Πανεπιστήμιο Θεσσαλίας
Τμήμα Μηχανικών Η/Υ, Τηλ & Δικτύων – Π.Μ.Σ

*Μεταπτυχιακή διατριβή:*

# *Μέθοδοι βελτιστοποίησης Λογισμικού και Υλικού Πολυμεσικού συστήματος σε ενσωματωμένες εφαρμογές.*

Βασίλης Ε. Σμαραΐδος

Νοέμβριος 2008

# Πρόλογος

Το J.P.E.G είναι ένα πρότυπο το οποίο γράφτηκε για την συμπίεση εικόνων. Το όνομα του προέρχεται από τα αρχικά του ονόματος της επιτροπής που το έγραψε (Joint Photographic Expert Group). Έγινε διεθνές το 1992. Ο στόχος της επιτροπής ήταν να εξελίξουν μια μέθοδο η οποία θα μπορούσε να συμπιέσει συνεχούς – τόνου σταθερές εικόνες η οποία θα κάλυπτε κάποιες συγκεκριμένες απαιτήσεις.

Ο *στόχος της παρούσης μεταπτυχιακής διατριβής* είναι να παρουσιάσει τεχνικές που πιθανόν βελτιώνουν την απόδοση του lossless JPEG κωδικοποιητή και στηρίζονται

- στους πίνακες Huffman
- στον τρόπο με τον οποίο γίνεται η κωδικοποίηση μιας εικόνας

και να ελέγξει εάν όντως πετυχαίνουν να βελτιώσουν την απόδοση του κωδικοποιητή.

# Περιεχόμενα

# Λίστα πινάκων

# Ευχαριστίες

Αρχικά, θα ήθελα να ευχαριστήσω τον καθηγητή και πρόεδρο του Τμήματος Μηχανικών Η/Υ , Τηλ & Δικτύων, κύριο *Γεώργιο Σταμούλη*. Η βοήθειά του όλα αυτά τα χρόνια ήταν ανεκτίμητη, η συμπαράσταση του αμέριστη, οι συμβουλές του πολύτιμες.

Επίσης, θα ήθελα να ευχαριστήσω τον καθηγητή του τμήματος, κύριο *Ιωάννη Κατσαβουνίδη* που μου έδωσε την ευκαιρία να πραγματοποιήσω αυτή την μεταπτυχιακή διατριβή. Σε όλη τη διάρκεια της διατριβής, με δίδαξε, με προβλημάτισε, με καθοδήγησε και με ενέπνευσε για να ολοκληρώσω το έργο μου.

Ακόμη θα ήθελα να ευχαριστήσω όλους τους μεταπτυχιακούς φοιτητές και υποψήφιους διδάκτορες του γραφείου Ε5 για την συνεργασία που είχαμε όλα αυτά τα χρόνια, φίλους και γνωστούς.

Τέλος, δεν έχω λόγια για την οικογένεια μου και τους γονείς μου που με στήριξαν όλα αυτά τα χρόνια, ως προπτυχιακό και μεταπτυχιακό φοιτητή στο τμήμα Μηχανικών Η/Υ ,Τηλ & Δικτύων, με όλους τους δυνατούς τρόπους.

*Στους γονείς μου,*
*Ευάγγελο κ Θεοδώρα που μου*
*έδωσαν την ευκαιρία να*
*σπουδάσω.*
*Σε όσους πίστευαν και υπέμειναν...*

# Εισαγωγή

# JPEG

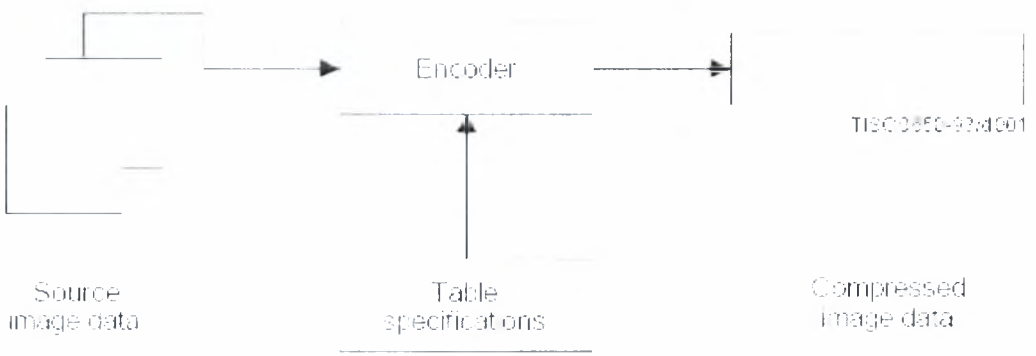
## Περιεχόμενα

Η ITU (International Telecommunication Union) είναι η ειδικευμένη αντιπροσωπεία των Ηνωμένων Εθνών στον τομέα των τηλεπικοινωνιών. Η CCITT (the International Telegraph and Telephone Consultative Committee) είναι ένα μόνιμο όργανο της ITU. Περίπου 166 χώρες μέλη, 163 επιστημονικές και βιομηχανικές οργανώσεις και 39 διεθνείς οργανισμοί συμμετέχουν στη CCITT που είναι το σώμα που καθορίζει τα πρότυπα παγκόσμιων τηλεπικοινωνιών. Η επιτροπή ISO/CCITT γνωστή ως JPEG έχει εργαστεί για να καθιερώσει τα πρώτα διεθνή πρότυπα συμπίεσης για τις συνεχούς τόνου σταθερές εικόνες, γκρι είτε έγχρωμες. Το πρότυπο JPEG στοχεύει να υποστηρίξει μια ευρεία ποικιλία των εφαρμογών για τις εικόνες συνεχούς τόνου καλύπτοντας τις εξής απατήσεις:

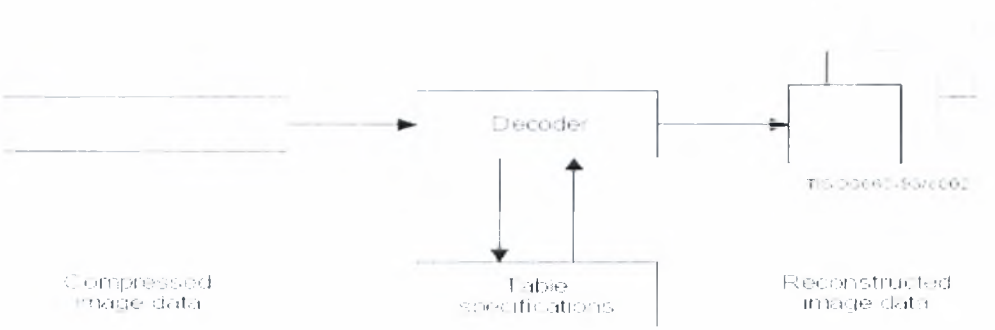
- Να μπορεί να παραμετροποιηθεί ο κωδικοποιητής έτσι ώστε να καλύπτει τις απαιτήσεις του εκάστοτε χρήστη ή εφαρμογής.
- Να μπορεί να εφαρμοστεί σε κάθε είδος συνεχούς τόνου ψηφιακή εικόνα ανεξαρτήτως διαστάσεων και εύρους χρωμάτων.
- Να έχει την κατάλληλη υπολογιστική πολυπλοκότητα για να πραγματοποιηθούν εφικτές υλοποιήσεις σε εφαρμογές λογισμικού, καθώς επίσης και σε εφαρμογές υλικού.
- Να καλύπτει τους ακόλουθους τρόπους εκτέλεσης:
  - Σειριακή κωδικοποίηση (Sequential encoding)
  - Προοδευτική κωδικοποίηση (Progressive encoding)
  - Ιεραρχική κωδικοποίηση (Hierarchical encoding)
  - Κωδικοποίηση χωρίς απώλειες δεδομένων (Lossless encoding).

## 2. Κωδικοποίηση και Αποκωδικοποίηση

Ένας κωδικοποιητής πραγματοποιεί τη διαδικασία της κωδικοποίησης. Όπως φαίνεται στην , ένας κωδικοποιητής παίρνει ως είσοδο μια εικόνα και κάποιους προκαθορισμένους πίνακες και με τη βοήθεια ενός διευκρινισμένου συνόλου διαδικασιών παράγει μια συμπιεσμένη εικόνα.




Ένας αποκωδικοποιητής πραγματοποιεί τη διαδικασία της αποκωδικοποίησης. Όπως φαίνεται στην ένας αποκωδικοποιητής παίρνει ως είσοδο μια συμπιεσμένη εικόνα και κάποιους προκαθορισμένους πίνακες, και με τη βοήθεια ενός διευκρινισμένου συνόλου διαδικασιών αναδημιουργεί την αρχική εικόνα.



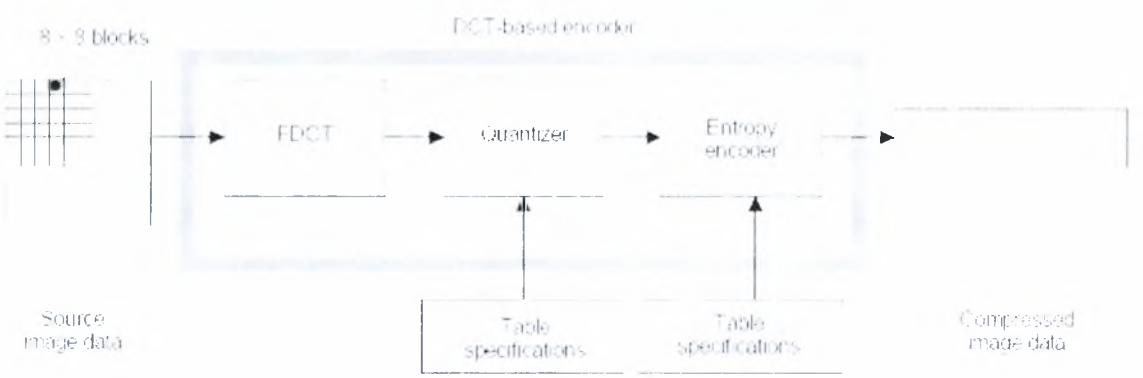
## 2.1 Μέθοδοι συμπίεσης JPEG

Το Jpeg περιλαμβάνει δυο διαφορετικές διαδικασίες κωδικοποίησης κ αποκωδικοποίησης.

- Lossy διαδικασία η οποία βασίζεται στον διακριτό μετασχηματισμό συνημίτονου (DCT) και είναι πολύ αποτελεσματική σε πολλές εφαρμογές.
- Lossless διαδικασία η οποία βασίζεται στην πρόβλεψη.

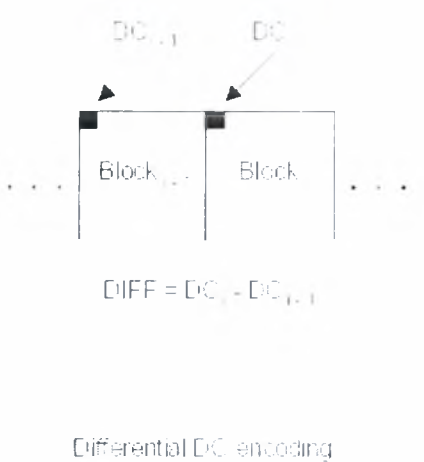
### 2.1.1 Κωδικοποίηση βασισμένη στο DCT

Η εικόνα 0.3 παρουσιάζει την διαδικασία κωδικοποίησης που βασίζεται στον DCT. Παρουσιάζεται η εκδοχή στην οποία η εικόνα αποτελείται μόνο από ένα συστατικό. Τα κύρια βήματα της κωδικοποίησης είναι αρχικά ευθύς μετασχηματισμός συνημίτονου, η κβαντοποίηση και τέλος κωδικοποίηση εντροπίας. Συγκεκριμένα, τα δείγματα της εικόνας εισόδου ομαδοποιούνται σε μπλοκ 8X8 και το κάθε μπλοκ μετασχηματίζεται με τον ευθύ διακριτό μετασχηματισμό συνημίτονου (FDCT) σε ένα σύνολο από 64 τιμές οι οποίες αναφέρονται ως συντελεστές του DCT. Ο πρώτος από αυτούς τους συντελεστές αναφέρεται ως DC συντελεστής ενώ οι άλλοι 63 ως AC συντελεστές.

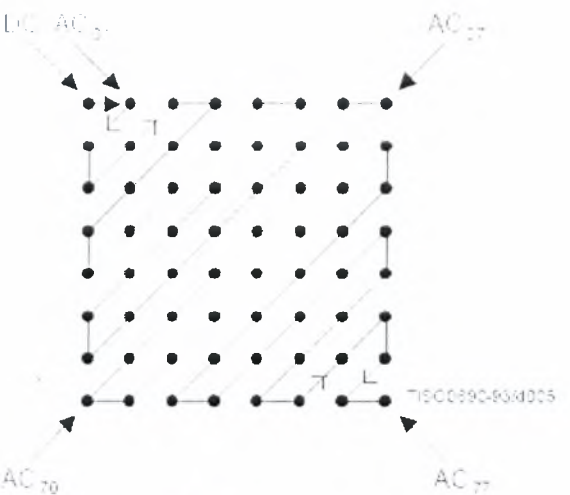


Κάθε ένας από αυτούς τους 64 συντελεστές κβαντοποιείται χρησιμοποιώντας την αντίστοιχη τιμή από τις εξήντα τέσσερις από έναν πίνακα κβαντοποίησης. Για αυτούς τους πίνακες κβαντοποίησης δεν υπάρχουν συγκεκριμένες τιμές αλλά ορίζονται κάθε φορά από την εφαρμογή προκειμένου να επιτύχει συγκεκριμένη ποιότητα εικόνας.

Μετά την κβαντοποίηση, οι συντελεστές προετοιμάζονται για κωδικοποίηση εντροπίας όπως φαίνεται στην




Differential DC encoding

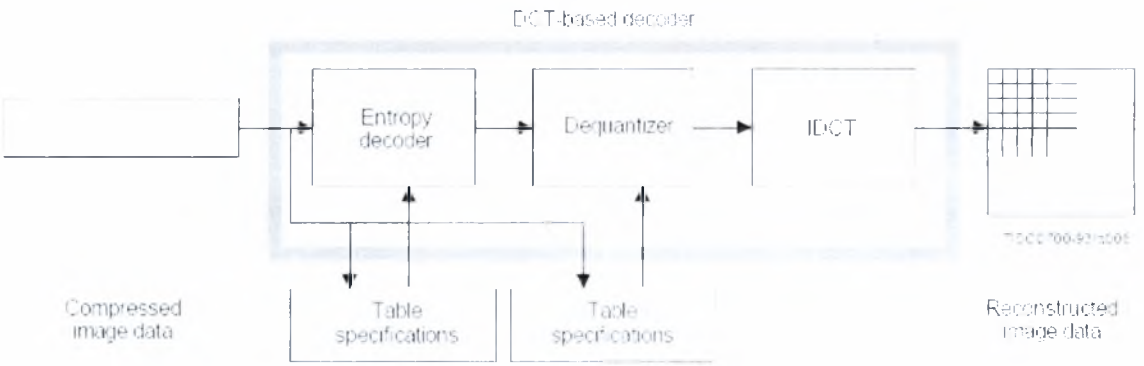


Zig-zag order

Ο κβαντοποιημένος DC συντελεστής του προηγούμενου μπλοκ χρησιμοποιείται για να προβλεφθεί ο κβαντοποιημένος DC συντελεστής του τρέχοντος μπλοκ και αυτό που κωδικοποιείται είναι η διαφορά τους. Όσον αφορά τους κβαντοποιημένους AC συντελεστές, αυτοί μετατρέπονται σε μια zig-zag ακολουθία μιας διάστασης όπως φαίνεται στην

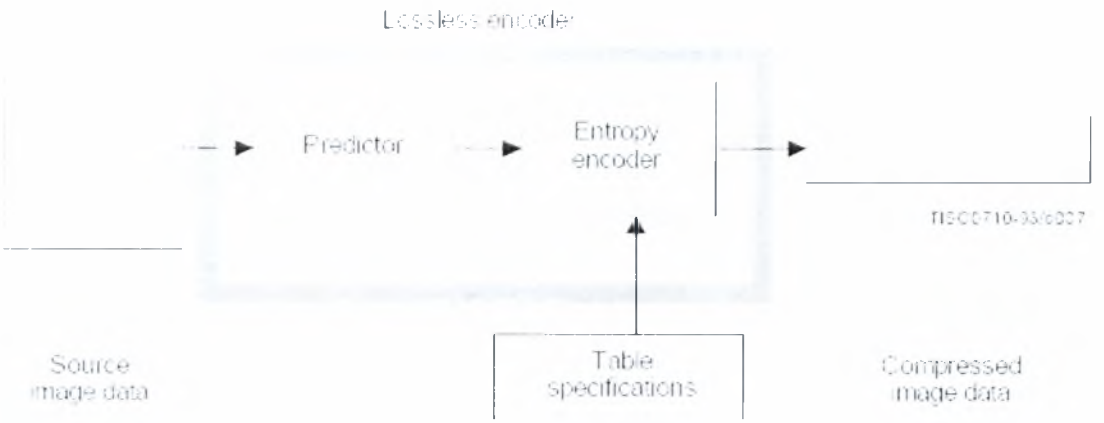
Οι κβαντοποιημένοι συντελεστές εισάγονται στην διαδικασία κωδικοποίησης εντροπίας η οποία συμπιέζει τα δεδομένα περισσότερο. Υπάρχουν δύο τρόποι κωδικοποίησης εντροπίας. Η κωδικοποίηση Huffman και η αριθμητική κωδικοποίηση.

