ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ψηφιακή Επεξεργασία Βίντεο Σε Πραγματικό Χρόνο

ΗΛΙΑΣ ΦΛΑΟΥΝΑΣ
ΑΜ: 1172

Επιβλέπων Καθηγητής: Δημήτριος Μαρούλης

ΑΘΗΝΑ
ΙΟΥΛΙΟΣ 2003

Περιεχόμενα

Πρόλογος

Η πτυχιακή αυτή εργασία με θέμα την «Ψηφιακή Επεξεργασία Βίντεο σε Πραγματικό Χρόνο» γράφτηκε κατά τη διάρκεια του ακαδημαϊκού έτους 2002-2003 υπό την εποπτεία του επίκουρου καθηγητή Δημήτριου Μαρούλη.

Θα ήθελα να εκφράσω τις ευχαριστίες μου στον κύριο Μαρούλη για την υποστήριξη του, τις συμβουλές του και για την γενικότερη καθοδήγηση του καθ' όλη τη διάρκεια της εκπόνησης αυτής της εργασίας. Παράλληλα θα ήθελα να ευχαριστήσω και τους Σταύρο Καρκάνη και Δημήτρη Ιακωβίδη για την καταλυτική βοήθεια που μου προσέφεραν.

Φλαούνας Ηλίας

1 Εισαγωγή

Στόχος αυτής της πτυχιακής εργασίας είναι η ψηφιακή επεξεργασία βίντεο σε πραγματικό χρόνο. Τα βίντεο με τα οποία ασχολείται είναι τα AVI. Τα AVI είναι ο τρόπος αποθήκευσης κινούμενης εικόνας βίντεο που έχει αναπτυχθεί από την Microsoft και χρησιμοποιείται ευρέως στα MS-Windows. Η επεξεργασία του βίντεο αφορά την σειριακή εφαρμογή φίλτρων επεξεργασίας εικόνας στα καρέ του και αναγνώρισης προτύπων σε αυτά. Για τις υλοποιήσεις των φίλτρων επεξεργασίας εικόνας έχει χρησιμοποιηθεί η βιβλιοθήκη IPL της Intel. Η αναγνώριση επιτυγχάνεται με χαρακτηριστικά υφής και με χρήση νευρωνικού δικτύου ως ταξινομητή.

Στα πλαίσια της εργασίας έχει αναπτυχθεί η εφαρμογή FPremier. Πρόκειται για ένα πρόγραμμα ψηφιακής επεξεργασίας βίντεο σε πραγματικό χρόνο. Παράλληλα έχουν δημιουργηθεί και τρεις βιβλιοθήκες δυναμικής διασύνδεσης, οι FAVI, FDIB και FIPL, οι οποίες προσφέρουν ένα αρκετά υψηλότερο επίπεδο εργασίας στο προγραμματιστή που ασχολείται με τα βίντεο AVI, τις εικόνες BMP και RAW καθώς και την ψηφιακή επεξεργασία εικόνας και βίντεο γενικότερα.

Οι εικόνες που παρουσιάζονται ή αναφέρονται στην εργασία αυτή μπορούν να βρεθούν στο συνοδευτικό CD.

1.1 Το Βίντεο ως Σήμα

Τα βίντεο όπως γνωρίζουμε είναι μια σειρά από εικόνες που διαδέχονται η μια την άλλη. Η διαδοχή αυτή γίνεται αρκετά γρήγορα και με σταθερό ρυθμό δίνοντας έτσι στο θεατή τους την ψευδαίσθηση της κινούμενης εικόνας. Οι εικόνες από τις οποίες αποτελείται ένα βίντεο ονομάζονται frames ή καρέ.

Μια εικόνα γενικά μπορεί να οριστεί ως μια δισδιάστατη συνάρτηση f(x,y). Τα x και y αποτελούν τις χωρικές συντεταγμένες και η τιμή της f σε ένα σημείο (x,y) του χώρου ορίζεται ως ένταση της φωτεινότητας στο συγκεκριμένο σημείο. Αν τα x, y και f(x,y) μιας εικόνας είναι πεπερασμένες διακριτές τιμές τότε είναι μια ψηφιακή εικόνα. Είναι φανερό λοιπόν ότι μια ψηφιακή εικόνα αποτελείται από ένα πεπερασμένο πλήθος στοιχείων, το καθένα με συγκεκριμένες συντεταγμένες και τιμή. Τα στοιχεία αυτά ονομάζονται pixels (Από τις λέξεις "**Pic**ture **el**ements") ή εικονοστοιχεία.

Με μια αντίστοιχη θεώρηση με αυτή που έγινε για τις εικόνες μπορούν να οριστούν και τα βίντεο ως σήματα τριών διαστάσεων. Απλά προστίθεται στη συνάρτηση μιας εικόνας f(x,y) μια ακόμα παράμετρος, ο χρόνος t. Ας θεωρήσουμε λοιπόν ότι έχουμε το βίντεο F(x,y,t). Η τιμή της F είναι η φωτεινότητα του εικονοστοιχείου με χωρικές συντεταγμένες (x,y) στο καρέ t του βίντεο. Τα x και y παίρνουν τιμές από 0 έως το πλάτος και ύψος αντίστοιχα του κάθε καρέ του βίντεο, ενώ ο χρόνος t παίρνει διακριτές τιμές από 0, που αποτελεί το πρώτο καρέ, έως T που αποτελεί το τελικό.

Αν η παράμετρος t είναι σταθερή και ίση με t_0, αναφερόμαστε σε ένα συγκεκριμένο καρέ. Έτσι το σήμα $F(x,y,t_0)$ μπορούμε να θεωρήσουμε ότι αποτελεί ένα δισδιάστατο σήμα $G_0(x,y)$. Αυτό αναπαριστά το συγκεκριμένο καρέ με αριθμό t_0, δηλαδή μία στατική εικόνα. Έτσι μπορούμε να εφαρμόσουμε πάνω στο G_0 κάθε γνωστό αλγόριθμο από την επεξεργασία εικόνας. Αυτή είναι γενικά από μαθηματική πλευρά η διαδικασία με την οποία προχωράμε στην ψηφιακή επεξεργασία του βίντεο βασιζόμενοι σε τεχνικές της ψηφιακής επεξεργασίας εικόνας.

1.2 Ψηφιοποίηση Βίντεο

Όλα τα σήματα που συναντώνται στη φύση είναι αναλογικά, δηλαδή συνεχή. Προκειμένου όμως να επεξεργαστούμε αναλογικά σήματα με ψηφιακά μέσα, όπως ένας υπολογιστής, θα πρέπει προηγουμένως να τα ψηφιοποιήσουμε. Το ίδιο ισχύει για εικόνες και βίντεο. Αν επεξεργαζόμαστε εικόνες αυτό γίνεται σε δυο βήματα: Το πρώτο είναι η δειγματοληψία και το δεύτερο η κβάντιση. Αν έχουμε βίντεο απαιτείται να καθοριστεί μια ακόμα παράμετρος, ο αριθμός των καρέ ανά δευτερόλεπτο.

1.2.1 Δειγματοληψία εικόνας

Η δειγματοληψία ενός μονοδιάστατου συνεχούς σήματος είναι να πάρουμε έναν αριθμό δειγμάτων του με σταθερό ρυθμό, δημιουργώντας έτσι μια σειρά από διακριτές τιμές. Κάθε δείγμα μπορεί να έχει οποιαδήποτε τιμή εντός διαστήματος αφού και το σήμα από το οποίο προήλθε είναι συνεχές.

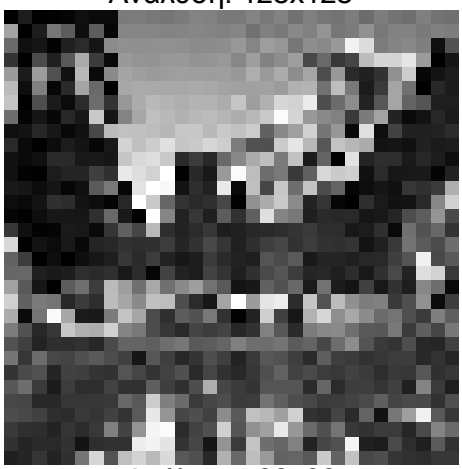
Κατά αναλογία με τα μονοδιάστατα σήματα, ένα δισδιάστατο σήμα δειγματοληπτείται βάζοντας ένα "πλέγμα" από τετράγωνα πάνω σε αυτό και ορίζοντας μια τιμή για κάθε ένα από αυτά που εξαρτάται από τις αντίστοιχες τιμές του σήματος που συναντώνται στην περιοχή του. Αν το δισδιάστατο σήμα είναι μια εικόνα το κάθε ένα από αυτά τα τετράγωνα ονομάζεται εικονοστοιχείο. Η ψηφιακή εικόνα λοιπόν αποτελείται από έναν δισδιάστατο πίνακα από εικονοστοιχεία. Η δειγματοληψία ή αλλιώς ανάλυση μετριέται με τον αριθμό των εικονοστοιχείων που χρησιμοποιούνται σε κάθε μία από τις διαστάσεις της εικόνας. Για παράδειγμα μια συνήθης τιμή ανάλυσης εικόνας είναι 256x256, που σημαίνει ότι έχουμε έναν τετράγωνο πίνακα με 256 στοιχεία σε κάθε διάσταση.

Ας θεωρήσουμε ότι έχουμε μια αρχική εικόνα με ανάλυση 256x256 (σχήμα 1). Αν μειώσουμε την ανάλυση της εικόνας κρατώντας σταθερό το μέγεθος με το οποίο την απεικονίζουμε είναι εύκολο να διαπιστώσουμε τα αποτελέσματα που έχει η μειωμένη δειγματοληψία. Ήδη από την μειωμένη ανάλυση 128x128 γίνεται εμφανές το φαινόμενο της σκακιέρας, γίνονται εμφανή δηλαδή τα εικονοστοιχεία από την οποία αποτελείται η εικόνα. Σε ακόμα μικρότερες αναλύσεις η εικόνα έχει πλέον παραμορφωθεί.

Ανάλυση: 256x256

Ανάλυση: 128x128

Ανάλυση: 64x64



Ανάλυση: 32x32

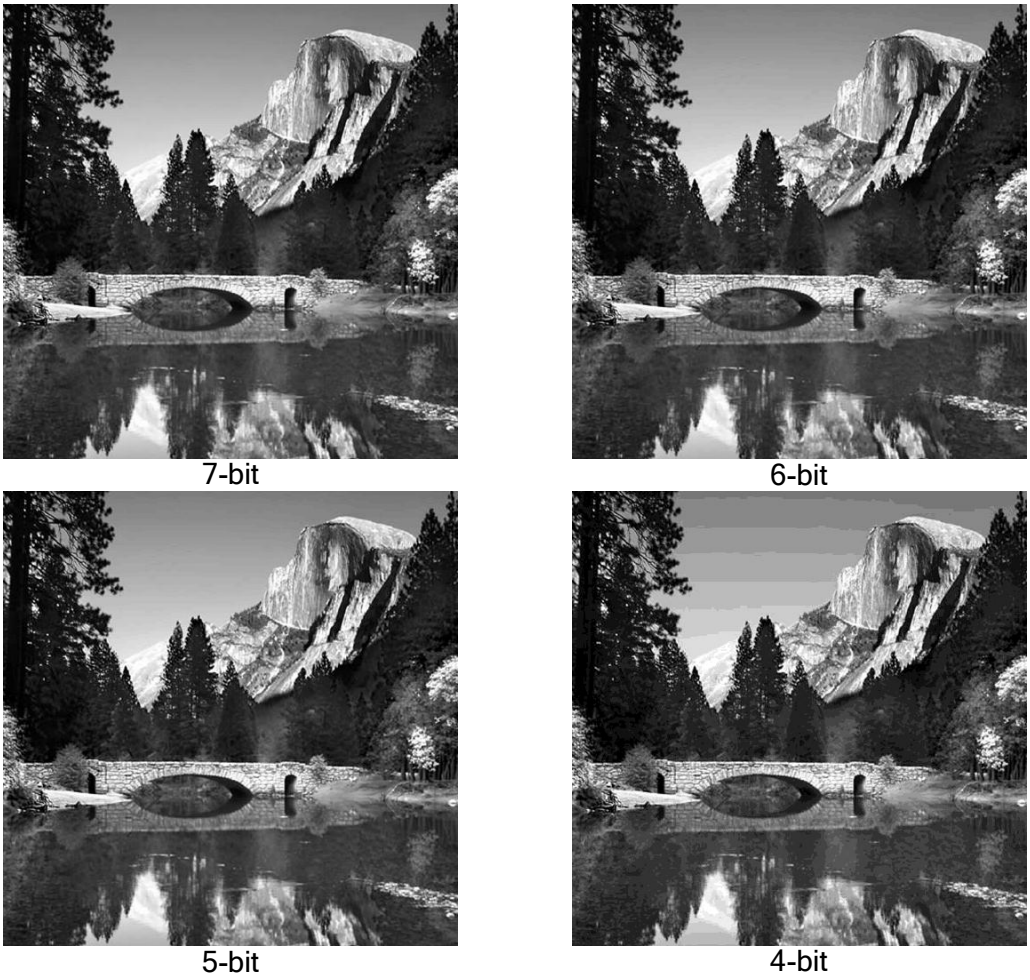
Σχήμα 1. Αποτελέσματα μείωσης της δειγματοληψίας στην αρχική εικόνα 'bridge256x256' ανάλυσης 256x256.

1.2.2 Κβάντιση εικόνας

Η κβάντιση αποσκοπεί στην αναπαράσταση των δειγμάτων που έχουν προκύψει από την δειγματοληψία, που μπορούν να έχουν οποιαδήποτε τιμή, με ένα πεπερασμένο πλήθος τιμών. Ο πιο συνήθης κβαντιστής είναι ο ομοιόμορφος κβαντιστής. Δηλαδή χωρίζουμε το πεδίο τιμών των εικονοστοιχείων σε έναν καθορισμένο αριθμό από ισοαπέχουσες στάθμες Στη συνέχεια κάθε εικονοστοιχείο ανάλογα με τη τιμή του κατατάσσεται και στην αντίστοιχη στάθμη. Αυτός ο κβαντιστής είναι ο πιο απλός. Έχει τα ίδια χαρακτηριστικά για κάθε εικόνα και δεν λαμβάνει υπ' όψιν του καθόλου το σφάλμα κβάντισης που υπεισέρχεται. Γενικά ανάλογα με τις απαιτήσεις της εφαρμογή μπορεί να επιλεγεί κάποιος άλλος κβαντιστής με διαφορετικά χαρακτηριστικά όπως για παράδειγμα την ελαχιστοποίηση του μέσου τετραγωνικού σφάλματος κβάντισης που υπεισέρχεται στην εικόνα (Κβαντιστής Loyd-Max). Αναλυτικές πληροφορίες για σχεδίαση κβαντιστών μπορούν να βρεθούν στα [1] και [2].

Το πλήθος των επιπέδων κβάντισης ονομάζεται και βάθος χρώματος μιας εικόνας. Αντιπροσωπεύει τον αριθμό των επιπέδων του γκρι αν είναι ασπρόμαυρη ή τον αριθμό των χρωμάτων της αν είναι έγχρωμη. Μετριέται με τον αριθμό των bits που χρησιμοποιούνται για την αναπαράσταση των χρωματικών επιπέδων. Για παράδειγμα

συναντάμε εικόνες με βάθος χρώματος 7-bit ή 8-bit όπου αντίστοιχα έχουμε 128 και 256 χρωματικές στάθμες.



Σχήμα 2. Αποτελέσματα μείωσης κβάντισης στην αρχική εικόνα 'bridge' βάθους χρώματος 7-bit.

Μια μικρή μείωση στο βάθος χρώματος της εικόνας στα 6-bit δεν γίνεται εύκολα αντιληπτή. Αυτό βέβαια είναι υποκειμενικό και εξαρτάται άμεσα από την εικόνα. Μικρότερες τιμές όμως προκαλούν παραμόρφωση και οι εικόνες είναι πλέον πολύ χαμηλής ποιότητας.

Όπως μπορούμε να παρατηρήσουμε από τα σχήματα 1 και 2 η πιστότητα της ψηφιακής εικόνας σε σχέση με την αναλογική είναι άμεση συνάρτηση τόσο του ρυθμού δειγματοληψίας όσο και της κβάντισης. Όσο μειώνονται οι τιμές των δύο αυτών παραμέτρων τόσο υποβαθμίζεται η ποιότητα της.

1.2.3 Αριθμός καρέ ανά δευτερόλεπτο

Κατά την ψηφιοποίηση ενός βίντεο πέρα από την ανάλυση και το βάθος χρώματος που έχουμε, υπάρχει ακόμα μια πολύ σημαντική παράμετρος που πρέπει να υπολογίζεται. Η τρίτη παράμετρος είναι ο αριθμός των καρέ ανά δευτερόλεπτο (FPS - **F**rames **P**er **S**ec) που χρησιμοποιούνται για την ψηφιοποίηση του βίντεο. Η τιμή αυτής της παραμέτρου συνήθως κυμαίνεται ανάμεσα στο 25 και 30 FPS. Τιμές

μικρότερες από τα 25 FPS ρίχνουν πολύ την ποιότητα του βίντεο αφού οι εναλλαγές των εικόνων γίνονται αντιληπτές από το ανθρώπινο μάτι. Αυτό έχει ως αποτέλεσμα μια ομαλή κίνηση να φαίνεται στο βίντεο σπασμωδική. Αντίστοιχα τιμές μεγαλύτερες από τα 30 FPS δεν γίνονται αντιληπτές από το ανθρώπινο μάτι. Δηλαδή όχι μόνο δεν ανεβάζουν την ποιότητα του βίντεο αλλά αντίθετα μεταφέρουν και μεγάλη πλεονάζουσα πληροφορία που δεν γίνεται αντιληπτή. Για παράδειγμα στο σύστημα τηλεόρασης NTSC έχουμε 30 FPS ενώ στο PAL έχουμε 25 FPS. Βέβαια σε ειδικές περιπτώσεις εφαρμογών μπορούν να απαιτούνται και τιμές μεγαλύτερες από τα 30 FPS. Γενικά η επιλογή της κατάλληλης τιμής του αριθμού των καρέ ανά δευτερόλεπτο εξαρτάται αποκλειστικά από την εφαρμογή.

1.3 Έγχρωμη Επεξεργασία Εικόνας

Μέχρι τώρα έχει γίνει αναφορά μόνο σε ασπρόμαυρες εικόνες. Σε αυτήν την παράγραφο θα παρουσιαστεί ο τρόπος αποθήκευσης και επεξεργασίας έγχρωμης εικόνας και των καρέ του βίντεο γενικότερα. Θα γίνει αναφορά στα χρωματικά μοντέλα RGB, YIQ και HSV που είναι από τα πλέον χρησιμοποιούμενα στην επεξεργασία εικόνας καθώς και στον τρόπο μετατροπής μιας RGB εικόνας σε ασπρόμαυρη εικόνα έντασης.

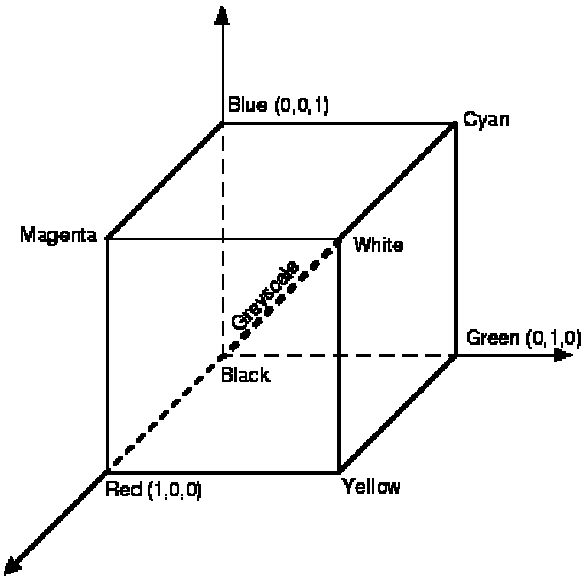
1.3.1 Χρωματικά Μοντέλα

Ο όρος χρωματικό μοντέλο χρησιμοποιείται για να περιγράψει ένα τρόπο κωδικοποίησης της χρωματικής πληροφορίας σε μια εικόνα. Γενικά ένα χρωματικό μοντέλο ορίζει ένα σύστημα συντεταγμένων και έναν υποχώρο μέσα σε αυτό όπου κάθε χρώμα αναπαρίσταται σε ένα μοναδικό σημείο [2]. Υπάρχουν πολλά χρωματικά μοντέλα και κάθε ένα από αυτά έχει ως στόχο να εξυπηρετήσει τις διαφορετικές ανάγκες συστημάτων και εφαρμογών που χρησιμοποιούν χρώμα.

Στην συνέχεια θα παρουσιάσουμε τα χρωματικά μοντέλα RGB, YIQ και HSV τα οποία είναι από τα πλέον συχνά χρησιμοποιούμενα στην επεξεργασία εικόνας.

1.3.1.1 Το χρωματικό μοντέλο RGB

Ένα από τα πιο βασικά χρωματικά μοντέλα είναι το RGB. Τα τρία γράμματα του ονόματός του αντιπροσωπεύουν τα τρία βασικά του χρώματα: **R**ed-Κόκκινο, **G**reen-Πράσινο και **B**lue-Μπλε. Ορίζει ένα τρισδιάστατο Καρτεσιανό σύστημα συντεταγμένων όπου κάθε άξονας αντιπροσωπεύει και ένα από τα βασικά του χρώματα (Σχήμα 3).



Σχήμα 3. Το χρωματικό μοντέλο RGB.

Κάθε χρώμα αντιπροσωπεύει και ένα διάνυσμα μέσα στο χώρο. Αν αναλυθεί διανυσματικά στις τρεις συνιστώσες των βασικών χρωμάτων, οι τιμές που παίρνουμε αντιπροσωπεύουν την συμμετοχή του αντίστοιχου βασικού χρώματος στο σχηματισμό του αρχικού. Έτσι για παράδειγμα το μοβ χρώμα (magenta) αντιστοιχείται στο διάνυσμα (R,G,B)=(1,0,1) και το κίτρινο στο (R,G,B)=(1,1,0). Στην αρχή των αξόνων (0,0,0) έχουμε το μαύρο χρώμα και στο σημείο (1,1,1) έχουμε το άσπρο. Κατά μήκος της ευθείας που διέρχεται από την αρχή των αξόνων και το σημείο (1,1,1) βρίσκονται όλες οι αποχρώσεις του γκρι που συναντούνται στις ασπρόμαυρες εικόνες.

Τα χρώματα όπως αναπαριστούνται στο σχήμα 3 είναι «κανονικοποιημένα» στο συνεχές διάστημα [0,1]. Πρακτικά δεν μπορούμε να έχουμε πραγματικούς αριθμούς για τις χρωματικές συνιστώσες οπότε κάνουμε ομοιόμορφη κβάντιση όπως και στην περίπτωση των ασπρόμαυρων εικόνων. Εδώ όμως έχουμε τρία χρωματικά κανάλια αντί για ένα. Έτσι αν έχουμε μια RGB εικόνα 24bit (που είναι και η πιο συνήθης περίπτωση έγχρωμων εικόνων) σημαίνει ότι χρησιμοποιούμε 8bit ή 256 στάθμες κβάντισης για το κάθε χρωματικό κανάλι. Συνολικά δηλαδή μπορούμε να αναπαραστήσουμε $256^3 \approx 16,7$εκατ. διαφορετικά χρώματα.

1.3.1.2 Το χρωματικό μοντέλο YIQ

Το YIQ χρησιμοποιήθηκε πρωταρχικά στις έγχρωμες τηλεοράσεις προκειμένου να εξασφαλιστεί η προς τα πίσω συμβατότητα με τις ασπρόμαυρες. Έτσι η πρώτη συνιστώσα του Y (Luminance) είναι αυτή που χρησιμοποιείται από τις ασπρόμαυρες τηλεοράσεις και αποτελεί τη φωτεινότητα του κάθε pixel. Δεν περιέχει καθόλου χρωματική πληροφορία. Αυτή είναι μοιρασμένη στα άλλα δυο κανάλια το I (Inphase) και Q (Quadrature). Σχεδιάστηκε για να εκμεταλλεύεται την μεγαλύτερη ευαισθησία του ανθρώπινου ματιού στις αλλαγές της φωτεινότητας σε σχέση με τις αλλαγές χρώματος και κορεσμού. Έτσι χρησιμοποιεί μεγαλύτερο εύρος ζώνης για την αναπαράσταση του Y παρά των I και Q.

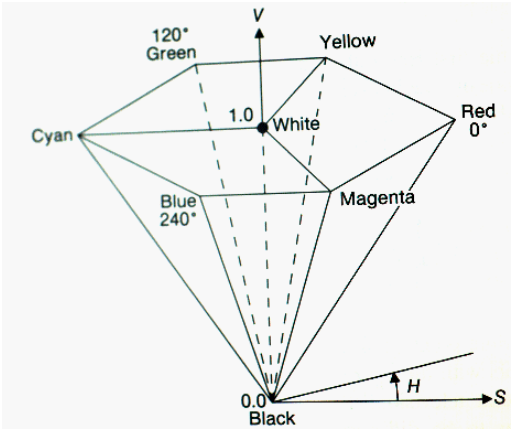
Ο τρόπος μετατροπής μιας RGB εικόνας σε YIQ γίνεται με τη χρήση του μετασχηματισμού που δίνεται στο σχήμα 4 και που εφαρμόζεται σε κάθε pixel της εικόνας.

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.275 & -0.321 \\ 0.212 & -0.523 & 0.311 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

Σχήμα 4. Μετασχηματισμός μετατροπής μιας εικόνας RGB σε YIQ.

1.3.1.3 Το χρωματικό μοντέλο HSV

Στο HSV μοντέλο τα τρία κανάλια αντιστοιχούν στις παραμέτρους Hue, Saturation και Value. Η πρώτη παράμετρος εκφράζει το χρώμα ενώ οι άλλες δύο συνδέονται με την προσθήκη και αφαίρεση άσπρου και μαύρου αντίστοιχα. Το μοντέλο αυτό γεωμετρικά αναπαρίσταται με μία εξάπλευρη πυραμίδα. Κάθε μια από τις έξι κορυφές της βάσης αντιστοιχεί ένα από τα χρώματα κόκκινο, κίτρινο, πράσινο κυανό, μπλε και μοβ. Ο προσδιορισμός του Hue γίνεται με τον υπολογισμό της γωνίας ανάμεσα στην ακτίνα του κόκκινου, που αντιστοιχεί στις 0 μοίρες, και το χρώμα που επιθυμούμε. Το Value εκφράζει την απόσταση του από την βάση της πυραμίδας και το Saturation την απόσταση από τον κεντρικό άξονα της πυραμίδας.



Σχήμα 5. Το χρωματικό μοντέλο HSV.

1.3.2 Μετατροπή Έγχρωμης RGB εικόνας σε ασπρόμαυρη

Το χρωματικό μοντέλο RGB δεν προσφέρει άμεσα τη δυνατότητα να γνωρίζουμε την ένταση της φωτεινότητας ενός εικονοστοιχείου. Μια χρήσιμη διαδικασία λοιπόν, είναι η μετατροπή μιας έγχρωμης RGB εικόνας σε ασπρόμαυρη εικόνα έντασης (Grayscale - Αποχρώσεων του γκρι). Αυτό γίνεται δημιουργώντας από την εικόνα των τριών καναλιών R, G και B του χρωματικού μοντέλου RGB μια νέα, ενός μόνο καναλιού I (**I**ntensity - Ένταση), που αντιστοιχεί στην φωτεινότητα του εικονοστοιχείου. Η μετατροπή γίνεται παίρνοντας τον μέσο όρο των τιμών των τριών καναλιών:

$$I = \frac{R + G + B}{3}$$

Η διαδικασία αυτή προφανώς δεν είναι δυνατό να αντιστραφεί. Όλη η χρωματική πληροφορία της εικόνας χάνεται μετά την μετατροπή.

Σχήμα 6. Μετατροπή έγχρωμης RGB εικόνας σε ασπρόμαυρη εικόνα έντασης.

1.4 Επεξεργασία Βίντεο

Ο τελικός στόχος μας στην εργασία αυτή είναι να εφαρμόσουμε αλγορίθμους αναγνώρισης προτύπων για να αναγνωρίσουμε διάφορες μορφές και αντικείμενα στα βίντεο που επεξεργαζόμαστε. Για να επιτευχθεί αυτό πρέπει να ακολουθηθεί μια σειρά από ενέργειες.

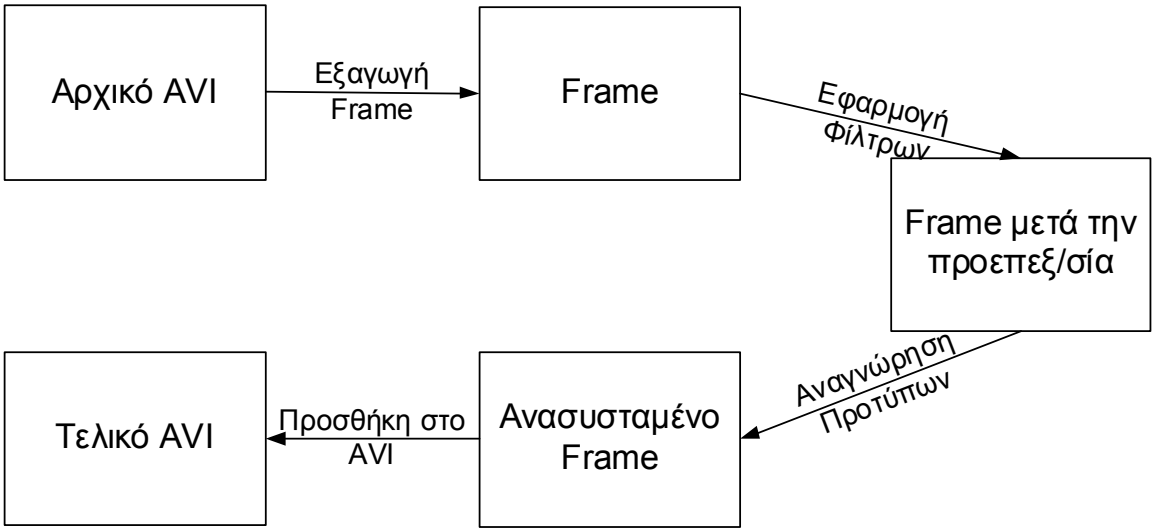
Το πρώτο βήμα που πρέπει να γίνει είναι ο χωρισμός του βίντεο στα καρέ από τα οποία αποτελείται. Στη συνέχεια γίνεται η επεξεργασία κάθε ενός από αυτά ξεχωριστά σαν να ήταν μια ανεξάρτητη εικόνα.

Πριν από την αναγνώριση προηγείται το στάδιο της προ-επεξεργασίας των εικόνων. Σε αυτό εφαρμόζονται μια σειρά από φίλτρα και μετασχηματισμοί στα καρέ του βίντεο προκειμένου να αποκτήσουν την επιθυμητή μορφή και χρήσιμες ιδιότητες. Για παράδειγμα απομακρύνεται ο θόρυβος, αναδεικνύονται οι ακμές, τα περιγράμματα, οι υφές και γενικότερα τα χαρακτηριστικά που μας ενδιαφέρουν.

Στη συνέχεια ακολουθεί το στάδιο της ανάλυση της εικόνας. Συγκεκριμένα εφαρμόζονται αλγόριθμοι αναγνώρισης προτύπων που βασίζονται στην αναγνώριση υφής.

Τα αποτελέσματα που παίρνουμε από το στάδιο της αναγνώρισης προτύπων είναι μια σειρά από εικόνες. Έτσι το τελικό στάδιο της επεξεργασίας μας είναι η δημιουργία ενός νέου βίντεο τα καρέ του οποίου αποτελούν οι εικόνες που έχουν δημιουργηθεί.

Η παραπάνω διαδικασία φαίνεται σχηματικά στο σχήμα 7.



