 Mbyte της BORLAND), γι' αυτό σύντομα τα προγράμματα μεταφέρθηκαν στο UNIX και στην GNU C++, στο οποίο δεν υπάρχει κανένας περιορισμός μνήμης (καθώς όλη η μνήμη του συστήματος είναι στη διάθεση του χρήστη).

8.2 ΠΡΟΓΡΑΜΜΑΤΑ

Μαζί με το κείμενο της διπλωματικής εργασίας έχει παραδωθεί και δισκέτα με τα προγράμματα τα οποία υλοποιήθηκαν. Στις επόμενες παραγράφους περιγράφουμε αναλυτικά τη λειτουργία του κάθε προγράμματος, τις εισόδους τις οποίες ζητάει το κάθε πρόγραμμα κατά την εκτέλεσή του και τον τρόπο με τον οποίο γίνεται το compilation του κάθε προγράμματος. Πριν περάσουμε όμως στην περιγραφή των προγραμμάτων θα αναφέρουμε κάποιες πληροφορίες που ισχύουν για όλα τα προγράμματα.

Πρώτα απ' όλα όλα τα προγράμματα παίρνουν σαν είσοδο grayscale εικόνες με format BMP. Όλα τα προγράμματα εκτυπώνουν κάποιες πληροφορίες και διαγνωστικά μηνύματα καθώς εκτελούνται. Επίσης, οτιδήποτε λάθος συμβεί (πρόβλημα μνήμης, πρόβλημα ανοίγματος-κλεισίματος αρχείου, πρόβλημα ανάγνωσης-εγγραφής σε αρχείο), το εκάστοτε πρόγραμμα τερματίζει και τυπώνεται στην οθόνη το ανάλογο μήνυμα λάθους. Μετά το τέλος της εκτέλεσης του εκάστοτε προγράμματος, το αποτέλεσμα βρίσκεται στο αρχείο εικόνας **"out.bmp"** το οποίο μπορούμε να δούμε πληκτρολογώντας:

xv out.bmp

8.2.1 Πρόγραμμα βέλτιστου χωρισμού (splitting) εικόνας σε πολύγωνα.

Το συγκεκριμένο πρόγραμμα υλοποιεί τον αλγόριθμο ROFS που περιγράψαμε στο 3ο Κεφάλαιο. Το αρχείο είναι το **"rofs.C"**, οι δομές που χρησιμοποιούμε ορίζονται μέσα στο αρχείο **"define.h"** ενώ ο τρόπος με τον οποίο γίνεται το compilation του αρχείου είναι πληκτρολογώντας:

g++ -O2 rofs.C

όπου η παράμετρος "O2" χρησιμοποιείται για την παραγωγή από τον compiler γρήγορου κώδικα. Ως γνωστόν, μετά το compilation, το εκτελέσιμο αρχείο θα είναι το **"a.out"**. Ωστόσο αν επιθυμούμε να δώσουμε στο εκτελέσιμο αρχείο ένα όνομα διαφορετικό από το "a.out", μπορούμε, αντί για την παραπάνω γραμμή, να πληκτρολογήσουμε:

g++ -O2 -o <όνομα εκτελέσιμου που επιθυμούμε> rofs.C

Κατόπιν, αν για παράδειγμα ονομάσαμε το εκτελέσιμο αρχείο **"ROFS"**, το εκτελούμε γράφοντας:

ROFS <αρχείο εικόνας το οποίο θέλουμε>

Οι παραπάνω πληροφορίες για τον τρόπο μετονομασίας του εκτελέσιμου αρχείου και εκτέλεσης του ισχύουν και για όλα τα προγράμματα που θα ακολουθήσουν (απλά μόνο το όνομα του προγράμματος θα αλλάζει κάθε φορά). Από δω και στο εξής, για όλα τα υπόλοιπα προγράμματα θα δίνουμε μόνο τον τρόπο με τον οποίο γίνεται το compilation του καθενός.

Η μοναδική είσοδος που μας ζητάει το πρόγραμμα στην αρχή της εκτέλεσής του είναι το κατώφλι ε (βλ. κεφάλαιο 3). Δίνοντας μία τιμή για το κατώφλι ε, το πρόγραμμα ξεκινάει την λειτουργία του.

8.2.2 Πρόγραμμα ανίχνευσης ακμών.

Το συγκεκριμένο πρόγραμμα είναι η επέκταση του προηγουμένου, καθώς αυτό το οποίο κάνει επιπλέον, είναι το merging των πολυγώνων που προέκυψαν από την εφαρμογή του αλγορίθμου ROFS. Το αρχείο είναι το **"edges.C"**, οι δομές που χρησιμοποιούμε ορίζονται μέσα στο αρχείο **"define.h"** ενώ ο τρόπος με τον οποίο γίνεται το compilation του αρχείου είναι πληκτρολογώντας:

g++ -O2 edges.C

Το πρόγραμμα κατά την εκκίνηση του, εκτός από το κατώφλι ε, μας ζητάει και μία τιμή για την **παράμετρο merging** (βλ. Κεφάλαιο 3).