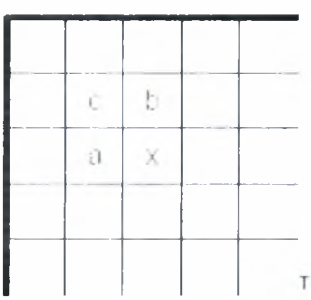
Η εικόνα 4.5 παρουσιάζει τις κύριες διαδικασίες αποκωδικοποίησης οι οποίες βασίζονται στον διακριτό μετασχηματισμό συνημίτονου. Κάθε βήμα που παρουσιάζεται εκτελεί ουσιαστικά την αντίστροφη διαδικασία που παρουσιάστηκε στον κωδικοποιητή. Ο αποκωδικοποιητής αποκωδικοποιεί την zig-zag ακολουθία από τους κβαντοποιημένους DCT συντελεστές. Μετά την αποκβαντοποίηση, οι DCT συντελεστές μετασχηματίζονται σε μπλοκ των 8X8 χρησιμοποιώντας τον αντίστροφο διακριτό μετασχηματισμό συνημίτονου(IDCT).




### 2.1.2 Κωδικοποίηση χωρίς απώλειες δεδομένων

Το διάγραμμα παρουσιάζει την κύρια διαδικασία για την κωδικοποίηση χωρίς απώλειες δεδομένων. Ο μηχανισμός πρόβλεψης (predictor) συνδυάζει τις ανακατασκευασμένες τιμές από τα τρία γειτονικά δείγματα στις θέσεις α, β, γ για να διαμορφώσει μία πρόβλεψη του δείγματος στη θέση x όπως φαίνεται στην
Αυτή η πρόβλεψη τότε αφαιρείται από την πραγματική τιμή του δείγματος στη θέση x και η διαφορά κωδικοποιείται είτε με Huffman είτε με αριθμητική κωδικοποίηση.




Υπάρχουν συγκεκριμένοι τρόποι εκτέλεσης κάτω από τις οποίες καθορίζονται διαδικασίας κωδικοποίησης. Αυτοί οι τρόποι είναι οι ακόλουθοι:

Α. σειριακή κωδικοποίηση η οποία εκτελείται από πάνω προς τα κάτω και από τα δεξιά προς τα αριστερά σε μια εικόνα.

Β. προοδευτική κωδικοποίηση η οποία στηρίζεται στις πολλαπλές σαρώσεις.

Γ. Ιεραρχική κωδικοποίηση

Δ. Κωδικοποίηση χωρίς απώλειες δεδομένων

Για τον σειριακό τρόπο ο οποίος βασίζεται σε DCT, τα 8x8 μπλοκ από δείγματα εισάγονται μπλοκ - μπλοκ από αριστερά προς τα δεξιά και από πάνω προς τα κάτω όπως φαίνεται στην εικόνα Π.1. Εφόσον ένα μπλοκ έχει μετασχηματιστεί με τον ευθύ μετασχηματισμό συνημίτονου, κβαντοποιείται και στη συνέχεια προετοιμάζεται για κωδικοποίηση εντροπίας .

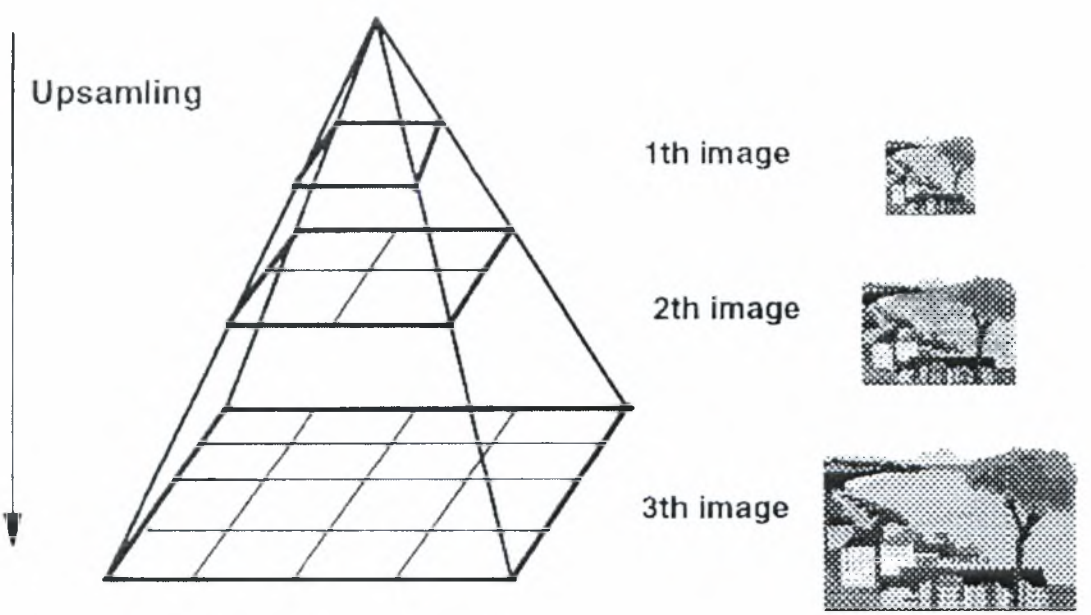
Sequential

Για τον προοδευτικό τρόπο βασισμένο σε DCT, τα μπλοκ κωδικοποιούνται με την ίδια σειρά όπως παρουσιάστηκε παραπάνω αλλά σε πολλαπλές σαρώσεις στην εικόνα όπως φαίνεται στην εικόνα Π.2. Αυτό επιτυγχάνεται τοποθετώντας έναν buffer ανάμεσα από τον κβαντοποιητή και την μηχανισμό που πραγματοποιεί η κωδικοποίηση εντοπίας. Καθώς κάθε μπλοκ μετασχηματίζεται με τον ευθύ μετασχηματισμό συνημίτονου και κβαντοποιείται, οι συντελεστές του αποθηκεύονται στον buffer. Οι DCT συντελεστές στον buffer κωδικοποιούνται μερικώς σε κάθε μια από τις πολλαπλές σαρώσεις.

Progressive

Υπάρχουν δυο διαδικασίες με τις οποίες οι κβαντοποιημένοι συντελεστές στον buffer κωδικοποιούνται μερικώς μέσα σε μια σάρωση. Πρώτα, μόνο μια συγκεκριμένη ζώνη από συντελεστές από την ακολουθία zig-zag χρειάζεται να κωδικοποιηθεί. Αυτή η διαδικασία καλείται φασματική επιλογή επειδή κάθε ζώνη περιέχει συντελεστές που καταλαμβάνουν χαμηλό ή υψηλό μέρος από το φάσμα συχνοτήτων για το κάθε 8x8 μπλοκ. Δεύτερον, οι συντελεστές μέσα στην τρέχουσα ζώνη δεν χρειάζεται να κωδικοποιηθούν με πλήρης ακρίβεια μέσα σε μια σάρωση. Κωδικοποιούνται αρχικά τα περισσότερο σημαντικά bits.

Στον ιεραρχικό τρόπο, μια εικόνα κωδικοποιείται σαν μια ακολουθία από καρέ. Αυτά τα καρέ συνήθως χρησιμοποιούνται για πρόβλεψη σε ακόλουθα καρέ. Η κωδικοποίηση ξεκινάει με μικρότερα καρέ και συνεχίζει με μεγαλύτερα. Το κάθε καρέ κλιμακώνεται και χρησιμοποιείται ως μηχανισμός πρόβλεψης στην επόμενη ανάλυση όπως φαίνεται στην

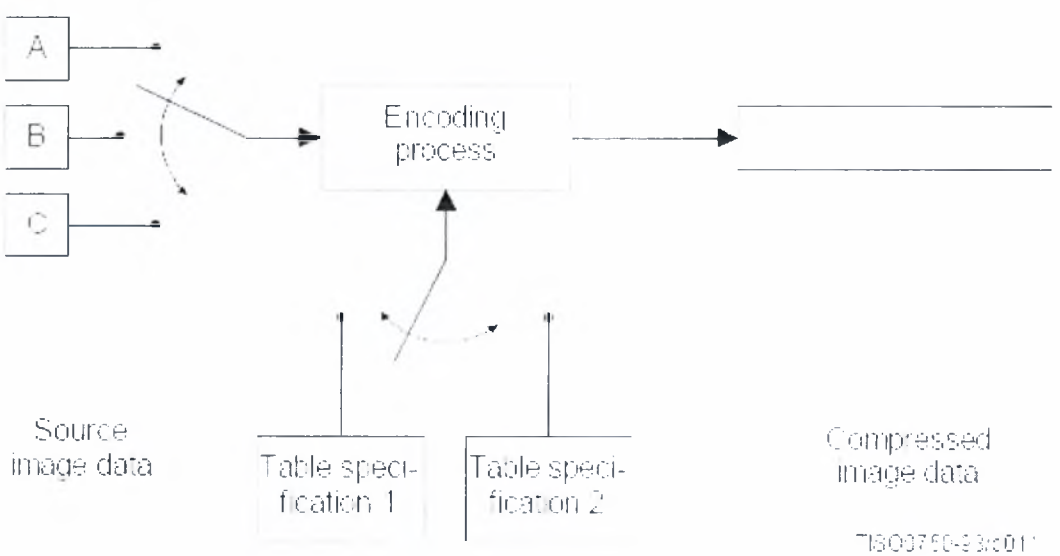


## 2.3 Ακρίβεια δείγματος

Για τις διαδικασίες που βασίζονται σε DCT, δυο διαφορετικές ακρίβειες δείγματος καθορίζονται: είτε 8 είτε 12 bits ανά δείγμα. Εφαρμογές οι ο ποίες χρησιμοποιούν δείγματα με άλλες ακρίβειες μπορούν να χρησιμοποιούσαν είτε 8 είτε 12 bits ακρίβεια ολισθαίνοντας τα δείγματα της εικόνας. Η baseline διαδικασία χρησιμοποιεί μόνο 8-bit ακρίβεια. Για τη διαδικασία χωρίς απώλειες δεδομένων, οι ακρίβειες δείγματος καθορίζονται από 2 έως 16 bits.

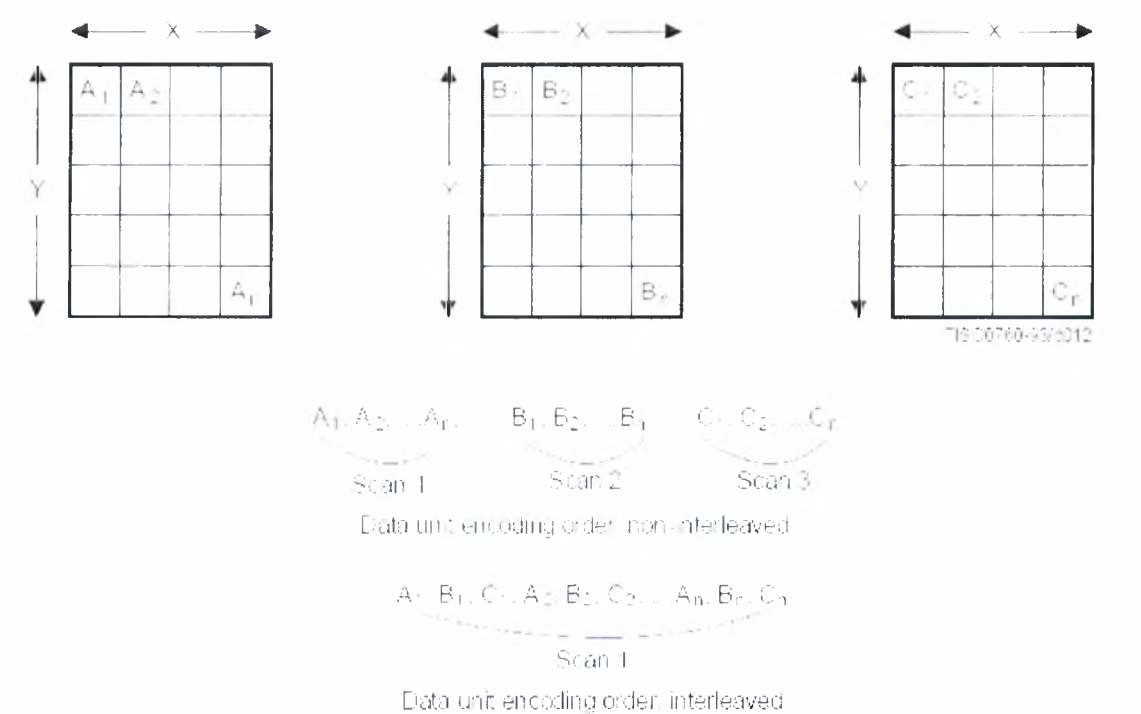
## 2.4 Interleaving πολλαπλά συστατικά

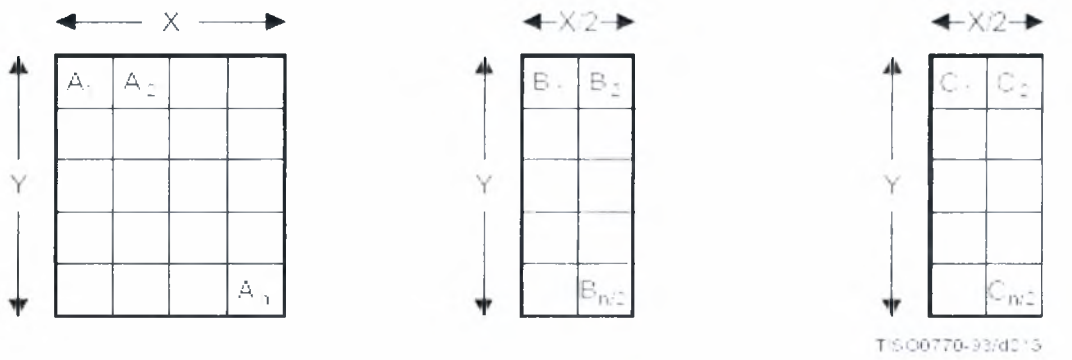
Η εικόνα 0.11 δείχνει ένα παράδειγμα για το πως μια διαδικασία κωδικοποίησης επιλέγει ανάμεσα από πολλαπλά στοιχεία μιας εικόνας καθώς επίσης και από μια σειρά από πίνακες δεδομένων. Στο συγκεκριμένο παράδειγμα, η εικόνα αποτελείται από 3 στοιχεία A,B,C και έχουν καθοριστεί 2 πίνακες δεδομένων για την κωδικοποίηση εντροπίας.