Σχήμα 7. Η επεξεργασία του βίντεο. Φαίνεται σχηματικά η διαδικασία που ακολουθείται για κάθε καρέ του.

1.5 Στάδιο Προ-επεξεργασίας

Οι τεχνικές της ψηφιακής επεξεργασίας εικόνας χωρίζονται, ανάλογα με το πεδίο στο οποίο εφαρμόζονται, σε δυο βασικές κατηγορίες. Έτσι υπάρχουν αυτές που εφαρμόζονται στο χωρικό πεδίο, δηλαδή απευθείας στα εικονοστοιχεία της εικόνας, και αυτές που εφαρμόζονται στο πεδίο των συχνοτήτων (Το πεδίο των συχνοτήτων μας δίνει τις «εναλλαγές» στις τιμές της φωτεινότητας στα εικονοστοιχεία μιας εικόνας αντί για τις ίδιες τις τιμές τους). Για τις τελευταίες απαιτείται μετατροπή της εικόνας από το πεδίο του χώρου στο πεδίο των συχνοτήτων. Αυτό γίνεται με την χρήση του δισδιάστατου διακριτού μετασχηματισμού Fourier. Μετά την εφαρμογή των φίλτρων στο πεδίο της συχνότητας απαιτείται ο αντίστροφος διακριτός μετασχηματισμός Fourier για την απεικόνιση του αποτελέσματος.

Όλες οι τεχνικές που παρουσιάζονται εφαρμόζονται στο πεδίο του χώρου. Όπου αυτό είναι δυνατό γίνεται σύγκριση και αναφορά με την ανάλογη διαδικασία που υπάρχει για το πεδίο των συχνοτήτων. Αναλυτικές πληροφορίες για τα φίλτρα που παρουσιάζονται εδώ και γενικότερα για την ψηφιακή επεξεργασία εικόνας μπορούν να βρεθούν στα βιβλία των Gonzales-Woods [2] και Πήτα [3]. Οι υλοποιήσεις των τεχνικών αυτών έγιναν με τη βοήθεια της βιβλιοθήκης "IPL - Image Processing Library" της Intel [4].

1.5.1 Κατωφλίωση

Η κατωφλίωση επιτρέπει την παραγωγή μιας δίτιμης εικόνας (Bitonal image), που αποτελείται μόνο από τιμές φωτεινότητας 0 και 255, από μια ασπρόμαυρη εικόνα. Αυτό γίνεται με τη χρήση του “κατωφλίου” δηλαδή μιας τιμής που επιλέγεται μεταξύ του 0 και 255. Όσα εικονοστοιχεία έχουν τιμή κάτω από το κατώφλι μηδενίζονται. Αντίστοιχα όσα έχουν πάνω από αυτό λαμβάνουν ως νέα τιμή το 255.

Η τεχνική αυτή εφαρμόζεται και σε έγχρωμες εικόνες. Σε αυτήν την περίπτωση το παραγόμενο αποτέλεσμα δεν είναι μια δίτιμη εικόνα. Το κατώφλι εφαρμόζεται σε κάθε κανάλι ξεχωριστά. Το αποτέλεσμα είναι να υπάρχουν οκτώ διαφορετικά δυνατά χρώματα για τα εικονοστοιχεία της νέας εικόνας:

(R,G,B) = { (0,0,0) (0,0,255) (0,255,0) (0,255,255)
(255,0,0) (255,0,255) (255,255,0) (255,255,255) }

Σχήμα 8. Παράδειγμα κατωφλίωσης στην ασπρόμαυρη εικόνα 'flowers_Gray' και στην έγχρωμη εικόνα 'flowers_RGB' με τιμή κατωφλίου 124. Οι αρχικές εικόνες φαίνονται στο σχήμα 6.

1.5.2 Αριθμητικά Φίλτρα

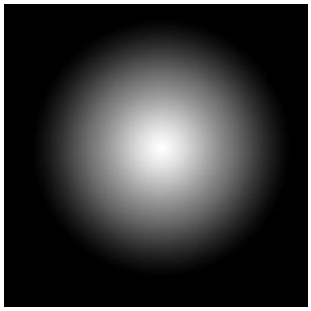
Τα αριθμητικά φίλτρα αποτελούν απλές αριθμητικές πράξεις που γίνονται στις τιμές των εικονοστοιχείων της εικόνας εισόδου. Χωρίζονται σε δυο κατηγορίες. Στην πρώτη υπάρχει μόνο η εικόνα εισόδου και ένας σταθερός αριθμός. Στη δεύτερη απαιτείται και μια δεύτερη εικόνα, πέρα της εικόνας εισόδου, που ονομάζεται μάσκα και πρέπει να έχει τις ίδιες διαστάσεις με την εικόνα εισόδου.

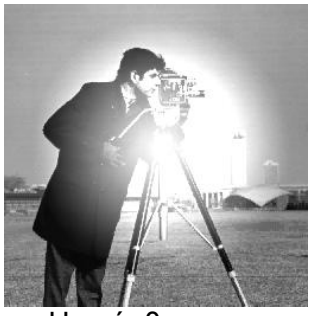
Κάθε εικονοστοιχείο της εικόνας εισόδου αποτελεί τον έναν τελεστέο της πράξης. Ο δεύτερος τελεστέος στην πρώτη κατηγορία είναι ο σταθερός αριθμός ενώ στη δεύτερη είναι το αντίστοιχο εικονοστοιχείο της μάσκας.

Οι επιτρεπτές πράξεις είναι:

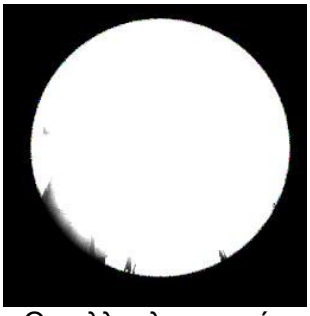
- Πρόσθεση. Προσθέτονται οι δυο τελεστέοι.
- Αφαίρεση. Αφαιρείται η εικόνα εισόδου από τον δεύτερο τελεστέο (ή το αντίστροφο).
- Πολλαπλασιασμός. Πολλαπλασιάζονται οι δυο τελεστέοι.
- Πολλαπλασιασμός με κανονικοποίηση. Πολλαπλασιάζονται οι δυο τελεστέοι και γίνεται κανονικοποίηση του αποτελέσματος.
- Τετράγωνο (Scare). Στην περίπτωση αυτή δεν υπάρχει δεύτερος τελεστέος. Απλά η τιμή του κάθε εικονοστοιχείου της εικόνας εισόδου υψώνεται στο τετράγωνο.


Αρχική εικόνα εισόδου.


Η μάσκα: 'mask_ar'.


Η πρόσθεση της εικόνας εισόδου και της μάσκας.


Η αφαίρεση της μάσκας από την εικόνα εισόδου.


Ο πολλαπλασιασμός της εικόνας εισόδου με την μάσκα.


Ο πολλαπλασιασμός με κανονικοποίηση της εικόνας εισόδου και της μάσκας.


Η ύψωση της εικόνας εισόδου στο τετράγωνο.

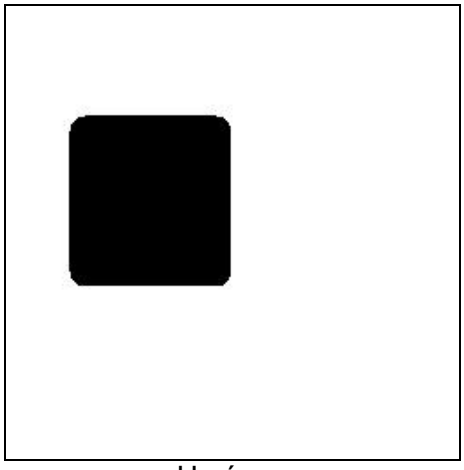
Σχήμα 9. Τα αποτελέσματα αριθμητικών πράξεων πάνω στην εικόνα 'cameraman' με μάσκα την εικόνα 'mask_ar.bmp'.

1.5.3 Δυαδικά Φίλτρα

Τα δυαδικά φίλτρα είναι παρόμοια περίπτωση με την δεύτερη κατηγορία των αριθμητικών φίλτρων. Η διαφορά που υπάρχει είναι ότι αντί για αριθμητικές πράξεις έχουμε λογικές πράξεις και συγκεκριμένα:

- Λογικό Και (AND)
- Λογικό Η΄ (OR)
- Αποκλειστικό Η΄ (XOR)

Οι υπολογισμοί γίνονται αφού πρώτα οι τιμές των εικονοστοιχείων μετατραπούν στις αντίστοιχες δυαδικές τους αναπαραστάσεις.



Η μάσκα

Λογικό ΚΑΙ

Λογικό Η'

Αποκλειστικό Η'

Σχήμα 10. Παράδειγμα εφαρμογής λογικών πράξεων στην εικόνα 'cameraman.bmp' με μάσκα την πρώτη εικόνα

1.5.4 Γραμμικά Φίλτρα - Συγκερασμός

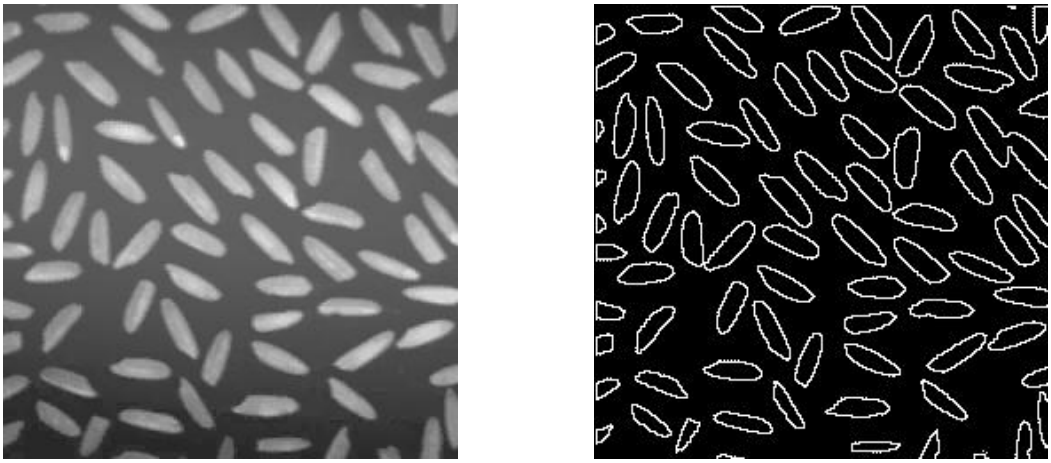
Ο συγκερασμός είναι μια πολύ συνηθισμένη τεχνική και είναι ανάλογη με την χρήση κάποιου φίλτρου στο πεδίο των συχνοτήτων. Επιλέγεται μια μάσκα (πίνακας) μεγέθους NxN με προκαθορισμένες σταθερές τιμές. Στην συνέχεια κάθε εικονοστοιχείο της εικόνας πολλαπλασιάζεται με το κεντρικό στοιχείο της μάσκας και τα γειτονικά του με τα αντίστοιχα στοιχεία της μάσκας. Τα γινόμενα αθροίζονται και το αποτέλεσμα δίνεται στο αντίστοιχο εικονοστοιχείο της νέας εικόνας.

w1	w2	w3
w4	w5	w6
w7	w8	w9

$$R = \sum_{i=1}^{N*N} w_i z_i$$

Σχήμα 11. Μάσκα συγκερασμού και τύπος υπολογισμού της απόκρισης. Με w_i συμβολίζονται οι τιμές της μάσκας και z_i οι τιμές φωτεινότητας των αντίστοιχων εικονοστοιχείων της εικόνας.

Με επιλογή κατάλληλης μάσκας μπορούμε να πάρουμε τα ίδια αποτελέσματα που θα είχαμε με ένα χαμηλοπερατό ή υψηλοπερατό φίλτρο στο πεδίο των συχνοτήτων. Μια απλή χρήση ενός χαμηλοπερατού φίλτρου είναι η απομάκρυνση θορύβου υψηλών συχνοτήτων (κρουστικού) και ενός υψηλοπερατού φίλτρου η ανίχνευση ακμών.



Σχήμα 12. Παράδειγμα συγκερασμού. Αρχική εικόνα η 'rice.bmp'. Πριν τον συγκερασμό έχει προηγηθεί κατωφλίωση. Η μάσκα που χρησιμοποιήθηκε έχει ανάλογα αποτελέσματα με ένα φίλτρο διέλευσης υψηλών συχνοτήτων.

1.5.4.1 Μέσος Όρος

Το φίλτρο μέσου όρου χρησιμοποιείται για την απομάκρυνση κρουστικού θορύβου από μια εικόνα. Σε κάθε εικονοστοιχείο της αρχικής εικόνας εφαρμόζουμε μια τετράγωνη μάσκα μεγέθους NxN με τιμές 1/(NxN). Έτσι επιτυγχάνεται η απομάκρυνση του θορύβου, ταυτόχρονα όμως προκαλείται θάμπωμα στην εικόνα ανάλογο του N. Για το λόγο αυτό αναφέρεται και ως φίλτρο θαμπώματος ('Blur'). Ανάλογα αποτελέσματα μπορούμε να επιτύχουμε στο πεδίο των συχνοτήτων με χρήση χαμηλοπερατού φίλτρου.

Σχήμα 13. Αρχική εικόνα με θόρυβο τύπου 'Salt & Pepper' 2% και απομάκρυνση του με χρήση φίλτρου μέσου όρου με N=5.

1.5.5 Μη γραμμικά φίλτρα

Μια χρήσιμη σειρά από φίλτρα είναι τα μη γραμμικά. Για κάθε εικονοστοιχείο της εικόνας εισόδου επιλέγεται μια γειτονιά από NxM εικονοστοιχεία γύρω του. Οι τιμές των εικονοστοιχείων της επιλεγμένης γειτονίας μαζί με το κεντρικό διατάσσονται κατά αύξουσα σειρά. Στη συνέχεια ανάλογα με το φίλτρο που υλοποιείται επιλέγεται μια από τις τιμές αυτές. Η επιλεγμένη τιμή αποτελεί τη νέα τιμή του εικονοστοιχείου.

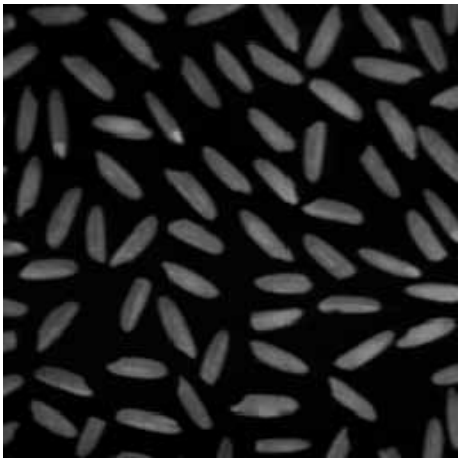
Εδώ θα παρουσιαστούν τα τρία βασικότερα φίλτρα της κατηγορίας. Η μοναδική διαφορά ανάμεσα τους είναι ως προς το κριτήριο επιλογής της νέας τιμής του εικονοστοιχείου.

- Ελάχιστου (Min). Το εικονοστοιχείο που επιλέγεται είναι αυτό με τη μικρότερη τιμή.
- Μεγίστου (Max). Το εικονοστοιχείο που επιλέγεται είναι αυτό με τη μεγαλύτερη τιμή.
- Ενδιάμεσου (Median). Το εικονοστοιχείο που επιλέγεται αυτό που βρίσκεται στη μέση αυτής της διάταξης.

Το φίλτρο ελάχιστου γενικά σκοτεινιάζει μια εικόνα αφού επιλέγει τις λιγότερο φωτεινές τιμές της. Αντίθετα το φίλτρο μεγίστου γενικά φωτίζει μια εικόνα. Αυτό συμβαίνει διότι επιλέγει τα πιο φωτεινά σημεία της εικόνας.

Σχήμα 14. Η εικόνα 'cameraman' μετά από πέρασμα από φίλτρο ελαχίστου και μεγίστου με παράθυρο γειτονίας μεγέθους 5x5

Ένα παράδειγμα πρακτικής χρήσης του φίλτρου ελάχιστου είναι η διόρθωση της φωτεινότητας υποβάθρου μιας εικόνας όταν αυτή δεν είναι ομοιόμορφη. Η διόρθωση γίνεται περνώντας την εικόνα από ένα φίλτρο ελαχίστου με αρκετά μεγάλο παράθυρο π.χ. 32x32 και στη συνέχεια αφαιρώντας το αποτέλεσμα από την αρχική εικόνα:



Σχήμα 15. Η εικόνα 'rice' μετά την διόρθωση της φωτεινότητάς της με χρήση φίλτρου ελαχίστου με παράθυρο 32x32.

Το φίλτρο ενδιάμεσου είναι το πιο ευρέως χρησιμοποιούμενο. Βασική του χρήση είναι η αποκατάσταση μιας εικόνας που έχει αλλοιωθεί από κρουστικό θόρυβο. Η μέθοδος είναι παρόμοια με αυτή του μέσου όρου. Τα αποτελέσματα όμως είναι πολύ καλύτερα διότι δεν υπάρχει το θάμπωμα που υπεισέρχεται στην εικόνα από το φίλτρο μέσου όρου.

Σχήμα 16. Απομάκρυνση θορύβου τύπου 'Salt & Pepper' 20% με φίλτρο Median.

1.5.6 Μορφολογικά

Τα μορφολογικά φίλτρα χρησιμοποιούνται για την εξαγωγή πληροφοριών σχετικά με την μορφή ενός αντικειμένου όπως το περίγραμμα του, τον σκελετό του το ελάχιστο κυρτό σχήμα που το περικλείει κλπ. Οι τεχνικές αυτές έχουν τις βάσεις τους στη θεωρία συνόλων. Μια δίχρωμη εικόνα θεωρείται ότι είναι ένα σύνολο από σημεία. Τα σημεία της είναι τα εικονοστοιχεία με άσπρο χρώμα. Έτσι για παράδειγμα ένα σύνολο S={(1,1) (2,1) (3,3)} θεωρείται ότι αναπαριστά μια δίχρωμη εικόνα με τρία άσπρα εικονοστοιχεία στις συντεταγμένες (1,1), (2,1) και (3,3). Αν η εικόνα ήταν

αποχρώσεων του γκρι θα έπρεπε να προστεθεί ένας ακόμα όρος σε κάθε αναπαράσταση εικονοστοιχείου που να αντιπροσωπεύει την φωτεινότητα του.

Οι βασικές πράξεις της θεωρίας συνόλων είναι η ένωση, η τομή, η διαφορά και το συμπλήρωμα. Σε αυτές συμπληρώνονται άλλες δυο η Μετατόπιση (Translation) και η Ανάκλαση (Reflection):

Η Μετατόπιση ενός συνόλου Α κατά ένα σημείο $z = (z_1,z_2)$ που συμβολίζεται με $(A)_z$ ορίζεται ως εξής:

$$(A)_z = \{w \mid w = a + z, \forall \alpha \in \mathrm{A}\}$$

Η φυσική σημασία της μετατόπισης είναι η μετακίνηση ενός αντικειμένου που ορίζει το σύνολο Α των σημείων του κατά z_1 στον οριζόντιο άξονα και κατά z_2 στον κατακόρυφο.

Η Ανάκλαση ενός συνόλου C συμβολίζεται με $\hat{C}$ και ορίζεται ως εξής:

$$\hat{C} = \{ w \mid w = -c, \forall c \in C \}$$

Οι πιο βασικοί μορφολογικοί μετασχηματισμοί είναι η συστολή (Erosion) και η Διαστολή (Dilation). Με βάση αυτούς ορίζονται άλλοι δυο, η ανοικτότητα (Open) και η κλειστότητα (Close).

1. Διαστολή (Dilation)

Η διαστολή του συνόλου Α από το σύνολο C, που συμβολίζεται με Α⊕ C ορίζεται ως εξής:

$$A \oplus C = \{ z \mid (\hat{C})_z \cap A \neq \emptyset \}$$

Μια απλή εφαρμογή της διαστολής που χρησιμοποιείται ευρέως είναι το κλείσιμο μικρών οπών σε ένα αντικείμενο και το "γεφύρωμα" μικρών κενών ανάμεσα σε εικονοστοιχεία. Για αυτές τις εργασίες χρησιμοποιείται ως σύνολο C ένας πίνακας της μορφής 3x3 με όλα τα στοιχεία του 1. Επίσης ο μετασχηματισμός αυτός έχει την ιδιότητα να διευρύνει το περίγραμμα ενός αντικειμένου.

2. Συστολή (Erosion)

Η συστολή ενός συνόλου Α από το σύνολο C, που συμβολίζεται με ΑΘC ορίζεται ως εξής:

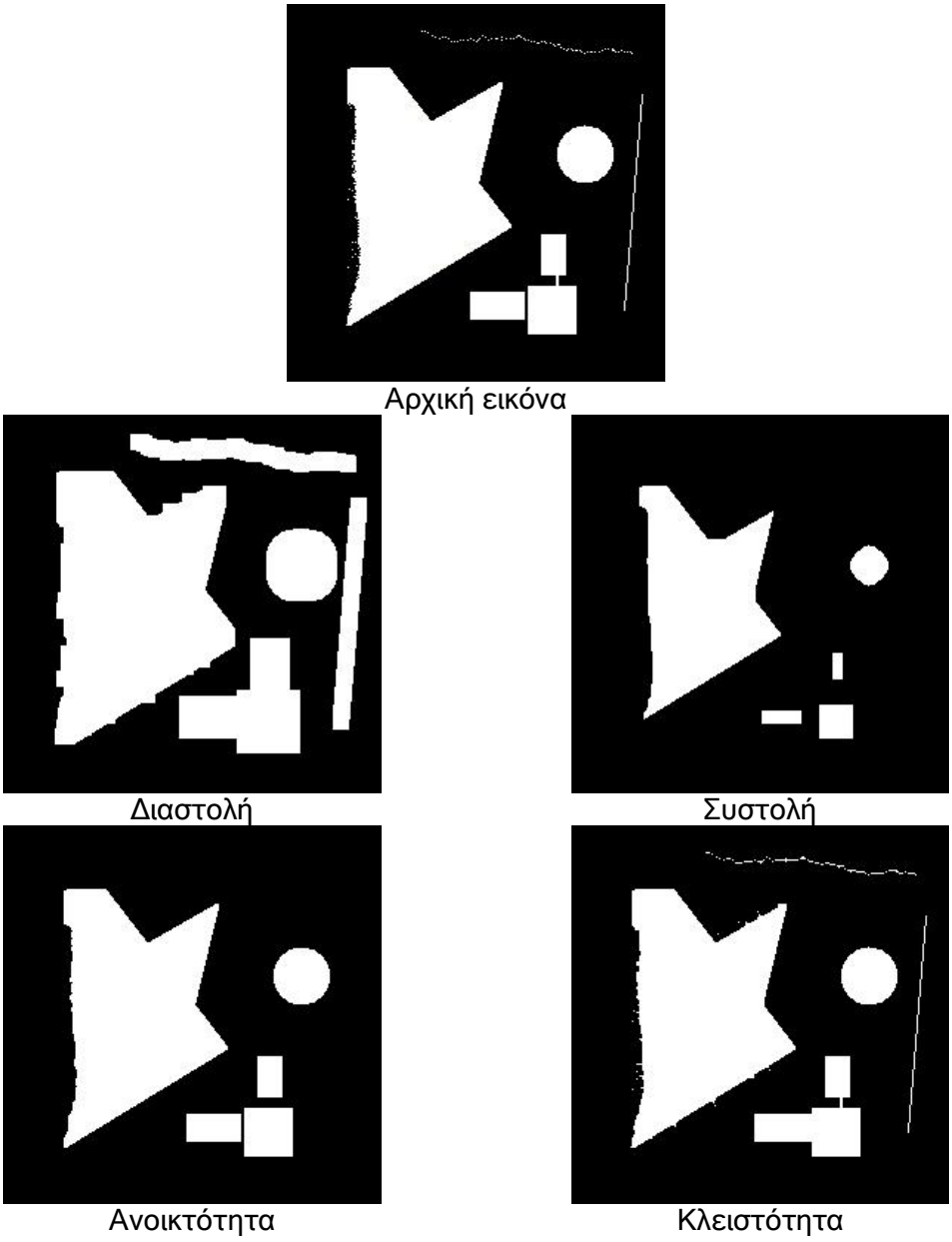
$$A \Theta \mathrm{C} = \{\mathrm{z} \mid (\mathrm{C})_{\mathrm{z}} \subseteq A\}$$

Η συστολή εξαλείφει απομονωμένα εικονοστοιχεία, χρησιμοποιείται για την απομάκρυνση άχρηστης λεπτομέρειας, κρουστικού θορύβου κλπ. Επίσης μικραίνει το περίγραμμα ενός αντικειμένου. Οι ιδιότητες του είναι συνάρτηση του συνόλου C που χρησιμοποιείται.

Οι μετασχηματισμοί της Διαστολής και της συστολής έχουν μια σχέση μεταξύ τους η οποία βασίζεται στο συμπλήρωμα και την ανάκλαση και φαίνεται στη σχέση που ακολουθεί:

$$(A \ominus B)^c = A^c \oplus \widehat{B}$$

Οι δυο μετασχηματισμοί που ακολουθούν του ανοίγματος και του κλεισίματος βασίζονται στην Διαστολή και την συστολή.



Σχήμα 17. Παράδειγμα αποτελεσμάτων των τεσσάρων βασικών μορφολογικών μετασχηματισμών που παρουσιάστηκαν. Το σύνολο C που χρησιμοποιήθηκε είναι ένας πίνακας 3x3 με όλα τα στοιχεία του ίσα με τη μονάδα και έγιναν σειριακά 5 επαναλήψεις του κάθε μετασχηματισμού.

3. Ανοικτότητα (Open)

Η ανοικτότητα ενός συνόλου A κατά ένα σύνολο C συμβολίζεται με A°C είναι αρχικά η συστολή του από το C και στη συνέχεια η Διαστολή του με το ίδιο σύνολο:

$$A \circ C = (A \ominus C) \oplus C$$

Το αποτέλεσμα που έχει η παραπάνω διαδικασία σε ένα αντικείμενο είναι η εξομάλυνση του περιγράμματος του, το σπάσιμο μικρών "ισθμών", η εξάλειψη απομονωμένων σημείων κλπ.

4. Κλειστότητα (Close)

Η κλειστότητα ενός συνόλου A από ένα σύνολο C συμβολίζεται με A•C και είναι αρχικά η Διαστολή του κατά C και στη συνέχεια η συστολή του με το ίδιο σύνολο:

$$A \bullet C = (A \oplus C) \ominus C$$

Αυτός ο μετασχηματισμός εξομαλύνει περιγράμματα, γεμίζει μικρά κενά, ενώνει μικρές ασυνέχειες κλπ.

Γενικά με τους μορφολογικούς μετασχηματισμούς μπορεί να υλοποιηθεί μια μεγάλη σειρά από αλγορίθμους επεξεργασίας εικόνας: Εξαγωγή περιγράμματος, γέμισμα μιας περιοχής, εύρεση κυρτού συνόλου, εκλέπτυνση ή χόντρεμα αντικείμένων, εξαγωγή σκελετού κλπ. Αναλυτικά αυτοί οι αλγόριθμοι μπορούν να βρεθούν στο [2].

1.5.7 Fast Fourier Transformation

Ένας από τους υλοποιημένους αλγορίθμους είναι ο Γρήγορος Μετασχηματισμός Fourier (Fast Fourier Transformation-FFT) ο οποίος μας προσφέρει τη δυνατότητα του μετασχηματισμού μιας εικόνας από το χωρικό πεδίο στο πεδίο των συχνοτήτων. Με αυτόν τον τρόπο μπορεί να αναπαρασταθεί το φάσμα μιας εικόνας και να γίνει η περαιτέρω επεξεργασία του. Υπάρχει επίσης ο αντίστροφος μετασχηματισμός Fourier για την μεταφορά μιας εικόνας από το πεδίο των συχνοτήτων στο πεδίο του χώρου.

1.6 Ανάλυση Εικόνας – Αναγνώριση Προτύπων

Με τον όρο ανάλυση εικόνας εννοείται ο χωρισμός της εικόνας σε κλάσεις. Για παράδειγμα σε ιατρικές εικόνες μπορεί να θεωρηθεί ότι υπάρχουν δυο κλάσεις: Υγιείς και μη υγιείς ιστοί. Προκειμένου να γίνει ο διαχωρισμός ανάμεσα στις κλάσεις αρχικά πρέπει να βρεθούν τα ουσιώδη χαρακτηριστικά που τις διαχωρίζουν με το μικρότερο δυνατό σφάλμα. Στη συνέχεια η ταξινόμηση γίνεται με κάποιον ταξινομητή.

Σε αυτήν την εργασία ως ουσιώδη χαρακτηριστικά χρησιμοποιήθηκαν χαρακτηριστικά υφής. Ακριβής ορισμός της υφής δεν υπάρχει. Επιχειρώντας έναν

πρέπει να πούμε ότι αναφέρεται στον τρόπο που αντιλαμβανόμαστε τις οπτικές ιδιότητες μιας επιφάνειας. Για παράδειγμα αναφερόμαστε σε τραχιά υφή, ινώδη, κοκκώδη, κανονικά επαναλαμβανόμενη κλπ. Η ποσοτικοποίηση της υφής γίνεται με τρεις βασικές τεχνικές: Την στατιστική, την φασματική και την δομική. Σε αυτή την εργασία χρησιμοποιείται η στατιστική προσέγγιση, δηλαδή ο υπολογισμός κάποιων στατιστικών στοιχείων ικανών να διαχωρίσουν διαφορετικές υφές. Πριν τον υπολογισμό τους η εικόνα εισόδου μεταφέρεται στο πεδίο των Wavelets. Τέλος τα χαρακτηριστικά ταξινομούνται σε κλάσεις από ένα νευρωνικό δίκτυο. Το νευρωνικό δίκτυο αρχικά πρέπει να εκπαιδευτεί προκειμένου να διακρίνει την μια κλάση από την άλλη. Αυτό είναι το στάδιο της εκπαίδευσης. Στην συνέχεια το εκπαιδευμένο νευρωνικό δίκτυο προχωράει στην ταξινόμηση των κλάσεων αναγνωρίζοντας τις υφές για τις οποίες έχει εκπαιδευτεί.

1.6.1 Το πεδίο των Wavelets

Η αναγνώριση μιας υφής με βάση τα χαρακτηριστικά που μελετάμε εξαρτάται από την κλίμακα της, δηλαδή το πόσο κοντά ή μακριά βρίσκεται από το σημείο παρατήρησης. Έτσι προκειμένου να απεξαρτηθούμε από την ανάγκη σταθερής κλίμακας πριν την εξαγωγή των χαρακτηριστικών μεταφέρουμε την εικόνα στο πεδίο των Wavelets. Αυτό επιτυγχάνεται με χρήση του Διακριτού Wavelet Μετασχηματισμού (Discrete Wavelet Transformation - DWT).

Τα Wavelets είναι ένα μαθηματικό εργαλείο για την ιεραρχική αποσύνθεση μιας εικόνας. Αρχικά θα περιγραφεί η μονοδιάστατη περίπτωση και στην συνεχεία η επέκταση της στις δυο διαστάσεις. Κάθε Wavelet μετασχηματισμός χρησιμοποιεί μια βάση η οποία μπορεί να διαφέρει ανάλογα με την εφαρμογή και χαρακτηρίζει τον μετασχηματισμό. Σε αυτήν την εργασία χρησιμοποιήθηκε ο Haar μετασχηματισμός.