8.2.3 Πρόγραμμα εφαρμογής του αλγορίθμου K-means σε εικόνα.

Το συγκεκριμένο πρόγραμμα εφαρμόζει τον αλγόριθμο K-means σε μία εικόνα. Το αρχείο είναι το **"Kmeans.C"** και ο τρόπος με τον οποίο γίνεται το compilation του προγράμματος είναι πληκτρολογώντας:

g++ -O2 Kmeans.C -lm

όπου η παράμετρος "lm" χρησιμοποιείται για να γίνει "link" το προγράμμα μας με την βιβλιοθήκη μαθηματικών της C, καθώς έχουμε χρησιμοποιήσει κάποιες μαθηματικές συναρτήσεις.

Το πρόγραμμα κατά την εκκίνηση του, αυτό που ζητά είναι ο αριθμός των περιοχών που θα ανιχνευθούν και κατόπιν τα κέντρα των clusters (συντεταγμένες και gray level) από τα οποία θα ξεκινήσει ο αλγόριθμος.

8.2.4 Πρόγραμμα εφαρμογής του adaptive αλγορίθμου σε εικόνα (εξαντλητική υλοποίηση).

Το πρόγραμμα αυτό υλοποιεί τον adaptive αλγόριθμο που παρουσιάσαμε στο Κεφάλαιο 4 (παρ. 4.3) υλοποιώντας τον εξαντλητικά. Το αρχείο είναι το **"exhaust.C"** ενώ το compilation γίνεται πληκτρολογώντας:

g++ -O2 exhaust.C -lm

Το πρόγραμμα κατά την εκκίνηση του ζητάει ότι ακριβώς και αυτό της προηγούμενης παραγράφου, και επιπλεόν την παράμετρο σ (τυπική απόκλιση του θορύβου).

8.2.5 Πρόγραμμα εφαρμογής του adaptive αλγορίθμου σε εικόνα (υλοποίηση με πλέγμα και χρήση bilinear interpolation).

Το πρόγραμμα αυτό υλοποιεί τον adaptive αλγόριθμο που παρουσιάσαμε στο Κεφάλαιο 4 (παρ. 4.3) υλοποιώντας τον όμως χρησιμοποιώντας το πλέγμα του σχήματος 4.3 και χρησιμοποιώντας bilinear interpolation για τα ενδιάμεσα σημεία. Το αρχείο είναι το **"fullgrid.C"** ενώ το compilation γίνεται πληκτρολογώντας:

```
g++ -O2  fullgrid.C  -lm
```

Το πρόγραμμα κατά την εκκίνηση του ζητάει ότι ακριβώς και αυτό της προηγούμενης παραγράφου. Από δω και στο εξής, όλα τα υπόλοιπα προγράμματα με μοναδική εξαίρεση αυτό της παραγράφου 8.2.7, ζητούν τις ίδιες εισόδους με το παρόν πρόγραμμα.

8.2.6 Πρόγραμμα εφαρμογής του ιεραρχικού αλγορίθμου σε εικόνα.

Το πρόγραμμα αυτό υλοποιεί τον ιεραρχικό αλγόριθμο που παρουσιάσαμε στο Κεφάλαιο 4 (παρ. 4.5). Το αρχείο είναι το **"ierarxik.C"** ενώ το compilation γίνεται πληκτρολογώντας:

```
g++ -O2  ierarxik.C  -lm
```

8.2.7 Πρόγραμμα εφαρμογής του μετασχηματισμού wavelet σε εικόνα.

Το πρόγραμμα αυτό υλοποιεί τον μετασχηματισμό wavelet που παρουσιάσαμε στο Κεφάλαιο 6 (παρ. 6.1.3). Το αρχείο είναι το **"wavtrans.C"** ενώ το compilation γίνεται πληκτρολογώντας:

```
g++ -O2  wavtrans.C  -lm
```

Το συγκεκριμένο πρόγραμμα δε ζητάει κάποια είσοδο κατά την εκτέλεση του.

8.2.8 Πρόγραμμα εφαρμογής του αλγορίθμου wavelet clustering σε εικόνα.

Το πρόγραμμα αυτό υλοποιεί τον αλγόριθμο wavelet clustering που παρουσιάσαμε στο Κεφάλαιο 6 (παρ. 6.3.1). Το αρχείο είναι το **"wavclust.C"** ενώ το compilation γίνεται πληκτρολογώντας:

```
g++ -O2  wavclust.C  -lm
```

Το πρόγραμμα ζητάει τις ίδιες ειδόδους με αυτές των προγραμμάτων των παραγράφων 8.2.4 - 8.2.6.

ΠΕΡΙΕΧΟΜΕΝΑ