Μονάδα πληροφορίας (data unit) είναι ένα δείγμα για την lossless διαδικασία και ένα μπλοκ δειγμάτων 8X8 για τις διαδικασίες που βασίζονται σε DCT.

Στον σειριακό τρόπο εκτέλεσης, η διαδικασία είναι μη interleaved εάν ο κωδικοποιητής συμπιέζει όλα τα data units πρώτα από το στοιχείο Α, ύστερα από Β και έπειτα από το Γ. Η διαδικασία είναι interleaved όταν ο κωδικοποιητής κωδικοποιεί πρώτα μια μονάδα πληροφορίας από το Α, ύστερα ένα από το Β και ένα από το Γ. Υπάρχει περίπτωση και τα 3 συστατικά εικόνας να έχουν τις ίδιες διαστάσεις (εικόνα 0.12). Υπάρχει όμως και περίπτωση να μην έχουν τις ίδιες(εικόνα 0.13) .Τότε οι μονάδες πληροφορίας έχουν την ακόλουθη μορφή:






## 2.7 Μικρότερη κωδικοποιημένη μονάδα (MCU)

Το minimum coded unit ορίζεται ως η μικρότερη μονάδα που μπορεί να κωδικοποιηθεί. Εάν τα συμπιεσμένα δεδομένα είναι μη interleaved, το MCU θα είναι μια απλή μονάδα πληροφορίας. Αντίθετα, εάν τα συμπιεσμένα δεδομένα είναι interleaved, το MCU θα περιέχει ένα ή περισσότερα data units για κάθε συστατικό εικόνας.

## 2.8 Δομή συμπιεσμένων δεδομένων

Τα συμπιεσμένα δεδομένα εικόνας έχουν συγκεκριμένη δομή και ένα σύνολο από παραμέτρους. Τα διάφορα μέρη της πληροφορίας μιας συμπιεσμένης εικόνας αναγνωρίζονται από ειδικούς κωδικούς 2-bytes τα οποία καλούνται markers.

### 3. Περίληψη

| Διαδικασία baseline (βασισμένη σε DCT) |
|---|
| ➢ Διαδικασία βασισμένη στο διακριτό μετασχηματισμό συνημίτονου |
| ➢ Τα δείγματα είναι 8-bit ανά συστατικό εικόνας |
| ➢ Σειριακή |
| ➢ Κωδικοποίηση Huffman: 2 AC και 2 DC πίνακες. |
| ➢ Οι αποκωδικοποιητές θα εκτελέσουν σαρώσεις με 1,2,3 και 4 συστατικά εικόνας. |
| ➢ Interleaved και non –interleaved σαρώσεις. |

| Εκτεταμένη διαδικασία βασισμένη σε DCT |
|---|
| ➢ Διαδικασία βασισμένη στο διακριτό μετασχηματισμό συνημίτονου |
| ➢ Τα δείγματα είναι 8-bit ή 12 bits |
| ➢ Σειριακή ή ακολουθιακή |
| ➢ Κωδικοποίηση Huffman ή Αριθμητική: 4 AC και 4 DC πίνακες. |
| ➢ Οι αποκωδικοποιητές θα εκτελέσουν σαρώσεις με 1,2,3 και 4 συστατικά εικόνας. |
| ➢ Interleaved και non –interleaved σαρώσεις. |

| Διαδικασία χωρίς απώλειες δεδομένων |
|---|
| ➢ Διαδικασία πρόβλεψης |
| ➢ Τα δείγματα έχουν ακρίβεια από 2 έως 16 bits. |
| ➢ Σειριακή |
| ➢ Κωδικοποίηση Huffman ή Αριθμητική: 4 DC πίνακες. |
| ➢ Οι αποκωδικοποιητές θα εκτελέσουν σαρώσεις με 1,2,3 και 4 συστατικά εικόνας. |
| ➢ Interleaved και non –interleaved σαρώσεις. |

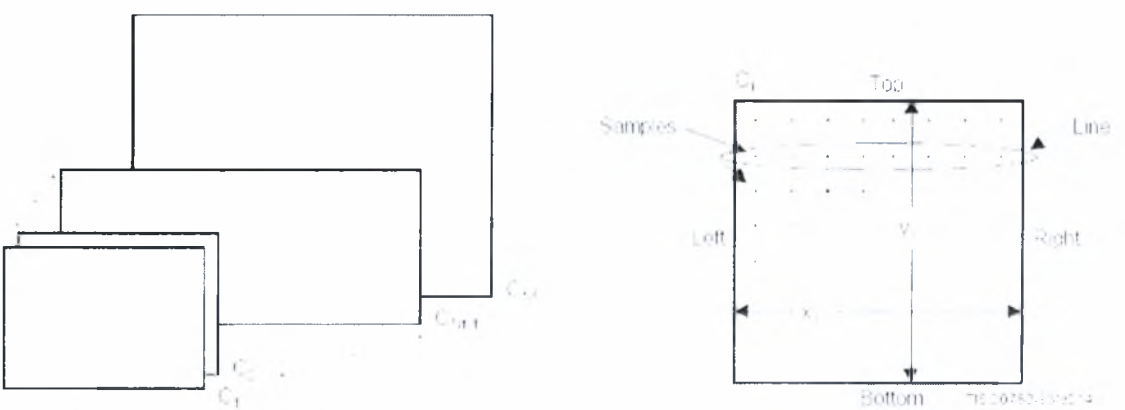
| **Ιεραρχική διαδικασία** |
| --- |
| ➢ Πολλαπλά πλαίσια. |
| ➢ Χρησιμοποιούνται εκτεταμένες διαδικασίες βασισμένες σε DCT ή σε διαδικασίες χωρίς απώλειες δεδομένων. |
| ➢ Οι αποκωδικοποιητές θα εκτελέσουν σαρώσεις με 1,2,3 και 4 συστατικά εικόνας. |
| ➢ Interleaved και non –interleaved σαρώσεις. |

# Κεφάλαιο 1ο

## Περιεχόμενα

Το κεφάλαιο αυτό έχει ως στόχο να παρουσιάσει χρήσιμες πληροφορίες σχετικά με τα χαρακτηριστικά της εικόνας και με την σειρά με την οποία τα συστατικά της εικόνας κωδικοποιούνται. Τέλος, έχει ως στόχο να παρουσιάσει κάποιους μαθηματικούς ορισμούς και να περιγράψει τον κύριο αλγόριθμο κωδικοποίησης εικόνας ο οποίος βασίζεται στον διακριτό μετασχηματισμό συνημίτονου και ονομάζεται baseline process.

Όπως φαίνεται στην , μια εικόνα αποτελείται από $N_f$ συστατικά. Το καθένα από αυτά έχει μοναδικό αναγνωριστικό $C_i$ και αποτελείται από ένα ορθογώνιο πίνακα από δείγματα με xi στήλες και $y_i$ γραμμές. Οι διαστάσεις των συστατικών μιας εικόνας παράγονται από δύο παραμέτρους, X και Y όπου X είναι η μεγαλύτερη τιμή από αυτές τις μεταβλητής xi σε όλα τα συστατικά του τρέχοντος πλαισίου και Y η μεγαλύτερη από τις τιμές της $y_i$ αντίστοιχα.




a. Source image with multiple components

b. Characteristics of an image component

Για κάθε συστατικό εικόνας, οι παράγοντες δειγματοληψίας καθορίζονται να σχετίζονται με τις διαστάσεις του κάθε συστατικού σύμφωνα με τις ακόλουθες εκφράσεις:

$$x_i = \left\lceil X \times \frac{H_i}{H_{max}} \right\rceil \text{ and } y_i = \left\lceil Y \times \frac{V_i}{V_{max}} \right\rceil.$$

Όπου $H_{max}$ και $V_{max}$ είναι οι μεγαλύτεροι παράγοντες δειγματοληψίας για όλα τα συστατικά στο τρέχον καρέ.

Για παράδειγμα, ας θεωρήσουμε μια εικόνα η οποία αποτελείται από τρία συστατικά με μέγιστες διαστάσεις 512 γραμμές και 512 δείγματα ανά γραμμή (X=512,Y=512) και με τους ακόλουθους παράγοντες δειγματοληψίας:

| | | |
|---|---|---|
| Component 0 | $H_0$=4 | $V_0$=1 |
| Component 1 | $H_1$=2 | $V_1$=2 |
| Component 2 | $H_2$=1 | $V_2$=1 |

Συνεπώς Hmax=4, Vmax=2. Το κάθε συστατικό εικόνας θα έχει τις ακόλουθες διαστάσεις

| | | |
|---|---|---|
| Component 0 | $x_0$=512 | $y_0$=256 |
| Component 1 | $x_1$=256 | $y_1$=512 |
| Component 2 | $x_2$=128 | $y_2$=256 |

### 2.2 Ακρίβεια δείγματος

Ένα δείγμα είναι ένας ακέραιος με ακρίβεια P bits, με τιμή οποιαδήποτε ανήκει στο εύρος τιμών από 0 έως $2^P$-1. Όλα τα δείγματα σε όλα τα συστατικά της εικόνας έχουν την ίδια ακρίβεια δείγματος P. Στην lossless κωδικοποίηση, η ακρίβεια είναι από 2 έως 16 bits.

### 2.3 Μονάδα πληροφορίας (unit)

Μια μονάδα πληροφορίας είναι ένα δείγμα στην κωδικοποίηση χωρίς απώλειες δεδομένων (lossless) και ένα 8x8 μπλοκ από συνεχές δείγματα στην κωδικοποίηση που βασίζεται στον DCT.

### 2.4 Παρεμβαστοίχιση

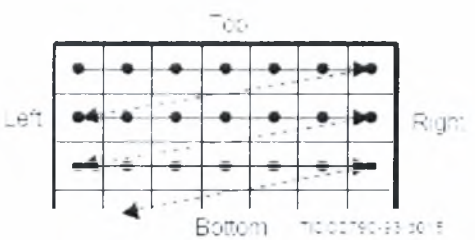
Η σειρά με την οποία οι μονάδες πληροφορίας κάθε συστατικού εικόνας εισέρχονται στην διαδικασία κωδικοποίησης είναι από αριστερά προς τα δεξιά και από πάνω προς τα κάτω μέσα σε ένα συστατικό εικόνας.

### 2.5 Σειρά κωδικοποίησης της πλάκας (scan)

Η επικεφαλίδα σάρωσης (scan header) καθορίζει την σειρά με την οποία οι μονάδες πληροφορίας της εικόνας θα κωδικοποιηθούν και θα τοποθετηθούν μέσα στη συμπιεσμένη μορφή. Για την τρέχουσα σάρωση, εάν η παράμετρος του scan header είναι $N_s$=1 τότε δεδομένα από μόνο ένα συστατικό εικόνας (το οποίο καθορίζεται από την παράμετρο $C_{s1}$) θα βρίσκονται μέσα στο πεδίο της σάρωσης. Αυτά τα δεδομένα είναι μη interleaved εξ ορισμού. Εάν η παράμετρος είναι $N_s$>1, τότε δεδομένα από τα $N_s$ συστατικά και συγκεκριμένα από το $C_{s1}$ έως το $C_{sNs}$ θα είναι παρούσα μέσα στο πεδίο της σάρωσης. Αυτά τα δεδομένα θα είναι συνεχώς interleaved. Η σειρά των συστατικών στη σάρωση θα είναι σύμφωνη με την σειρά που έχει καθοριστεί στην επικεφαλίδα καρέ (frame header).

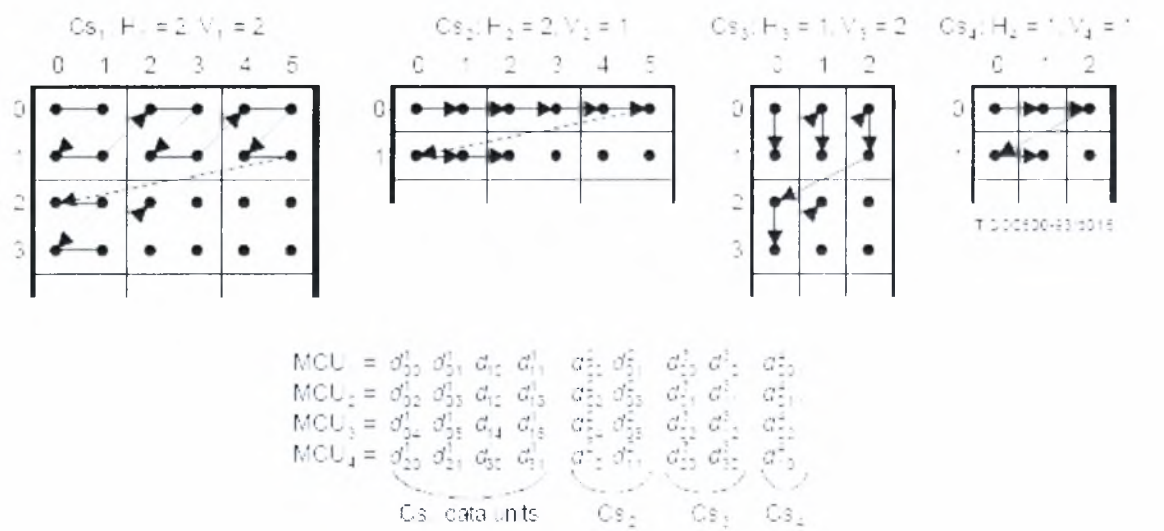
Για τα μη interleaved δεδομένα, το MCU είναι μια μονάδα πληροφορίας. Για τα interleaved δεδομένα, το MCU είναι μια ακολουθία από μονάδες πληροφορίας οι οποίες προσδιορίζονται από τους παράγοντες δειγματοληψίας των συστατικών εικόνων στη σάρωση.

Όταν η παράμετρος $N_s$ ισούται με ένα (όπου $N_s$ είναι ο αριθμός των συστατικών σε μια σάρωση), η σειρά των μονάδων πληροφορίας σε μια σάρωση θα είναι από αριστερά προς τα δεξιά και από πάνω προς τα κάτω όπως φαίνεται στην




Όταν η $N_s$ είναι μεγαλύτερη από ένα, κάθε συστατικό σάρωσης διαμερίζεται σε μικρούς ορθογώνιους πίνακες από $H_k$ οριζόντιες μονάδες πληροφορίας και $V_k$ κάθετες μονάδες πληροφορίας. Όπως φαίνεται στην εικόνα 1.3, όταν ο αριθμός των συστατικών εικόνας είναι τέσσερα ($N_s$=4) το $MCU_1$ αποτελείται από μονάδες πληροφορίας που προέρχονται από πάνω αριστερά από κάθε συστατικό $Cs_i$. Αντίστοιχα, στο $MCU_2$ οι μονάδες πληροφορίας θα προέρχονται από τις διπλανές

περιοχές σε σχέση με αυτές του $MCU_1$ με φορά από αριστερά προς τα δεξιά και από πάνω προς τα κάτω.



Τέλος, αξίζει να σημειώσουμε κάτι ακόμη. Στις διαδικασίες κωδικοποίησης που στηρίζονται στον DCT, εάν το $x_i$ δεν είναι πολλαπλάσιο του 8, η διαδικασία κωδικοποίησης θα επεκτείνει τον αριθμό των στηλών για να συμπληρωθούν τα δεξιότερα μπλοκ δειγμάτων. Εάν τα συστατικό εικόνας είναι interleaved, η διαδικασία κωδικοποίησης θα επεκτείνει επίσης των αριθμό των δειγμάτων με ένα ή περισσότερα μπλοκ εάν χρειαστεί έτσι ώστε ο αριθμός των μπλοκ να είναι ακέραιο πολλαπλάσιο του $H_i$ . Το ίδιο ισχύει και για την παράμετρο $y_i$ επεκτείνοντας τώρα των γραμμών.