Μετασχηματισμός Haar

Θα περιγράψουμε τον Haar μετασχηματισμό με ένα παράδειγμα. Έστω ότι έχουμε το μονοδιάστατο σήμα τεσσάρων δειγμάτων: "9 7 3 5". Παίρνουμε ανά δυο τις τιμές και υπολογίζουμε το μέσο όρο. Οπότε το αρχικό σήμα γίνεται "8 4". Προφανώς από το μετασχηματισμό αυτό ένα ποσό πληροφορίας έχει χαθεί. Προκειμένου να είναι δυνατός ο υπολογισμός του αρχικού σήματος χρειάζεται να υπολογιστούν και δύο συντελεστές "λεπτομέρειας" (detail coefficients). Ο πρώτος από αυτούς τους συντελεστές είναι το 1 διότι 8+1=9 και 8-1=7 οπότε είναι δυνατός ο υπολογισμός των αρχικών τιμών του σήματος. Ο δεύτερος είναι το -1 διότι 4-1=3 και 4-(-1)=5. Επομένως το αρχικό σήμα μπορεί να αναπαρασταθεί ως "8 4 1 -1". Στη συνέχεια προχωράμε στο δεύτερο επίπεδο του μετασχηματισμού: Υπολογίζουμε το μέσο όρο των 8 και 4 που είναι το 6 και έχουμε έναν συντελεστή λεπτομέρειας το 2 αφού 6+2=8 και 6-2=4. Επομένως η πλήρη αποσύνθεση δυο επιπέδων του αρχικού σήματος "9 7 3 5" δίνει το "6 2 1 -1".

Η μέθοδος λοιπόν αποτελείται από τον αναδρομικό υπολογισμό μέσων όρων και συντελεστών λεπτομέρειας. Καμία πληροφορία δεν χάνεται και είναι δυνατός ο υπολογισμός του αρχικού σήματος χωρίς απώλειες. Παράλληλα με προσθαφαιρέσεις συντελεστών λεπτομέρειας μπορεί να επιτευχθεί η αναπαράσταση του σήματος με οποιοδήποτε επιθυμητό ποσοστό λεπτομέρειας: Προσθέτοντας συντελεστές αυξάνουμε τις λεπτομέρειες ενώ αφαιρώντας τις μειώνουμε.

Ο μετασχηματισμός που περιγράφηκε μπορεί να εφαρμοστεί και σε δισδιάστατα σήματα, δηλαδή εικόνες: Αρχικά θεωρείται ότι οι γραμμές της εικόνας είναι ένα σύνολο από μονοδιάστατα σήματα. Εφαρμόζεται λοιπόν ο μετασχηματισμός σε κάθε μια από αυτές. Στην συνέχεια θεωρείται ότι οι στήλες της εικόνας που παράγεται είναι ένα σύνολο από μονοδιάστατα σήματα οπότε εφαρμόζεται σε κάθε ένα από αυτά ο μετασχηματισμός. Το αποτέλεσμα που παίρνουμε με την εφαρμογή του μετασχηματισμού αυτού φαίνεται στο σχήμα 18.

Σχήμα 18. Αρχική εικόνα "lenna" και ο Δισδιάστατος Διακριτός Wavelet Μετασχηματισμός της.

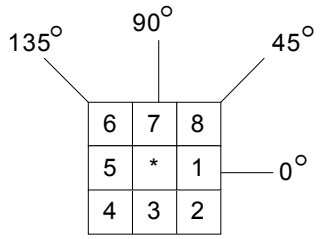
Όπως φαίνεται στο πάνω αριστερά τεταρτημόριο της εικόνας έχει δημιουργηθεί μια αναπαράσταση της αρχικής εικόνας με τη μισή ανάλυση και στα άλλα τρία τεταρτημόρια φαίνονται οι συντελεστές λεπτομέρειας που απαιτούνται για την ανακατασκευή της αρχικής εικόνας. Αν στο πάνω αριστερά τεταρτημόριο εφαρμοστεί ο μετασχηματισμός για δεύτερη φορά θα πάρουμε το δεύτερο επίπεδο του μετασχηματισμού. Το αποτέλεσμα φαίνεται στο σχήμα 19.

Σχήμα 19. Δεύτερο επίπεδο του Wavelet μετασχηματισμού. Έχει αυξηθεί η αντίθεση και η φωτεινότητα της εικόνας για να φανούν οι λεπτομέρειες.

Μια γενική εισαγωγή στον Wavelet μετασχηματισμό και τις εφαρμογές του σε εικόνες μπορεί να βρεθεί στα [2,5,6].

1.6.2 Τα στατιστικά χαρακτηριστικά υφής

Η ταξινόμηση μιας υφής σε κλάσεις, σε αυτήν την εργασία γίνεται με τη χρήση στατιστικών δεύτερης τάξης. Αυτά υπολογίζονται με χρήση των πινάκων συνεμφάνισης (Cooccurrence Matrix). Οι πίνακες αυτοί είναι τετράγωνοι και το πλήθος των στηλών και των γραμμών τους είναι ίσο με τον αριθμό των αποχρώσεων του γκρι στην εικόνα (Θεωρούμε ότι έχουμε ασπρόμαυρες εικόνες). Το κάθε στοιχείο C(i,j) του πίνακα αυτού δείχνει τη πιθανότητα εμφάνισης στην εικόνα ενός ζεύγους εικονοστοιχείων με τιμές φωτεινότητας i και j, για δοσμένη γωνία και απόσταση μεταξύ τους. Σε μια εικόνα ή περιοχή εικόνας μπορούμε να ορίσουμε τέσσερις τέτοιους πίνακες, για σταθερή απόσταση, ανάλογα με τη γωνία γειτνίασης από την οποία έχουν προέλθει.



Σχήμα 20. Οι γωνίες γειτνίασης ως προς το κεντρικό εικονοστοιχείο για απόσταση d ίση με 1.

Οι τέσσερις δυνατές γωνίες γειτνίασης είναι των 0°, 45°, 90° και 135° μοιρών. Έτσι σύμφωνα με το σχήμα 20 τα εικονοστοιχεία 5 και 1 έχουν γωνία 0° ως προς το κεντρικό, τα 4 και 8 έχουν γωνία 45°, τα 3 και 7 έχουν γωνία 90° και τα 2 και 6 γωνία 135°. Το κάθε στοιχείο C(i,j) του πίνακα γωνίας α και απόστασης d, υπολογίζεται μετρώντας το πλήθος των ζευγαριών των εικονοστοιχείων που έχουν φωτεινότητες i και j, γωνία α μοιρών μεταξύ τους και απόσταση d. Στην συνέχεια διαιρούμε με το πλήθος των δυνατών ζευγών που υπάρχουν στην εικόνα για την δοσμένη γωνία.

Σε κάθε έναν από τους τέσσερις αυτούς πίνακες υπολογίζονται τέσσερα ουσιώδη χαρακτηριστικά από τα 14 που προτάθηκαν για αναγνώριση υφής από τον Haralick [7], [8]: Η ενέργεια, η συσχέτιση, η ροπή αντίστροφης διαφοράς και η εντροπία. Στις σχέσεις που ακολουθούν για τον υπολογισμό των χαρακτηριστικών αυτών, με p(i,j) συμβολίζουμε το (i,j) στοιχείο του πίνακα συνεμφάνισης, με N_g τον αριθμό των αποχρώσεων του γκρι στην εικόνα, με μ_x, σ_x, τον μέσο όρο και διασπορά $p_x(i)$ των τιμών των γραμμών του πίνακα p(i,j) και με μ_y ,σ_y τον μέσο όρο και διασπορά $p_y(j)$ των τιμών των στηλών του πίνακα p(i,j).

1.6.2.1 Ενέργεια (Energy ή Angular Second Moment):

Η ενέργεια υπολογίζεται από την σχέση:

$$f_1 = \sum_{i=1}^{Ng} \sum_{j=1}^{Ng} p(i,j)^2$$

Η ενέργεια είναι ένα μέτρο της ομοιογένειας που υπάρχει στην εικόνα. Μια ομοιογενής εικόνα έχει λίγα μη μηδενικά στοιχεία στον πίνακα συνεμφάνισης αλλά με μεγάλες τιμές. Μια μη ομοιογενής εικόνα έχει περισσότερα μη μηδενικά στοιχεία στον πίνακα συνεμφάνισης αλλά με μικρότερες τιμές. Στις ομοιογενείς εικόνες η ενέργεια έχει γενικά μικρότερες τιμές από τις μη ομοιογενείς.

1.6.2.2 Συσχέτιση (Correlation)

Η συσχέτιση υπολογίζεται από την σχέση:

$$f_2 = \frac{\sum_{i=1}^{Ng}\sum_{j=1}^{Ng}(i * j)\,p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y}$$

Η συσχέτιση μας δίνει ένα μέτρο της γραμμικής εξάρτησης των επιπέδων του γκρι στην εικόνα.

1.6.2.3 Ροπή Αντίστροφης Διαφοράς (Inverse Difference Moment)

Η ροπή αντίστροφης διαφοράς υπολογίζεται από την σχέση:

$$f_3 = \sum_{i=1}^{Ng}\sum_{j=1}^{Ng}\frac{1}{1 + (i - j)}\,p(i,j)$$

1.6.2.4 Εντροπία (Entropy)

Η εντροπία υπολογίζεται από την σχέση:

$$f_4 = -\sum_{i=1}^{Ng}\sum_{j=1}^{Ng} p(i,j)\log(\ p(i,j))$$

Γενικά η εντροπία παρουσιάζει μεγάλες τιμές όταν η δομή της υφής είναι περίπλοκη.

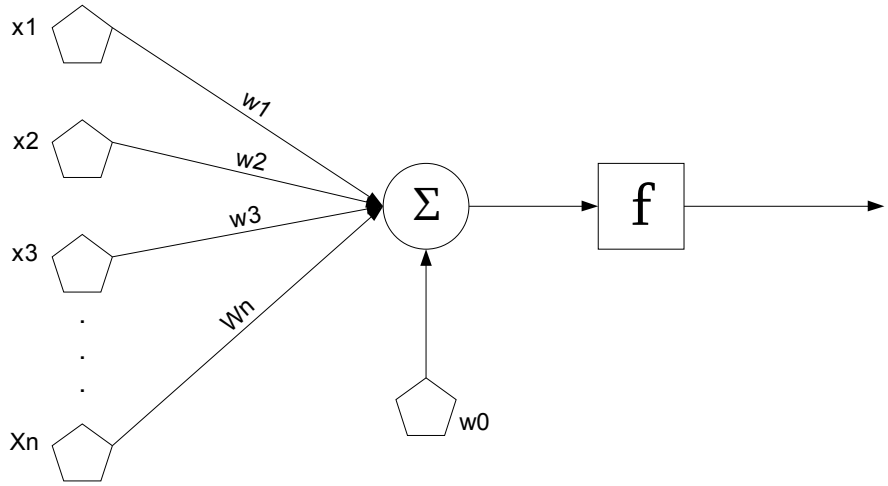
1.6.3 Το Νευρωνικό Δίκτυο

Τα τεχνητά νευρωνικά δίκτυα είναι συστήματα τα οποία παρομοιάζουν την λειτουργία του νευρικού συστήματος και του εγκεφάλου ενός οργανισμού. Βασικό στοιχείο ενός νευρωνικού δικτύου είναι ο νευρώνας ή αρχιτεκτονική Perceptron. Ένας νευρώνας αποτελείται από κάποιες εισόδους και μια έξοδο. Σε κάθε είσοδο

οδηγείται και ένα από τα ουσιώδη χαρακτηριστικά βάση των οποίων θέλουμε να γίνει ο διαχωρισμός των κλάσεων. Έστω x_1, x_2 έως x_n τα χαρακτηριστικά αυτά. Κάθε είσοδος έχει ένα βάρος w_i, που ονομάζεται σύνοψη, και το οποίο πολλαπλασιάζεται με το αντίστοιχο x_i. Όλα τα γινόμενα οδηγούνται σε έναν αθροιστή. Μαζί με αυτά προστίθεται και μια ακόμα τιμή η w_0 που ονομάζεται κατώφλι. Στην συνέχεια το αποτέλεσμα z του αθροιστή περνάει από μια μη γραμμική συνάρτηση f της μορφής:

$$f(z) = \begin{cases} 1, z > 0 \\ 0, z < 0 \end{cases}$$

Η συνάρτηση αυτή είναι γνωστή ως συνάρτηση ενεργοποίησης. Σε άλλες περιπτώσεις η f μπορεί να δίνει διαφορετική έξοδο όπως το 1 και -1.



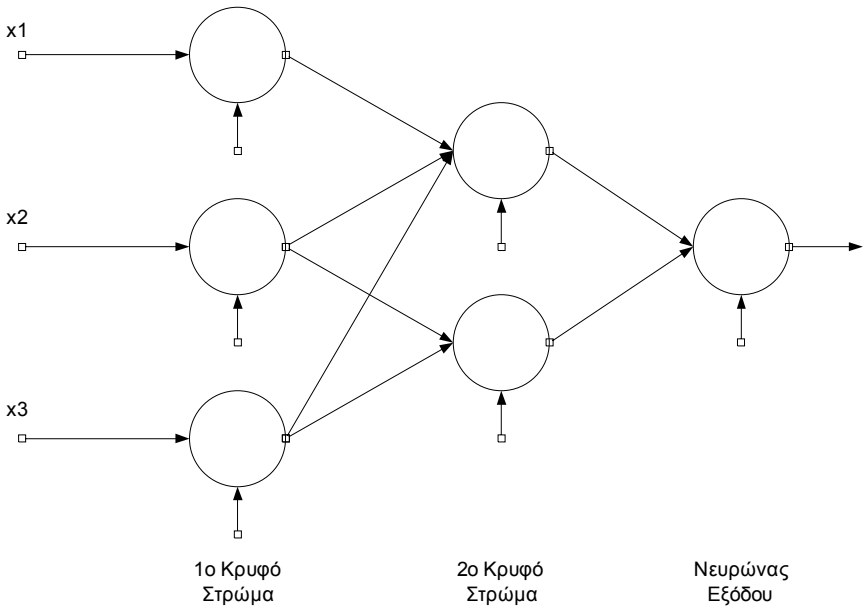
Σχήμα 21. Αρχιτεκτονική Perceptron (Νευρώνα).

Ο υπολογισμός των τιμών των συνάψεων **w** και του κατωφλίου w_0 γίνεται μέσω του αλγορίθμου Perceptron. Αυτός ο αλγόριθμος απαιτεί να υπάρχει μια σειρά από διανύσματα εκπαίδευσης. Τα διανύσματα αυτά είναι ουσιώδη χαρακτηριστικά από πρότυπα που γνωρίζουμε εκ των προτέρων σε πια κλάση ανήκουν. Ο αλγόριθμος λειτουργεί αναδρομικά και σε κάθε βήμα που κάνει προσπαθεί να ελαχιστοποιήσει μια συνάρτηση κόστους η οποία μετράει πόσα από τα διανύσματα εκπαίδευσης έχουν ταξινομηθεί λάθος. Αποδεικνύεται ότι ο αλγόριθμος αυτός συγκλίνει μετά από πεπερασμένο αριθμό βημάτων για γραμμικά διαχωρίσιμες κλάσεις.

Ο απλός νευρώνας που αναφέρθηκε υλοποιεί ένα υπερ-επίπεδο $\mathbf{w}^T\mathbf{x}+w_0=0$ και δίνει έξοδο 1 ή 0 ανάλογα με το αν το διάνυσμα των χαρακτηριστικών **x** βρίσκεται από την μια ή την άλλη πλευρά του υπερ-επιπέδου. Με άλλα λόγια καταφέρνει να διαχωρίσει δύο κλάσεις μεταξύ τους, αρκεί αυτές να είναι γραμμικά διαχωρίσιμες.

Συνήθως τα προβλήματα ταξινόμησης δεν μπορούν να λυθούν με την χρήση ενός μόνο νευρώνα διότι δεν είναι γραμμικά διαχωρίσιμες οι κλάσεις τους. Σε αυτή την περίπτωση απαιτείται να μετασχηματίσουμε το χώρο των χαρακτηριστικών σε ένα άλλο χώρο στον οποίο οι κλάσεις να είναι γραμμικά διαχωρίσιμες. Ο μετασχηματισμός αυτός επιτυγχάνεται με την χρήση περισσότερων νευρώνων οργανωμένους σε επίπεδα. Δυο νευρώνες επικοινωνούν μεταξύ τους ενώνοντας την έξοδο του ενός σε μια από τις συνάψεις του άλλου. Μια τέτοια αρχιτεκτονική ονομάζεται νευρωνικό δίκτυο. Στο σχήμα 22 φαίνεται ένα παράδειγμα τέτοιου δικτύου τριών στρωμάτων. Τα πρώτα δυο στρώματα νευρώνων ονομάζονται και

κρυφά στρώματα. Οι έξοδοι του δεύτερου κρυφού στρώματος οδηγούνται σε έναν νευρώνα που ονομάζεται νευρώνας εξόδου.



Σχήμα 22. Παράδειγμα νευρωνικού δικτύου τριών στρωμάτων.

Αν είχαμε περισσότερες από δυο κλάσεις να διαχωρίσουμε θα έπρεπε να χρησιμοποιήσουμε περισσότερους νευρώνες εξόδου και να αντιστοιχήσουμε έναν σε κάθε κλάση. Τότε ο αντίστοιχος νευρώνας εξόδου θα έδινε 1 και όλοι οι υπόλοιποι 0.

Οι λόγοι για τους οποίους στην εργασία αυτή προτιμήθηκε ένα νευρωνικό δίκτυο ως ταξινομητής είναι: Η ικανότητα του εκπαίδευσης με μικρό αριθμό δειγμάτων και γενικότερα ελλιπείς πληροφορίες καθώς και η καλή ανοχή του στο θόρυβο.

Αρχικά απαιτείται να εκπαιδεύσουμε το νευρωνικό δίκτυο προκειμένου να «μάθει» τα χαρακτηριστικά της προς αναγνώριση υφής. Στη συνέχεια το εκπαιδευμένο δίκτυο είναι ικανό με βάση τα ουσιώδη χαρακτηριστικά που περιγράψαμε να αναγνωρίζει την επιθυμητή υφή.

1.7 Πραγματικός Χρόνος

Η εφαρμογή που έχει αναπτυχθεί χαρακτηρίζεται ως σύστημα πραγματικού χρόνου [9]. Με τον όρο ‘πραγματικός χρόνος’ εννοούμε ένα σύστημα του οποίου η ορθότητα δεν χαρακτηρίζεται μόνο από τα λογικά αποτελέσματα των υπολογισμών του αλλά και από το χρόνο στον οποίο παράγονται αυτά. Τα συστήματα πραγματικού χρόνου πρέπει να αντεπεξέρχονται στους χρονικούς περιορισμούς μέσα στους οποίους πρέπει να παράγουν τα αποτελέσματα τους.

Υπάρχουν δυο βασικές κατηγορίες συστημάτων πραγματικού χρόνου:

- Hard real-time. Είναι ένα σύστημα το οποίο είναι σε θέση να ανταποκρίνεται μέσα στα προκαθορισμένα χρονικά πλαίσια σε κάθε περίπτωση, ανεξάρτητα από τους διάφορους συνδυασμούς των γεγονότων που μπορούν να συμβούν.

- Soft real-time. Είναι ένα σύστημα με μειωμένους χρονικούς περιορισμούς το οποίο όμως πρέπει να λειτουργεί γρήγορα μέσα σε αυτούς. Πρέπει να είναι αρκετά ικανό και να ανταποκρίνεται γρήγορα στα διάφορα γεγονότα. Επιτρέπει την πιθανή μη τήρηση των χρονικών περιορισμών, το οποίο έχει όμως ως αποτέλεσμα την μείωση της απόδοσης του συστήματος.

1.8 Το βίντεο AVI

Το AVI (**A**udio **V**ideo **I**nterleaved / Ήχος Βίντεο Πεπλεγμένα) είναι ένας τρόπος αποθήκευσης κινούμενης εικόνας βίντεο και ήχου ανεπτυγμένος από τη Microsoft. Τα αρχεία βίντεο AVI ακολουθούν το γενικότερο πρότυπο των αρχείων RIFF. Η εσωτερική οργάνωση των RIFF μπορεί να βρεθεί στο παράρτημα 2.9.

Γενικά ένα AVI αποτελείται από ένα σύνολο διάφορων τύπων ροών δεδομένων (Streams). Οι ροές αυτές μπορεί να είναι βίντεο, ήχος και γενικών δεδομένων. Για παράδειγμα ένα τυπικό βίντεο με στερεοφωνικό ήχο περιέχει συνολικά τρεις ροές: Μια βασική ροή βίντεο και δύο ήχου μια για το αριστερό και μια δεξιό κανάλι. Η πιο απλή περίπτωση βίντεο περιέχει μόνο μια ροή αυτή της κινούμενης εικόνας. Αυτού του είδους είναι και τα βίντεο με τα οποία θα ασχοληθούμε.

Τα δεδομένα στο AVI μπορούν να είναι συμπιεσμένα ή και ασυμπίεστα. Τα ασυμπίεστα δεδομένα έχουν το προτέρημα ότι είναι άμεσα προσπελάσιμα και η επεξεργασία τους γίνεται άμεσα. Το μειονέκτημα τους είναι ότι καταλαμβάνουν πάρα πολύ χώρο στο δίσκο. Αντίθετα τα συμπιεσμένα βίντεο είναι ναι μεν πολύ μικρότερα, απαιτούν όμως πιο περίπλοκες διαδικασίες στο χειρισμό τους και χάνουν μέρος της αρχικής πληροφορίας που φέρουν αφού η πλειονότητα των προτύπων συμπίεσης χρησιμοποιούν απολεστικούς αλγόριθμους. Η συμπίεση και αποσυμπίεση των βίντεο επιτυγχάνεται με τη χρήση των συμπιεστών (Video Codec). Αυτοί είναι προγράμματα του συστήματος που υλοποιούν πρότυπα συμπίεσης βίντεο. Για παράδειγμα υπάρχουν συμπιεστές για MPEG-1, MPEG-2, Intel Indeo κλπ.

2 Η Εφαρμογή 'FPremier'

Το βασικό πρόγραμμα που αναπτύχθηκε και εφαρμόζει όλες τις τεχνικές που παρουσιάστηκαν στο πρώτο μέρος είναι η εφαρμογή "FPremier". Είναι ένα πρόγραμμα ψηφιακής επεξεργασίας βίντεο σε πραγματικό χρόνο. Τα βίντεο που υποστηρίζονται είναι τα AVI. Στις δυνατότητες του προγράμματος περιλαμβάνονται:

- Προβολή βίντεο AVI.
- Σειριακή εφαρμογή φίλτρων πάνω σε αυτό σε πραγματικό χρόνο για προεπεξεργασία των καρέ του βίντεο.
- Εξαγωγή χαρακτηριστικών υφής από τα καρέ του βίντεο.
- Αναγνώριση προτύπων με βάση τα χαρακτηριστικά αυτά από προ-εκπαιδευμένο νευρωνικό δίκτυο.
- Δημιουργία νέου βίντεο με τα αποτελέσματα της επεξεργασίας.

Στην συνέχεια παρουσιάζεται η βασική διεπαφή της εφαρμογής και τα διάφορα στοιχεία από τα οποία αποτελείται, ενώ παράλληλα φαίνονται και οι διάφορες δυνατότητες της. Στην παράγραφο 2.4 δίνεται ένα αναλυτικό παράδειγμα χρήσης της εφαρμογής με ένα βίντεο ενδοσκόπησης ως είσοδο, στο οποίο διαχωρίζονται αυτόματα οι υγιείς από τους καρκινικούς ιστούς.

2.1 Η διεπαφή

Το βασικό παράθυρο της διεπαφής της εφαρμογής αφού έχει ανοιχθεί ένα αρχείο βίντεο προς επεξεργασία διαμορφώνεται όπως φαίνεται στο σχήμα 23.

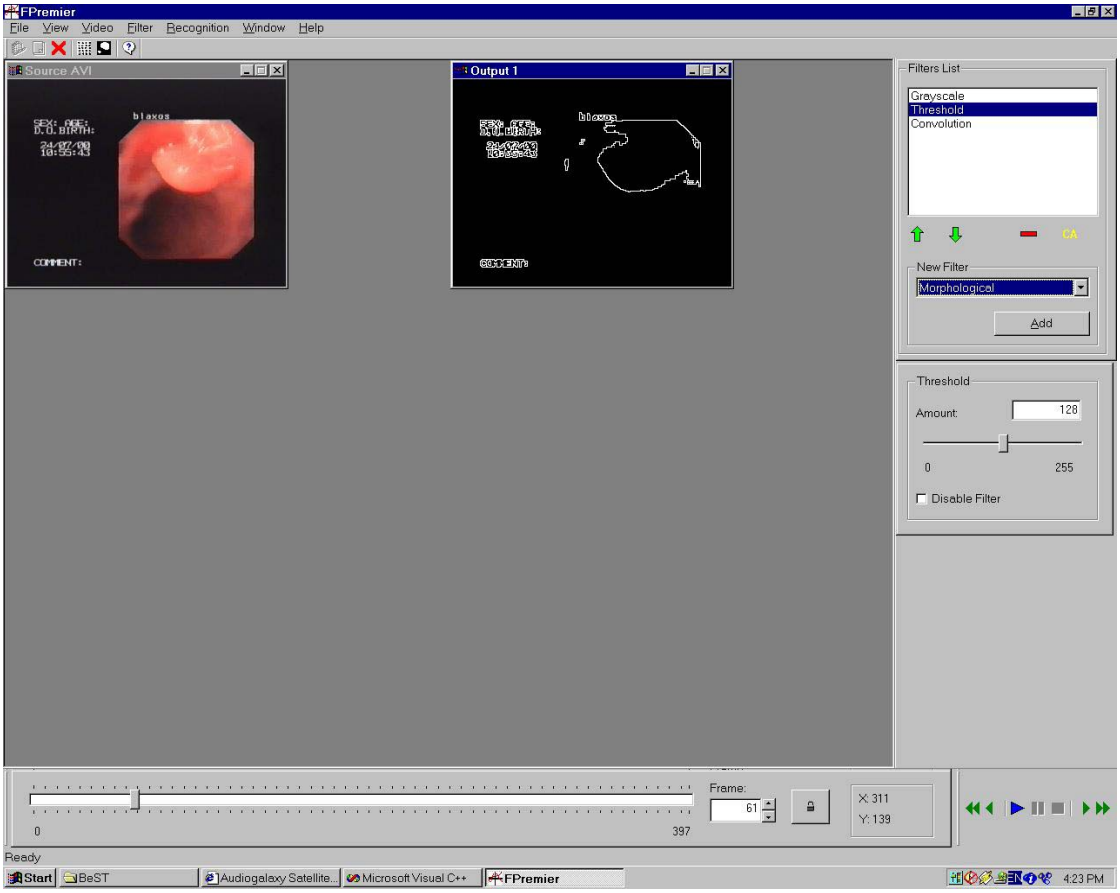
Τα στοιχεία από τα οποία αποτελείται η διεπαφή της εφαρμογής είναι:

1) Το παράθυρο του βίντεο εισόδου με τίτλο "Source AVI".
2) Τα παράθυρο του βίντεο εξόδου με τίτλο "Output 1".
3) Το παράθυρο αναπαραγωγής του βίντεο.
4) Το παράθυρο επιλογής καρέ.
5) Το παράθυρο επιλογής φίλτρων.
6) Το παράθυρο ιδιοτήτων του επιλεγμένου φίλτρου.

Στο πάνω μέρος της διεπαφής βρίσκονται τα μενού από τα οποία γίνονται κάποιες βασικές λειτουργίες της εφαρμογής.

Στο μενού "File" υπάρχουν οι παρακάτω λειτουργίες:

1) "Open": Άνοιγμα ενός αρχείου AVI προς επεξεργασία.
2) "Close": Κλείσιμο ενός ανοιγμένου αρχείου.
3) "Properties": Ιδιότητες του ανοιγμένου αρχείου.
4) "Export→Frame": Εξαγωγή ενός καρέ του βίντεο σε αρχείο εικόνας BMP.
5) "Export→AVI": Αποθήκευση του βίντεο με εφαρμοσμένα πάνω του όλα τα επιλεγμένα φίλτρα.
6) "Exit": Έξοδος από την εφαρμογή.



Σχήμα 23. Η εφαρμογή FPremier.

Στο μενού "Video" έχουμε τις ίδιες επιλογές με αυτές που έχουμε και στο παράθυρο αναπαραγωγής του βίντεο (παρ. 2.1.2). Στο μενού "Filter" έχουμε την δυνατότητα να προσθέσουμε ένα φίλτρο στο βίντεο από αυτά που υποστηρίζονται από την εφαρμογή (παρ. 2.1.3). Από το μενού "Recognition" έχουμε τις εξής δυνατότητες:

1) "Video Recognition": Ξεκινά την διαδικασία αναγνώρισης υφής σε ένα βίντεο.
2) "Frame Extract Features": Εξάγει τα χαρακτηριστικά υφής σε ένα συγκεκριμένο καρέ.
3) "Frame Recognition": Αναγνωρίζει μια υφή σε ένα καρέ με βάση τα χαρακτηριστικά που έχουμε εξάγει και ένα προ-εκπαιδευμένο νευρωνικό δίκτυο για την συγκεκριμένη υφή.

Το βίντεο που έχουμε ανοίξει εμφανίζεται στο παράθυρο με τίτλο “Source AVI”. Τα αποτελέσματα κάθε φίλτρου που εφαρμόζουμε σε αυτό εμφανίζεται σε ένα παράθυρο εξόδου με τίτλο “Output x”, όπου x ο αριθμός του παραθύρου εξόδου. Σημειώνεται ότι μπορούμε να έχουμε παραπάνω από ένα βίντεο εξόδου με διαφορετική λίστα φίλτρων σε κάθε ένα από αυτά.

2.1.1 Το παράθυρο επιλογής καρέ.

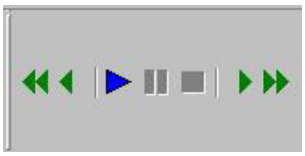
Σχήμα 24. Το παράθυρο επιλογής καρέ.

Το παράθυρο επιλογής καρέ εκτελεί δυο βασικές λειτουργίες. Καταρχάς δείχνει ποιος είναι ο αριθμός του καρέ του βίντεο που εμφανίζεται αυτή τη στιγμή στο ενεργό παράθυρο βίντεο. Επιπλέον μετακινώντας την ανάλογη μπάρα είτε γράφοντας έναν συγκεκριμένο αριθμό στο πεδίο “Frame:” μπορούμε να μετακινηθούμε στο αντίστοιχο καρέ του βίντεο.

Τα πεδία “X:” και “Y:” μας δίνουν τις συντεταγμένες του εικονοστοιχείου πάνω από το οποίο βρίσκεται ο δείκτης του ποντικιού.

Το πλήκτρο με το σύμβολο του μικρού λουκέτου όταν είναι πατημένο «κλειδώνει» τον αριθμό του καρέ του παραθύρου του βίντεο εισόδου με του ενεργοποιημένου παράθυρου του βίντεο εξόδου με αποτέλεσμα να έχουμε ταυτόχρονη αναπαραγωγή του βίντεο και στα δυο παράθυρα και να μπορούν να γίνονται εύκολα συγκρίσεις.

2.1.2 Το παράθυρο αναπαραγωγής του βίντεο.



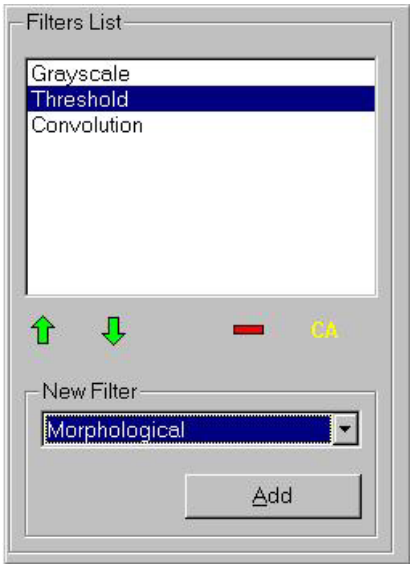
Σχήμα 25. Το παράθυρο αναπαραγωγής του βίντεο.