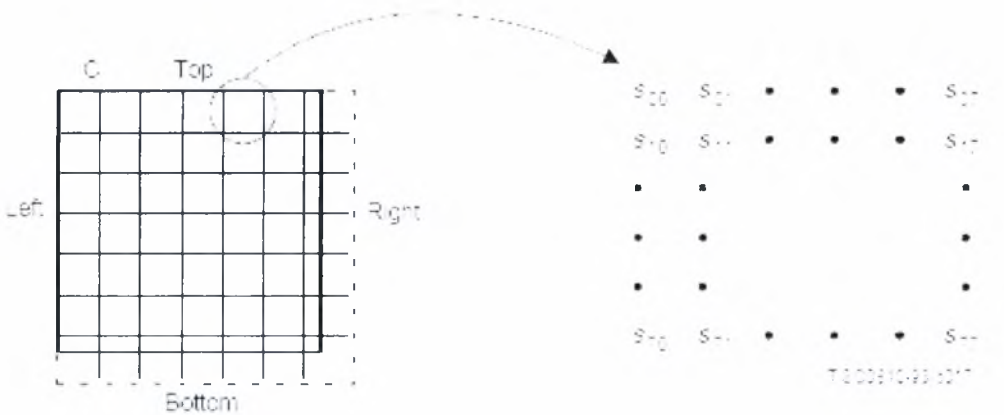
Για την διαδικασία χωρίς απώλειες δεδομένων, η μονάδα πληροφορίας είναι ένα δείγμα. Εάν τα συστατικά εικόνας είναι interleaved , η διαδικασία κωδικοποίησης θα επεκτείνει τον αριθμό των δειγμάτων εάν χρειαστεί έτσι ώστε ο αριθμός να είναι πολλαπλάσιο του $H_i$. Παρομοίως, η διαδικασία κωδικοποίησης θα επεκτείνει των

αριθμό των γραμμών εάν χρειαστεί έτσι ώστε ο αριθμός των γραμμών να είναι πολλαπλάσιο του $V_i$ .
Κάθε δείγμα που θα προστεθεί κατά την κωδικοποίηση για να συμπληρώσει το όχι πλήρες MCU θα αφαιρεθεί από την διαδικασία αποκωδικοποίησης.

Ο βασικός αλγόριθμος κωδικοποίησης ονομάζεται baseline αλγόριθμος και βασίζεται στον διακριτό μετασχηματισμό συνημίτονου. Στην συνέχεια περιγράφεται λεπτομερώς.

Πριν εφαρμοστεί ο ευθύς διακριτός μετασχηματισμός συνημίτονου (FDCT) σε ένα καρέ, τα δείγματα θα αφαιρεθούν κατά $2^{p-1}$ ,όπου p είναι η παράμετρος της ακρίβειας. Έτσι, όταν το P=8, η αφαίρεση θα είναι κατά 128. Κατά την διαδικασία αποκωδικοποίησης, εφόσον εφαρμοστεί ο αντίστροφος διακριτός μετασχηματισμός συνημίτονου (IDCT), στα δείγματα θα προστεθεί η τιμή 128.

Η εικόνα 1.4 δείχνει ένα συστατικό εικόνας το οποίο έχει διαμεριστεί σε 8x8 μπλοκς για τον υπολογισμό του ευθύ μετασχηματισμού συνημίτονου. Η εικόνα 1.4 επίσης καθορίζει την διάταξη των δειγμάτων μέσα σε ένα μπλοκ υποδεικνύοντας τους δείκτες που χρησιμοποιούνται στους υπολογισμούς του μετασχηματισμού συνημίτονου.




*Γ.FDCT and IDCT*

Οι παρακάτω ισότητες καθορίζουν τους υπολογισμούς του ευθύ και αντίστροφου μετασχηματισμού συνημίτονου.

FDCT $$S_{vu} = \frac{1}{4} C_u C_v \sum_{x=0}^{7} \sum_{y=0}^{7} s_{yx} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

IDCT $$s_{yx} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_u C_v S_{vu} \cos \frac{(2x+1)u\pi}{16} \cos \frac{(2y+1)v\pi}{16}$$

Όπου

$$C_u, C_v = 1/\sqrt{2} \text{ for } u, v = 0$$

$$C_u, C_v = 1 \text{ otherwise}$$

*Δ. κβαντοποίηση και αποκβαντοποίηση DCT συντελεστών*

Εφόσον ο ευθύς μετασχηματισμός συνημίτονου έχει υπολογιστεί για κάθε μπλοκ, κάθε ένας από τους 64 συντελεστές που παράγονται κβαντοποιείται. Το ποσό της κβαντοποίησης για κάθε συντελεστή $S_{vu}$ είναι το στοιχείο $Q_{vu}$ από τον πίνακα κβαντοποίησης και καθορίζεται από την παράμετρο $T_{qi}$. Ο κβαντοποιητής καθορίζεται από την ακόλουθη εξίσωση:

$$Sq_{vu} = round\left(\frac{S_{vu}}{Q_{vu}}\right)$$

Στρογγυλοποιημένος στον πλησιέστερο ακέραιο.

Στον αποκωδικοποιητή γίνεται η αντίστροφη διαδικασία

$$R_{vu} = Sq_{vu} \times Q_{vu}$$

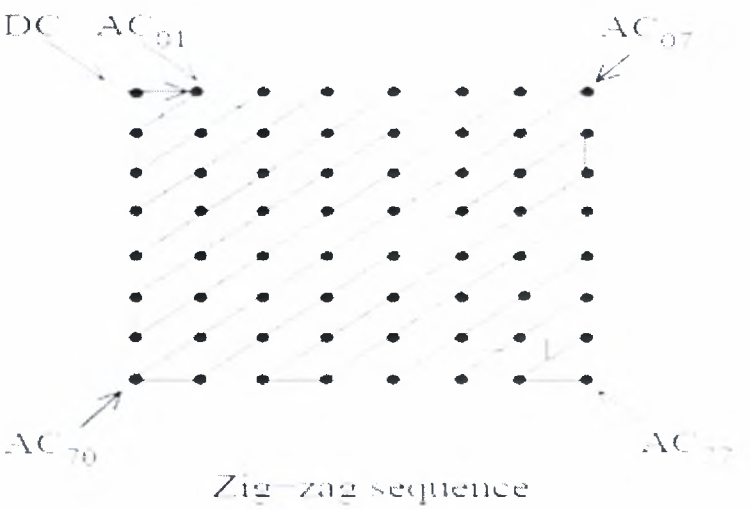
*Ε. κωδικοποίηση DC συντελεστή*

Μετά την κβαντοποίηση και κατά την κωδικοποίηση εντροπίας, ο κβαντοποιημένος DC συντελεστής $Sq_{00}$ επεξεργάζεται ξεχωριστά από τους υπόλοιπους 63 κβαντοποιημένους AC συντελεστές. Η τιμή η οποία θα κωδικοποιηθεί είναι η διαφορά(DIFF) μεταξύ του κβαντοποιημένου συντελεστή του τρέχοντος μπλοκ($DC_i$ ή $Sq_{00}$) από αυτόν του προηγούμενου μπλοκ του ίδιου συστατικού εικόνας (PRED).

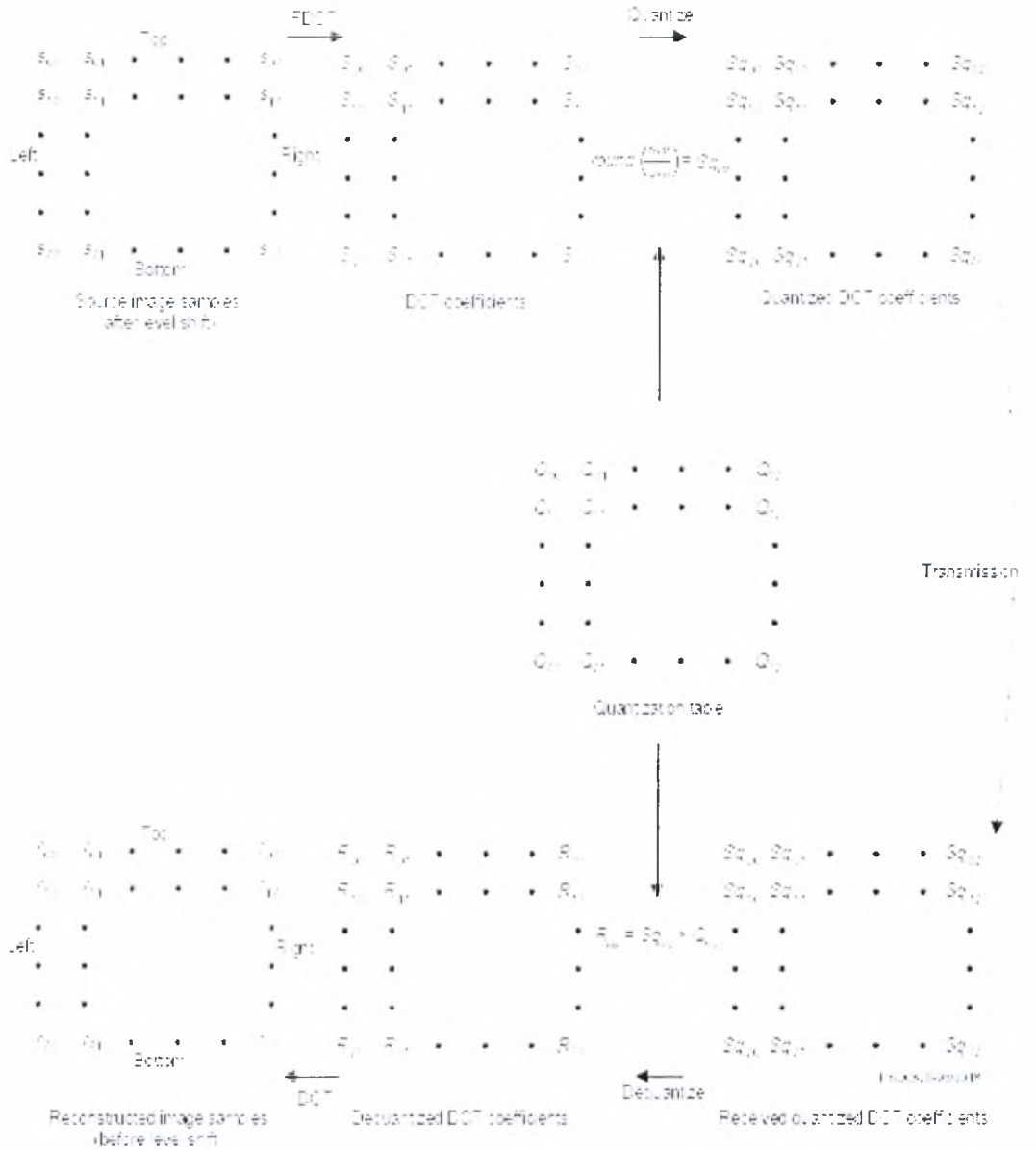
$$DIFF = DC_i - PRED$$

*Ζ. Ακολουθία Zig-zag*

Μετά την κβαντοποίηση και πριν την κωδικοποίηση εντροπίας, οι κβαντοποιημένοι AC συντελεστές μετατρέπονται σε μία zig-zag ακολουθία με την παρακάτω σειρά( ).




Στην παρουσιάζεται η σχέση μεταξύ του μπλοκ 8χ8 και των συντελεστών DCT.




Παρακάτω ακολουθεί ένα αριθμητικό παράδειγμα του baseline αλγορίθμου (

).

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 52 | 55 | 61 | 66 | 70 | 61 | 64 | 73 |
| 63 | 59 | 66 | 90 | 109 | 85 | 69 | 72 |
| 62 | 59 | 68 | 113 | 144 | 104 | 66 | 73 |
| 63 | 58 | 71 | 122 | 154 | 106 | 70 | 69 |
| 67 | 161 | 68 | 104 | 126 | 88 | 68 | 70 |
| 79 | 65 | 60 | 70 | 77 | 68 | 58 | 75 |
| 85 | 71 | 64 | 59 | 55 | 61 | 65 | 83 |
| 87 | 79 | 69 | 68 | 65 | 76 | 78 | 94 |

-128

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -76 | -73 | -67 | -62 | -58 | -67 | -64 | -55 |
| -65 | -69 | -62 | -38 | -19 | -43 | -59 | -56 |
| -66 | -69 | -60 | -15 | 16 | -24 | -62 | -55 |
| -65 | -70 | -57 | -6 | 26 | -22 | -58 | -59 |
| -61 | -67 | -60 | -24 | -2 | -40 | -60 | -58 |
| -49 | -63 | -68 | -58 | -51 | -65 | -70 | -53 |
| -43 | -57 | -64 | -69 | -73 | -67 | -63 | -45 |
| -41 | -49 | -59 | -60 | -63 | -52 | -50 | -34 |

FDCT

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| -415 | -29 | -62 | 25 | 55 | -20 | -1 | 3 |
| 7 | -21 | -62 | 9 | 11 | -7 | -6 | 6 |
| -46 | 8 | 77 | -25 | -30 | 10 | 7 | -5 |
| -50 | 13 | 35 | -15 | -9 | 6 | 0 | 3 |
| 11 | -8 | -13 | -2 | -1 | 1 | -4 | 1 |
| -10 | 1 | 3 | -3 | -1 | 0 | 2 | -1 |
| -4 | -1 | 2 | -1 | 2 | -3 | 1 | -2 |
| -1 | -1 | -1 | -2 | -1 | -1 | 0 | -1 |

Με κάποιον πίνακα κβαντοποίησης

[-26 -3 1 -3 2 6 2 -4 1 -4
1 4 1 1 5 0 2 0 0 1
2 0 0 0 0 0 -1 -1 EOB]

# Κεφάλαιο 2ο

# Κωδικοποίηση χωρίς απώλειες δεδομένων – Lossless encoding

**Περιεχόμενα**

## 1. Εισαγωγή

Το κεφάλαιο αυτό έχει ως στόχο να παρουσιάσει την μέθοδο κωδικοποίησης μιας εικόνας χωρίς απώλειες δεδομένων (lossless mode of operation). Η κωδικοποίηση μπορεί να πραγματοποιήσει είτε Huffman είτε αριθμητική κωδικοποίηση. Ο τρόπος αυτός κωδικοποίησης δεν στηρίζεται στον διακριτό μετασχηματισμό συνημίτονου αλλά στην πρόβλεψη η οποία πηγάζει από την σχέση ενός δείγματος με τα γειτονικά του.
Στην παρούσα εργασία, ο συγκεκριμένος τρόπος κωδικοποίησης αποτελεί το υπόβαθρο. Η κωδικοποίηση εντροπίας γίνεται με κωδικοποίηση Huffman.

## 2. Διαδικασίες lossless κωδικοποιητή

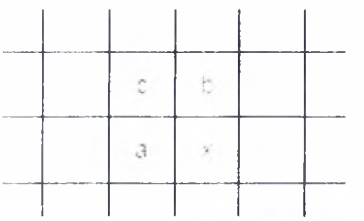
Σε αυτόν τον τρόπο κωδικοποίησης, η μονάδα πληροφορίας (data unit) είναι ένα δείγμα. Η ακρίβεια των δειγμάτων της εικόνας κυμαίνεται από 2 έως 16 bits. Τα δείγματα της εικόνας παρουσιάζονται ως μη προσημασμένοι ακέραιοι και δεν υφίστανται μετατόπιση επίπεδου πριν την κωδικοποίηση. Όταν ο κωδικοποιητής επανεκκινείται στην διαδικασία έλεγχου restart interval, η πρόβλεψη παίρνει την προκαθορισμένη της τιμή.

### 2.1 Lossless μοντέλο κωδικοποίησης

Το μοντέλο κωδικοποίησης που εξελίχθηκε για να κωδικοποιήσει τους DC συντελεστές στις διαδικασίες κωδικοποίησης βασισμένες σε DCT, επεκτείνεται για να επιτρέψει μια επιλογή από ένα σύνολο 7 μιας διάστασης και δυο διαστάσεων μηχανισμούς πρόβλεψης (predictor). Ο predictor γράφεται στην επικεφαλίδα σάρωσης. Ο ίδιος predictor χρησιμοποιείται για όλα τα συστατικά της σάρωσης. Κάθε συστατικό στη σάρωση μοντελοποιείται ξεχωριστά χρησιμοποιώντας προβλέψεις από τα γειτονικά δείγματα του ίδιου συστατικού.

## 2.2 Πρόβλεψη

Η δείχνει την σχέση ανάμεσα στις θέσεις a,b,c των αναδημιουργημένων γειτονικών δειγμάτων που χρησιμοποιούνται για πρόβλεψη και της θέσης του δείγματος x που πρόκειται να κωδικοποιηθεί.




Ορίζουμε το $P_x$ να είναι η πρόβλεψη της θέσης x και Ra ,Rb ,Rc να είναι τα αναδημιουργημένα δείγματα ακριβώς αριστερά, διαγώνια πάνω και πάνω από το τρέχον δείγμα. Οι επιτρεπόμενοι predictors, ένας εκ των οποίων επιλέγεται στην επικεφαλίδα σάρωσης παρουσιάζονται πιο κάτω ( ).

| Selection-value | Prediction |
|---|---|
| 0 | No prediction (See Annex J) |
| 1 | Px = Ra |
| 2 | Px = Rb |
| 3 | Px = Rc |
| 4 | Px = Ra + Rb – Rc |
| 5 | Px = Ra + (Rb – Rc) $2^{-1}$ |
| 6 | Px = Rb + (Ra – Rc) $2^{-1}$ |
| 7 | Px = (Ra + Rb) $2^{-1}$ |
| (1) Shift right arithmetic operation | |

## 2.3 Predictors

Οι επιλογές 1, 2 και 3 είναι για predictors μιας διάστασης και οι επιλογές 4,5,6 και 7 είναι δυο διαστάσεων. Ο οριζόντιος predictor χρησιμοποιείται στην πρώτη γραμμή των δειγμάτων στην αρχή της σάρωσης και στην αρχή κάθε restart interval. Ο επιλεγμένος predictor χρησιμοποιείται για όλες τις υπόλοιπες γραμμές. Το δείγμα που βρίσκεται ακριβώς από πάνω από το τρέχον (Rβ) χρησιμοποιείται στην αρχή κάθε γραμμής εκτός από την πρώτη. Στην αρχή κάθε γραμμής και στην αρχή κάθε restart interval, η τιμή πρόβλεψης $2^{p-1}$ χρησιμοποιείται , όπου P είναι η ακρίβεια του δείγματος. Εάν η παράμετρος point transform δεν είναι μηδέν, η τιμή πρόβλεψης στην αρχή των πρώτων γραμμών και στην αρχή κάθε restart interval είναι $2^{P-Pt-1}$ .

Για απλότητα της υλοποίησης, η διαίρεση με το 2 στους predictors 5 και 6 γίνεται με αριθμητική δεξιά ολίσθηση κατά ένα bit.

Η διαφορά μεταξύ της τιμής πρόβλεψης και της τιμής του δείγματος υπολογίζεται με υπόλοιπο 16. Στον αποκωδικοποιητή, η διαφορά αποκωδικοποιείται και προστίθεται το υπόλοιπο 16 στην πρόβλεψη.

## 2.4 Παράμετρος point transform

Στην κωδικοποίηση χωρίς απώλειες δεδομένων υπάρχει περίπτωση τα δείγματα της εικόνας να διαιρεθούν με μια ποσότητα $2^{Pt}$ όπου Pt η παράμετρος point transform. Στην lossless κωδικοποίηση επιλέγουμε να γίνει η διαίρεση αυτή με δεξιά ολίσθηση και όχι με αριθμητική πράξη. Ο λόγος είναι ότι με την ολίσθηση επηρεάζονται μόνο τα LSB (λιγότερα σημαντικά ψηφία ) ενώ με αριθμητική πράξη επηρεάζεται όλη η ποσότητα.

| SSSS | Difference values |
|---|---|
| 0 | 0 |
| 1 | −1,1 |
| 2 | −3,−2,2,3 |
| 3 | −7,...,−4,4,...,7 |
| 4 | −15,...,−8,8,...,15 |
| 5 | −31,...,−16,16,...,31 |
| 6 | −63,...,−32,32,...,63 |
| 7 | −127,...,−64,64,...,127 |
| 8 | −255,...,−128,128,...,255 |
| 9 | −511,...,−256,256,...,511 |
| 10 | −1 023,...,−512,512,...,1 023 |
| 11 | −2 047,...,−1 024,1 024,...,2 047 |
| 12 | −4 095,...,−2 048,2 048,...,4 095 |
| 13 | −8 191,...,−4 096,4 096,...,8 191 |
| 14 | −16 383,...,−8 192,8 192,...,16 383 |
| 15 | −32 767,...,−16 384,16 384,...,32 767 |
| 16 | 32 768 |

## 3. *Διαδικασίες lossless κωδικοποιητή*

Οι lossless αποκωδικοποιητές πρέπει να είναι ικανοί να αποκωδικοποιήσουν κωδικοποιημένα δεδομένα εικόνας με ακρίβεια από 2 έως 16 bits. Η μονάδα πληροφορίας είναι ένα δείγμα. Όταν ο αποκωδικοποιητής επανεκκινείται εξαιτίας της διαδικασίας ελέγχου restart interval η πρόβλεψη παίρνει την ίδια τιμή με αυτή που χρησιμοποίησε ο κωδικοποιητής. Τέλος, αποκωδικοποιούν τη διαφορά και προσθέτουν σε αυτή το υπόλοιπο 16 για να δημιουργηθεί το δείγμα εξόδου. Εάν είναι ενεργοποιημένη η παράμετρος Pt, τα δείγματα πολλαπλασιάζονται με ένα παράγοντα $2^{Pt}$.

# Κεφάλαιο 3ο

## Σύνταξη συμπιεσμένων δεδομένων

### Περιεχόμενα

## 1. Εισαγωγή

Η δομή των συμπιεσμένων δεδομένων αποτελείται από μια ακολουθία από κώδικες που διατάσσονται σε bytes.

## 2. Βασικά μέρη δομής συμπιεσμένων δεδομένων

Τα συμπιεσμένα δεδομένα αποτελούνται κυρίως από παραμέτρους, markers και κομμάτια κωδικοποιημένων δεδομένων. Οι παράμετροι και οι markers συχνά οργανώνονται σε marker κομμάτια.

### 2.1 Παράμετροι

Οι παράμετροι είναι μη προσημασμένοι ακέραιοι με συγκεκριμένο μέγεθος και συγκεκριμένη τιμή η οποία καθορίζεται από την διαδικασία κωδικοποίησης και από τα χαρακτηριστικά της εικόνας. Το μέγεθος τους είναι συνήθως 4-bit, 1-byte είτε 2-bytes. Η ύπαρξη τους είναι πάρα πολύ σημαντική. Χωρίς αυτές, θα ήταν αδύνατον να ανακατασκευαστεί η εικόνα από τον αποκωδικοποιητή.

Για τις παραμέτρους με μέγεθος 2 bytes, το περισσότερο σημαντικό byte γράφεται πρώτα και ύστερα το λιγότερο σημαντικό. Οι παράμετροι που έχουν μήκος 4 bits, συνήθως έρχονται σε ζευγάρια και κωδικοποιούνται ως ένα απλό byte με την πρώτη 4-bit παράμετρο να καταλαμβάνει τα τέσσερα περισσότερο σημαντικά bit. Το ίδιο συμβαίνει και για τα εναπομείναντα μεγέθη παραμέτρων, τα περισσότερα σημαντικά bits γράφονται πρώτα.

### 2.2 Markers

Οι markers χρησιμοποιούνται στο να αναγνωρίζονται τα διάφορα δομικά μέρη των συμπιεσμένων δεδομένων. Πολλοί markers ακολουθούνται από μια λίστα από παραμέτρους όπως περιγράψαμε ενώ άλλοι όχι. Όλοι τους είναι δυο bytes. Το πρώτο byte είναι το X'FF το οποίο ακολουθείται από ένα άλλο byte το οποίο δεν πρέπει να είναι το 0 ή το X'FF.

Ακολουθεί ο πίνακας 3.1 ο οποίος παρουσιάζει όλους τους πιθανούς markers. Με αστερίσκο είναι όσοι στέκονται μόνοι τους, τα οποία δεν είναι η αρχή ενός marker κομματιού.

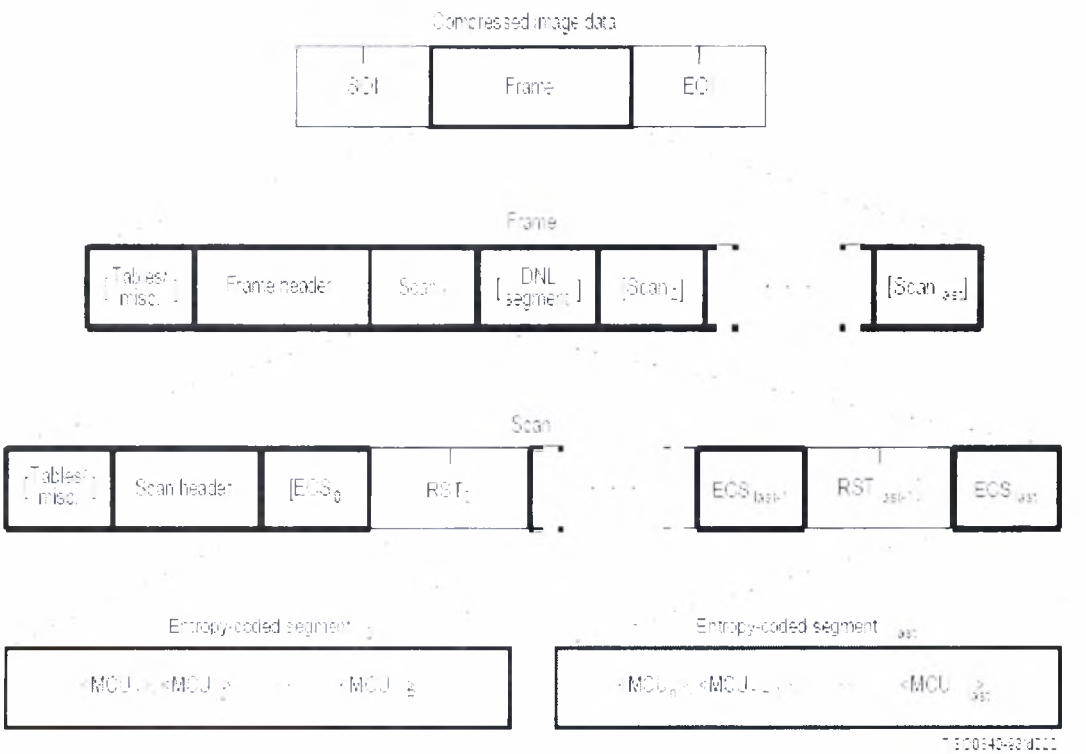
| Code Assignment | Symbol | Description |
|---|---|---|
| Start Of Frame markers, non-differential Huffman coding | | |
| X'FFC0'<br>X'FFC1'<br>X'FFC2'<br>X'FFC3' | $SOF_0$<br>$SOF_1$<br>$SOF_2$<br>$SOF_3$ | Baseline DCT<br>Extended sequential DCT<br>Progressive DCT<br>Lossless (sequential) |
| Start Of Frame markers, differential Huffman coding | | |
| X'FFC5'<br>X'FFC6'<br>X'FFC7' | $SOF_5$<br>$SOF_6$<br>$SOF_7$ | Differential sequential DCT<br>Differential progressive DCT<br>Differential lossless (sequential) |
| Start Of Frame markers, non-differential arithmetic coding | | |
| X'FFC8'<br>X'FFC9'<br>X'FFCA'<br>X'FFCB' | JPG<br>$SOF_9$<br>$SOF_{10}$<br>$SOF_{11}$ | Reserved for JPEG extensions<br>Extended sequential DCT<br>Progressive DCT<br>Lossless (sequential) |
| Start Of Frame markers, differential arithmetic coding | | |
| X'FFCD'<br>X'FFCE'<br>X'FFCF' | $SOF_{13}$<br>$SOF_{14}$<br>$SOF_{15}$ | Differential sequential DCT<br>Differential progressive DCT<br>Differential lossless (sequential) |
| Huffman table specification | | |
| X'FFC4' | DHT | Define Huffman table(s) |
| Arithmetic coding conditioning specification | | |
| X'FFCC' | DAC | Define arithmetic coding conditioning(s) |
| Restart interval termination | | |
| X'FFD0' through X'FFD7' | $RST_m$* | Restart with modulo 8 count "m" |
| Other markers | | |
| X'FFD8'<br>X'FFD9'<br>X'FFDA'<br>X'FFDB'<br>X'FFDC'<br>X'FFDD'<br>X'FFDE'<br>X'FFDF'<br>X'FFE0' through X'FFEF'<br>X'FFF0' through X'FFFD'<br>X'FFFE' | SOI*<br>EOI*<br>SOS<br>DQT<br>DNL<br>DRI<br>DHP<br>EXP<br>$APP_n$<br>JPG<br>COM | Start of image<br>End of image<br>Start of scan<br>Define quantization table(s)<br>Define number of lines<br>Define restart interval<br>Define hierarchical progression<br>Expand reference component(s)<br>Reserved for application segments<br>Reserved for JPEG extensions<br>Comment |
| Reserved markers | | |
| X'FF01'<br>X'FF02' through X'FFBF' | TEM*<br>RES | For temporary private use in arithmetic coding<br>Reserved |

### 2.3 Marker Κομμάτια

Αποτελούνται από marker ο οποίος ακολουθείται από σχετικές παραμέτρους. Η πρώτη παράμετρος είναι η παράμετρος που καθορίζει το μήκος του συγκεκριμένου κομματιού και είναι 2 bytes. Αυτή η παράμετρος είναι υπεύθυνη για την κωδικοποίηση του αριθμού των bytes στο κάθε κομμάτι εκτός βέβαια από τον 2-byte marker. Τα κομμάτια marker που αναγνωρίζονται από τους markers SOF και SOS αναφέρονται σε κεφαλίδες, στην επικεφαλίδα καρέ και σάρωσης αντίστοιχα.

## 3. Σύνταξη Lossless κωδικοποίησης

Παρακάτω περιγράφεται η σύνταξη των συμπιεσμένων δεδομένων για lossless κωδικοποίηση. Η δείχνει την υψηλού επιπέδου σύνταξη για όλα τα μέρη της συμπιεσμένης μορφής.



Διακρίνονται τρεις markers:

SOI: Σηματοδοτεί την αρχή της πληροφορίας για την συμπιεσμένη εικόνα.

EOI: Σηματοδοτεί το τέλος της πληροφορίας για την συμπιεσμένη εικόνα.

$RST_m$: marker επανεκκίνησης. Ένας υπό συνθήκη marker ο οποίος τοποθετείται ανάμεσα από τα marker κομμάτια που προέρχονται από κωδικοποίηση εντροπίας μόνο εάν η παράμετρος επανεκκίνησης είναι ενεργοποιημένη. Υπάρχουν οχτώ μοναδικοί markers επανεκκίνησης (m=0-7)οι οποίοι επαναλαμβάνονται σε ακολουθία από 0 έως 7 αρχίζοντας από το 0 για κάθε σάρωση.

Αξίζει να σημειώσουμε ότι υπάρχει μόνο ένα καρέ το οποίο όταν συμπιεστεί πρέπει να ακολουθηθεί από το τον marker EOI.