Από το παράθυρο αυτό γίνεται ο έλεγχος της αναπαραγωγής του βίντεο. Περιέχει επτά πλήκτρα, τα οποία με την σειρά που εμφανίζονται είναι:

- Fast Backward – Μας μεταφέρει στην αρχή του βίντεο.
- Backward – Μας μεταφέρει ένα καρέ πίσω από αυτό που βρισκόμαστε.
- Play – Ξεκινάει την αναπαραγωγή του βίντεο.
- Pause – Διακόπτει προσωρινά την αναπαραγωγή του βίντεο.
- Stop – Διακόπτει την αναπαραγωγή του βίντεο και μας μεταφέρει στο πρώτο του καρέ.
- Forward – Μας μεταφέρει στο επόμενο καρέ από αυτό που βρισκόμαστε.
- Fast Forward – Μας μεταφέρει στο τελευταίο καρέ του βίντεο.

Τα πλήκτρα αυτά γίνονται ανενεργά αν δεν μπορούν να χρησιμοποιηθούν. Έτσι για παράδειγμα τα “Stop” και “Pause” δεν λειτουργούν αν δεν έχει πατηθεί προηγουμένως το “Play” και το βίντεο αναπαράγεται, τα “Fast Forward” και “Forward” δεν λειτουργούν αν το καρέ στο οποίο βρισκόμαστε είναι το τελευταίο.

2.1.3 Το παράθυρο επιλογής φίλτρου.



Σχήμα 26. Το παράθυρο επιλογής φίλτρου.

Στο παράθυρο αυτό βρίσκεται η λίστα των φίλτρων από τα οποία περνάμε το βίντεο. Τα φίλτρα εφαρμόζονται σειριακά ξεκινώντας από αυτό που είναι στην κορυφή της λίστας. Στην σχήμα 26 βλέπουμε ένα παράδειγμα μιας τέτοιας λίστας: Υπάρχουν τρία φίλτρα τα "Grayscale", "Threshold" και "Convolution". Στην κορυφή της λίστας είναι το "Grayscale" και είναι αυτό από το οποίο θα ξεκινήσει η εισαγωγή των φίλτρων.

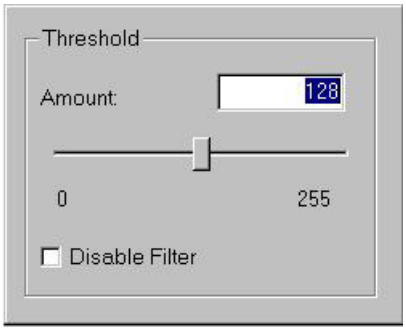
Κάτω από την λίστα βρίσκονται τέσσερα πλήκτρα ελέγχου της λίστας και με τη σειρά που εμφανίζονται είναι:

- Βέλος προς τα πάνω – Μετακινεί το επιλεγμένο φίλτρο μια θέση πιο πάνω στην λίστα.
- Βέλος προς τα κάτω – Μετακινεί το επιλεγμένο φίλτρο μια θέση πιο κάτω στην λίστα.
- Διαγραφή – Διαγράφει το επιλεγμένο φίλτρο από την λίστα.
- Διαγραφή όλων (CA) – Διαγράφει όλη την λίστα των φίλτρων.

Στο πεδίο "New Filter" γίνεται η επιλογή του νέου φίλτρου που θέλουμε να προσθέσουμε στην λίστα. Αρχικά το επιλέγουμε από το πτυσσόμενο μενού και το προσθέτουμε πατώντας το πλήκτρο "Add".

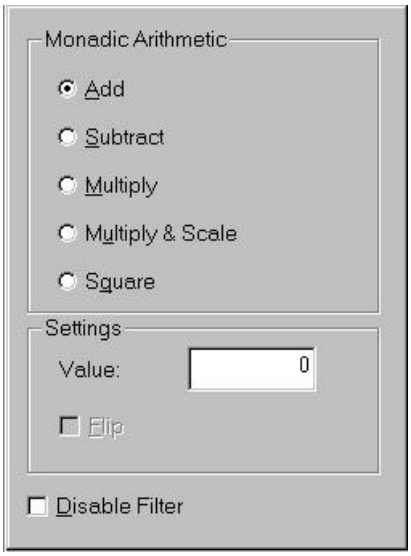
2.1.4 Τα παράθυρα ιδιοτήτων των φίλτρων.

Σε αυτή την παράγραφο παρουσιάζονται όλα τα φίλτρα επεξεργασίας εικόνας που υποστηρίζονται από την εφαρμογή. Το αποτέλεσμα της εφαρμογής του κάθε φίλτρου πάνω σε ένα καρέ από το βίντεο εισόδου έχουν ήδη παρουσιαστεί στην παράγραφο 1.5.



Κατώφλι

Η μοναδική επιλογή που θέτουμε είναι η τιμή του κατωφλίου. Η τιμή αυτή πρέπει να είναι μεταξύ του 0 και του 255.



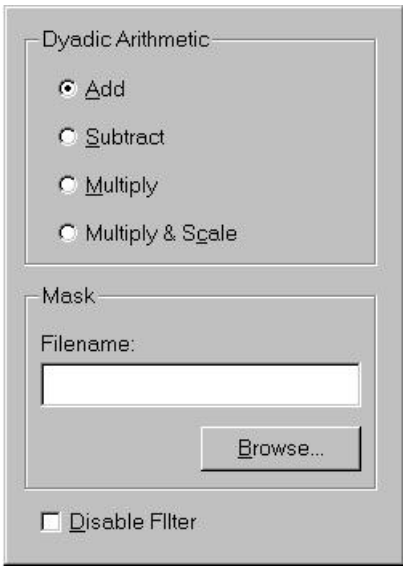
Αριθμητικές Πράξεις

Στο πρώτο πλαίσιο επιλέγουμε την αριθμητική πράξη που θέλουμε να εκτελέσουμε:

- Add – Πρόσθεση
- Subtract – Αφαίρεση
- Multiply – Πολλαπλασιασμός
- Multiply & Scale – Πολ/σμός με κλίμακα
- Square - Τετράγωνο

Στο πεδίο "Value" θέτουμε τον δεύτερο τελεστέο της πράξης.

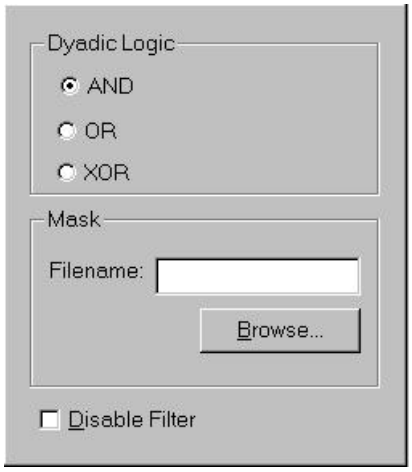
Στην περίπτωση της αφαίρεσης μπορούμε να αφαιρέσουμε τις τιμές των εικονοστοιχείων της εικόνας από το Value επιλέγοντας το "Flip".



Αριθμητικές Πράξεις μεταξύ του Βίντεο και μιας Εικόνας Εισόδου

Στο πρώτο πλαίσιο επιλέγουμε την αριθμητική πράξη που θέλουμε να εκτελέσουμε όπως και στην περίπτωση των αριθμητικών πράξεων με μια εικόνα.

Στο πεδίο "Filename" εισάγουμε το όνομα του αρχείου εικόνας που θα αποτελέσει τον δεύτερο τελεστέο της αριθμητικής πράξης. Με το πλήκτρο "Browse" μπορούμε να εντοπίσουμε το αρχείο αυτό στο δίσκο.

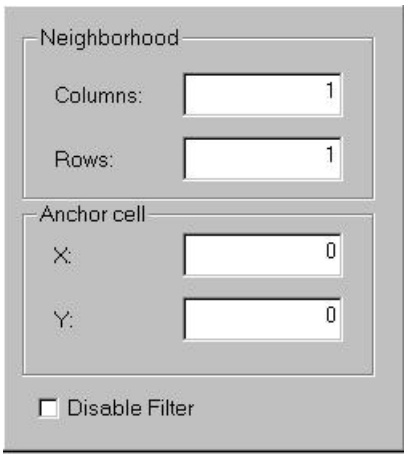


Λογικές Πράξεις μεταξύ του Βίντεο και μιας Εικόνας Εισόδου

Στο πρώτο πλαίσιο επιλέγουμε την λογική πράξη που θέλουμε να εκτελέσουμε:

- AND - Λογικό ΚΑΙ
- OR – Λογικό Η'
- XOR – Αποκλειστικό Η'

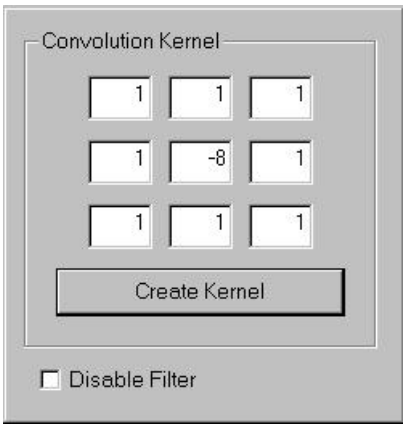
Στο πεδίο "Filename" εισάγουμε το όνομα του αρχείου εικόνας που θα αποτελέσει τον δεύτερο τελεστέο της λογικής πράξης. Με το πλήκτρο "Browse" μπορούμε να εντοπίσουμε το αρχείο αυτό στο δίσκο



Μέσος Όρος

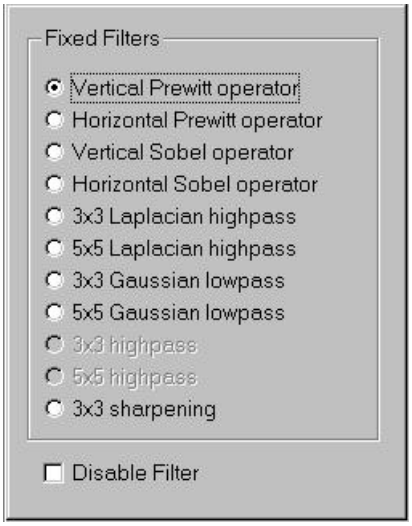
Στο πρώτο πλαίσιο γίνεται επιλογή του μεγέθους της γειτονίας. Στο πεδίο "Columns" επιλέγουμε το πλάτος της και στο "Rows" το ύψος της.

Με τα πεδία "X" και "Y" επιλέγουμε πιθανή μετακίνηση της τελικής εικόνας.



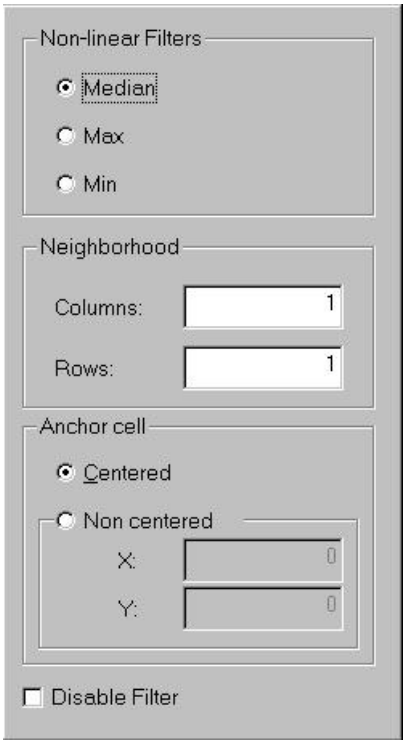
Συνέλιξη

Στα εννέα μικρά τετράγωνα του πλαισίου "Convolution Kernel" επιλέγουμε την μάσκα του συγκερασμού. Πατώντας το πλήκτρο "Create Kernel" οριστικοποιούμε τις αλλαγές μας.



Προκατασκευασμένοι Πυρήνες Συνέλιξης

Εδώ επιλέγουμε ένα από τα προκατασκευασμένα φίλτρα που είναι διαθέσιμα από την IPL.



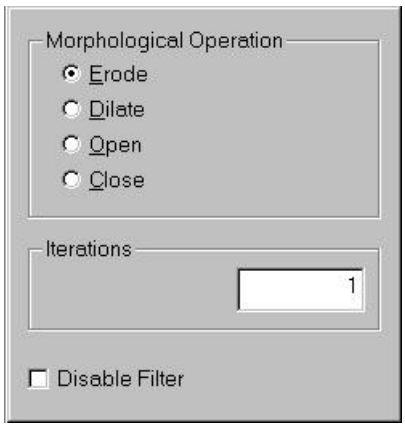
Μη Γραμμικά Φίλτρα

Στο πρώτο πλαίσιο γίνεται η επιλογή του μη γραμμικού φίλτρου:

- Median – Ενδιάμεσου
- Max – Μεγίστου
- Min – Ελαχίστου

Στο δεύτερο επιλέγεται η γειτονιά από NxM εικονοστοιχεία στα οποία θα εφαρμοστεί το φίλτρο.

Στο τελευταίο πλαίσιο δίνεται η δυνατότητα μετατόπισης των καρέ εξόδου του βίντεο ή μπορούμε να επιλέξουμε να παραμείνουν κεντραρισμένα.

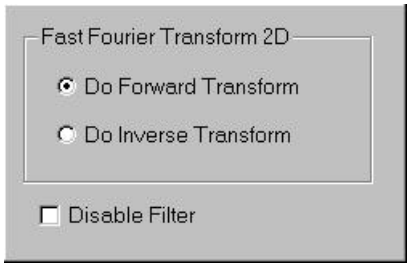


Μορφολογικά

Στο πρώτο πλαίσιο επιλέγομε το είδος του μορφολογικού μετασχηματισμού που θέλουμε να εκτελέσουμε:

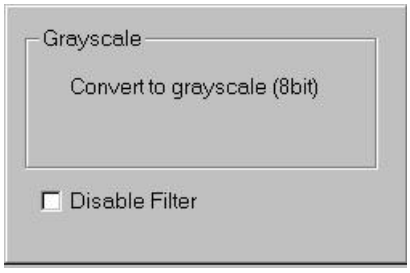
- Erode – Συστολή
- Dilate – Διαστολή
- Open – Ανοικτότητα
- Close – Κλειστότητα

Στο πεδίο “Iterations” επιλέγουμε τον αριθμό των φορών που θέλουμε να εκτελεστεί ο μετασχηματισμός.



Μετασχηματισμός Fourier

Η μοναδική επιλογή που έχουμε είναι το αν θέλουμε να εκτελέσουμε τον ευθύ ή τον αντίστροφο δισδιάστατο διακριτό μετασχηματισμό Fourier.



Μετατροπή σε Ασπρόμαυρο

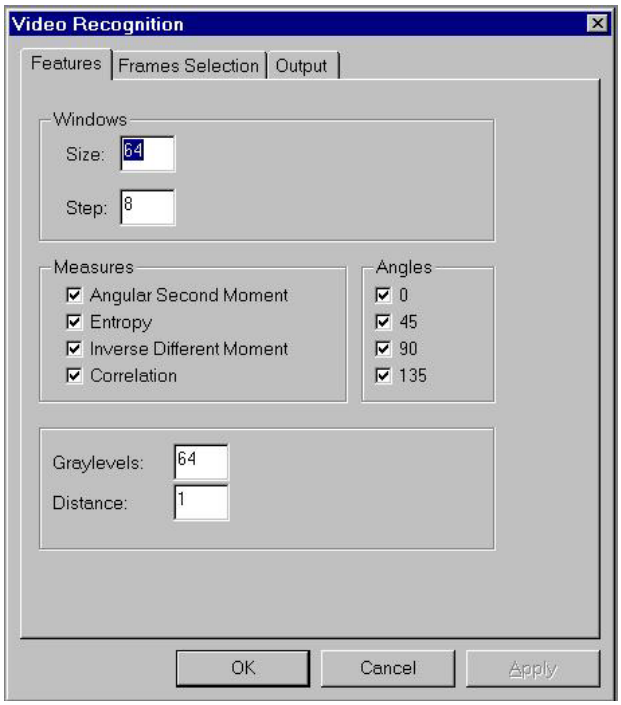
Σε αυτό το φίλτρο δεν έχουμε καμία επιλογή ρύθμισης. Απλά η εικόνα εισόδου γίνεται grayscale 8bit.

Σε όλα τα φίλτρα υπάρχει η επιλογή "Disable Filter". Επιλέγοντάς την απενεργοποιούμε το φίλτρο προσωρινά. Οι ρυθμίσεις που του έχουμε κάνει δεν χάνονται.

2.2 Η αναγνώριση προτύπων στο βίντεο.

Η αναγνώριση προτύπων στο βίντεο γίνεται από το μενού Recognition→Video Recognition. Εμφανίζεται ένα βασικό παράθυρο που περιέχει τρεις καρτέλες ρυθμίσεων σχετικές με τα χαρακτηριστικά, τα καρέ του βίντεο εισόδου και τα αρχεία εξόδου που θα δημιουργηθούν.

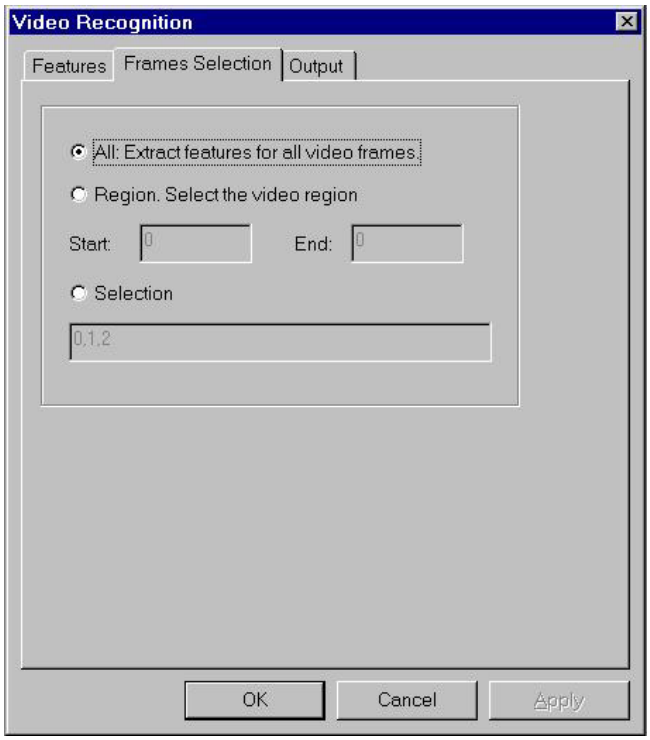
2.2.1 Επιλογή χαρακτηριστικών υφής



Σχήμα 27. Καρτέλα επιλογής χαρακτηριστικών.

Στην καρτέλα αυτή (Σχήμα 27) επιλέγουμε τα χαρακτηριστικά βάση των οποίων θα γίνει η ταξινόμηση. Στο πεδίο "Windows Size" επιλέγουμε το μέγεθος του παραθύρου με το οποίο θα «προσπελαθεί» η εικόνα. Το "Step" είναι το βήμα σε εικονοστοιχεία με το οποίο θα ολισθαίνει το παράθυρο. Στο πεδίο "Measures" επιλέγουμε τα χαρακτηριστικά που θέλουμε να εξάγουμε και στο "Angles" τις γωνίες για τις οποίες θα εξάγονται αυτά. Στο "Graylevels" μπορούμε να ορίσουμε ένα διαφορετικό βάθος χρώματος για την εικόνα εισόδου από αυτό που έχει, αφού με λιγότερα χρώματα, ανάλογα με την εφαρμογή, μπορεί να γίνεται πιο εύκολα και γρήγορα η αναγνώριση της επιθυμητής υφής.

2.2.2 Επιλογή καρέ

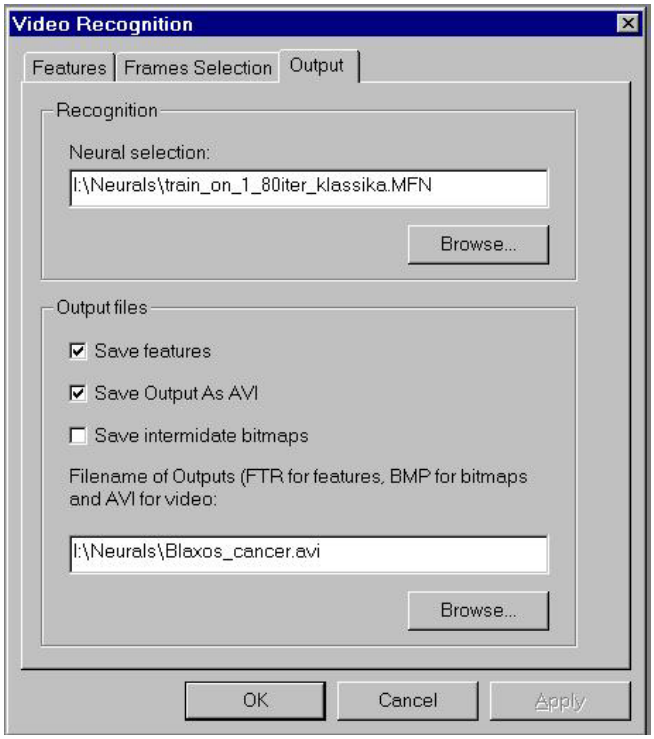


Σχήμα 28. Καρτέλα επιλογής καρέ.

Στην καρτέλα αυτή (Σχήμα 28) επιλέγουμε τα καρέ του βίντεο στα οποία θέλουμε να γίνει η αναγνώριση. Έχουμε τρεις επιλογές:

- All – Όλα τα καρέ
- Region – Μια περιοχή του βίντεο. Δίνουμε αρχικό και τελικό καρέ.
- Selection – Συγκεκριμένα καρέ. Στο ανάλογο πεδίο γράφουμε τους αριθμούς των καρέ χωρίζοντας τους με κόμματα.

2.2.3 Επιλογή νευρωνικού δικτύου και αρχείων εξόδου



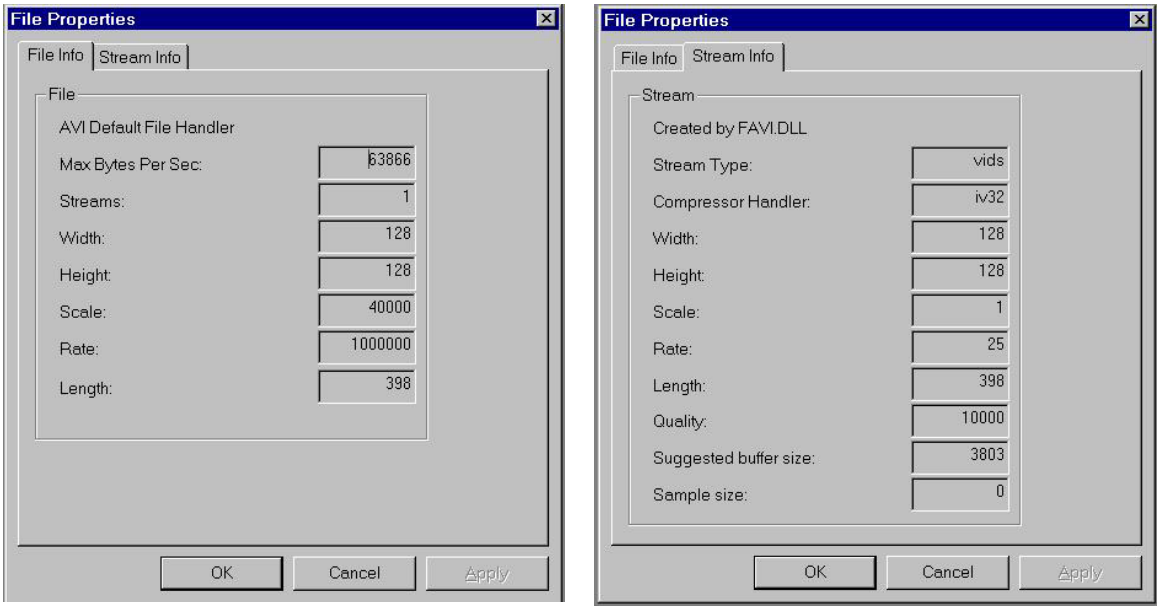
Σχήμα 29. Καρτέλα επιλογής νευρωνικού δικτύου και αρχείων εξόδου.

Εδώ κάνουμε δυο βασικές επιλογές (Σχήμα 29). Καταρχάς επιλέγουμε το προ-εκπαιδευμένο νευρωνικό δίκτυο που θα κάνει την ταξινόμηση των χαρακτηριστικών. Το νευρωνικό δίκτυο πρέπει να είναι αποθηκευμένο στο δίσκο σε μορφή αρχείου τύπου "MFN". Τέτοια αρχεία μπορεί να παράγει η εφαρμογή CoLD 10

Στη συνέχεια επιλέγουμε τις εξόδους που επιθυμούμε να πάρουμε. Επιλέγοντας το πεδίο "Save features" αποθηκεύουμε τα χαρακτηριστικά του κάθε καρέ σε αρχείο κειμένου με κατάληξη ".FTR". Επιλέγοντας το πεδίο "Save Output As AVI" δημιουργούμε ένα νέο βίντεο τύπου AVI με τις ανασυσταμένες εικόνες που παίρνουμε μετά την αναγνώριση. Επιλέγοντας το πεδίο "Save intermediate bitmaps" αποθηκεύουμε όλα τα καρέ του βίντεο ως ξεχωριστές εικόνες bitmap τύπου ".BMP". Τέλος καθορίζουμε το γενικό όνομα που θα έχουν τα αρχεία εξόδου (Θα διαφοροποιούνται ως προς την κατάληξη "FTR" για χαρακτηριστικά, "BMP" για εικόνες και "AVI" για βίντεο.

2.3 Ιδιότητες του βίντεο

Από το μενού “File→Properties…” έχουμε την δυνατότητα να πάρουμε πληροφορίες σχετικά με ένα αρχείο βίντεο που έχουμε ανοίξει. Το παράθυρο που εμφανίζεται χωρίζεται σε δυο «καρτέλες». Η μια περιέχει γενικές πληροφορίες για το αρχείο που ανοίχτηκε και η άλλη σχετικές με τη ροή δεδομένων βίντεο μέσα σε αυτό.

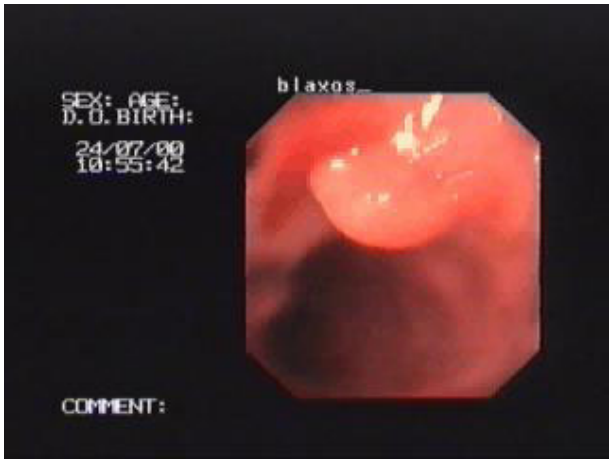


Σχήμα 30. Ιδιότητες του βίντεο.

2.4 Παράδειγμα χρήσης σε ιατρικό βίντεο

Σε αυτή την παράγραφο θα παρουσιάσουμε πώς είναι δυνατή η χρήση της εφαρμογής 'FPremier' σε ιατρικά βίντεο. Συγκεκριμένα τα βίντεο εισόδου που έχουμε είναι από ενδοσκοπήσεις ασθενών και στόχος μας είναι να εντοπίσουμε σε αυτά πιθανούς καρκινικούς ιστούς. Η αναγνώριση γίνεται με βάση προ-εκπαιδευμένο νευρωνικό δίκτυο και βασίζεται στα χαρακτηριστικά υφής των ιστών.

Το βίντεο που παίρνουμε από μια ενδοσκόπηση είναι της μορφής που παρουσιάζεται στην σχήμα 31.

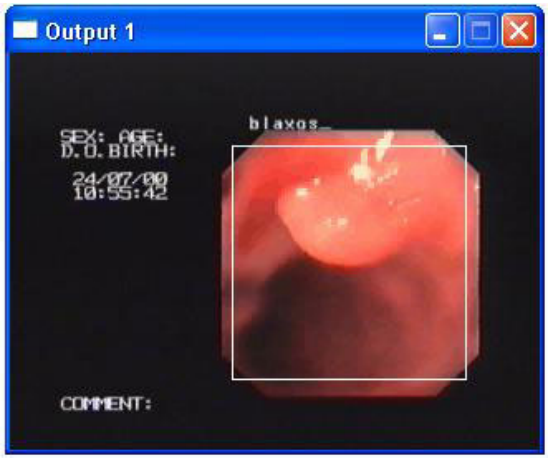


Σχήμα 31. Αρχικό βίντεο ενδοσκοπίου.

Όπως μπορούμε εύκολα να παρατηρήσουμε γύρω από την εικόνα του ενδοσκοπίου υπάρχει ένα «μαύρο» πλαίσιο με διάφορες πληροφορίες σχετικά με τον ασθενή που προστίθεται κατά την εξέταση του. Αυτό το πλαίσιο είναι περιττό για την αναγνώριση που θέλουμε να επιτύχουμε και πρέπει να απομακρυνθεί από το βίντεο.

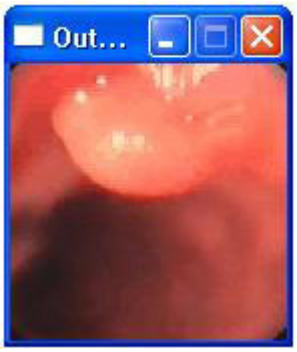
Αυτή η δυνατότητα μας δίνεται από την εφαρμογή μέσω της διαδικασίας επιλογής περιοχής ενδιαφέροντος στο βίντεο και εξαγωγής αυτής ως νέο βίντεο.

Αφού ανοίξουμε το βίντεο του ενδοσκοπίου στην εφαρμογή παίρνουμε την σχήμα 23 (χωρίς τις προσθέσεις των φίλτρων). Στην συνέχεια μπορούμε στο παράθυρο του βίντεο εξόδου με τίτλο 'Output 1' να επιλέξουμε ως περιοχή ενδιαφέροντος μόνο εκείνη που περιέχει την εικόνα του ενδοσκοπίου. Αυτό γίνεται κάνοντας δεξί 'κλικ' στο παράθυρο και επιλέγοντας από αναδιπλούμενο μενού την επιλογή 'Set ROI' (ROI = '**R**egion **O**f **I**nterest'). Στην συνέχεια με το ποντίκι επιλέγουμε την περιοχή 'σχεδιάζοντας' ένα ορθογώνιο γύρω της.



Σχήμα 32. Επιλογή περιοχής ενδιαφέροντος.

Μετά την επιλογή της περιοχής ενδιαφέροντος, δημιουργούμε ένα νέο βίντεο μόνο από αυτή: Κάνουμε δεξί 'κλικ' στο παράθυρο του βίντεο εξόδου και επιλέγουμε την τώρα ενεργοποιημένη εντολή 'Save ROI as AVI…'. Αποθηκεύουμε το βίντεο και ανοίγοντας το βλέπουμε ότι πράγματι περιλαμβάνει μόνο την περιοχή ενδιαφέροντος που είχαμε επιλέξει.

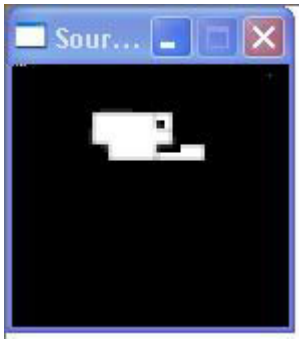


Σχήμα 33. Το νέο βίντεο περιλαμβάνει μόνο την περιοχή ενδιαφέροντος.

Στην συνέχεια μπορούμε να εφαρμόσουμε στο βίντεο, αν κρίνεται απαραίτητο, κάποια από τα φίλτρα προ-επεξεργασίας. Για παράδειγμα μπορεί να απαιτείται απομάκρυνση θορύβου, ενίσχυση υψηλών συχνοτήτων (ακμών) κλπ.