Το δεύτερο επίπεδο καθορίζει ότι το καρέ πρέπει να αρχίζει με μια κεφαλίδα καρέ και θα περιέχει μια ή περισσότερες σαρώσεις. Διάφοροι ορισμοί πινάκων και marker κομμάτια μπορεί να προηγούνται της επικεφαλίδας καρέ. Εάν το κομμάτι DNL είναι παρών, πρέπει να ακολουθεί την πρώτη σάρωση. Κάθε σάρωση θα περιέχει πληροφορία από ένα έως τέσσερα συστατικά εικόνας.

Το τρίτο επίπεδο καθορίζει ότι μια σάρωση θα ξεκινάει με μια επικεφαλίδα σάρωσης και θα περιέχει ένα η περισσότερα κομμάτια που έχουν υποστεί κωδικοποίηση εντροπίας. Διάφοροι ορισμοί πινάκων και marker κομμάτια μπορεί να προηγούνται της επικεφαλίδας σάρωσης. Εάν η παράμετρος επανεκκίνησης δεν είναι ενεργοποιημένη, θα υπάρχει μόνο ένα κομμάτι που έχει υποστεί κωδικοποίηση εντροπίας και δεν θα εμφανίζονται markers επανεκκίνησης. Εάν η παράμετρος επανεκκίνησης είναι ενεργοποιημένη, ο αριθμός των κομματιών που έχουν υποστεί κωδικοποίηση εντροπίας καθορίζεται από το μέγεθος της εικόνας και το καθορισμένο restart interval.

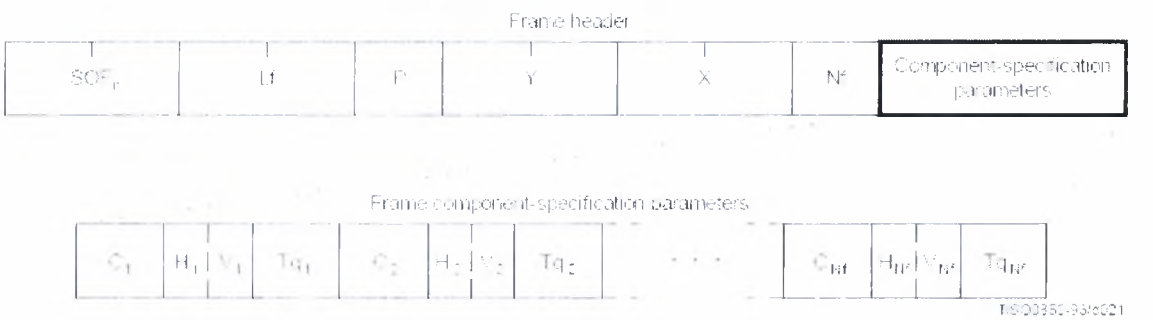
Το τέταρτο επίπεδο καθορίζει ότι το κωδικοποιημένο κομμάτι αποτελείται από μια ακολουθία από κωδικοποιημένα MCUs. Εάν η παράμετρος επανεκκίνησης είναι ενεργοποιημένη και έχει την τιμή $R_i$, κάθε κωδικοποιημένο κομμάτι εκτός από το

πρώτο θα περιέχει $R_i$ MCUs. Το τελευταίο θα περιέχει όσα MCUs συμπληρώνουν την σάρωση.

Τέλος παρατηρούμε ότι ο καθορισμός των πινάκων που είναι απαραίτητοι για την αποκωδικοποίηση γράφονται σε συγκεκριμένες τοποθεσίες της κωδικοποιημένης πληροφορίας.

### *3.1 Σύνταξη επικεφαλίδας καρέ*

Η καθορίζει την επικεφαλίδα καρέ η οποία θα παρουσιάζεται στην αρχή του καρέ. Η επικεφαλίδα καθορίζει τα χαρακτηριστικά της εικόνας, τα συστατικά εικόνας στο καρέ και τους παράγοντες δειγματοληψίας για κάθε συστατικό εικόνας και καθορίζει τον προορισμό από όπου οι κβαντοποιημένοι πίνακες θα χρησιμοποιηθούν κατά την αποκωδικοποίηση.



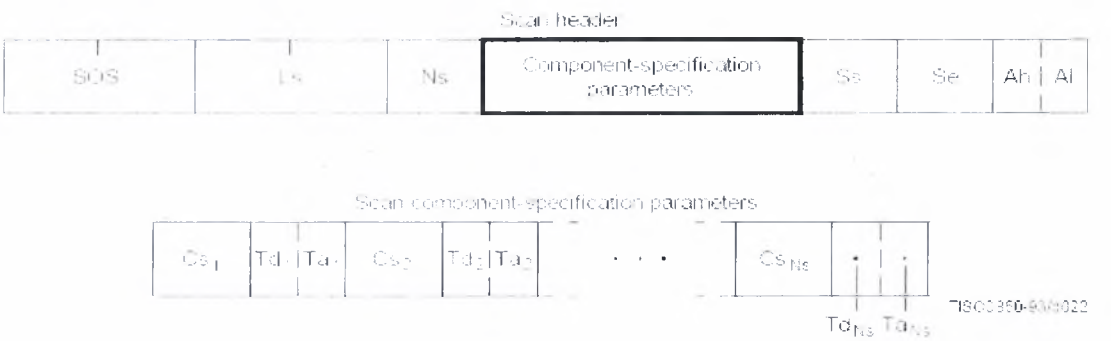

Οι markers και οι παράμετροι που χρησιμοποιούνται επεξηγούνται στον . Το μέγεθος και οι επιτρεπόμενες τιμές παρουσιάζονται στον .

| SOF3: marker για Lossless τρόπο κωδικοποίηση, κωδικοποίηση Huffman |
|---|
| Lf: μέγεθος επικεφαλίδας καρέ |
| P: ακρίβεια δείγματος σε bits |
| Y: αριθμός γραμμών |
| X: αριθμός δειγμάτων ανά γραμμή |
| Nf: αριθμός συστατικών εικόνας στο καρέ |
| Ci: αναγνωριστής συστατικού εικόνας |
| Hi: οριζόντιος παράγοντας δειγματοληψίας |
| Vi: κάθετος παράγοντας δειγματοληψίας |
| Tqi: δείκτης πίνακα κβαντοποίησης |

| Parameter | Size (bits) | Values | | | |
|---|---|---|---|---|---|
| | | Sequential DCT | | Progressive DCT | Lossless |
| | | Baseline | Extended | | |
| Lf | 16 | 8 + 3 × Nf | | | |
| P | 8 | 8 | 8, 12 | 8, 12 | 2-16 |
| Y | 16 | 0-65 535 | | | |
| X | 16 | 1-65 535 | | | |
| Nf | 8 | 1-255 | 1-255 | 1-4 | 1-255 |
| $C_i$ | 8 | 0-255 | | | |
| $H_i$ | 4 | 1-4 | | | |
| $V_i$ | 4 | 1-4 | | | |
| $T_{qi}$ | 8 | 0-3 | 0-3 | 0-3 | 0 |

## 3.2 Σύνταξη επικεφαλίδας σάρωσης

Η καθορίζει την επικεφαλίδα σάρωσης η οποία θα παρουσιάζεται στην αρχή της σάρωσης. Αυτή η επικεφαλίδα καθορίζει ποια συστατικά καθορίζονται στη σάρωση και καθορίζει από πού οι πίνακες εντροπίας θα χρησιμοποιηθούν κατά την αποκωδικοποίηση. Για την διαδικασία αποκωδικοποίησης χωρίς απώλειες, οι παράμετροι σάρωσης καθορίζουν την πρόβλεψη και το point transform.




Το μέγεθος και οι επιτρεπόμενες τιμές των παραμέτρων παρουσιάζονται στον

| Parameter | Size (bits) | Values | | | |
|---|---|---|---|---|---|
| | | Sequential DCT | | Progressive DCT | Lossless |
| | | Baseline | Extended | | |
| Ls | 16 | 6 + 2 × Ns | | | |
| Ns | 8 | 1–4 | | | |
| $Cs_j$ | 8 | 0–255[1] | | | |
| Td | 4 | 0–1 | 0–3 | 0–3 | 0–3 |
| $Ta_j$ | 4 | 0–1 | 0–3 | 0–3 | 0 |
| Ss | 8 | 0 | 0 | 0–63 | 1–7[2] |
| Se | 8 | 63 | 63 | Ss–63[3] | 0 |
| Ah | 4 | 0 | 0 | 0–13 | 0 |
| Al | 4 | 0 | 0 | 0–13 | 0–15 |

[1] $Cs_j$ shall be a member of the set of $Cs_j$ specified in the frame header.
[2] 0 for lossless differential frames in the hierarchical mode (see B.3.1).
[3] 0 if Ss equals zero.

Οι markers και οι παράμετροι που χρησιμοποιούνται επεξηγούνται στον

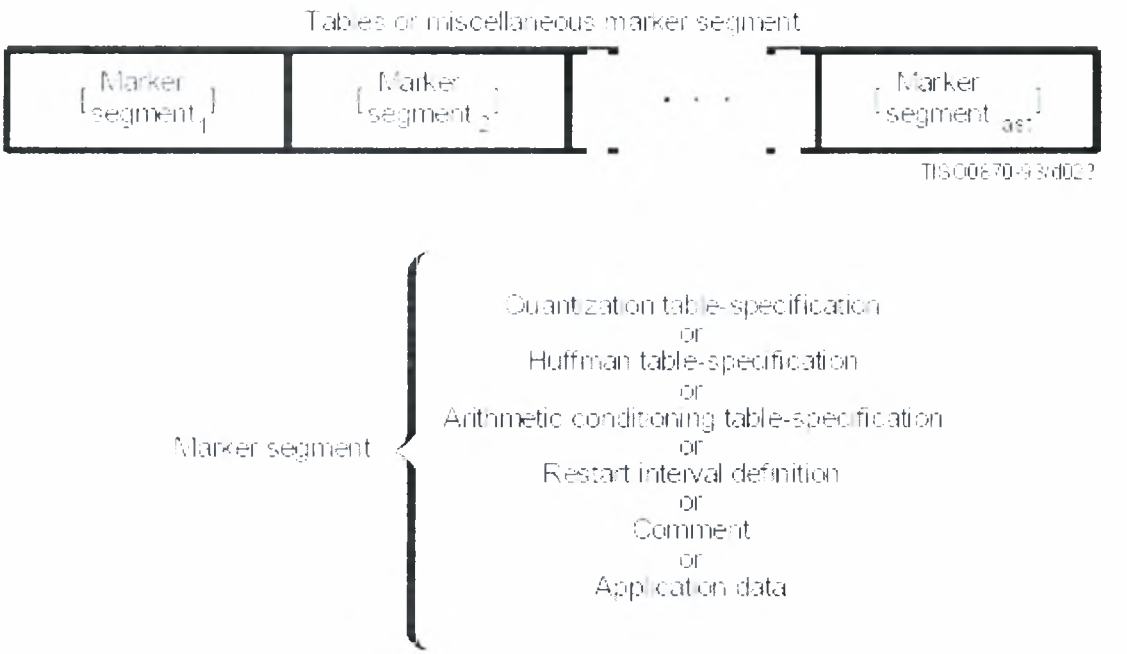
| |
|---|
| SOS : αρχή marker σάρωσης |
| Ls: μήκος επικεφαλίδας σάρωσης |
| Ns: αριθμός συστατικών εικόνας στη σάρωση |
| $Cs_j$: Καθορίζει ποίο από τα Nf συστατικά εικόνας που καθορίστηκαν στη επικεφαλίδα σάρωσης θα είναι το j-οστό συστατικό στη σάρωση. Κάθε $Cs_j$ θα πρέπει να ταιριάξει με ένα από τα Ci που καθορίστηκαν στην επικεφαλίδα σάρωσης με τη σωστή σειρά. |
| $Td_j$: δείκτης για πίνακες DC κωδικοποίησης εντροπίας. Καθορίζει έναν από τους τέσσερις πίνακες κωδικοποίησης εντροπίας. |
| $Ta_j$: δείκτης για πίνακες AC κωδικοποίησης εντροπίας. Καθορίζει έναν από τους τέσσερις πίνακες κωδικοποίησης εντροπίας. |
| Ss: επιλογή πρόβλεψης |
| Se: στην lossless κωδικοποίηση είναι μηδέν. |
| Ah: στην lossless κωδικοποίηση είναι μηδέν. |
| Al: point transform |

4. *Σύνταξη καθορισμού πινάκων και marker κομματιών*

Στην παράγραφο αυτή περιγράφονται ποια marker κομμάτια υπάρχουν ( ) και πώς αυτά συντάζονται στην συμπιεσμένη μορφή. Τα κομμάτια αυτά είναι

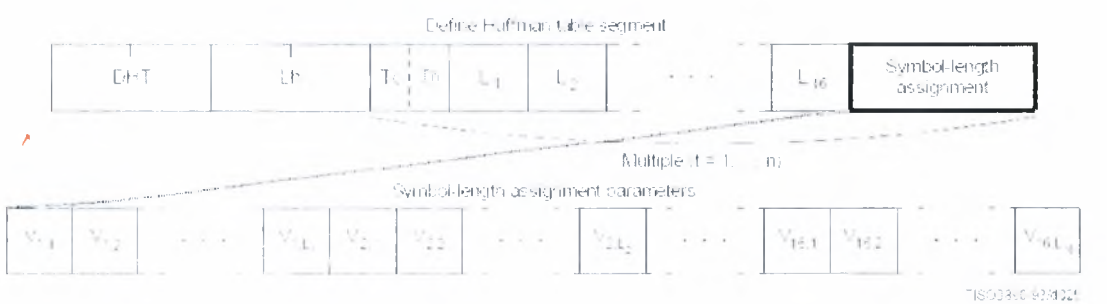
1. Καθορισμός πινάκων κβαντοποίησης
2. Καθορισμός πινάκων Huffman
3. Καθορισμός πινάκων αριθμητικής κωδικοποίησης
4. Καθορισμός restart interval
5. σχόλια
6. πληροφορίες εφαρμογής

Από αυτά, θα εξετάσουμε τον καθορισμό πινάκων Huffman και τον καθορισμό του restart interval.




## 4.1 *Σύνταξη καθορισμού πινάκων Huffman*

Η καθορίζει το κομμάτι εκείνο της συμπιεσμένης εικόνας το οποίο εμπεριέχει τους πίνακες Huffman.



Οι markers και οι παράμετροι που χρησιμοποιούνται επεξηγούνται στον

| |
|---|
| DHT: marker που καθορίζει το Huffman Table |
| Lh: μήκος όλων των παραμέτρων του πίνακα Huffman |
| Tc:κλάση πίνακα .Μηδέν γαι DC ,1 για AC |
| Th: δείκτης για κάθε ένα πίνακα κωδικοποίησης. |
| $L_i$:αριθμός κωδίκων Huffman μήκους i(1<= I <=16).Στοιχεία του πίνακα BITS |
| $V_{i,j}$:τιμές οι οποίες σχετίζονται με κάθε κώδικα Huffman.Στοιχεία του πίνακα HUFFVAL |

Το μέγεθος και οι επιτρεπόμενες τιμές των παραμέτρων παρουσιάζονται στον

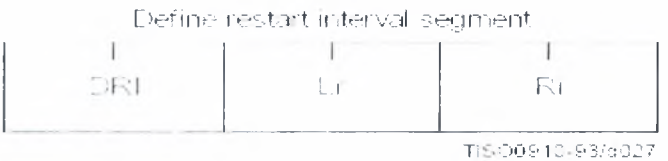
| Parameter | Size (bits) | Values | | | |
|---|---|---|---|---|---|
| | | Sequential DCT | | Progressive DCT | Lossless |
| | | Baseline | Extended | | |
| Lh | 16 | $2 + \sum_{i=1}^{n} (1^{-} + m_i)$ | | | |
| Tc | 4 | 0, 1 | | | 0 |
| Th | 4 | 0, 1 | 0-3 | | |
| $L_i$ | 8 | 0-255 | | | |
| $V_{i,j}$ | 8 | 0-255 | | | |

- Το n καθορίζει τον αριθμό των πινάκων Huffman που καθορίζονται στο DHT .
- Η τιμή $m_t$ καθορίζει τον αριθμό των παραμέτρων οι οποίες ακολουθούν τους 16 Li παραμέτρους για τον πίνακα t και δίνεται από τη παρακάτω σχέση και είναι διαφορετικό για κάθε πίνακα.

$$m_i = \sum_{i=1}^{15} L_i$$

## 4.2 Σύνταξη Restart interval

Η καθορίζει το κομμάτι εκείνο της συμπιεσμένης εικόνας το οποίο καθορίζει το restart interval.