Τέλος προχωράμε στην αναγνώριση της επιθυμητής υφής μέσω της επιλογής 'Recognition→Video Recognition' όπως παρουσιάστηκε στην παράγραφο 2.2. Επιλέγουμε δηλαδή τα χαρακτηριστικά που μας ενδιαφέρουν, το προ-εκπαιδευμένο

νευρωνικό δίκτυο και αποθηκεύουμε το τελικό βίντεο εξόδου. Αυτό είναι της μορφής που φαίνεται στην σχήμα 34.



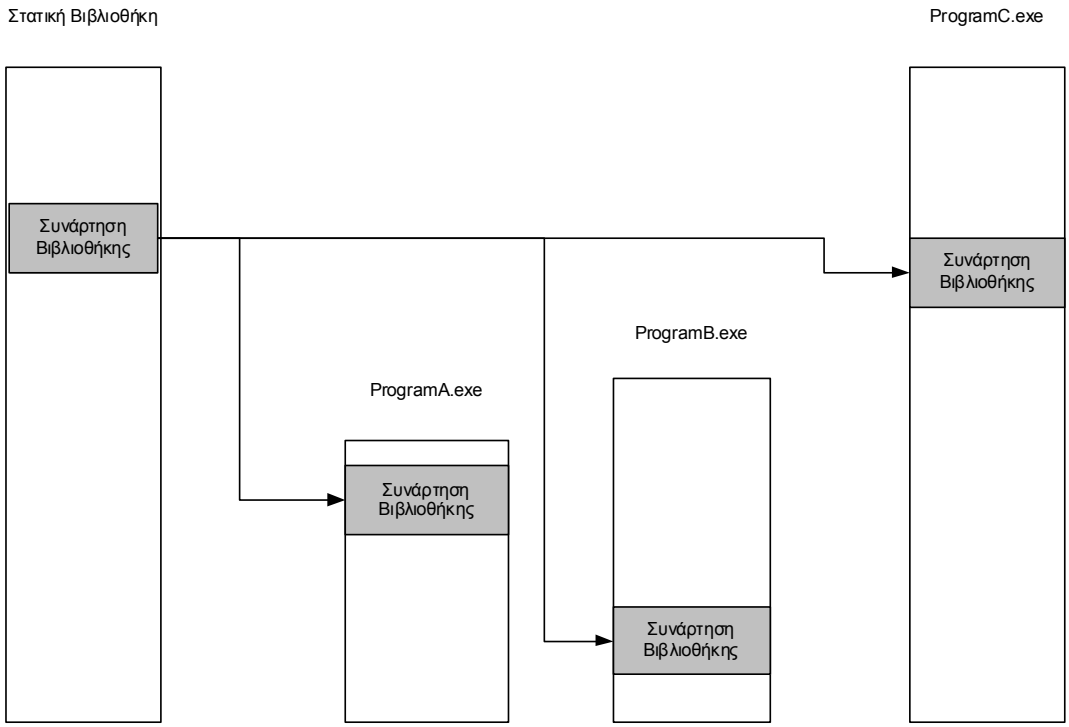
Σχήμα 34. Το τελικό βίντεο εξόδου.

Με άσπρο χρώμα αντιπροσωπεύεται η κλάση της υφής που θέλαμε να αναγνωρίσουμε. Με μαύρο χρώμα είναι όλες οι άλλες. Έτσι με κατάλληλη επιλογή χαρακτηριστικών και εκπαίδευση του νευρωνικού δικτύου μπορούμε να εντοπίσουμε τους καρκινικούς ιστούς. Αν υπάρχει τέτοιος ιστός στο βίντεο εισόδου, τότε θα φανεί στο βίντεο εξόδου ως περιοχές χρωματισμένες με άσπρο χρώμα. Με περαιτέρω επεξεργασία, όπως για παράδειγμα με την χρήση ενός υψηλοπερατού φίλτρου, μπορεί να παραχθεί το περίγραμμα του καρκινικού ιστού που έχει βρεθεί, και με πρόσθεση στο αρχικό βίντεο να φαίνεται ο ιστός αυτός με έντονο το περίγραμμα του.

3 Οι Βιβλιοθήκες Δυναμικής Διασύνδεσης

Στη γλώσσα C/C++ είναι αρκετά συνήθης η χρήση βιβλιοθηκών για κοινά χρησιμοποιούμενες συναρτήσεις. Για παράδειγμα μαθηματικές συναρτήσεις βρίσκονται στη βιβλιοθήκη “math”. Ο κώδικας αυτών των συναρτήσεων βρίσκεται σε ένα αρχείο με κατάληξη “lib”. Όταν δημιουργούμε ένα πρόγραμμα που κάνει χρήση συναρτήσεων αυτής της βιβλιοθήκης ο κώδικάς τους συμπεριλαμβάνεται στο τελικό εκτελέσιμο αρχείο “exe” της εφαρμογής μας. Αν δημιουργήσουμε ένα νέο πρόγραμμα που κάνει επίσης χρήση συναρτήσεων της ίδιας βιβλιοθήκης ένα αντίγραφό τους θα συμπεριληφθεί και στο νέο εκτελέσιμο.

Για παράδειγμα ας θεωρήσουμε ότι η συνάρτηση pow() της βιβλιοθήκης math χρησιμοποιείται σε τρία προγράμματα τα ProgramA.exe, ProgramB.exe και ProgramC.exe. Κάθε ένα από αυτά περιλαμβάνει στο κώδικα του και τον κώδικα της συνάρτησης pow(). Ο κώδικας αυτός συμπεριλήφθηκε στα εκτελέσιμα από την βιβλιοθήκη math κατά το στάδιο της διασύνδεσης (linking). Η συνάρτηση pow() λέμε ότι είναι στατικά συνδεδεμένη (Statically linked) με κάθε εκτελέσιμο και η βιβλιοθήκη "math" ότι είναι βιβλιοθήκη στατικής διασύνδεσης.



Σχήμα 35. Η λειτουργία μιας βιβλιοθήκης στατικής διασύνδεσης.

Η στατική διασύνδεση έχει το πλεονέκτημα της εύκολης ενσωμάτωσης συναρτήσεων σε προγράμματα. Ένα άμεσο μειονέκτημα της όπως φαίνεται είναι η σπατάλη μνήμης και χώρου στο δίσκο. Στα Windows, που είναι ένα πολυδιεργασιακό περιβάλλον, η στατική διασύνδεση πολύ κοινά χρησιμοποιούμενων συναρτήσεων, όπως αυτές εισόδου - εξόδου που βρίσκονται σε κάθε πρόγραμμα, θα προκαλούσε μεγάλη σπατάλη μνήμης αφού κάθε εφαρμογή θα είχε στην μνήμη τα δικά της αντίγραφα των ίδιων συναρτήσεων. Παράλληλα κάθε εκτελέσιμο αρχείο που

βρίσκεται στον δίσκο θα είχε τα δικά του αντίγραφα των ίδιων συναρτήσεων με αποτέλεσμα μεγάλη σπατάλη και αποθηκευτικού χώρου.

Για τους παραπάνω λόγους έχει δημιουργηθεί ένας διαφορετικός τύπος βιβλιοθήκης η Βιβλιοθήκη Δυναμικής Διασύνδεσης (**D**ynamic **L**ink **L**ibrary - DLL).

Οι βιβλιοθήκες δυναμικής διασύνδεσης είναι αρχεία που περιέχουν μια συλλογή από υπομονάδες, όπως συναρτήσεις, κλάσεις, εικόνες, εικονίδια (Icons) κλπ., που μπορούν να χρησιμοποιηθούν από οποιοδήποτε πρόγραμμα. Η συνήθης κατάληξη των αρχείων αυτών είναι ".dll" χωρίς αυτό να είναι υποχρεωτικό.

Η βασική διαφορά με τις βιβλιοθήκες στατικής διασύνδεσης είναι ότι ο κώδικας των συναρτήσεων των βιβλιοθηκών δυναμικής διασύνδεσης δεν προστίθεται στο τελικό εκτελέσιμο που τις χρησιμοποιεί. Αντίθετα ο κώδικας τους βρίσκεται αποκλειστικά μέσα στις βιβλιοθήκες. Όταν εκτελείται ένα πρόγραμμα που χρειάζεται μια από τις συναρτήσεις αυτές φορτώνει την αντίστοιχη βιβλιοθήκη στη μνήμη και κάνει την σύνδεση με την επιθυμητή συνάρτηση. Αν η βιβλιοθήκη βρίσκεται ήδη στη μνήμη δεν την ξαναφορτώνει αλλά χρησιμοποιεί το υπάρχον αντίγραφο της. Η βιβλιοθήκη απομακρύνεται από την μνήμη όταν όλα τα προγράμματα που την χρησιμοποιούν την αποδεσμεύσουν.

Τα πλεονεκτήματα των DLL είναι πολλά. Δεν σπαταλιέται ούτε χώρος στη μνήμη (Όταν έχουμε πολλά προγράμματα να χρησιμοποιούν συγχρόνως την βιβλιοθήκη) ούτε στο δίσκο. Επίσης είναι εύκολη η αναβάθμιση ενός προγράμματος. Μπορούμε να αλλάξουμε μια συνάρτηση του που βρίσκεται σε βιβλιοθήκη δυναμικής διασύνδεσης αλλάζοντας την βιβλιοθήκη και αφήνοντας ανεπηρέαστο το κυρίως εκτελέσιμο. Αυτή η περίπτωση βέβαια έχει και τον κίνδυνο ένα πρόγραμμα να χρησιμοποιεί λάθος έκδοση μιας βιβλιοθήκης που "αναβαθμίσαμε" με αποτέλεσμα να μην δουλεύει.

Οι βιβλιοθήκες δυναμικής διασύνδεσης μπορούν να φορτωθούν στη μνήμη είτε όταν και το πρόγραμμα που τις χρησιμοποιεί φορτώνεται είτε εκ των υστέρων όταν δηλαδή απαιτηθεί η χρήση συναρτήσεων τους. Η πρώτη περίπτωση ονομάζεται "load-time dynamic linking" ή "early binding". Η δεύτερη περίπτωση ονομάζεται "run-time dynamic linking" ή "late binding".

3.1 Η Βιβλιοθήκη Δυναμικής Διασύνδεσης FDIB

Για τον εύκολο χειρισμό των bitmaps έχει δημιουργηθεί η βιβλιοθήκη δυναμικής διασύνδεσης FDIB. Περιλαμβάνει μια μεγάλη γκάμα συναρτήσεων για τον πλήρη χειρισμό αρχείων εικόνας BMP καθώς και RAW. Οι εικόνες αποθηκευμένες σε μορφή RAW είναι στην ουσία bitmaps τα οποία περιλαμβάνουν μόνο τα δεδομένα της εικόνας και καθόλου επικεφαλίδες, παλέτα χρωμάτων ή στοιχίσεις όπως τα BMP.

3.1.1 Αρχικοποίηση Βιβλιοθήκης

Η συνάρτηση αυτή αρχικοποιεί την βιβλιοθήκη FDIB. Πρέπει να καλείται μια φορά πριν από την κλήση κάθε άλλης συνάρτησης της βιβλιοθήκης.

```
void FDIB_Init();
```

3.1.2 Άνοιγμα Αρχείων BMP

Για το άνοιγμα εικόνων BMP και αυτόματη μεταφορά τους στη μνήμη δίνονται δυο συναρτήσεις. Η μια επιστρέφει τα δεδομένα σε μορφή DIB για εύκολο χειρισμό από τα Windows, διασύνδεση με την IPL κλπ. και η άλλη σε μορφή RAW για εύκολη επεξεργασία.

Οι συναρτήσεις αυτές έχουν τα εξής πρότυπα:

```
void FDIB_LoadBMP2DIB(  const char* filename,
                        BITMAPINFOHEADER** pDIB);


void FDIB_LoadBMP2RAW(  const char* filename,
                        char** pRAW,
                        unsigned int* width,
                        unsigned int* height,
                        unsigned int* channels);
```

Η LoadBMP2DIB ανοίγει το αρχείο BMP με όνομα filename, το ανοίγει και δημιουργεί δυναμικά αντίστοιχο DIB που δείχνει το *pDIB. Αντίστοιχα η LoadBMP2RAW ανοίγει το αρχείο BMP με όνομα filename και δεσμεύει χώρο στο *pRAW όπου επιστρέφει τα δεδομένα της εικόνας και στα height, width και channels, το ύψος, το πλάτος και τον αριθμό των χρωματικών καναλιών της εικόνας αντίστοιχα (3 για έγχρωμη RGB εικόνα και 1 για μονόχρωμη). Όταν καλούμε τις συναρτήσεις αυτές τα πεδία *pDIB και *pRAW πρέπει να έχουν τιμή NULL.

Τα αρχεία που ανοίγονται από το δίσκο διαβάζονται και μεταφέρονται στη μνήμη ολόκληρα. Το αρχείο στη συνέχεια κλείνει αυτόματα και δεν απαιτείται χρήση κάποιας άλλης συνάρτησης για το κλείσιμο του. Μετά το πέρας της επεξεργασίας του όμως πρέπει να απελευθερώσουμε τη μνήμη που έχει δεσμευτεί. Σχετικά με την αποδέσμευση της μνήμης βλέπε την παράγραφο 1.7.

3.1.3 Άνοιγμα Αρχείων RAW

Δίνονται και δυο συναρτήσεις για το άνοιγμα αρχείων εικόνας μορφής RAW και μεταφορά τους στη μνήμη. Η πρώτη ανοίγει ένα αρχείο RAW και το μετατρέπει αυτόματα σε μορφή DIB για περαιτέρω επεξεργασία. Η δεύτερη απλά το ανοίγει και μεταφέρει τα δεδομένα του στη μνήμη.

```
void FDIB_LoadRAW2DIB(  const char* filename,
                        const unsigned int width,
                        const unsigned int height,
                        const unsigned int channels,
                        BITMAPINFOHEADER** pDIB);

void FDIB_LoadRAW2RAW(  const char* filename,
                        const unsigned int width,
                        const unsigned int height,
                        const unsigned int channels,
                        char** pRAW);
```

Οι δυο συναρτήσεις παίρνουν ως είσοδο το όνομα του αρχείου ως 'filename', τις διαστάσεις του - ύψος 'height' και πλάτος 'width', καθώς και τον αριθμό των καναλιών του-'channels'. Αυτό το πεδίο έχει τιμή 3 για έγχρωμες εικόνες και 1 για ασπρόμαυρες. Δίνουν ως έξοδο ένα δείκτη σε εικόνα DIB-'pDIB' ή RAW-'pRAW' ανάλογα με την συνάρτηση. Όταν καλούμε τις συναρτήσεις αυτές τα ορίσματα '*pDIB' και '*pRAW' πρέπει να έχουν τιμή NULL.

Όπως και με τις συναρτήσεις ανοίγματος αρχείων BMP έτσι και σε αυτές το αρχείο RAW διαβάζεται, μεταφέρεται στη μνήμη ολόκληρο και κλείνει αυτόματα. Πρέπει όμως να μεριμνήσουμε για την αποδέσμευση της μνήμης που καταλαμβάνεται από τις συναρτήσεις αυτές. Η διαδικασία αυτή αναλύεται στη παράγραφο 1.7.

3.1.4 Αποθήκευση σε Αρχείο BMP

Αυτές οι συναρτήσεις χρησιμοποιούνται για την δημιουργία ενός νέου αρχείου BMP. Υπάρχουν δυο εκδόσεις ανάλογα με το είδος της εισόδου. Αυτή μπορεί να είναι είτε μια εικόνα σε μορφή DIB, είτε μια εικόνα σε μορφή RAW. Στην δεύτερη περίπτωση πρέπει να δώσουμε και τις διαστάσεις της εικόνας, καθώς και τον αριθμό των καναλιών της. Το πεδίο 'filename' και στις δυο συναρτήσεις είναι το όνομα του αρχείου που θα δημιουργηθεί.

```
void FDIB_SaveDIB2BMP(  const BITMAPINFOHEADER* pDIB,
                        const char* filename);

void FDIB_SaveRAW2BMP(  const char* pRAW,
                        const unsigned int width,
                        const unsigned int height,
                        const unsigned int channels,
                        const char* filename);
```

Υπάρχει και η συνάρτηση SaveRAW2BMPForce24 η οποία αποτελεί μια παραλλαγή της SaveRAW2BMP: Αυτό που κάνει επιπλέον είναι να 'υποχρεώνει' το BMP που θα δημιουργηθεί να έχει βάθος χρώματος 24Bit ανεξάρτητα με το αν η RAW εικόνα εισόδου είναι έγχρωμη ή ασπρόμαυρη (8 ή 24bit).

```
void FDIB_SaveRAW2BMPForce24( const char* pRAW,
                              const unsigned int width,
                              const unsigned int height,
                              const unsigned int channels,
                              const char* filename);
```

3.1.5 Αποθήκευση σε Αρχείο RAW

Ανάλογες με τις προηγούμενες συναρτήσεις δημιουργίας αρχείων BMP είναι και οι συναρτήσεις δημιουργίας εικόνων μορφής RAW:

```
void FDIB_SaveDIB2RAW(  const BITMAPINFOHEADER* pDIB,
                        const char* filename);


void FDIB_SaveRAW2RAW(  const char* pRAW,
                        const unsigned int width,
```

```
                        const unsigned int height,
                        const unsigned int channels,
                        const char* filename);
```

Τα πεδία pDIB και pRAW είναι η εικόνα που θέλουμε να αποθηκεύσουμε στο δίσκο. Το όνομα του νέου αρχείου είναι το πεδίο 'filename'. Η συνάρτηση FDIB_SaveRAW2RAW απαιτεί ως είσοδο τις διαστάσεις της εικόνας καθώς και τον αριθμό των καναλιών της.

3.1.6 Μετατροπές ανάμεσα στις μορφές RAW και DIB

Δίνονται δυο συναρτήσεις για την μετατροπή μιας εικόνας που βρίσκεται στη μνήμη από την μορφή RAW στην DIB (ConvertRAW2DIB) και από την DIB στην RAW (ConvertDIB2RAW).

```
void FDIB_ConvertRAW2DIB(const char* pRAW,
                         const unsigned int width,
                         const unsigned int height,
                         const unsigned int channels,
                         BITMAPINFOHEADER** pDIB);

void FDIB_ConvertDIB2RAW(const BITMAPINFOHEADER* pDIB,
                         char** pRAW);
```

Το περιεχόμενο των διπλών δεικτών 'char** pRAW' και 'BITMAPINFOHEADER** pDIB' πρέπει να είναι τιμή NULL όταν καλούμε τις συναρτήσεις αυτές. Στην συνάρτηση ConvertRAW2DIB που παίρνει ως είσοδο τη μορφή RAW πρέπει να δίνουμε επιπλέον τις διαστάσεις της εικόνας και το πλήθος των καναλιών της.

Παρόμοια συνάρτηση με την ConvertRAW2DIB είναι και η ConvertRAW2DIBForce24 η οποία όμως 'υποχρεώνει' την εικόνα DIB που θα δημιουργηθεί να έχει βάθος χρώματος 24bit ανεξάρτητα με το βάθος χρώματος της RAW εικόνας εισόδου.

```
void FDIB_ConvertRAW2DIBForce24( const char* pRAW,
                                 const unsigned int width,
                                 const unsigned int height,
                                 const unsigned int channels,
                                 BITMAPINFOHEADER** pDIB);
```

3.1.7 Αποδέσμευση μνήμης

Οι συναρτήσεις ανοίγματος αρχείων εικόνας και μετατροπής τους ανάμεσα στις μορφές DIB και RAW δεσμεύουν αυτόματα μνήμη για να αποθηκεύσουν την έξοδο τους. Μετά την επεξεργασία που εκτελείτε σε αυτές η μνήμη αυτή πρέπει να αποδεσμευτεί. Για το λόγο αυτό έχουν δημιουργηθεί οι παρακάτω δυο συναρτήσεις:

```
void FDIB_DeallocateDIB(BITMAPINFOHEADER* pDIB);

void FDIB_DeallocateRAW(char* pRAW);
```

Ανάλογα με την μορφή της εικόνας που θέλουμε να αποδεσμεύσουμε χρησιμοποιούμε και την αντίστοιχη συνάρτηση. Για εικόνα μορφής DIB εκτελούμε την πρώτη με το δείκτη pDIB να δείχνει σε αυτή. Για εικόνα μορφής RAW εκτελούμε την δεύτερη με τον δείκτη pRAW να δείχνει στην αυτή.

3.1.8 Εμφάνιση εικόνας DIB σε παράθυρο

Η απεικόνιση μιας εικόνας μορφής DIB σε ένα παράθυρο των windows γίνεται με την συνάρτηση FDIB_DrawDIB.

```
void FDIB_DrawDIB(BITMAPINFOHEADER* pDIB,
                  HDC hDC,
                  HWND hWnd);
```

Η εικόνα που θέλουμε να απεικονίσουμε είναι η pDIB. Το παράθυρο στο οποίο θέλουμε να γίνει η απεικόνιση έχει handle ‘hWnd’ και επιφάνεια σχεδίασης με handle hDC. Μπορούμε εύκολα να πάρουμε την επιφάνεια σχεδίασης για ένα παράθυρο με κλήση της συνάρτησης του WinAPI GetDC().

Η συνάρτηση αυτή δεν υλοποιεί ένα διπλό βρόχο για την απεικόνιση του DIB. Καλεί συναρτήσεις του WinAPI με αποτέλεσμα να έχουμε πολύ μεγάλη βελτίωση στην ταχύτητα του αλγορίθμου.

Ένας επιπλέον περιορισμός της συνάρτησης αυτής είναι το DIB να έχει βάθος χρώματος 24bit.

Για την απεικόνιση εικόνων μορφής RAW απαιτείται πρώτα η μετατροπή τους σε DIB με την συνάρτηση FDIB_ConvertRAW2DIBForce24 (Αν η εικόνα γνωρίζουμε ότι είναι έγχρωμη με βάθος χρώματος 24bit μπορεί να χρησιμοποιηθεί και η FDIB_ConvertRAW2DIB).

3.1.9 Συναρτήσεις Πληροφοριών Εικόνας

Τέλος υπάρχει μια σειρά από βοηθητικές συναρτήσεις που μας δίνουν εύκολα διάφορες χρήσιμες πληροφορίες για εικόνες μορφής DIB.

3.1.9.1 Ύψος Εικόνας

Η συνάρτηση αυτή επιστρέφει το ύψος της εικόνας ‘pDIB’ σε εικονοστοιχεία.

```
int FDIB_GetHeight(const BITMAPINFOHEADER* pDIB);
```

3.1.9.2 Πλάτος Εικόνας

Η συνάρτηση αυτή μας επιστρέφει το πλάτος της εικόνας ‘pDIB’ σε εικονοστοιχεία.

```
int FDIB_GetWidth(const BITMAPINFOHEADER* pDIB);
```

3.1.9.3 Αριθμός Καναλιών Εικόνας

Η συνάρτηση αυτή μας επιστρέφει τον αριθμό των καναλιών της εικόνας 'pDIB'. Για παράδειγμα 3 για έγχρωμη εικόνα 24bit και 1 για ασπρόμαυρη εικόνα 8bit.

```
int FDIB_GetChannels(const BITMAPINFOHEADER* pDIB);
```

3.1.9.4 Έγχρωμη – Ασπρόμαυρη Εικόνα

Η συνάρτηση αυτή μας επιστρέφει true για μονόχρωμη εικόνα και false για έγχρωμη.

```
bool FDIB_IsGray(const BITMAPINFOHEADER* pDIB);
```

3.1.9.5 Τα Δεδομένα της εικόνας

Η συνάρτηση αυτή μας επιστρέφει έναν δείκτη στην αρχή των δεδομένων των εικονοστοιχείων της εικόνας 'pDIB'. Τα δεδομένα αυτά είναι στοιχισμένα.

```
unsigned char* FDIB_GetData(BITMAPINFOHEADER* pDIB);
```

3.1.9.6 Πληροφορίες Στοίχισης

Αυτές οι συναρτήσεις μας δίνουν τον αριθμό των επιπλέον bytes που απαιτούνται για κάθε γραμμή μιας εικόνας DIB. Τα bytes αυτά χρησιμοποιούνται για λόγους στοίχισης (DWORD) των δεδομένων στην μνήμη και έχουν μηδενική τιμή.

```
int FDIB_ExtraAlignmentBytes(const BITMAPINFOHEADER* pDIB);

int FDIB_ExtraAlignmentBytes( const unsigned int width,
                              const unsigned int channels);
```

Η πρώτη συνάρτηση παίρνει ως είσοδο μια εικόνα DIB και κάνει τον υπολογισμό. Η δεύτερη υπολογίζει θεωρητικά τον αριθμό αυτό σύμφωνα με το πλάτος μιας εικόνας και τον αριθμό των καναλιών της.

3.2 Η Βιβλιοθήκη Δυναμικής Διασύνδεσης FAVI

Βασικός στόχος της βιβλιοθήκης FAVI είναι πρόσβαση στα δεδομένα ενός βίντεο AVI. Τα δεδομένα αυτά είναι τα καρέ του βίντεο που λαμβάνουμε σε μορφή εικόνων τύπου DIB. Η βιβλιοθήκη μας δίνει επίσης τη δυνατότητα της αναπαραγωγής του βίντεο σε ένα παράθυρο ή την απεικόνιση ενός συγκεκριμένου καρέ σε αυτό. Μια άλλη δυνατότητά της είναι η δημιουργία ενός νέου βίντεο AVI από μια σειρά από εικόνες τύπου DIB. Το βίντεο αυτό μπορεί να είναι ασυμπίεστο ή και συμπιεσμένο σύμφωνα με τις επιλογές του τελικού χρήστη. Η μεγάλη ευκολία που κερδίζουμε με την χρήση της βιβλιοθήκης είναι ότι δεν χρειάζεται καμία περαιτέρω γνώση σχετικά

με τα AVI, της εσωτερικής αρχιτεκτονικής τους ή του WinAPI για την επεξεργασία τους.

3.2.1 Η δομή SFAVI

Οι συναρτήσεις της βιβλιοθήκης αυτής χρησιμοποιούν μια δομή, την SFAVI, η οποία περιέχει όλες τις απαραίτητες πληροφορίες σχετικά με ένα αρχείο AVI που απαιτούνται από το WinAPI.

```
struct SFAVI {
      char filename[FILENAMELENGTH];
      PAVIFILE pAVIFile;
      AVIFILEINFO fi;
      AVISTREAMINFO     si;
      PAVISTREAM pAVIStream;
      PGETFRAME pGetFrame;
      BITMAPINFOHEADER bmpheader;
      COMPVARS com;}
```

Όλα τα πεδία της συμπληρώνονται αυτόματα και δεν απασχολούν το χρήστη της βιβλιοθήκης. Το μόνο πεδίο που πρέπει να αρχικοποιείται είναι το 'filename' το οποίο περιέχει το όνομα του αρχείου.

Για την επεξεργασία ενός AVI πρέπει να δημιουργήσουμε αρχικά ένα στιγμιότυπο αυτής της δομής και να ορίσουμε το πεδίο του 'filename'. Στην συνέχεια περνάμε ένα δείκτη σε αυτό στις συναρτήσεις της βιβλιοθήκης που θέλουμε να χρησιμοποιήσουμε. Κάθε διαφορετικό AVI απαιτεί και ένα νέο στιγμιότυπο της SFAVI.

3.2.2 Αρχικοποίηση της Βιβλιοθήκης

Η συνάρτηση FAVI_Init() αρχικοποιεί την βιβλιοθήκη FAVI. Πρέπει να καλείται μια φορά πριν από την κλήση κάθε άλλης συνάρτησης της βιβλιοθήκης.

```
void FAVI_Init();
```

Επιπλέον υπάρχει η συνάρτηση FAVI_Exit() που είναι υπεύθυνη για την αποδέσμευση της βιβλιοθήκης και πρέπει να καλείται μια φορά όταν έχει τελειώσει η χρήση της.

```
void FAVI_Exit();
```

3.2.3 Άνοιγμα Αρχείου AVI

Η συνάρτηση FAVI_OpenFile ανοίγει ένα αρχείο AVI που βρίσκεται στο δίσκο για περαιτέρω επεξεργασία από τις συναρτήσεις της βιβλιοθήκης.

```
void  FAVI_OpenFile(SFAVI *sf);
```

Η μοναδική παράμετρος που παίρνει είναι ένας δείκτης σε ένα στιγμιότυπο της δομής SFAVI που θα πρέπει από πριν να έχει δημιουργηθεί. Το μοναδικό πεδίο της δομής που θα πρέπει να έχει αρχικοποιηθεί είναι το 'filename'. Το πεδίο αυτό ορίζει το όνομα του αρχείου AVI που θέλουμε να ανοίξουμε προς επεξεργασία.

3.2.4 Κλείσιμο Αρχείου AVI

Τα αρχεία που έχουν ανοιχθεί με την συνάρτηση FAVI_OpenFile πρέπει μετά το τέλος της επεξεργασία τους να κλείσουν. Αυτό γίνεται με τη συνάρτηση FAVI_CloseFile.

```
void  FAVI_CloseFile(SFAVI *sf);
```

Το μοναδικό της όρισμα είναι ο δείκτης στο στιγμιότυπο της δομής SFAVI που έχει χρησιμοποιηθεί για το AVI που θέλουμε να κλείσουμε. Μετά την εκτέλεση αυτής της συνάρτησης μπορούμε να το αποδεσμεύσουμε το στιγμιότυπο που χρησιμοποιήσαμε.

3.2.5 Διάβασμα Καρέ από AVI

Η συνάρτηση FAVI_GetFrame διαβάζει ένα καρέ από ένα βίντεο που έχει προηγουμένως ανοιχθεί με την FAVI_OpenFile. Για κάθε AVI δημιουργείται αυτόματα ένας μοναδικός buffer που περιέχει το τελευταίο καρέ που έχει διαβαστεί από το αρχείο. Έτσι η συνάρτηση αυτή το μόνο που κάνει είναι να ενημερώνει τον buffer αυτόν με το καρέ που της ζητήσαμε.

```
void  FAVI_GetFrame(    SFAVI *sf,
                        int FrameNum,
                        void** buffer);
```

Το πεδίο 'sf' είναι ο δείκτης στη δομή SFAVI και αναπαριστά το βίντεο στο οποίο αναφερόμαστε. Το πεδίο 'FrameNum' είναι ο αριθμός του καρέ που θέλουμε να διαβάσουμε. Το αποτέλεσμα του διαβάσματος αποθηκεύεται προσωρινά στον buffer η διεύθυνση του οποίου είναι στο πεδίο buffer. Για περαιτέρω χρήση και 'μόνιμη' αποθήκευση ενός καρέ πρέπει να δεσμεύεται ανάλογος χώρος και να αντιγράφεται ο buffer εκεί. Το περιεχόμενο του buffer είναι σε μορφή DIB και με μετατροπή σε BITMAPINFOHEADER* μπορεί εύκολα να χρησιμοποιηθεί ως bitmap.

3.2.6 Εμφάνιση καρέ σε παράθυρο

Η εμφάνιση ενός καρέ σε ένα παράθυρο των windows γίνεται με την συνάρτηση FAVI_ShowFrame. Το βίντεο θα πρέπει προηγουμένως να έχει ανοιχθεί με τη συνάρτηση FAVI_OpenFile.

```
void  FAVI_ShowFrame(   SFAVI *sf,
                        int FrameNum,
                        HDC dc);          //Show Frame to DC
```

Η πρώτη παράμετρος που παίρνει είναι η δομή SFAVI που αναπαριστά το βίντεο στο οποίο αναφερόμαστε. Το 'FrameNum' είναι ο αριθμός του καρέ που θέλουμε να σχεδιάσουμε. Το 'dc' είναι ο handle στην επιφάνεια σχεδίασης ενός παραθύρου πάνω στην οποία θα γίνει η σχεδίαση. Το handle αυτό το παίρνουμε εύκολα για ένα δεδομένο παράθυρο με κλήση της συνάρτησης GetDC του API των Windows [11].