Οι markers και οι παράμετροι που χρησιμοποιούνται επεξηγούνται στον .

| |
|---|
| DRI: καθροίζει τον marker toy restart interval |
| Lr:καθορίζει το μήκος |
| Ri:καθορίζει τον αριθμό των MCU |

Η τιμή του MCUR είναι ο αριθμός των MCUs τα οποία απαιτούνται για make up μιας γραμμής κάθε συστατικού στη σάρωση.

Το μέγεθος και οι επιτρεπόμενες τιμές των παραμέτρων παρουσιάζονται στον

| Parameter | Size (bits) | Values | | | |
|---|---|---|---|---|---|
| | | Sequential DCT | | Progressive DCT | Lossless |
| | | Baseline | Extended | | |
| Lr | 16 | 4 | | | |
| Ri | 16 | | | | 1 - MCUR |

# Κεφάλαιο 4ο

# Διαδικασίες lossless κωδικοποιητή & αποκωδικοποιητή.

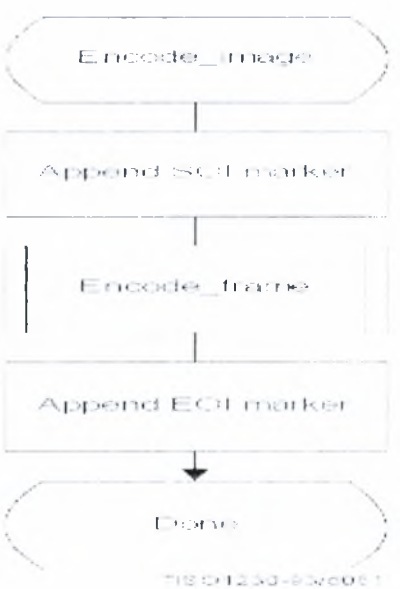
**Περιεχόμενα**

## 1. Εισαγωγή

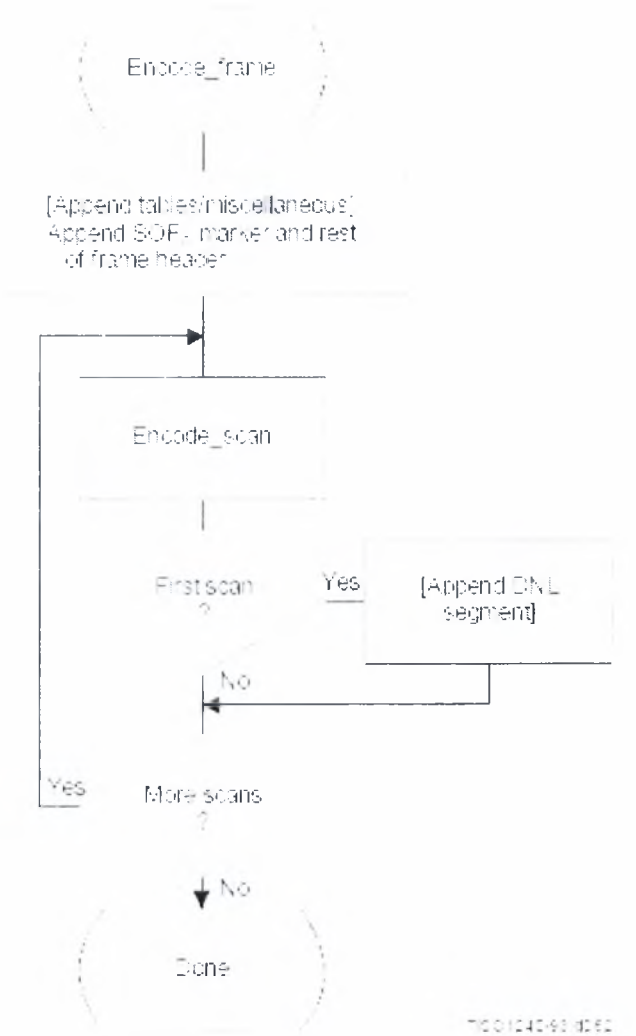
Σε αυτό το κεφάλαιο θα περιγράψουμε τις διαδικασίες με τις οποίες γίνεται η κωδικοποίηση μιας εικόνας. Στην ουσία, η κωδικοποίηση μιας εικόνας είναι μια ιεραρχική διαδικασία η οποία απαιτεί τις ακόλουθες υπορουτίνες: Αρχικά απαιτείται η κωδικοποίηση ενός καρέ. Η κωδικοποίηση ενός καρέ περιλαμβάνει την κωδικοποίηση μιας σάρωσης και αυτή με την σειρά της την κωδικοποίηση ενός restart interval εάν υπάρχει. Τέλος, φθάνοντας στο κατώτερο επίπεδο κωδικοποιείται ένα MCU, το οποίο περιλαμβάνει μια ακολουθία από δείγματα.

## 2. Κωδικοποίηση εικόνας

Η κωδικοποίηση εικόνας ( ) περιλαμβάνει αρχικά την προσάρτηση του κατάλληλου marker (SOI) καθώς επίσης και την κωδικοποίηση κάθε καρέ (ένα καρέ υπάρχει στην lossless κωδικοποίηση) η οποία θα παρουσιαστεί πιο κάτω. Τέλος, εφόσον ολοκληρωθεί η κωδικοποίηση του καρέ, γίνεται προσάρτηση του marker ο οποίος υποδηλώνει το τέλος κωδικοποίησης της εικόνας (EOI) .



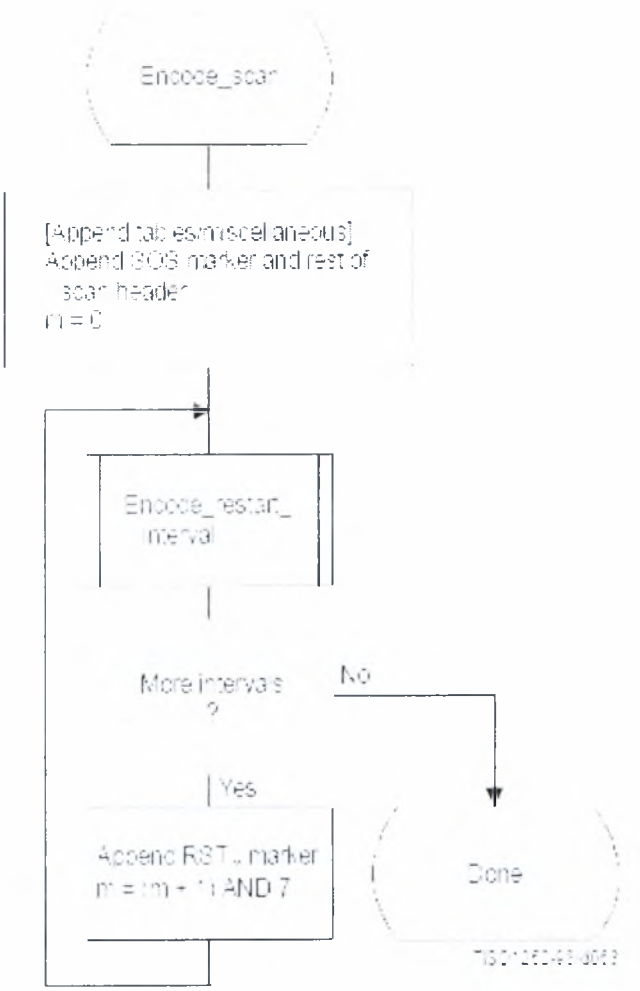
Η διαδικασία για την κωδικοποίηση ενός καρέ ( ) προσανατολίζεται γύρω από τις σαρώσεις στο καρέ. Στην αρχή προσαρτείται η επικεφαλίδα του καρέ και στη συνέχεια κωδικοποιούνται οι σαρώσεις. Καθορισμοί πινάκων και άλλα marker κομμάτια προηγούνται των $SOF_n$ markers (από τα αρχικά Start of Frame). Η διαδικασία τερματίζει όταν ολοκληρωθούν οι σαρώσεις.



## 2.2 Κωδικοποίηση σάρωσης

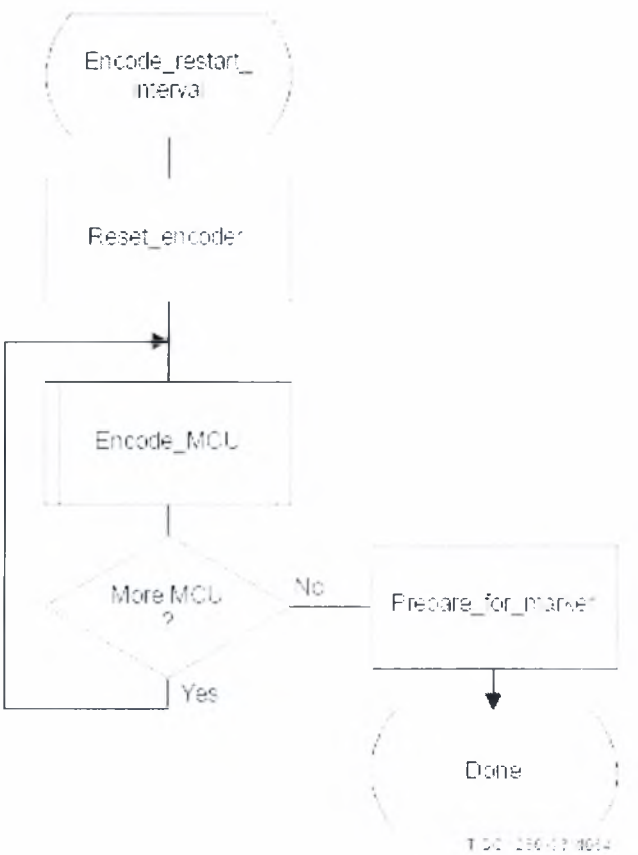
Μια σάρωση αποτελείται από μια απλή διαπέραση στα δεδομένα κάθε συστατικού εικόνας στη σάρωση. Προσδιορισμοί πινάκων και άλλα marker κομμάτια προηγούνται του SOS marker( ). Εάν περισσότερα από ένα συστατικά κωδικοποιούνται στη σάρωση, τα δεδομένα είναι interleaved. Εάν η παράμετρος που καθορίζει το restart interval είναι ενεργοποιημένη, τα δεδομένα τμηματοποιούνται μέσα στα restart intervals. Επίσης, ο $RST_m$ τοποθετούνται στα κωδικοποιημένα δεδομένα ανάμεσα από τα restart intervals. Εάν η παράμετρος που καθορίζει το restart interval είναι απενεργοποιημένη, η διαδικασία ελέγχου είναι ίδια εκτός από το ότι ολόκληρη η σάρωση περιέχει μοναδικό restart interval. Τα συμπιεσμένα δεδομένα εικόνας που δημιουργούνται από μια σάρωση πάντα ακολουθούνται από ένα marker, είτε τον EOI είτε τον marker από το επόμενο marker τμήμα.

Η δείχνει την διαδικασία ελέγχου για την κωδικοποίηση μιας σάρωσης. Ο βρόχος τερματίζεται όταν η διαδικασία κωδικοποίησης έχει κωδικοποιήσει τον αριθμό από τα restart intervals. Ο δείκτης m είναι ο μετρητής υπολοίπου του restart interval που απαιτείται για τον $RST_m$ marker.



## 2.3 Κωδικοποίηση restart interval

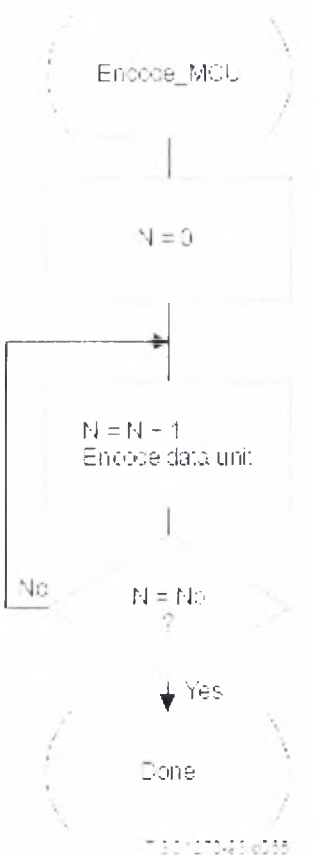
Στην εικόνα 4.4 παρουσιάζεται η διαδικασία ελέγχου για την κωδικοποίηση του restart interval. Ο βρόχος τερματίζεται είτε όταν η διαδικασία κωδικοποίησης έχει κωδικοποιήσει τον αριθμό των MCU είτε όταν έχει τελειώσει η σάρωση .




- Η διαδικασία ελέγχου "Reset_encoder" στον lossless τρόπο εκτέλεσης, επαναφέρει την πρόβλεψη στην προκαθορισμένη τιμή για όλα τα συστατικά στη σάρωση .
- Η διαδικασία "Prepare_for_marker" προσθέτει bits εάν χρειαστούν για να ολοκληρωθεί το τελευταίο byte.

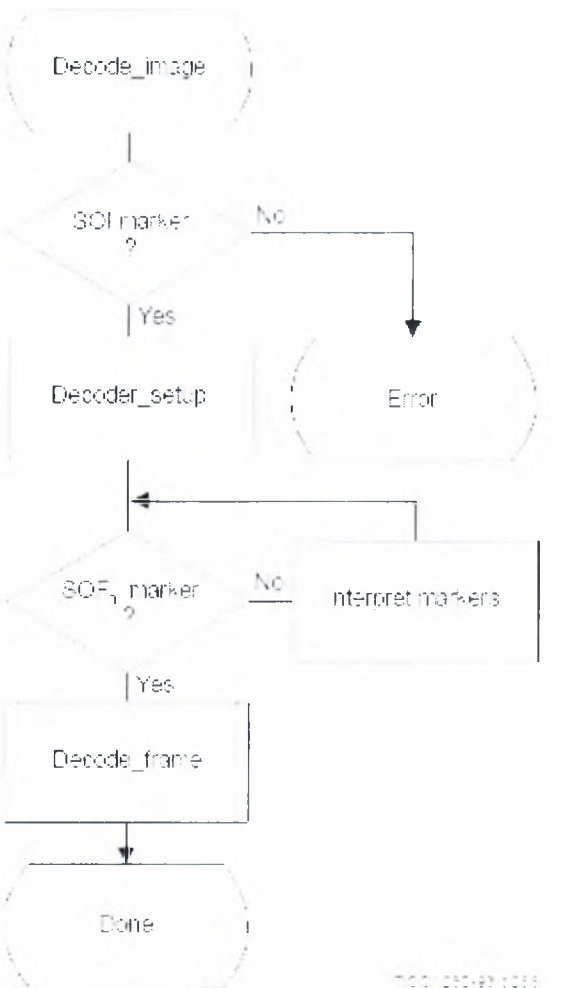
Μέσα στο δοθέν MCU, οι μονάδες πληροφορίας κωδικοποιούνται με την σειρά στην οποία βρίσκονται μέσα στο MCU ( ). Η διαδικασία ελέγχου κωδικοποίησης παρουσιάζεται στο παρακάτω σχήμα.

Το $N_b$ αναφέρεται στον αριθμό των μονάδων πληροφορίας που βρίσκονται μέσα στο MCU.