3.2.7 Αναπαραγωγή του βίντεο σε παράθυρο

Μπορούμε να αναπαραγάγουμε ένα βίντεο σε ένα παράθυρο με την συνάρτηση FAVI_PlayVideo3.

```
int FAVI_PlayVideo3(SFAVI *sf,
                    HWND hWnd,
                    HDC dc,
                    unsigned int* StartFrame,
                    char* DoStop,
                    int msg=0);
```

Το πεδίο 'sf 'είναι η δομή SFAVI που αναπαριστά το βίντεο που θέλουμε να αναπαραγάγουμε. Το 'hWnd' είναι το handle του παραθύρου στο οποίο θα γίνει η αναπαραγωγή και το 'dc' είναι το handle της επιφάνειας σχεδίασης μέσα στο παράθυρο αυτό. Η αναπαραγωγή του βίντεο θα ξεκινήσει από το καρέ με αριθμό 'StartFrame'. Όταν τελειώσει ή διακοπεί η αναπαραγωγή, το πεδίο αυτό θα έχει τον αριθμο του τελευταίου καρέ που προβλήθηκε.

Το πεδίο 'DoStop' καθορίζει την τρέχουσα κατάσταση της αναπαραγωγής και μας δίνει την δυνατότητα να διακόψουμε την αναπαραγωγή όποτε το επιθυμούμε. Όταν καλούμε την συνάρτηση πρέπει να του έχουμε δώσει την τιμή V_PLAYING η οποία διατηρείται κατά τη διάρκεια όλης της αναπαραγωγής του βίντεο. Αν θελήσουμε να διακόψουμε την αναπαραγωγή αλλάζουμε την τιμή του σε V_STOP. Η συνάρτηση τότε θα την διακόψει και με την σειρά της θα αλλάξει την τιμή του πεδίου σε V_NORM.

Παράλληλα με την αναπαραγωγή η συνάρτηση στέλνει στο παράθυρο 'hWnd' και ένα μήνυμα με αριθμό 'msg' κάθε φορά που εμφανίζεται και ένα νέο καρέ. Στη παράμετρο 'wParam' του μηνύματος που στέλνεται βρίσκεται ο αριθμός του καρέ. Έτσι η τελική εφαρμογή ελέγχοντας τα μηνύματα 'msg' μπορεί να ξέρει συνεχώς σε ποιό καρέ βρίσκεται η αναπαραγωγή.

Τέλος πρέπει να αναφερθεί ότι η αναπαραγωγή του βίντεο με την συνάρτηση FAVI_PlayVideo3 γίνεται σε διαφορετικό νήμα από αυτό της κυρίως εφαρμογής.

3.2.8 Δημιουργία-Αποθήκευση Νέου AVI

Συχνά δημιουργείται η ανάγκη δημιουργίας ενός καινούργιου βίντεο από μια σειρά από εικόνες. Η βιβλιοθήκη FAVI μας δίνει την δυνατότητα αυτή. Μπορούμε να δημιουργήσουμε ένα καινούριο βίντεο από εικόνες μορφής DIB. Όλες θα πρέπει να έχουν το ίδιο μέγεθος, το ίδιο βάθος χρώματος και στην περίπτωση πού είναι συμπιεσμένες τον ίδιο τύπο συμπίεσης.

Η διαδικασία που ακολουθούμε για την δημιουργία του βίντεο έχει ως εξής: Αρχικά καλούμε μια φορά την συνάρτηση FAVI_SaveAVICreate:

```
void FAVI_SaveAVICreate(SFAVI *sf,
                        UINT nTotalFrames,
                        BITMAPINFOHEADER* pFrame);
```

Αυτή παίρνει ως παράμετρο μια δομή τύπου SFAVI που θα πρέπει να έχουμε δημιουργήσει. Το μοναδικό πεδίο στη δομή που θα πρέπει να συμπληρώσουμε είναι το όνομα του νέου αρχείου που θέλουμε να δημιουργηθεί. Η δεύτερη παράμετρος που παίρνει η συνάρτηση είναι ο συνολικός αριθμός των καρέ από τα οποία θα αποτελείται το τελικό βίντεο. Το τελευταίο πεδίο της συνάρτησης είναι ένα από τα καρέ αυτά. Μπορούμε να δώσουμε οποιοδήποτε από τα καρέ και όχι απαραίτητα το πρώτο. Η συνάρτηση χρησιμοποιεί την παράμετρο αυτή μόνο για να πάρει πληροφορίες σχετικές με το μέγεθος του καρέ, τον τύπο του, το βάθος χρώματος κλπ αλλά δεν το προσθέτει στο βίντεο. Αφού την καλέσουμε θα δημιουργήσει αυτόματα ένα παράθυρο που θα ζητάει από τον χρήστη της εφαρμογής να επιλέξει, αν επιθυμεί, τον τύπο και τα χαρακτηριστικά του συμπιεστή που θα χρησιμοποιηθεί για την συμπίεση του βίντεο. Η συμπίεση θα γίνει αυτόματα ανάλογα με τις επιλογές του χρήστη.

Αφού καλέσαμε τη συνάρτηση FAVI_SaveAVICreate θα πρέπει να καλέσουμε την FAVI_SaveAVIAddDIB μια φορά για κάθε καρέ που θέλουμε να προσθέσουμε στο βίντεο:

```
void FAVI_SaveAVIAddDIB(SFAVI *sf,
                        UINT nFrameNum,
                        BITMAPINFOHEADER* pFrame);
```

Ο αριθμός των φορών που την καλούμε θα πρέπει να είναι ο ίδιος με τον αριθμό nTotalFrames που δηλώσαμε στην FAVI_SaveAVICreate. Οι παράμετροι που παίρνει η συνάρτηση FAVI_SaveAVIAddDIB είναι ο δείκτης στη δομή SFAVI που έχουμε δημιουργήσει και χρησιμοποιήσει, ο αριθμός του καρέ που προσθέτουμε και τέλος ο δείκτης pFrame δείχνει στην ίδια την εικόνα. Τα καρέ του βίντεο πρέπει να τα προσθέτουμε με την σωστή σειρά με την οποία θέλουμε να εμφανίζονται στο τελικό βίντεο. Ο αριθμός του πρώτου καρέ είναι το 0.

Μετά την ολοκλήρωση της προσθήκης όλων των καρέ στο βίντεο θα πρέπει να καλέσουμε μια φορά την συνάρτηση FAVI_SaveAVIClose η οποία θα κλείσει το νέο αρχείο που έχει δημιουργηθεί:

```
void FAVI_SaveAVIClose(SFAVI *sf);
```

Μοναδική παράμετρος της συνάρτησης είναι ο δείκτης στη δομή SFAVI του αρχείου που δημιουργήθηκε.

3.2.9 Βοηθητικές Συναρτήσεις

Τέλος υπάρχει μια σειρά από βοηθητικές συναρτήσεις που μας δίνουν εύκολα διάφορες χρήσιμες πληροφορίες για αρχεία AVI που έχουν ανοιχθεί με την FAVI_OpenFile().

1) Συνολικός Αριθμός Καρέ

Η συνάρτηση αυτή μας επιστρέφει τον συνολικό αριθμό των καρέ του βίντεο.

```
UINT FAVI_GetTotalFrames(SFAVI *sf);
```

2) Πλάτος Βίντεο

Η συνάρτηση αυτή μας επιστρέφει το πλάτος του βίντεο σε εικονοστοιχεία.

```
int FAVI_GetWidth(SFAVI *sf);
```

3) Ύψος Βίντεο

Η συνάρτηση αυτή μας επιστρέφει το ύψος του βίντεο σε εικονοστοιχεία.

```
int FAVI_GetHeight(SFAVI *sf);
```

3.3 Η Βιβλιοθήκη Δυναμικής Διασύνδεσης FIPL

Η βιβλιοθήκη δυναμικής διασύνδεσης FIPL βασίζεται άμεσα στην βιβλιοθήκη IPL της Intel. Στην πραγματικότητα αποτελεί ένα υποσύνολο της και ενσωματώνει μέρος των δυνατοτήτων της IPL. Από την άλλη πλευρά όμως είναι πολύ ευκολότερη στη χρήση και παρουσιάζει εύκολη διασύνδεση με τις εικόνες τύπου DIB και RAW. Έτσι ο χρήστης της δεν χρειάζεται να γνωρίζει τίποτα για την περίπλοκη δομή των εικόνων της IPL.

3.3.1 Αρχικοποίηση Βιβλιοθήκης

Η συνάρτηση αυτή αρχικοποιεί την βιβλιοθήκη FIPL. Πρέπει να καλείται μια φορά πριν από την κλήση κάθε άλλης συνάρτησης της βιβλιοθήκης.

```
void FIPL_Init();
```

3.3.2 Μετατροπές ανάμεσα σε εικόνες IPL και RAW ή DIB

Οι παρακάτω συναρτήσεις χρησιμοποιούνται για την δημιουργία μιας εικόνας IPL από εικόνα τύπου RAW ή DIB που βρίσκεται στη μνήμη καθώς και την μετατροπή μιας εικόνας IPL σε μορφή RAW ή DIB.

- IPL → RAW

```
void FIPL_ConvertIPL2RAW(const IplImage *im,
                         unsigned char **pRAW);
```

- IPL → DIB

```
void FIPL_ConvertIPL2DIB(IplImage* im,
```

```
                        BITMAPINFOHEADER** pDIB);
```

- RAW → IPL

```
void FIPL_ConvertRAW2IPL(const char* pRAW,
                        const unsigned int width,
                        const unsigned int height,
                        const unsigned int channels,
                        IplImage** pIm);
```

- DIB → IPL

```
void FIPL_ConvertDIB2IPL(BITMAPINFOHEADER* pDIB,
                        IplImage** pIm);
```

3.3.3 Αντιγραφή Εικόνας

Η συνάρτηση αυτή δημιουργεί μια νέα εικόνα, με νέα επικεφαλίδα και δεδομένα, ακριβές αντίγραφο της εικόνας εισόδου.

```
IplImage* FIPL_CloneImage(IplImage* pIplImage);
```

Η εικόνα εισόδου είναι η pIplImage. Η συνάρτηση επιστρέφει έναν δείκτη στην επικεφαλίδα της νέας εικόνας που έχει δημιουργηθεί.

3.3.4 Δημιουργία νέας εικόνας από ROI

Η συναρτήση αυτή δημιουργεί μια νέα εικόνα IPL από μια υπάρχουσα. Η νέα εικόνα αποτελείται από την περιοχή ενδιαφέροντος (ROI) που έχει οριστεί στην αρχική. Αυτό που κάνει είναι να "κόβει" την ROI και να την κάνει νέα εικόνα.

```
void FIPL_CutROI2IPL(   IplImage* im,
                        IplImage** newIm);
```

Η παράμετρος "im" είναι η αρχική εικόνα και το περιεχόμενο του δείκτη newIm είναι η νέα εικόνα που δημιουργήθηκε.

3.3.5 Αποδέσμευση εικόνας IPL

Η παρακάτω συνάρτηση παίρνει ως είσοδο μια εικόνα IPL και αποδεσμεύει πλήρως την περιοχή που καταλαμβάνει στη μνήμη. Δηλαδή απομακρύνει τόσο τα δεδομένα της εικόνας, όσο και την επικεφαλίδα της.

```
void FIPL_DeallocateIPL(IplImage* pIplImage);
```

3.3.6 Βοηθητικές Συναρτήσεις

3.3.6.1 Ύψος Εικόνας

Η συνάρτηση αυτή μας επιστρέφει το ύψος της εικόνας 'pIplImage' σε εικονοστοιχεία.

```
int FIPL_GetHeight(const IplImage* pIplImage);
```

3.3.6.2 Πλάτος Εικόνας

Η συνάρτηση αυτή μας επιστρέφει το πλάτος της εικόνας 'pIplImage' σε εικονοστοιχεία.

```
int FIPL_GetWidth(const IplImage* pIplImage);
```

3.3.6.3 Αριθμός Καναλιών Εικόνας

Η συνάρτηση αυτή μας επιστρέφει τον αριθμό των καναλιών της εικόνας 'pDIB'. Για παράδειγμα 3 για έγχρωμη εικόνα 24bit και 1 για ασπρόμαυρη εικόνα 8bit.

```
int FIPL_GetChannels(const IplImage* pIplImage);
```

3.3.6.4 Τα δεδομένα της εικόνας

Η συνάρτηση αυτή μας επιστρέφει έναν δείκτη στα δεδομένα της εικόνας pIplImage. Τα δεδομένα αυτά ακολουθούν την οργάνωση των δεδομένων μιας εικόνας IPL.

```
unsigned char* FIPL_GetData(IplImage* pIplImage);
```

3.3.6.5 Έγχρωμη – Ασπρόμαυρη Εικόνα

Η συνάρτηση αυτή επιστρέφει 0 αν η εικόνα pIplImage είναι ασπρόμαυρη (8bit) και 1 αν η εικόνα είναι έγχρωμη.

```
int FIPL_IsGray(const IplImage* pIplImage);
```

3.3.7 Χρωματικές Μετατροπές

Η μετατροπή μιας έγχρωμης εικόνας σε ασπρόμαυρη γίνεται με την συνάρτηση FIPL_ColorToGray. Είναι δυνατή και η αντίστροφη διαδικασία με την συνάρτηση FIPL_GrayToColor. Σε αυτή την περίπτωση απλά αλλάζει η εσωτερική δομή της εικόνας ώστε να υποστηρίζονται 3 κανάλια (τα R,G,B) αντί για το ένα της ασπρόμαυρης, ενώ η εικόνα παραμένει ασπρόμαυρη. Τα τρία κανάλια που δημιουργούνται έχουν τιμές και τα τρία ίσες με το αρχικό κανάλι της ασπρόμαυρης εικόνας.

```
void FIPL_ColorToGray(IplImage* iplIm);
```

```
void FIPL_GrayToColor(IplImage* iplIm);
```

Η μοναδική παράμετρος που παίρνουν οι δυο συναρτήσεις είναι η iplIm που αντιπροσωπεύει την εικόνα που θέλουμε να μετατρέψουμε.

3.3.8 Φίλτρα

Σε όλα τα φίλτρα η εικόνα εισόδου είναι η παράμετρος image. Αυτή αποτελεί και την εικόνα εξόδου αφού όλες οι αλλαγές θα γίνουν πάνω της.

3.3.8.1 Κατώφλι

Η συνάρτηση FIPL_Threshold Βάζει ένα κατώφλι σε μια εικόνα με αποτέλεσμα κάθε κανάλι της να αποκτά μόνο δυο δυνατές τιμές, την μέγιστη 255 και την ελάχιστη 0.

```
 void FIPL_Threshold(IplImage* image,
                          int value);
```

Η παράμετρος value είναι η τιμή του κατωφλίου.

3.3.8.2 Αριθμητικά

Οι παρακάτω συναρτήσεις υλοποιούν τα αριθμητικά φίλτρα ανάμεσα σε μια εικόνα και έναν αριθμό (Εκτός την ύψωση στο τετράγωνο που δεν δέχεται αριθμό):

- Πρόσθεση

```
void FIPL_AddS(IplImage* image,
                   int x);
```

- Αφαίρεση

```
void FIPL_SubtractS(IplImage* image,
                    int value,
                    bool flip);
```

- Πολλαπλασιασμός

```
void FIPL_MultiplyS(IplImage* image,
                         int value);
```

- Πολλαπλασιασμός με κλίμακα

```
void FIPL_MultiplySScale(IplImage* image,
                              int value);
```

- Τετράγωνο

```
void FIPL_Square(IplImage* image);
```

Η αριθμητική τιμή είναι η value. Ειδικά η αφαίρεση έχει και την παράμετρο flip η οποία όταν έχει τιμή true αντί να αφαιρείται η value από την εικόνα γίνεται το αντίστροφο δηλαδή αφαιρείται από την τιμή value η εικόνα.

3.3.8.3 Δυαδικά Αριθμητικά

Οι παρακάτω συναρτήσεις υλοποιούν αριθμητικές πράξεις ανάμεσα σε δυο εικόνες.

- Πρόσθεση

```
void FIPL_Add(    IplImage* image,
                  IplImage* mask);
```

- Αφαίρεση

```
void FIPL_Subtract(IplImage* image,
                  IplImage* mask);
```

- Πολλαπλασιασμός

```
void FIPL_Multiply(IplImage* image,
                  IplImage* mask);
```

- Πολλαπλασιασμός με κλίμα

```
void FIPL_MultiplyScale(IplImage* image,
                        IplImage* mask);
```

Η παράμετρος mask είναι η εικόνα που θα προσθέσουμε, αφαιρέσουμε ή πολλαπλασιάσουμε με την εικόνα εισόδου image.

3.3.8.4 Δυαδικά Λογικά

Οι συναρτήσεις αυτές μας προσφέρουν τις λογικές δυαδικές πράξεις AND, OR και XOR:

- Λογικό ΚΑΙ (AND)

```
void FIPL_AND(    IplImage* image,
                  IplImage* mask);
```

- Λογικό Η΄ (OR)

```
void FIPL_OR(     IplImage* image,
                  IplImage* mask);
```

- Λογικό Αποκλειστικό Η΄ (XOR)

```
void FIPL_XOR(    IplImage* image,
                  IplImage* mask);
```

Η παράμετρος mask είναι η μάσκα δηλαδή ο δεύτερος τελεστής της λογικής πράξης που κάνουμε.

3.3.8.5 Γραμμικά

Τα γραμμικά φίλτρα που υλοποιούνται είναι ο μέσος όρος και ο συγκερασμός.

- Μέσος Όρος

```
void FIPL_Blur(   IplImage* image,
                  int cols,
                  int rows,
                  int x,
                  int y);
```

Οι παράμετροι cols (Μήκος) και raws (Ύψος) καθορίζουν τις διαστάσεις του παραθύρου. Επίσης οι παράμετροι x, y μας δίνουν την δυνατότητα της μετακίνησης της εικόνας εισόδου κατά ανάλογο αριθμό εικονοστοιχείων σε κάθε άξονα. Αυτό είναι χρήσιμο διότι μετά από την εφαρμογή αυτού του φίλτρου γίνεται μια μετατόπιση της εικόνας ανάλογη με το μέγεθος του παραθύρου. Έτσι δίνοντας για παράδειγμα τιμές στα x και y ίσες με cols/2 και raws/2 έχουμε την δυνατότητα να "κεντράρουμε" την μετατοπισμένη εικόνα εξόδου.

- Συγκερασμός

```
void FIPL_Convolution(  IplImage* image,
                        int kCols,
                        int kRows,
                        int* kernel);
```

Η παράμετρος kernel είναι ο πίνακας του συγκερασμού. Πρέπει να έχει kCols*kRows πλήθος στοιχείων, όπου kCols είναι ο αριθμός των στηλών του και kRaws ο αριθμός των γραμμών του.

3.3.8.6 Μη γραμμικά

Οι παρακάτω συναρτήσεις υλοποιούν τα μη γραμμικά φίλτρα Μέγιστου, Ελάχιστου και Ενδιάμεσης Τιμής.

- Μέγιστου (Max)

```
void FIPL_Max(    IplImage* image,
                  int cols,
                  int rows,
                  int x,
                  int y);
```

- Ελάχιστου (Min)

```
void FIPL_Min(    IplImage* image,
                  int cols,
```

```
                  int rows,
                  int x,
                  int y);
```

- Ενδιάμεσης Τιμής (Median)

```
void FIPL_Median( IplImage* image,
                  int cols,
                  int rows,
                  int x,
                  int y);
```

Οι παράμετροι cols (Μήκος) και raws (Ύψος) καθορίζουν τις διαστάσεις του παραθύρου. Δίνεται επίσης η δυνατότητα μετακίνησης της εικόνας εξόδου κατά x εικονοστοιχεία στον οριζόντιο άξονα και y εικονοστοιχεία στον κατακόρυφο.

3.3.8.7 Μορφολογικά

Τα μορφολογικά φίλτρα Erode, Dilate, Open και Close υλοποιούνται από τις παρακάτω τέσσερις συναρτήσεις.

- Συστολή (Erode)

```
void FIPL_Erode(  IplImage* image,
                  int nIterations);
```

- Διαστολή (Dilate)

```
void FIPL_Dilate( IplImage* image,
                  int nIterations);
```

- Ανοικτότητα (Open)

```
void FIPL_Open(   IplImage* image,
                  int nIterations);
```

- Κλειστότητα (Close)

```
void FIPL_Close(  IplImage* image,
                  int nIterations);
```

Η παράμετρος nIterations είναι ο αριθμός των επαναλήψεων που θέλουμε να εφαρμοστεί το φίλτρο. Το σύνολο με το οποίο γίνεται ο κάθε μετασχηματισμός (Το σύνολο C που παρουσιάζεται στην παρ. 1.5.6) είναι ένας πίνακας 3x3 με όλα τα στοιχεία του ίσα με την μονάδα.

3.3.8.8 Μετασχηματισμός Fourier

Η συνάρτηση FIPL_FFT μας επιτρέπει την μεταφορά μιας εικόνας στο πεδίο των συχνοτήτων. Αυτό γίνεται με χρήση του δισδυάστατου Διακριτού Μετασχηματισμού Fourier (DFT) που υλοποιείται με τον αλγόριθμο του Γρήγορου Μετασχηματισμού Fourier (FFT). Επίσης είναι δυνατός και ο αντίστροφος μετασχηματισμός δηλαδή το πέρασμα από το πεδίο των συχνοτήτων στο χωρικό.

```
void FIPL_FFT(    IplImage* image,
                  int Forward);
```

Η παράμετρος Forward είναι μια σημαία. Όταν έχει τιμή 1 γίνεται ο ευθύς μετασχηματισμός ενώ όταν είναι 0 γίνεται ο αντίστροφος.

3.3.8.9 Προσθήκη τεχνητού θορύβου

Πολλές φορές για να προσομοιωθούν οι πραγματικές συνθήκες είναι χρήσιμο να υπάρχει η δυνατότητα πρόσθεσης στις εικόνες τεχνητού θορύβου. Η βιβλιοθήκη FIPL καλύπτει αυτήν την ανάγκη προσφέροντας τρεις συναρτήσεις προσθήκης τριών διαφορετικών τύπων θορύβου. Το πρώτο όρισμα που παίρνουν είναι η εικόνα τύπου IPL που θέλουμε να αλλοιωθεί. Οι τρεις τύποι θορύβου που έχουν υλοποιηθεί είναι:

- Κρουστικός θόρυβος.

```
void FIPL_NoiseSaltAndPepper( IplImage* im,
                              double m);
```

Ο θόρυβος αυτός έχει ως αποτέλεσμα ένα ποσοστό m των εικονοστοιχείων της εικόνας εισόδου να αλλάζει τιμή και να αποκτούν τα μισά από αυτά την μέγιστη δυνατή τιμή (λευκό = 255) και τα υπόλοιπα μισά την ελάχιστη δυνατή (μαύρο - 0).

- Gaussian λευκός θόρυβος.

```
void FIPL_NoiseGaussian(IplImage* im,
                        double m,
                        double v);
```

Ο θόρυβος αυτός είναι προσθετικός. Σε κάθε εικονοστοιχείο της εικόνας προστίθεται μια τιμή, θετική ή αρνητική. Η συνολική ποσότητα που προστίθεται ακολουθεί Gaussian κατανομή μέσης τιμής m και διασποράς v.

- Θόρυβος Βολής

```
void FIPL_NoiseSpeckle( IplImage* im,
                        double v);
```

Ο θόρυβος αυτός είναι πολλαπλασιαστικού τύπου. Ο αλγόριθμος που χρησιμοποιεί για τον υπολογισμό της εικόνας εξόδου βασίζεται στον τύπο:

$$J = I + n*I$$

όπου J είναι η εικόνα εξόδου, I η εικόνα εισόδου και n είναι μια τιμή ομοιόμορφα κατανεμημένου θορύβου μέσης τιμής 0 και διασποράς v [12]. Η υλοποίηση του

κώδικα σε έγινε με χρήση της βιβλιοθήκης SPL (**S**ignal **P**rocessing **L**ibrary) της Intel [13].


Salt & Pepper


Gaussian


Speckle

Σχήμα 36. Τα αποτελέσματα των τριών τύπου θορύβου στην εικόνα 'cameraman' με τιμή παραμέτρου 0.05 (μέση τιμή 0 στον Gaussian).

3.4 Παράδειγμα χρήσης των βιβλιοθηκών

Στην παράγραφο αυτή δίνεται ένα πλήρες πρόγραμμα Windows γραμμένο σε C που κάνει χρήση των τριών βιβλιοθηκών που παρουσιάστηκαν. Το πρόγραμμα ονομάζεται "Easy2". Αυτό που κάνει είναι να ανοίγει το βίντεο με όνομα "Confused4.avi", να παίρνει το πέμπτο καρέ του και αφού του περάσει δυο φίλτρα της FIPL (Blur και Dilate) το εμφανίζει σε ένα παράθυρο. Η πλήρης κατανόηση του παραδείγματος προϋποθέτει βασικές γνώσεις προγραμματισμού με το API των Windows. Ο στόχος είναι να γίνει κατανοητός ο τρόπος κλήσεως των συναρτήσεων των βιβλιοθηκών.

```
#include <windows.h>                        //Genika gia windows
#include <wingdi.h>                         //Gia ta BITMAP

#include "FAVI.h"       //Sxetika me AVI
#include "FDIB.h"       //Sxetika me DIB
#include "FIPL.h"       //Sxetika me IPL

//Protypo Diekperoti minimatwn
long WINAPI WindowProc( HWND hwnd,UINT uMsg,
                        WPARAM wParam,LPARAM lParam);

SFAVI myavi;                    //To Video pou tha xrisimopoiisoume
BITMAPINFOHEADER* pFrame=NULL;//To frame
IplImage* pIplImage;            //H iplImage

int WINAPI WinMain(HINSTANCE hInstance,  // handle to current
instance
            HINSTANCE hPrevInstance,// handle to previous instance
            LPSTR lpCmdLine,          // pointer to command line
            int nCmdShow   )           // window show state
{
      FAVI_Init();                  //Arxikopoisi FAVI
      FIPL_Init();                  //Arxikopoisi FIPL
```

```
      FDIB_Init();                  //Arxikopoisi FDIB

      strcpy(myavi.filename,"Confused4.avi");//To AVI pou tha
anoixtei
      FAVI_OpenFile(&myavi);              //Anoigma tou AVI

      FAVI_GetFrame(&myavi,5,
                  (void**)&pFrame);//Pairnoume to frame se morfi
                  DIB

      WNDCLASS wndClass;//H klasi tou parathirou pou tha
dimiourgisoume
      memset(&wndClass,0,sizeof(wndClass));//Midenismos
axrisimopoiitwn pediwn
      wndClass.hInstance=hInstance; //To instance tou parathirou
      wndClass.lpfnWndProc=WindowProc;//H synartisi twn minimatwn
      wndClass.lpszClassName="Easy2Class";//To onoma ths classis

      RegisterClass(&wndClass);//Kanoume register th classi

      HWND hwndMain =CreateWindow("Easy2Class", //Dimiourgeia tou
                                                //parathirou
            "Easy2",                          //Titlos
            WS_OVERLAPPEDWINDOW|WS_VISIBLE,   //Normal parathiro
            100,                  //Thesi tou parathirou X
            100,                  //Thesi tou parathirou Y
            320,                  //Platos
            240,                  //Ypsos
            NULL,                 //Goneas
            NULL,                 //Menu
            hInstance,            //To hInstance
            NULL);                //CreateStruct

      try
      {
            pIplImage=NULL;         //Arxikopoiisi tis IPL eikonas
            FIPL_ConvertDIB2IPL(pFrame,&pIplImage);// DIB->IPL
            FIPL_Blur(pIplImage,3,3,2,2); //Perasma filtrou Blur
            FIPL_Dilate(pIplImage,3);     //Perasma filtrou Dilate
      }
      catch(char* msg)//Elegxos lathwn
      {
            MessageBox(hwndMain,msg,"Error",MB_OK);
            exit(1);
      }
      UpdateWindow(hwndMain); //Epanasxediasi tou parathirou

      MSG msg; //Broxos minimatwn
      while (GetMessage(&msg, (HWND) NULL,
                        0, 0))      //An yparxei minima to pernoume
    {
        TranslateMessage(&msg);
        DispatchMessage(&msg);      //Diekperewsi minimatos
    }
      try
      {
            FIPL_DeallocateIPL(pIplImage);//Apodesmeusi mnimis
            FAVI_CloseFile(&myavi);       //Kleisimo tou arxeiou
      }
      catch(char* msg)
      {
```

```
            MessageBox(hwndMain,msg,"Error",MB_OK);
            exit(1);
      }

      return 0;
}

long WINAPI WindowProc( HWND hwnd,      // handle to window
                        UINT uMsg,      // message identifier
                        WPARAM wParam,  // first message parameter
                        LPARAM lParam)   // second message parameter
{
      switch(uMsg)
      {
      case WM_PAINT:    //Epanasxediasmos tou parathirou
            {
            HDC hDC=GetDC(hwnd);//Desmeusi epifanias sxediasis
            try   {
                  FIPL_ConvertToDIB(pIplImage,pFrame);      //IPL-
>DIB
                  FDIB_DrawDIB(pFrame,hDC,hwnd);//Sxediasi telikou
DIB                     }
                  catch(char* msg)  {
                        MessageBox(hwnd,msg,"Error",MB_OK);
                        exit(1);    }
                  ReleaseDC(hwnd,hDC);//Apeleutherwsi epifanias
                                      //sxediasis
                  break;
            }
      case WM_DESTROY:
            PostQuitMessage(0);
            return 0;
      }
      return DefWindowProc(hwnd,uMsg,wParam,lParam);
}
```

Παραρτήματα

Στα παραρτήματα που ακολουθούν θα καλυφθούν τα τεχνικά ζητήματα των θεμάτων που συζητήθηκαν από την πλευρά του προγραμματιστή. Συγκεκριμένα θα γίνει αναφορά στις μεθόδους κωδικοποίησης και αποθήκευσης εικόνας BMP και DIB, στα βίντεο AVI και στη βιβλιοθήκη IPL.

1 Εικόνες DIB και BMP

1.1 Οι εικόνες DIB και τα αρχεία εικόνας BMP

Τα καρέ των βίντεο αντιμετωπίζονται από τα Windows ως εικόνες bitmap. Τα διαχειρίζονται με τον ίδιο τρόπο με τον κλασικό τύπο εικόνων των Windows, τα αρχεία BMP (Bitmap). Είναι σκόπιμο λοιπόν να μελετήσουμε τα BMP αφού αποτελούν το δομικό λίθο των AVI. Τα αρχεία αυτά συναντώνται αποθηκευμένα με κατάληξη BMP ή DIB (Device Independent Bitmap).

Χωρίζονται σε δυο κλάσεις. Τα ανεξάρτητα συσκευής (Device-Independent Bitmaps - DIB) και τα εξαρτημένα συσκευής (Device-Dependent Bitmaps (DDB)). Τα πρώτα σχεδιάστηκαν με σκοπό να μπορούν να αναπαρασταθούν και διαχειριστούν σωστά από κάθε εφαρμογή ανεξάρτητα από αυτήν στην οποία δημιουργήθηκαν. Τα DDB χρησιμοποιούνταν ευρύτατα στο παρελθόν και εξαρτιόντουσαν άμεσα από τις δυνατότητες και τις ρυθμίσεις τόσο του λογισμικού όσο και του υλικού του συστήματος που τα διαχειριζόταν. Στην συνέχεια λόγο των πολλών προβλημάτων τους αντικαταστάθηκαν από τα DIB με τα οποία θα ασχοληθούμε αποκλειστικά.

Όταν αναφέρεται ένα "DIB" εννοούμε μια εικόνα bitmap που βρίσκεται στην μνήμη. Όταν αναφέρεται ένα BMP εννοούμε μια εικόνα bitmap που βρίσκεται ως αρχείο τύπου *.BMP αποθηκευμένη στο δίσκο.

1.2 Εσωτερική Οργάνωση των DIB

Τα bitmaps αποτελούνται από ένα structure ως επικεφαλίδα και τα δεδομένα της εικόνας αμέσως μετά. Το structure αυτό είναι το 'BITMAPINFO' το οποίο ορίζεται ως εξής:

```
typedef struct tagBITMAPINFO {
    BITMAPINFOHEADER    bmiHeader;
    RGBQUAD             bmiColors[1];
} BITMAPINFO;
```

Παρατηρούμε ότι αποτελείται από:

- Ένα header, το BITMAPINFOHEADER, που περιέχει πληροφορίες σχετικά με την ανάλυση της εικόνας, τις διαστάσεις της, τον τύπο της συμπίεσης που πιθανόν να υπάρχει κλπ.
- Μια παλέτα χρωμάτων από τα οποία συντίθεται η εικόνα.

Το BITMAPINFOHEADER ορίζεται ως εξής:

```
typedef struct tagBITMAPINFOHEADER{
    DWORD  biSize;
    LONG   biWidth;
    LONG   biHeight;
    WORD   biPlanes;
    WORD   biBitCount
    DWORD  biCompression;
    DWORD  biSizeImage;
    LONG   biXPelsPerMeter;
    LONG   biYPelsPerMeter;
    DWORD  biClrUsed;
    DWORD  biClrImportant;
} BITMAPINFOHEADER;
```

Τα πεδία που το αποτελούν είναι:
biSize: Το μέγεθος του struct.
biWidth: Το πλάτος του Bitmap.
biHeight: Το ύψος του.
biplanes: Ο αριθμός των πλάνων. Πρέπει να είναι ίσος με 1.
biBitCount : Ο αριθμός των Bits ανά εικονοστοιχείο.
biCompression: Υποδηλώνει το είδος της συμπίεσης. Έχει τιμή BI_RGB για ασυμπίεστες εικόνες.
biSizeImage: Το μέγεθος της εικόνας σε Bytes. Για ασυμπίεστες εικόνες το πεδίο αυτό έχει τιμή 0.
biXPelsPerMeter: Οριζόντια ανάλυση της εικόνας.
biYPelsPerMeter :Κατακόρυφη ανάλυση της εικόνας.
biClrUsed: Έχει τιμή 0. Χρησιμοποιείται για την παλέτα χρωμάτων.
biClrImportant : Έχει τιμή 0. Χρησιμοποιείται για την παλέτα χρωμάτων.

Η εικόνα μπορεί να είναι έγχρωμη ή ασπρόμαυρη. Σε κάθε εικονοστοιχείο της, αντιστοιχούμε έναν αριθμό από bits για να περιγράψουμε το χρώμα του. Οι μονόχρωμες εικόνες για παράδειγμα έχουν ακριβώς 1 bit ανά εικονοστοιχείο (Bit per pixel - bpp) ενώ μια εικόνα 256 χρωμάτων (ή αποχρώσεων του γκρι) έχει 8bpp.

Η παλέτα χρωμάτων είναι στην ουσία ένας πίνακας. Κάθε στοιχείο του πίνακα είναι ένα struct τύπου RGBQUAD.

```
typedef struct tagRGBQUAD {
    BYTE    rgbBlue;
    BYTE    rgbGreen;
    BYTE    rgbRed;
    BYTE    rgbReserved;
} RGBQUAD;
```

Η δομή αυτή περιγράφει ένα συγκεκριμένο χρώμα στο χρωματικό μοντέλο RGB το οποίο χρησιμοποιείται στην εικόνα. Περιέχει δηλαδή την ένταση του κάθε χρωματικού καναλιού Κόκκινο - Πράσινο - Μπλε

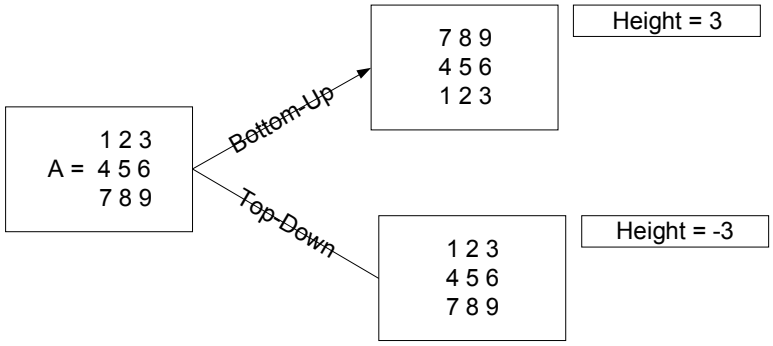
Εμείς θα ασχοληθούμε με έγχρωμα Bitmap 32bpp. Σε αυτή την περίπτωση δεν χρησιμοποιείται παλέτα χρωμάτων αφού όλα τα χρώματα χρησιμοποιούνται και όχι ένα υποσύνολο από αυτά.

1.3 Δομή των δεδομένων

Τα δεδομένα της εικόνας αποθηκεύονται με δυο τρόπους ανάλογα με το που θεωρούμε την «αρχή» της. Έτσι έχουμε:

- Τα bottom-up DIB, στα οποία η αρχή είναι κάτω αριστερά
- Τα top-down DIB, στα οποία η αρχή είναι πάνω αριστερά

Οι δυο διαφορετικοί τρόποι αναπαράστασης διακρίνονται μεταξύ τους ανάλογα με το αν θεωρήσουμε το ύψος της εικόνας θετικό ή αρνητικό. Αν είναι θετικό τότε έχουμε την bottom-up μορφή που είναι και η προκαθορισμένη, ενώ αν είναι αρνητικό έχουμε την top-down. Για παράδειγμα στο παρακάτω σχήμα θεωρούμε την εικόνα Α υπό μορφή πίνακα και παρατηρούμε τις δυο δυνατές αναπαραστάσεις των δεδομένων της:



Σχήμα 37. Τρόποι αποθήκευσης δεδομένων στο DIB.

1.4 Στοίχιση Δεδομένων

Τα δεδομένα των BMP είναι αποθηκευμένα με στοίχιση 4Bytes (DWORD). Αυτό σημαίνει ότι κάθε γραμμή της εικόνας όταν αποθηκεύεται πρέπει να αποτελείται από L bytes τέτοια ώστε:

$$L \bmod 4 = 0$$

Αν τα bytes της γραμμής δεν ικανοποιούν αυτή τη συνθήκη τότε συμπληρώνονται με κατάλληλο αριθμό μηδενικών bytes.

Για παράδειγμα έστω ότι μια εικόνα RGB έχει πλάτος 10 εικονοστοιχεία. Με άλλα λόγια κάθε γραμμή του αποτελείται από 10*3=30Bytes. Το 30 όμως δεν είναι πολλαπλάσιο του 4. Έτσι στο αρχείο θα είναι γραμμένα τα 30Bytes της γραμμής της εικόνας και στην συνέχεια ακόμα 2 με μηδενική τιμή. Έτσι έχουμε συνολικό μήκος γραμμής 32Bytes και ικανοποιούμε την συνθήκη στοίχισης.

Τα επιπλέον μηδενικά που βάζουμε προφανώς δεν λαμβάνονται υπ' όψιν κατά την απεικόνιση της εικόνας. Ο λόγος που μπαίνουν είναι για λόγους ταχύτητας ανάγνωσης του αρχείου.

Στην περίπτωση των έγχρωμων εικόνων 24bit για κάθε εικονοστοιχείο αποθηκεύονται 3Bytes, ένα για κάθε κανάλι. Η σειρά αποθήκευσης τους είναι Μπλε – Πράσινο – Κόκκινο (B-G-R) και όχι R-G-B!

1.5 Αποθήκευση της εικόνας στο δίσκο

Όταν αποθηκεύουμε ένα bitmap στο δίσκο, δηλαδή δημιουργούμε ένα αρχείο ".BMP" χρειάζεται να αποθηκεύσουμε κάποιες επιπλέον πληροφορίες σχετικά με την εικόνα μας. Αυτές αποτελούν την δομή 'BITMAPFILEHEADER':

```
typedef struct tagBITMAPFILEHEADER{
    WORD    bfType;
    DWORD   bfSize;
    WORD    bfReserved1;
    WORD    bfReserved2;
    DWORD   bfOffBits;
} BITMAPFILEHEADER;
```

Τα πεδία αυτής της δομής αναλυτικά είναι:

bfType: Σταθερά. Έχει την τιμή 'BM'.
bfSize: Το μέγεθος όλου του αρχείου σε Bytes.
bfReserved1: Σταθερά. Έχει την τιμή 0.
bfReserved2: Σταθερά. Έχει την τιμή 0.
bfOffBits: Η απόσταση σε bytes από το BITMAPINFOHEADER μέχρι την αρχή των εικονοστοιχείων της εικόνας.

Έτσι λοιπόν η πραγματική δομή ενός αρχείου BMP στο δίσκο έχει την μορφή:

BITMAPFILEHEADER
BITMAPINFO
BITMAPINFOHEADER
COLORMAP (Προαιρετικό)
BITMAP DATA

Σχήμα 38. Δομή αρχείου BMP.

2 Επεξεργασία αρχείων AVI με το WinAPI

2.1 Αντιμετώπιση των AVI από το WinAPI

Το API των Windows προσφέρει μια σειρά από συναρτήσεις και μακροεντολές στον προγραμματιστή ειδικά για τον χειρισμό αρχείων τύπου RIFF στα οποία περιλαμβάνονται και τα AVI. Αποτελούν την βιβλιοθήκη συναρτήσεων 'AVIFile'. Μας βοηθούν στη δημιουργία, αναπαραγωγή, συμπίεση και αποσυμπίεση και γενικότερα την επεξεργασία ενός βίντεο χωρίς να απαιτούν από τον προγραμματιστή

την γνώση της αρχιτεκτονικής των αρχείων RIFF. Συμπεριφέρονται στα αρχεία σαν ένα σύνολο από ροές δεδομένων (Data Streams). Για παράδειγμα μια τέτοια ροή μπορεί να είναι η κινούμενη εικόνα και άλλες τρεις να είναι τρεις διαφορετικές εκδοχές του ήχου που την συνοδεύει, για τρεις διαφορετικές γλώσσες.

Οι συναρτήσεις AVIFile είναι οργανωμένες σε μια βιβλιοθήκη δυναμικής διασύνδεσης. Για να τις χρησιμοποιήσουμε πρέπει προηγουμένως να αρχικοποιήσουμε την βιβλιοθήκη με την συνάρτηση AVIFileInit.

```
STDAPI_(VOID) AVIFileInit(VOID);
```

Αφού την χρησιμοποιήσουμε θα πρέπει να την αποδεσμεύσουμε καλώντας την συνάρτηση AVIFileExit.

```
STDAPI_(VOID) AVIFileExit(VOID);
```

Η δυο αυτές συναρτήσεις δεν παίρνουν καμία παράμετρο και δεν επιστρέφουν καμία τιμή.

2.2 Άνοιγμα και κλείσιμο Αρχείων AVI

Πριν την ανάγνωση ή την εγγραφή ενός αρχείου AVI η εφαρμογή πρέπει προηγουμένως να το ανοίξει όπως γίνεται και με τα συνήθη αρχεία δεδομένων. Το άνοιγμα του αρχείου γίνεται με την συνάρτηση AVIFileOpen:

```
STDAPI AVIFileOpen(
  PAVIFILE * ppfile,
  LPCTSTR szFile,
  UINT mode,
  CLSID * pclsidHandler
);
```

Η πρώτη παράμετρος της συνάρτησης είναι ο δείκτης αρχείου που επιστρέφει η συνάρτηση και χρησιμοποιείται σε επόμενες αναγνώσεις ή εγγραφές του αρχείου. Είναι σε πλήρη αναλογία με τον τύπο δεδομένων FILE* της C). Η δεύτερη παράμετρος είναι το όνομα του αρχείου. Στη παράμετρο mode, για άνοιγμα υπάρχοντος αρχείου, δίνουμε την τιμή OF_READ. Η τελευταία παράμετρος `pclsidHandler` έχει τιμή NULL. Χρησιμοποιείται μόνο αν το όνομα του αρχείου που προσπελαύνουμε δεν έχει μια γνωστή και προκαθορισμένη κατάληξη π.χ. '.AVI'.

Κάθε αρχείο που ανοίγεται με την AVIFileOpen πρέπει μετά την επεξεργασία του να κλείνει. Το κλείσιμο των αρχείων AVI γίνεται με την συνάρτηση AVIFileRelease.

```
STDAPI_(ULONG) AVIFileRelease(
   PAVIFILE pfile
);
```

Η μοναδική παράμετρος που παίρνει αυτή η συνάρτηση είναι ο δείκτης αρχείου που επέστρεψε η AVIFileOpen κατά το άνοιγμα του αρχείου.

2.3 Γενικές Πληροφορίες Αρχείου AVI

Μετά το άνοιγμα ενός αρχείου AVI είναι χρήσιμο να καλέσουμε τη συνάρτηση AVIFileInfo προκειμένου να πάρουμε διάφορες γενικές πληροφορίες για αυτό όπως το πλήθος των ροών δεδομένων του, οι διαστάσεις του AVI σε εικονοστοιχεία κλπ. Η συνάρτηση αυτή έχει το εξής πρότυπο:

```
STDAPI AVIFileInfo(
  PAVIFILE pfile,
  AVIFILEINFO * pfi,
  LONG lSize
);
```

Η πρώτη παράμετρος είναι ο δείκτης αρχείου για το οποίο θέλουμε να ανακτήσουμε πληροφορίες. Η δεύτερη είναι ένας δείκτης σε δομή τύπου AVIFILEINFO που θα πρέπει να έχουμε ορίσει. Η συνάρτηση AVIFileInfo θα συμπληρώσει τα πεδία της δομής με τις τιμές που ανταποκρίνονται στα χαρακτηριστικά του AVI. Τέλος στη παράμετρος lSize δίνουμε το μέγεθος της δομής AVIFILEINFO (Αυτό γίνεται για λόγους προς τα πίσω συμβατότητας). Η δομή με τις γενικές πληροφορίες του AVI είναι η εξής:

```
typedef struct {
    DWORD dwMaxBytesPerSec;
    DWORD dwFlags;
    DWORD dwCaps;
    DWORD dwStreams;
    DWORD dwSuggestedBufferSize;
    DWORD dwWidth;
    DWORD dwHeight;
    DWORD dwScale;
    DWORD dwRate;
    DWORD dwLength;
    DWORD dwEditCount;
    char  szFileType[64];
} AVIFILEINFO;
```

Αναλυτικά τα πεδία της δομής είναι:
dwMaxBytesPerSec: Ο ρυθμός δεδομένων του βίντεο.
dwFlags: Σημαίες με ειδικές ιδιότητες του βίντεο. Βλέπε το [11] για περισσότερες πληροφορίες.
dwCaps: Επιπλέον σημαίες με ιδιότητες του βίντεο. Βλέπε το [11] για περισσότερες πληροφορίες.
dwStreams: Ο αριθμός των ροών δεδομένων.
dwSuggestedBufferSize: Προτεινόμενο μέγεθος προσωρινής μνήμης για τμηματική ανάγνωση του βίντεο.
dwWidth: Πλάτος σε εικονοστοιχείας.
dwHeight: Ύψος σε εικονοστοιχείας.
dwScale: Χρονική κλίμακα του βίντεο. Διαιρώντας το dwRate με αυτό το πεδίο έχουμε τον αριθμό των δειγμάτων του βίντεο ανά δευτερόλεπτο.
dwRate: Βλέπε πεδίο dwScale.
dwLength: Το μήκος του βίντεο. Οι μονάδες εξαρτώνται από τα πεδία dwRate και dwScale.
dwEditCount: Αριθμός των ροών που έχουν προστεθεί ή αφαιρεθεί από το βίντεο.
szFileType: Συμβολοσειρά που περιγράφει τον τύπο του αρχείου.

2.4 Άνοιγμα και κλείσιμο Ροών Δεδομένων

Όπως έχουμε αναφέρει τα AVI αποτελούνται εσωτερικά από ροές δεδομένων. Εδώ θα δείξουμε πως μπορούμε να έχουμε πρόσβαση στις ροές αυτές. Η διαδικασία που ακολουθείται είναι παρόμοια με αυτή των αρχείων. Οι ροές πρέπει να ανοίγονται πριν χρησιμοποιηθούν και στο τέλος της επεξεργασίας τους να κλείνουν. Το άνοιγμα μιας ροής γίνεται με τη συνάρτηση AVIFileGetStream:

```
STDAPI AVIFileGetStream(
  PAVIFILE pfile,
  PAVISTREAM * ppavi,
  DWORD fccType,
  LONG lParam
);
```

Η πρώτη παράμετρος είναι ο δείκτης αρχείου στο οποίου θέλουμε να ανοίξουμε μια ροή. Είναι ο δείκτης που επιστρέφει η συνάρτηση AVIFileOpen με την οποία θα πρέπει προηγουμένως να έχουμε ανοίξει το αρχείο. Η παράμετρος ppavi είναι ο δείκτης σε μια ροή δεδομένων και είναι αυτό που μας επιστρέφει η συνάρτηση. Το fccType καθορίζει τον τύπο της ροής που θέλουμε να ανοίξουμε. Για ροή κινούμενης εικόνας το πεδίο αυτό έχει τιμή 'streamtypeVIDEO' ενώ αν θέλουμε μια ροή ήχου η τιμή του είναι 'streamtypeAUDIO'. Το lparam δηλώνει τον αριθμό της ροής τύπου fccType που θέλουμε να ανοίξουμε. Συνήθως στα AVI έχόυμε μόνο μια ροή βίντεο οπότε για να την ανοίξουμε το πεδίο αυτό παίρνει τιμή 0.

Το κλείσιμο των ανοιγμένων ροών δεδομένων γίνεται με τη συνάρτηση AVIStreamRelease:

```
STDAPI_(LONG) AVIStreamRelease(
  PAVISTREAM pavi
);
```

Η μοναδική παράμετρος που παίρνει η συνάρτηση αυτή είναι ο δείκτης ροής που θέλουμε να κλείσουμε. Είναι η τιμή που μας έχει επιστρέψει η συνάρτηση AVIFileGetStream.

Αν θέλουμε να έχουμε πρόσβαση σε παραπάνω από μια ροές δεδομένων ενός αρχείου AVI θα πρέπει να κάνουμε μια μοναδική κλήση στη συνάρτηση AVIFileOpen για να ανοίξουμε το αρχείο και πολλαπλές κλήσεις της AVIFileGetStream, μια για κάθε ροή που θέλουμε να ανοίξουμε.

2.5 Πληροφορίες Ροής Δεδομένων

Αφού ανοίξουμε μια ροή δεδομένων μπορούμε να πάρουμε διάφορες πληροφορίες σχετικά με αυτήν με την συνάρτηση AVIStreamInfo:

```
STDAPI AVIStreamInfo(
      PAVISTREAM pavi,
      AVISTREAMINFO * psi,
      LONG lSize );
```

Η πρώτη παράμετρος είναι ο δείκτης στη ροή δεδομένων για την οποία θέλουμε να πάρουμε πληροφορίες. Η δεύτερη είναι ένας δείκτης σε δομή τύπου

AVISTREAMINFO την οποία θα πρέπει να έχουμε ορίσει και lSize είναι το μέγεθος της. Τα πεδία της δομής μετά την κλήση της συνάρτησης θα ανανεωθούν ώστε οι τιμές τους να αντιπροσωπεύουν τη ροή pavi. Η δομή AVISTREAMINFO είναι η εξής:

```
typedef struct {
      DWORD fccType;
      DWORD fccHandler;
      DWORD dwFlags;
      DWORD dwCaps;
      WORD wPriority;
      WORD wLanguage;
      DWORD dwScale;
      DWORD dwRate;
      DWORD dwStart;
      DWORD dwLength;
      DWORD dwInitialFrames;
      DWORD dwSuggestedBufferSize;
      DWORD dwQuality;
      DWORD dwSampleSize;
      RECT rcFrame;
      DWORD dwEditCount;
      DWORD dwFormatChangeCount;
      char szName[64]; } AVISTREAMINFO;
```

Αναλυτικά τα πεδία της δομής είναι τα εξής:

fccType: Ένας κωδικός τεσσάρων Bytes που υποδηλώνει το τύπο της ροής. Για βίντεο έχει τιμή streamtypeVideo ενώ για ήχο streamtypeAudio.
fccHandler: Ένας κωδικός τεσσάρων Bytes που υποδηλώνει το τύπο του συμπιεστή που χρησιμοποιήθηκε για την συμπίεση της δομής (0 για ασυμπίεστη ροή δεδομένων).
dwFlags: Σημαίες σχετικές με τη ροή.
dwCaps: Δυνατότητες της ροής.
wPriority: Προτεραιότητα της ροής.
wLanguage: Η γλώσσα της ροής.
dwScale: Διαιρώντας την τιμή του πεδίου dwRate με αυτό παίρνουμε τον αριθμό των δειγμάτων ανά δευτερόλεπτο. Αν έχουμε ροή βίντεο ο αριθμός αυτός είναι ο αριθμός των καρέ ανά δευτερόλεπτο.
dwRate: Βλέπε το πεδίο dwScale.
dwStart: Αριθμός του πρώτου καρέ. Συνήθως είναι το 0.
dwLength: Το μήκος της ροής, δηλαδή ο αριθμός των καρέ αν είναι βίντεο.
dwInitialFrames: Αριθμός αρχικών καρέ που πρέπει να προηγηθεί η ροή του βίντεο από αυτή του ήχου.
dwSuggestedBufferSize: Προτεινόμενο μέγεθος προσωρινής μνήμης για τμηματική ανάγνωση του βίντεο (Μπορεί να έχει τιμή 0 οπότε δεν προτείνεται κάποιο μέγεθος).
dwQuality: Ποιότητα της ροής. Είναι ένα νούμερο μεταξύ του 0 και του 10000.
dwSampleSize: Μέγεθος του κάθε καρέ σε bytes. Αν είναι 0 τότε τα δείγματα μπορούν να μεταβάλλονται σε μέγεθος.
rcFrame: Διαστάσεις του βίντεο.
dwEditCount: Ο αριθμός των φορών που έχει επεξεργασθεί η ροή.
dwFormatChangeCount: Ο αριθμός των φορών που έχει αλλαχτεί η μορφή του βίντεο.
szName: Συμβολοσειρά με την περιγραφή της ροής.

2.6 Διάβασμα καρέ

Η ανάγνωση ενός καρέ από μια ροή βίντεο γίνεται αρκετά εύκολα με τη χρήση τριών συναρτήσεων που μας παρέχει το WinAPI. Αυτές αναγνωρίζουν αυτόματα την πιθανή συμπίεση που έχει υποστεί το βίντεο και το αποσυμπιέζουν χωρίς περαιτέρω παρέμβαση του χρήστη. Τα καρέ που επιστρέφουν είναι τύπου DIB. Τα χαρακτηριστικά του καθορίζονται από τον προγραμματιστή.

Αρχικά καλούμε μια φορά την συνάρτηση AVIStreamGetFrameOpen:

```
STDAPI_(PGETFRAME) AVIStreamGetFrameOpen(
      PAVISTREAM pavi,
      LPBITMAPINFOHEADER lpbiWanted );
```

Η πρώτη παράμετρος που παίρνει αυτή η συνάρτηση είναι ο δείκτης στη ροή δεδομένων που θέλουμε να διαβάσουμε. Η δεύτερη παράμετρος είναι ένας δείκτης σε δομή BITMAPINFOHEADER η οποία περιέχει τις επιθυμητές ιδιότητες που θέλουμε να έχουν τα καρέ που διαβάζουμε (Μέγεθος, συμπίεση, βάθος χρώματος κλπ.). Αυτές δεν είναι απαραίτητο να είναι οι ίδιες με τις ιδιότητες των καρέ όπως βρίσκονται στο βίντεο αλλά μπορούμε να τις αλλάζουμε σύμφωνα με τις απαιτήσεις μας. Η συνάρτηση επιστρέφει έναν δείκτη σε αντικείμενο τύπου PGETFRAME που χρησιμοποιείται στις υπόλοιπες συναρτήσεις της παραγράφου.

Στη συνέχεια καλούμε μια φορά την συνάρτηση AVIStreamGetFrame για κάθε καρέ που θέλουμε να διαβάσουμε.

```
STDAPI_(LPVOID) AVIStreamGetFrame(
      PGETFRAME pgf,
      LONG lPos );
```

Η πρώτη παράμετρος που παίρνει είναι ο δείκτης που μας έχει επιστρέψει η συνάρτηση AVIStreamGetFrameOpen. Η δεύτερη είναι ο αριθμός του καρέ που θέλουμε να διαβάσουμε. Το αρχικό καρέ είναι συνήθως το 0. Αν είναι διαφορετικό μπορούμε να το βρούμε από τις πληροφορίες που παίρνουμε από την συνάρτηση AVIStreamInfo. Η συνάρτηση μας επιστρέφει έναν δείκτη σε void που δείχνει την αρχή μιας προσωρινής μνήμης που θα αποθηκευτεί το καρέ που διαβάζουμε. Κάνοντας αυτό το δείκτη αυτό μετατροπή σε BITMAPINFOHEADER* έχουμε το καρέ σε μορφή DIB σύμφωνα με τις απαιτήσεις που είχαμε θέσει. Η προσωρινή μνήμη που χρησιμοποιείται είναι η ίδια για κάθε κλήση της συνάρτησης AVIStreamGetFrame και κάθε φορά έχει αποθηκευμένο το αποτέλεσμα από την τελευταία κλήση της συνάρτησης.

Αφού διαβάσουμε τα δεδομένα που θέλουμε από ένα βίντεο θα πρέπει να καλέσουμε την συνάρτηση AVIStreamGetFrameClose η οποία θα αποδεσμεύσει την προσωρινή που έχει χρησιμοποιηθεί.

```
STDAPI AVIStreamGetFrameClose(
      PGETFRAME pget );
```

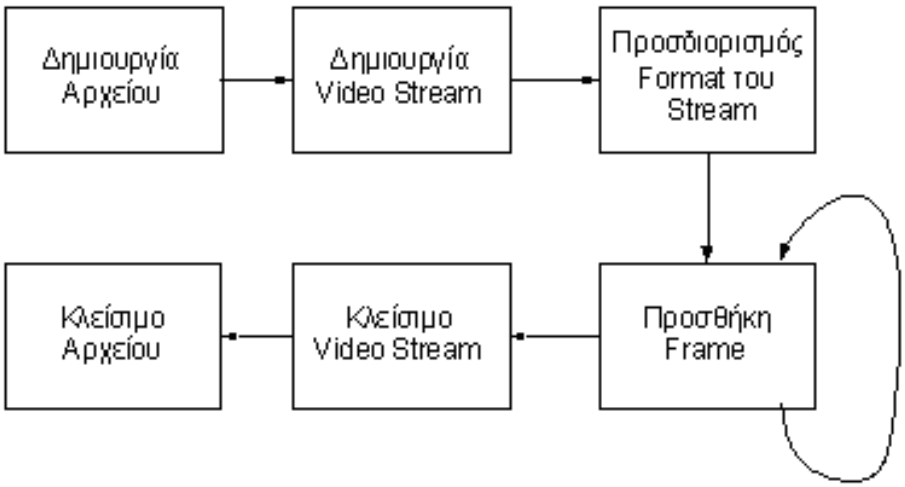
Η μοναδική παράμετρος της συνάρτησης είναι ο δείκτης στο αντικείμενο PGETFRAME που μας επέστρεψε η συνάρτηση AVIStreamGetFrameOpen.

2.7 Δημιουργία και Αποθήκευση ασυμπίεστου AVI

Σε αυτή τη παράγραφο θα δείξουμε πως μπορούμε με τη χρήση του API να δημιουργήσουμε ένα καινούριο αρχείο βίντεο AVI. Για να γίνει πιο κατανοητή η διαδικασία θα αναλύσουμε ως παράδειγμα την δημιουργία του βίντεο από μια σειρά από εικόνες DIB.

Μια από τις παραμέτρους που πρέπει να ορίσουμε κατά την διαδικασία δημιουργίας νέου βίντεο είναι το αν θα χρησιμοποιήσουμε συμπιεστή (Codec) και αν ναι, ποιον και με τι χαρακτηριστικά. Η βασική μεθοδολογία που ακολουθούμε είναι η ίδια και στις δυο περιπτώσεις με λίγα παραπάνω βήματα αν θέλουμε συμπίεση. Αρχικά θα μελετήσουμε την περίπτωση του ασυμπίεστου βίντεο που είναι και η απλούστερη.

Τα βήματα που θα ακολουθήσουμε φαίνονται στο σχήμα 39.



Σχήμα 39. Διαδικασία δημιουργίας ασυμπίεστου αρχείου AVI.

Το πρώτο βήμα που κάνουμε είναι να καλέσουμε τη συνάρτηση AVIFileOpen (Βλέπε παρ. 2.2). Η συνάρτηση αυτή έχει σκοπό να δημιουργήσει και να ανοίξει προς περαιτέρω επεξεργασία ένα νέο αρχείο βίντεο με όνομα που δίνουμε στη παράμετρο szFile. Η προέκταση του αρχείου πρέπει να είναι ".AVI". Για mode δίνουμε τιμή OF_CREATE | OF_WRITE αφού θέλουμε να δημιουργήσουμε αρχείο και να γράψουμε δεδομένα σε αυτό. Την τελευταία παράμετρο την αφήνουμε με τιμή NULL.

Στη συνέχεια θα δημιουργήσουμε ένα νέο stream τύπου βίντεο μέσα στο αρχείο που δημιουργήσαμε με την συνάρτηση AVIFileCreateStream:

```
STDAPI STDAPI AVIFileCreateStream(
  PAVIFILE pfile,
  PAVISTREAM * ppavi,
  AVISTREAMINFO * psi
);
```

Η πρώτη παράμετρος pfile είναι ο περιγραφέας αρχείου που μας έχει επιστρέψει η AVIFileOpen. Το ppavi είναι ο περιγραφέας stream που δημιουργεί η συνάρτηση. Τέλος το psi είναι δείκτης σε ένα struct τύπου AVISTREAMINFO (Βλέπε παρ.2.5).

Το struct αυτό περιέχει τα χαρακτηριστικά του stream που θέλουμε να δημιουργήσουμε.

Αφού δημιουργήσουμε το stream πρέπει να ορίσουμε την μορφή με την οποία δέχεται τα δεδομένα. Η μορφή αυτή είναι DIB. Έτσι λοιπόν πρέπει να του περάσουμε έναν δείκτη σε δομή τύπου BITMAPINFOHEADER με τα χαρακτηριστικά του κάθε καρέ του βίντεο. Αυτό γίνεται με την συνάρτηση AVIStreamSetFormat:

```
STDAPI AVIStreamSetFormat(
      PAVISTREAM pavi,
      LONG lPos,
      LPVOID lpFormat,
      LONG cbFormat );
```

Η πρώτη παράμετρος που δέχεται η συνάρτηση είναι ο δείκτης στη ροή δεδομένων που δημιουργήσαμε. Η παράμετρος lPos πρέπει να έχει τιμή 0. Ο δείκτης lpFormat δείχνει στη δομή DIB και η παράμετρος cbFormat πρέπει να έχει τιμή ίση με το μέγεθος της δομής.

Στη συνέχεια προσθέτουμε σειριακά τα καρέ του βίντεο καλώντας την συνάρτηση AVIStreamWrite μια φορά για κάθε καρέ:

```
STDAPI AVIStreamWrite(
      PAVISTREAM pavi,
      LONG lStart,
      LONG lSamples,
      LPVOID lpBuffer,
      LONG cbBuffer,
      DWORD dwFlags,
      LONG * plSampWritten,
      LONG * plBytesWritten );
```

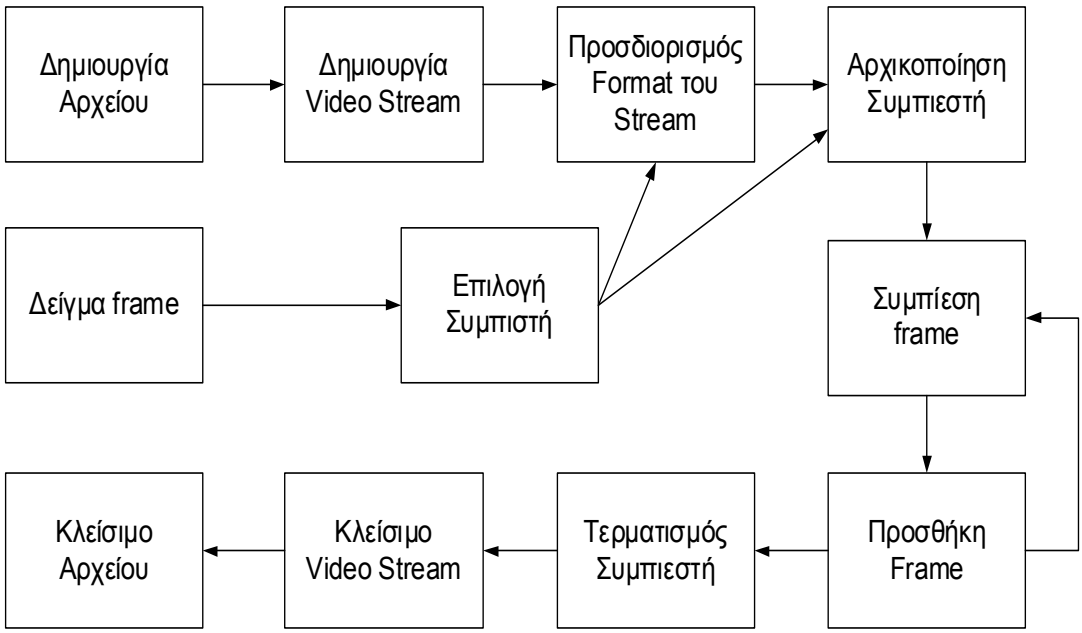
Η πρώτη παράμετρος είναι η ροή στην οποία θα γίνει η εγγραφή. Η δεύτερη είναι ο αριθμός του καρέ που προσθέτουμε. Στην παράμετρο lSamples δίνουμε τιμή ίση με 1 αφού γράφουμε ένα καρέ κάθε φορά που καλούμε την συνάρτηση. Ο lpBuffer είναι ο δείκτης στο DIB που θέλουμε να προσθέσουμε στο βίντεο και cbBuffer είναι το μέγεθος του DIB (Επικεφαλίδα και δεδομένα). Στην παράμετρο dwFlags δίνουμε τιμή AVIIF_KEYFRAME. Οι τελευταίες δυο παράμετροι μας επιστρέφουν το πλήθος των εγγραφών που έγιναν και τον αριθμό των bytes που γράφτηκαν. Αν δεν θέλουμε να χρησιμοποιήσουμε αυτές τις δυο παραμέτρους τους δίνουμε τιμή NULL.

Αφού προσθέσουμε όλα τα καρέ στο βίντεο πρέπει να κλείσουμε τη ροή δεδομένων που δημιουργήσαμε με την συνάρτηση AVIStreamRelease και στη συνέχεια και το αρχείο με την συνάρτηση AVIFileRelease (Βλέπε παρ. 2.4 και 2.2)

2.8 Δημιουργία και Αποθήκευση συμπιεσμένου AVI

Στην περίπτωση που θέλουμε να δημιουργήσουμε ένα συμπιεσμένο βίντεο ακολουθούμε μια διαφορετική προσέγγιση σε σχέση με το ασυμπίεστο. Απαιτούνται μια σειρά από παραπάνω βήματα όπως η επιλογή του συμπιεστή και των χαρακτηριστικών του, η συμπίεση του κάθε καρέ κλπ.

Στο σχήμα που ακολουθεί φαίνεται αναλυτικά η διαδικασία της δημιουργίας συμπιεσμένου βίντεο από εικόνες



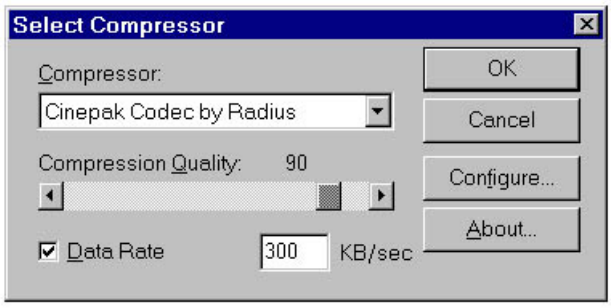
Σχήμα 40. Διαδικασία δημιουργίας συμπιεσμένου αρχείου AVI.

Αρχικά πρέπει να γίνει η επιλογή του συμπιεστή (Codec). Ο συμπιεστής λαμβάνει εικόνες ως είσοδο και τις βγάζει στην έξοδο του συμπιεσμένες σύμφωνα με τις ρυθμίσεις (ως προς το λόγο συμπίεσης, την επιθυμητή ταχύτητα αποσυμπίεσης, το μέγεθος της εξόδου κλπ) που του έχουμε κάνει. Από όλους τους συμπιεστές που μπορούν να βρίσκονται σε ένα σύστημα δεν μπορούμε να επιλέξουμε πάντα οποιονδήποτε επιθυμούμε. Αυτό συμβαίνει γιατί η είσοδος που δέχεται κάθε ένας είναι με συγκεκριμένες προδιαγραφές και αν δεν πληρούνται η συμπίεση θα αποτύχει. Επίσης είναι σύνηθες η επιλογή του συμπιεστή για ένα βίντεο να μην είναι επιλογή του προγραμματιστή μιας εφαρμογής αλλά του τελικού χρήστη της.

Στο WinAPI υπάρχει μια συνάρτηση με την οποία δίνεται η επιλογή στον τελικό χρήστη μιας εφαρμογής του συμπιεστή που επιθυμεί να χρησιμοποιήσει καθώς και η πλήρης ρύθμιση του. Η συνάρτηση αυτή είναι η ICCompressorChoose:

```
BOOL ICCompressorChoose(
  HWND hwnd,
  UINT uiFlags,
  LPVOID pvIn,
  LPVOID lpData,
  PCOMPVARS pc,
  LPSTR lpszTitle
);
```

Μόλις γίνει κλήση αυτής της συνάρτησης δημιουργείται αυτόματα από το WinAPI ένα καινούργιο παράθυρο επιλογής και ρύθμισης συμπιεστή:



Σχήμα 41. Παράθυρο επιλογής και ρύθμισης συμπιεστή της ICCompressorChoose.

Η πρώτη παράμετρος hWnd της συνάρτησης είναι το handle του παράθυρου που δημιουργεί το νέο παράθυρο. Μπορεί να έχει και τιμή NULL. Με την παράμετρο uiFlags μπορούμε να θέσουμε κάποιους περιορισμούς στις επιλογές του χρήστη. Συνήθης τιμή αυτού του πεδίου είναι η "ICMF_CHOOSE_DATARATE" και η "CMF_CHOOSE_KEYFRAME " έτσι ώστε να μπορεί να γίνει επιλογή του ρυθμού της ροής των δεδομένων του βίντεο και η επιλογή της συχνότητας των Keyframes αντίστοιχα. Στην παράμετρο pvIn πρέπει να περάσουμε ένα οποιοδήποτε καρέ (δείκτης σε BITMAPINFOHEADER) από τις εικόνες που θα αποτελέσουν το τελικό βίντεο, έτσι ώστε να γίνει η επιλογή των συμπιεστών που μπορούν να τα χειριστούν. Μόνο αυτοί οι συμπιεστές που είναι ικανοί να τις χειριστούν θα εμφανιστούν στη λίστα επιλογής του παραθύρου. Το πεδίο lpData το αφήνουμε NULL και στο πεδίο lpszTitle περνάμε το δείκτη στη συμβολοσειρά που θα αποτελέσει το τίτλο του παραθύρου.

Η παράμετρος 'pc' είναι δείκτης σε μια δομή τύπου PCOMPVARS, που θα πρέπει να έχει οριστεί, και αποτελεί την έξοδο της συνάρτησης ICCompressorChoose. Σε αυτή τη δομή θα αποθηκευτούν όλες οι επιλογές του τελικού χρήστη που θα χρησιμοποιηθούν για την συμπίεση του βίντεο. Η δομή PCOMVARS έχει ως εξής:

```
typedef struct {
    LONG          cbSize;
    DWORD         dwFlags;
    HIC           hic;
    DWORD         fccType;
    DWORD         fccHandler;
    LPBITMAPINFO lpbiIn;
    LPBITMAPINFO lpbiOut;
    LPVOID        lpBitsOut;
    LPVOID        lpBitsPrev;
    LONG          lFrame;
    LONG          lKey;
    LONG          lDataRate;
    LONG          lQ;
    LONG          lKeyCount;
    LPVOID        lpState;
    LONG          cbState;
} COMPVARS;
```

Πριν καλέσουμε την συνάρτηση το μοναδικό πεδίο της δομής που θα πρέπει να έχουμε αρχικοποιήσει είναι το ΄cbSize' με τιμή το μέγεθος της δομής. Θα πρέπει εδώ να διευκρινιστεί ότι με την κλήση της ICCompressorChoose το μόνο που πραγματικά

γίνεται είναι η συμπλήρωση της δομής που θα μας βοηθήσει στην περαιτέρω διαδικασία.

Αφού λοιπόν γίνει η επιλογή του συμπιεστή δημιουργούμε ένα νέο αρχείο AVI και μια νέα ροή δεδομένων βίντεο μέσα σε αυτό (όπως ακριβώς περιγράφτηκε στην προηγούμενη παράγραφο). Στην επιλογή της μορφής των δεδομένων που δέχεται η ροή χρησιμοποιούμε και πάλι την συνάρτηση AVIStreamSetFormat αλλά την παράμετρο της lpFormat την κάνουμε να ίση με το πεδίο lpbiOut της δομής COMPVARS που μας συμπλήρωσε η συνάρτηση ICCompressorChoose.

Στην συνέχεια πρέπει να αρχικοποιήσουμε τον συμπιεστή που έχει επιλέξει ο χρήστης. Αυτό γίνεται καλώντας την συνάρτηση ICSeqCompressFrameStart:

```
BOOL ICSeqCompressFrameStart(
  PCOMPVARS pc,
  LPBITMAPINFO lpbiIn
);
```

Η πρώτη της παράμετρος είναι ένας δείκτης στη δομή COMPVARS και η δεύτερη ένας δείκτης σε ένα από τα ασυμπίεστα καρέ τύπου DIB που θέλουμε να συμπεριλάβουμε στο βίντεο (Χρησιμοποιείται μόνο για λόγους αρχικοποίησης και δεν συμπιέζεται).

Αφού αρχικοποιήσουμε τον συμπιεστή ξεκινάμε την διαδικασία συμπίεσης του κάθε καρέ ξεχωριστά και της εγγραφής της συμπιεσμένης μορφής του στη ροή δεδομένων. Η συμπίεση γίνεται με την συνάρτηση ICSeqCompressFrame:

```
LPVOID ICSeqCompressFrame(
  PCOMPVARS pc,
  UINT uiFlags,
  LPVOID lpBits,
  BOOL * pfKey,
  LONG * plSize
);
```

Η πρώτη παράμετρος είναι και πάλι ο δείκτης στη δομή COMPVARS. Στη δεύτερη βάζουμε τιμή 0. Ο δείκτης lpBits δείχνει στα δεδομένα του DIB που θέλουμε να συμπιέσουμε (Δηλαδή όχι στην αρχή της επικεφαλίδας BITMAPINFOHEADER αλλά αμέσως μετά). Το *pfKey παίρνει τιμή true αν το καρέ που συμπιέστηκε ήταν Key frame (Δηλαδή ασυμπίεστο). Το *plSize πρέπει να έχει τιμή 0 όταν καλούμε την συνάρτηση και παίρνει τιμή ίση με το τελικό μέγεθος του συμπιεσμένου καρέ. Η συνάρτηση μας επιστρέφει ένα δείκτη τύπου void* στην προσωρινή μνήμη που βρίσκεται το συμπιεσμένο καρέ.

Στην συνέχεια πρέπει να γράψουμε το συμπιεσμένο καρέ στη ροή βίντεο που είχαμε δημιουργήσει με την συνάρτηση AVIStreamWrite όπως έχει περιγραφεί στην προηγούμενη παράγραφο. Για δεδομένα εγγραφής θα περάσουμε αυτό που μας επιστρέφει η ICSeqCompressFrame και για μέγεθος τους το plSize.

Αφού κάνουμε αυτή την διαδικασία για όλα τα καρέ του βίντεο αποδεσμεύουμε τον συμπιεστή καλώντας την συνάρτηση ICSeqCompressFrameEnd και αμέσως μετά την ICSeqCompressorFree:

```
void ICSeqCompressFrameEnd(
  PCOMPVARS pc
);

void ICCompressorFree(
```

```
  PCOMPVARS pc
);
```

Οι δυο συναρτήσεις παίρνουν ως μοναδικό τους όρισμα τον δείκτη στη δομή COMPVARS.

Τέλος κλείνουμε τη ροή δεδομένων που έχουμε δημιουργήσει με την AVIStreamRelease και το αρχείο με την AVIFileRelease. Το νέο αρχείο AVI έχει δημιουργηθεί.

2.9 Σχετικά με τα αρχεία RIFF

Το *RIFF (Resource Interchange File Format)* είναι ένα πρότυπο αποθήκευσης αρχείων πολυμέσων. Τα δεδομένα των αρχείων αυτών είναι οργανωμένα εσωτερικά σε ανεξάρτητα μεταξύ τους κομμάτια. Το κάθε κομμάτι περιέχει τη δική του επικεφαλίδα που το χαρακτηρίζει και τα δικά του δεδομένα. Από την επικεφαλίδα καθορίζεται το είδος των δεδομένων του. Έτσι μπορούν να αποφεύγονται κάποια προβλήματα συμβατότητας όταν αλλάζουν οι ορισμοί των τύπων δεδομένων ή δημιουργούνται νέοι. Για παράδειγμα μια εφαρμογή μπορεί να μην αναγνωρίζει το είδος των δεδομένων σε κάποιο κομμάτι αλλά αυτό δεν την εμποδίζει να διαβάσει τα υπόλοιπα. Παραδείγματα αρχείων RIFF αποτελούν τα AVI για αποθήκευση βίντεο, τα WAVE (Waveform audio) και MIDI (Musical Instrument Digital Interface) για ήχο, τα RDIB για εικόνες ανεξάρτητες συσκευής.

Πιο συγκεκριμένα τώρα τα κομμάτια δεδομένων που αναφέραμε αποτελούν το δομικό στοιχείο των RIFF και ονομάζονται *chunks*. Κάθε ένα αποτελείται από τα εξής πεδία:

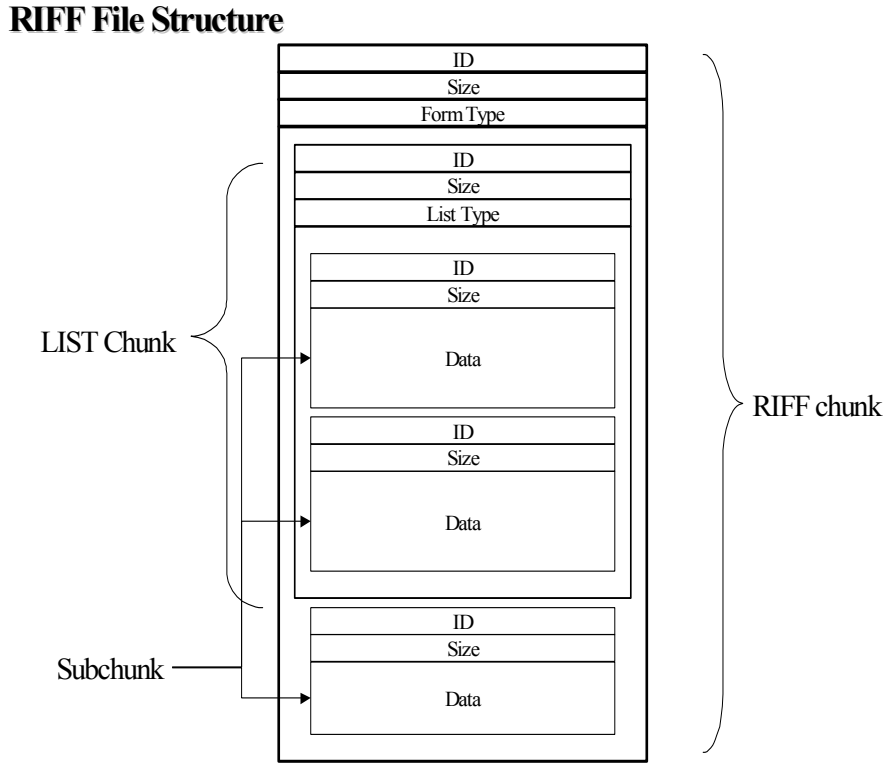
- Ένα κωδικό τεσσάρων χαρακτήρων που δείχνει την ταυτότητα του chunk (*chunk identifier*)
- Ένα DWORD που δίνει το μήκος των δεδομένων
- Τα δεδομένα

Ένα chunk μπορεί να βρίσκεται μέσα σε ένα άλλο chunk. Τότε αποκαλείται *subchunk*. Τα μόνα chunk που μπορούν να περιέχουν subchunks είναι αυτά με ταυτότητα “RIFF” ή “LIST”. Σε ένα αρχείο RIFF το πρώτο chunk είναι χαρακτηρισμένο ως “RIFF”. Όλα τα υπόλοιπα είναι subchunks σε αυτό.

Ένα chunk “RIFF” περιέχει στην επικεφαλίδα του ένα ακόμα πεδίο, το *form type*, μήκους τεσσάρων χαρακτήρων που περιγράφει τον τύπο των δεδομένων που είναι αποθηκευμένα στο αρχείο. Για παράδειγμα στα αρχεία DirectMusic το πεδίο form type έχει τιμή “DMST”.

Τα chunk τύπου “LIST” παρέχουν δυνατότητα ομαδοποίησης chunks. Έχουν επίσης ένα επιπλέον πεδίο τεσσάρων χαρακτήρων, το *List Type* που χαρακτηρίζει τον τύπο των δεδομένων της λίστας.

Στο παρακάτω σχήμα φαίνεται ένα απλό παράδειγμα αρχείου τύπου RIFF. Περιέχει το αρχικό chunk τύπου “RIFF” το οποίο αποτελείται από δυο subchunks. Το ένα subchunk είναι τύπου “LIST” και το άλλο είναι απλό. Το chunk “LIST” περιέχει με τη σειρά του δυο subchunks με δεδομένα.



Σχήμα 42. Παράδειγμα δομής αρχείου τύπου RIFF.

Στον προγραμματιστή παρέχονται μια σειρά από συναρτήσεις διαχείρισης αρχείων πολυμέσων τύπου RIFF. Οι συναρτήσεις αυτές αποτελούν την οικογένεια mmio (Multimedia Input/Output). Εμείς όμως ενδιαφερόμαστε μόνο για αρχεία AVI που αποτελούν ένα υποσύνολο των RIFF. Για αυτόν το λόγο δεν μελετούμε τις συναρτήσεις αυτές αλλά ένα άλλο υποσύνολο του WinAPI που αφορά σχεδόν αποκλειστικά τα βίντεο AVI. Οι συναρτήσεις αυτές, που αποτελούν την βιβλιοθήκη "AVIFile", παρουσιάστηκαν στην παράγραφο 2.

3 Image Processing Library

3.1 Η βιβλιοθήκη IPL

Η IPL (**I**mage **P**rocessing **L**ibrary) είναι μια βιβλιοθήκη επεξεργασίας εικόνας που έχει αναπτυχθεί από την Intel. Οι συναρτήσεις της είναι βελτιστοποιημένες για τους επεξεργαστές της Intel και εκμεταλλεύονται τον παραλληλισμό που προσφέρουν οι τελευταίες γενεές αυτών μέσω των εντολών SIMD (**S**ingle-**I**nstruction, **M**ultiple-**D**ata). Έτσι έχουν πολύ αυξημένη απόδοση στους επεξεργαστές αυτούς ενώ παράλληλα διατηρούν και προς τα πίσω συμβατότητα αφού μπορούν να εκτελεστούν και σε παλαιότερους.

Η βιβλιοθήκη δεν υποστηρίζει τους διάφορους γνωστούς τύπους εικόνας (JPG, GIF,TIFF κλπ.) για ανάγνωση και επεξεργασία. Ο μοναδικός τύπος εικόνας που

υποστηρίζεται άμεσα είναι τα DIB με βάθος χρώματος 24bit. Η βιβλιοθήκη ορίζει έναν δικό της τύπο δεδομένων την εικόνα IPL και όλες οι συναρτήσεις της χρησιμοποιούν αυτόν για είσοδο και έξοδο. Παρέχονται εκτός των άλλων και συναρτήσεις μετατροπής εικόνων DIB σε IPL καθώς και το αντίστροφο, εικόνων IPL σε DIB.

Η βιβλιοθήκη είναι φτιαγμένη για τα λειτουργικά συστήματα της Microsoft: Windows 95/98/NT. Μπορεί να ενσωματωθεί σε προγράμματα που έχουν φτιαχτεί σε C ή C++.

3.2 Η δομή της εικόνας IPL

Μια εικόνα στην IPL έχει μια επικεφαλίδα η οποία την περιγράφει σαν μια λίστα από ιδιότητες και δείκτες σε δεδομένα που σχετίζονται με αυτήν. Η συναρτήσεις της βιβλιοθήκης χρησιμοποιούν αυτήν την επικεφαλίδα για να λάβουν τα χαρακτηριστικά της εικόνας και τη διεύθυνση των δεδομένων της για να τα επεξεργαστούν αναλόγως. Γενικά μια εικόνα IPL μπορεί να έχει πολλές μορφές και διαφορετικούς τύπους οργάνωσης των εσωτερικών δεδομένων της. Αυτό συμβαίνει για πολλούς λόγους. Καταρχάς υποστηρίζονται πολλά διαφορετικά χρωματικά μοντέλα όπως RGB, RGBA, CMYK, HSV κλπ. με διαφορετικό πλήθος καναλιών το καθένα (Για παράδειγμα το RGB έχει τρία ενώ το CMYK τέσσερα). Επίσης το βάθος χρώματος δεν είναι το ίδιο σε κάθε εικόνα, όπως και η σειρά εμφάνισης τους.

Ανεξάρτητα από την εσωτερική οργάνωση μιας εικόνας, η βασική δομή της επικεφαλίδας της είναι σταθερή σε κάθε εικόνα και είναι η εξής:

```
typedef struct _IplImage {
    int             nSize;   /* size of iplImage struct
*/
    int             ID;      /* version
*/
    int             nChannels;
    int             alphaChannel;
    int             depth;   /* pixel depth in bits
*/
    char            colorModel[4];
    char            channelSeq[4];
    int             dataOrder;
    int             origin;
    int             align;    /* 4 or 8 byte align
*/
    int             width;
    int             height;
    struct _IplROI *roi;
    struct _IplImage
                   *maskROI;  /* poiner to maskROI if any
*/
    void           *imageId;  /* use of the application
*/
    struct
      _IplTileInfo *tileInfo; /* contains information on tiling
*/
    int             imageSize;/* useful size in bytes
*/
    char           *imageData;/* pointer to aligned image
*/
```

```
    int              widthStep;/* size of aligned line in bytes
*/
    int              BorderMode[4]; /*
*/
    int              BorderConst[4];/*
*/
    char            *imageDataOrigin;/* ptr to full, nonaligned image
*/
} IplImage;
```

Ο χρήστης της βιβλιοθήκης μπορεί να την ορίσει μόνος του και να κάνει τις ανάλογες αναθέσεις τιμών ή να χρησιμοποιήσει συναρτήσεις της βιβλιοθήκης σχετικές με την δημιουργία εικόνας IPL. Οι βασικές για αυτό τον σκοπό είναι οι iplCreateImageHeader για την δημιουργία της επικεφαλίδας και η iplAllocateImage για την δέσμευση μνήμης για τα δεδομένα της. Αναλυτικές πληροφορίες μπορούν να βρεθούν στο [4].

3.3 Μετατροπές ανάμεσα σε εικόνες DIB και IPL

Η μετατροπή μιας εικόνας DIB σε εικόνα της IPL γίνεται με την συνάρτηση iplConvertFromDIB:

```
void iplConvertFromDIB( BITMAPINFOHEADER* dib,
                        IplImage* image);
```

Η είσοδος της συνάρτησης είναι η παράμετρος "dib" η οποία είναι ο δείκτης σε μια εικόνα DIB. Η έξοδός της είναι η εικόνα "image". Η συνάρτηση θα μετατρέψει την εικόνα DIB σε μορφή IPL σύμφωνα με τα χαρακτηριστικά που θα βρει στην επικεφαλίδα της. Προϋποθέσεις για να λειτουργήσει αυτή η συνάρτηση είναι να έχει δημιουργηθεί η επικεφαλίδα της νέας εικόνας αλλά και να έχει δεσμευθεί η ανάλογη μνήμη (Με την συνάρτηση iplAllocateImage) που θα δεχθεί τα δεδομένα της εικόνας.

Η αντίστροφη μετατροπή μιας εικόνας IPL σε μορφή DIB γίνεται με την συνάρτηση iplConvertToDIB:

```
void iplConvertToDIB(   iplImage* image,
                        BITMAPINFOHEADER* dib,
                        int dither,
                        int paletteConversion);
```

Η παράμετρος "image" είναι η εικόνα εισόδου και η "dib" η εικόνα εξόδου. Η μετατροπή γίνεται σύμφωνα με τα χαρακτηριστικά της επικεφαλίδας της εικόνας DIB την οποία θα πρέπει να έχουμε δημιουργήσει καθώς και δεσμεύσει αρκετό χώρο αμέσως μετά από αυτή για να "χωρέσουν" τα δεδομένα της νέας εικόνας.

3.4 Φίλτρα της IPL

Η IPL περιλαμβάνει μια μεγάλη σειρά από φίλτρα και συναρτήσεις επεξεργασίας εικόνας. Καταρχάς παρέχει πλήρη υποστήριξη όλων των τεχνικών που παρουσιάστηκαν στο πρώτο μέρος. Πέρα από αυτές συμπεριλαμβάνονται και οι εξής:

- Διακριτός Μετασχηματισμός Συνημίτονου - Discrete Cosine Transform

- Μείωση δειγματοληψίας και κβάντισης
- Μετατροπές ανάμεσα σε διάφορα χρωματικά μοντέλα: RGB, HSV, HLS, LUV, XYZ, YUW.
- Ψευδοχρωματισμός
- Εξαγωγή πληροφοριών ιστογράμματος
- Συγκρίσεις ανάμεσα σε εικόνες
- Γεωμετρικοί μετασχηματισμοί
- Υπολογισμός ροπών

Συντομογραφίες

AVI (**A**udio **V**ideo **I**nterleaved / Ήχος Βίντεο Πεπλεγμένα): Είναι ένας τρόπος αποθήκευσης βίντεο και ήχου ανεπτυγμένος από τη Microsoft.

BMP (**B**it**map**): Τύπος αρχείου εικόνας.

DDB (**D**evice-**D**ependent **B**itmap): Κατηγορία αρχείων BMP που η μορφή τους εξαρτάται από την συσκευή στην οποία προορίζονται να απεικονιστούν.

DIB (**D**evice-**I**ndependent **B**itmap): Κατηγορία αρχείων BMP που η μορφή τους δεν εξαρτάται από την συσκευή στην οποία προορίζονται να απεικονιστούν.

DLL (**D**ynamic **L**ink **L**ibrary): Βιβλιοθήκη Δυναμικής Διασύνδεσης.

FFT (**F**ast **F**ourier **T**ransformation): Ο Γρήγορος Μετασχηματισμός Fourier μας προσφέρει τη δυνατότητα της μεταφοράς μιας εικόνας από το χωρικό πεδίο στο πεδίο των συχνοτήτων.

FPS (**F**rames **P**er **S**ec): Ο αριθμός των καρέ ανά δευτερόλεπτο σε ένα βίντεο.

HSV (**H**ue-**S**aturation-**V**alue): Χρωματικό μοντέλο που αναπαρίσταται με μια εξάπλευρη πυραμίδα.

IPL (**I**mage **P**rocessing **L**ibrary): Βιβλιοθήκη δυναμικής διασύνδεσης της Intel με συναρτήσεις επεξεργασίας εικόνας.

Pixel (**Pic**ture **el**ement – Εικονοστοιχείο): Τα στοιχεία από τα οποία αποτελείται μια ψηφιακή εικόνα.

RAW: Τύπος αρχείου εικόνας που δεν περιέχει καθόλου επικεφαλίδες αλλά μόνο τις τιμές των εικονοστοιχείων της.

RGB (**R**ed-**G**reen-**B**lue): Ένα από τα πιο βασικά χρωματικά μοντέλα. Ορίζει ένα τρισδιάστατο Καρτεσιανό σύστημα συντεταγμένων όπου κάθε άξονας αντιπροσωπεύει και ένα από τα τρία βασικά του χρώματα

RIFF (**R**esource **I**nterchange **F**ile **F**ormat): Ένα πρότυπο αποθήκευσης αρχείων πολυμέσων. Αυτό το πρώτυπο ακολουθούν τα βίντεο AVI.

ROI (**R**egion **O**f **I**nterest): Περιοχή ενδιαφέροντος μιας εικόνας.

SIMD (**S**ingle-**I**nstruction, **M**ultiple-**D**ata): Ένα σύνολο εντολών που υποστηρίζονται από τους επεξεργαστές της Intel και προσφέρουν δυνατότητες μεγάλης παραλληλίας σε εφαρμογές πολυμέσων.

SPL (**S**ignal **P**rocessing **L**ibrary): Βιβλιοθήκη δυναμικής διασύνδεσης της Intel με συναρτήσεις επεξεργασίας σήματος.

YIQ (Luminance **I**nphase **Q**uadrature): Χρωματικό μοντέλο που χρησιμοποιήθηκε πρωταρχικά στις έγχρωμες τηλεοράσεις.

WinAPI (**Win**dows **A**pplication **P**rogramming **I**nterface): Το σύνολο των συναρτήσεων πυρήνα των Windows.

Βιβλιογραφία

[1] K.R. Rao, J.J. Hwang, “Techniques & Standards for Image, Video & Audio Coding,” *Prentice Hall*

[2] Rafael C. Gonzales, Richard E. Woods, "Digital Image Processing," Second Edition, *Prentice Hall*

[3] Ιωάννης Πήτας, "Ψηφιακή Επεξεργασία Εικόνας," 2001

[4] "Intel® Image Processing Library Reference Manual". Δικτυακός τόπος: "http://www.intel.com"

[5] Eric J. Stollnitz, Tony D. DeRose, David H. Salesin, “Wavelets for Computer Graphics: A Primer Part 1”

[6] Eric J. Stollnitz, Tony D. DeRose, David H. Salesin, “Wavelets for Computer Graphics: A Primer Part 2”

[7] R.M. Haralick, K. Shanmugam, I. Dinstein, "Textural features for image classification,” IEEE Trans. Syst., Man Cybernet. 3 (1973) 610–621

[8] R.M. Haralick, "Statistical and structural approaches to texture," IEEE Proc. 67 (1979), pp. 786–804

[9] Διπλωματική εργασία της Μόκα Γεωργίας. “Πραγματικός-Χρόνος(Real-Time) και Πολυνηματισμός (Multithreading)”

[10] D.E. Maroulis, D.K. Iakovidis, S.A. Karkanis, D.A. Karras, "CoLD: A versatile detection system for colorectal lesions in endoscopy video-frames," accepted for publication to the Journal Computer Methods and Programs in Biomedicine, Elsevier.

[11] Microsoft Developers Network (MSDN) Library, July 1999. Δικτυακός τόπος: "http://msdn.microsoft.com/library/"

[12] MATLAB Help Desk και δικτυακός τόπος: "http://www.mathworks.com/"

[13] "Intel® Signal Processing Library Reference Manual". Δικτυακός τόπος: "http://developer.intel.com/software/products/perflib/"