### 3. Αποκωδικοποίηση εικόνας

Η διαδικασία αποκωδικοποίησης βασίζεται στην αναγνώριση των διαφόρων markers. Ο πρώτος marker πρέπει να είναι ο SOI. Η διεργασία "Decoder_setup" επανεκκινεί το restart interval . Ο επόμενος marker είναι συνήθως ο $SOF_n$ . Εάν αυτός δεν βρεθεί τότε ένα από τα marker τμήματα που αναφέρονται στον έχουν ληφθεί. Όταν ο $SOF_n$ ανιχνευτεί, ακολουθεί η αποκωδικοποίηση ενός καρέ ( ).



| Marker | Purpose |
|---|---|
| DHT | Define Huffman Tables |
| DAC | Define Arithmetic Conditioning |
| DQT | Define Quantization Tables |
| DRI | Define Restart Interval |
| $APP_n$ | Application defined marker |
| COM | Comment |

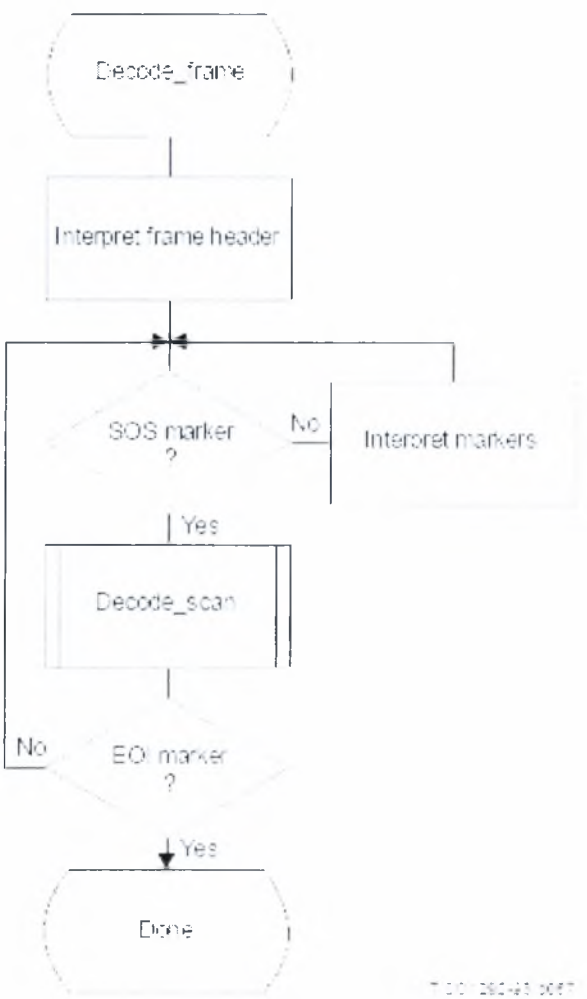
Η επιπρόσθετη λογική να μεταφράσεις αυτούς τους markers εμπεριέχεται στην διεργασία "Interpret markers".

- Ο DHT θα μεταφραστεί από τις διεργασίες που χρησιμοποιούν κωδικοποίηση Huffman.
- Ο DAC θα μεταφραστεί από τις διεργασίες που χρησιμοποιούν αριθμητική κωδικοποίηση.
- Ο DQT θα μεταφραστεί από αποκωδικοποιητές που στηρίζονται στον διακριτό μετασχηματισμό συνημίτονου.
- Ο DRI θα μεταφραστεί από όλους τους αποκωδικοποιητές. Εξ ορισμού, οι διαδικασίες του "interpret markers" αφήνουν στο σύστημα τον επόμενο marker.

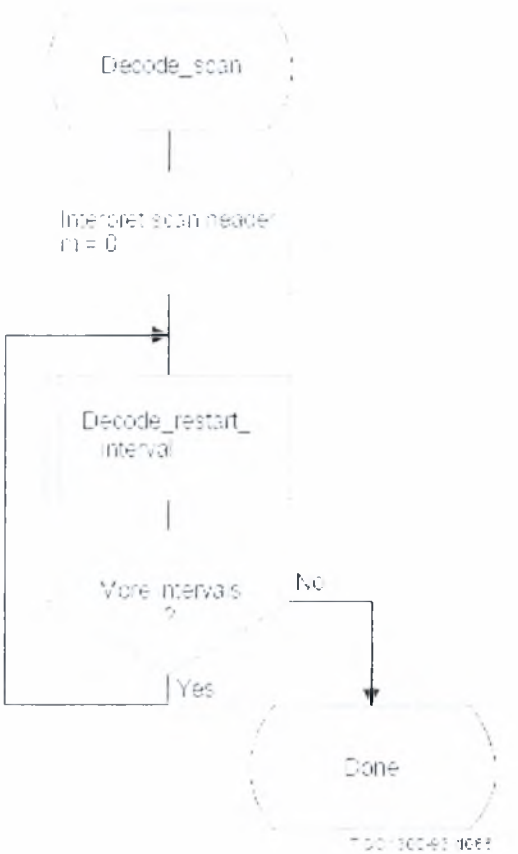
Αξίζει να σημειώσουμε ότι εάν ο marker SOI λείπει από την αρχή των κωδικοποιημένων δεδομένων, συνθήκη λάθους θα έχει ανιχνευτεί.

### 3.1 Αποκωδικοποίηση καρέ

Στην Εικόνα 4.7 απεικονίζεται η διαδικασία αποκωδικοποίησης ενός καρέ. Ο βρόχος τερματίζεται όταν ο marker EOI εντοπίζεται στο τέλος της σάρωσης. Όταν ο SOS marker ανιχνευτεί, ακολουθεί η αποκωδικοποίηση μιας σάρωσης.

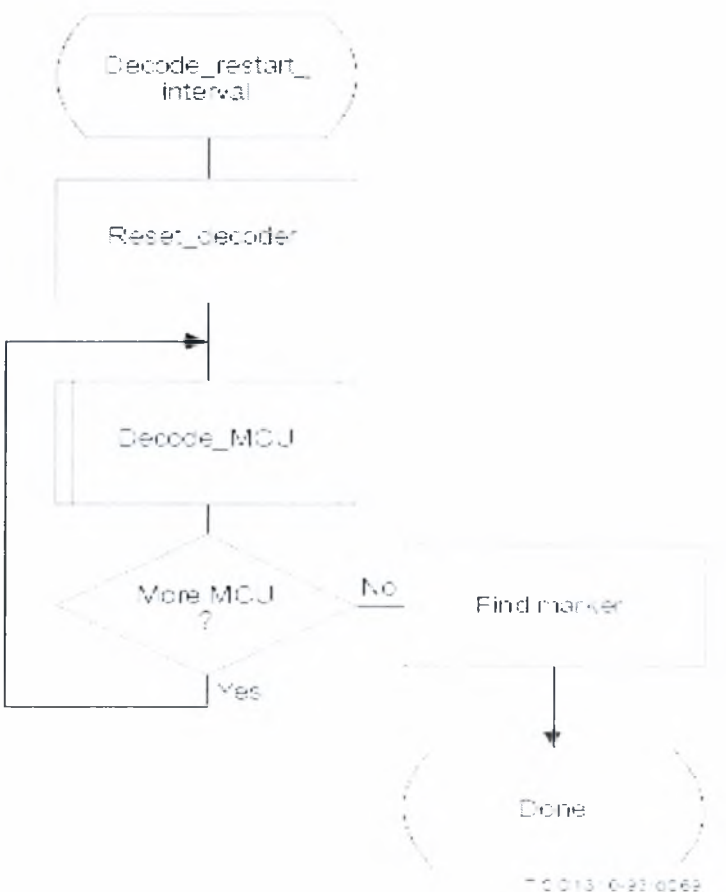


Η 4.8 δείχνει την διαδικασία αποκωδικοποίησης μιας σάρωσης. Ο βρόχος τερματίζει όταν ο απαιτούμενος αριθμός από τα restart intervals έχει αποκωδικοποιηθεί.



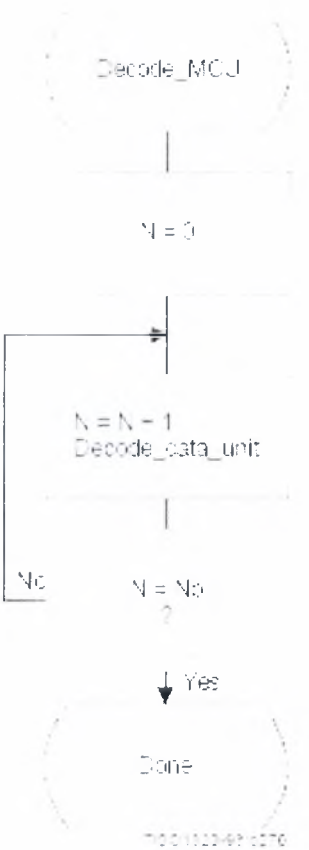
## 3.3 *Αποκωδικοποίηση restart interval*

Η διαδικασία αποκωδικοποίησης ενός restart interval παρουσιάζεται στην παρακάτω εικόνα ( ). Η διαδικασία "Reset_decoder" επαναφέρει την πρόβλεψη στην καθορισμένη τιμή για όλα τα συστατικά εικόνας μέσα στη σάρωση και αποκωδικοποιεί κάθε MCU . Στο τέλος του restart interval , ο επόμενος marker τοποθετείται. Εάν κάποιο λάθος ανιχνευτεί, διορθώνεται.



## 3.4 Αποκωδικοποίηση MCU

Η διαδικασία αποκωδικοποίησης ενός MCU παρουσιάζεται στην . Η παράμετρος $N_b$ περιέχει τον αριθμό των μονάδων πληροφορίας μέσα στο MCU.



# Κεφάλαιο 5ο

# Καθορισμός προσαρμοσμένων Huffman πινάκων

**Περιεχόμενα**

### *Εισαγωγή*

Η κωδικοποίηση Huffman χρησιμοποιείται στην κωδικοποίηση εντροπίας σε όλους τους τρόπους κωδικοποίησης.

Οι πίνακες Huffman καθορίζονται από μια λίστα 16 bytes(BITS) η οποία δίνει των αριθμό των κωδίκων για κάθε μέγεθος κώδικα από ένα έως δεκαέξι. Ακολουθείται από μια λίστα με 8 bit τιμές συμβόλων (HUFFVAL) κάθε μια από τις οποίες ανατίθενται σε κάθε κώδικα Huffman. Οι τιμές των συμβόλων τοποθετούνται στη λίστα για να αυξήσουν το μέγεθος των κωδίκων. Μεγέθη κωδίκων μεγαλύτερα από 16 bits δεν επιτρέπονται. Επιπρόσθετα, οι κώδικες που δημιουργούνται έτσι ώστε όλοι οι 1-bit κώδικες κάθε μεγέθους να είναι προθέματα για μεγαλύτερους κώδικες.

Παρακάτω παρουσιάζεται η διαδικασία κατά την οποία οι πίνακες Huffman παράγονται από δυο λίστες (BITS και HUFFVAL). Ο τόπος με τον οποίο αυτές οι δυο λίστες προσδιορίζονται δεν καθορίζεται. Όμως ακολουθούν τις βασικές αρχές της κωδικοποίησης Huffman.

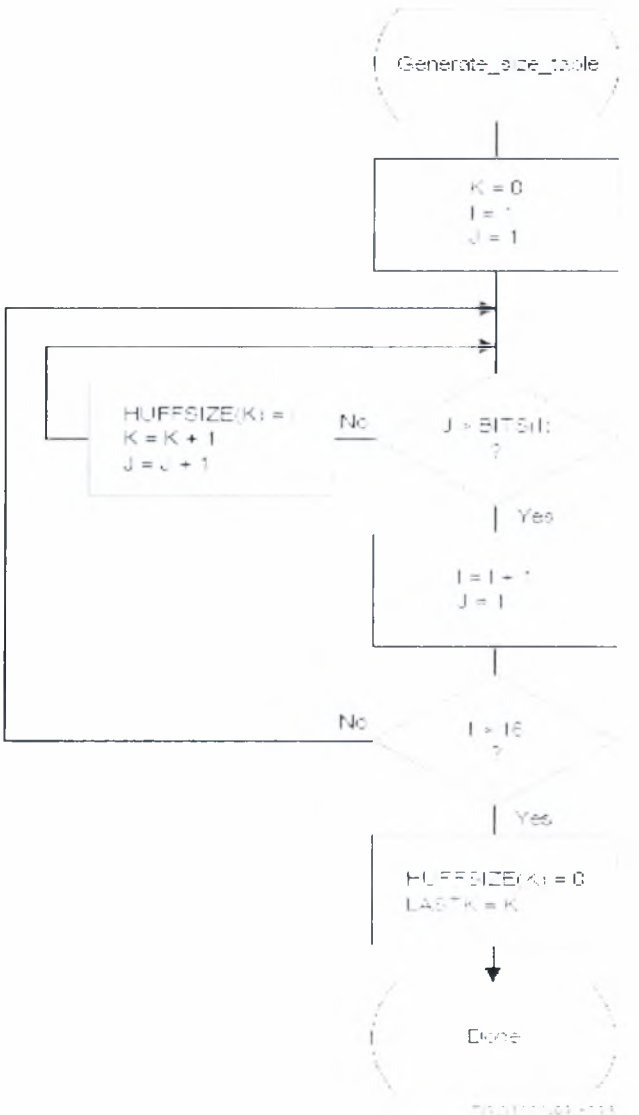
Ο DHT marker μέσα στα συμπιεσμένα δεδομένα υποδεικνύει την αρχή του ορισμού των πινάκων Huffman.

### *Παραγωγή Huffman πινάκων*

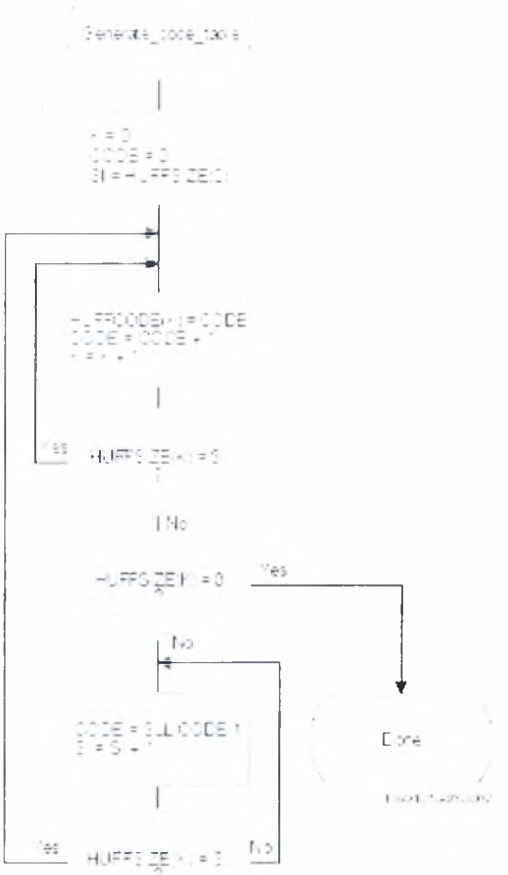
Για την μετατροπή σε πίνακες από κωδικούς και μεγέθη κωδικών χρησιμοποιούνται 3 διαδικασίες.

1. Η πρώτη δημιουργεί έναν πίνακα με μεγέθη κωδικών Huffman .Δηλαδή δημιουργείται ο πίνακας HUFFSIZE ( ).
2. Η δεύτερη παράγει τους κωδικούς από τον πίνακα που δημιουργήθηκε στο πρώτο βήμα. Δηλαδή δημιουργείται ο πίνακας HUFFCODE ( ).
3. Η Τρίτη δημιουργεί τους κωδικούς Huffman σε σειρά τιμών συμβόλου. Δηλαδή, δοθείσας μιας λίστας BITS (από 1 έως 16) η οποία περιέχει τους αριθμούς των

κωδίκων ανά μέγεθος και μια λίστα HUFFVAL η οποία περιέχει τις τιμές συμβόλων που σχετίζονται με τους κώδικες παραπάνω, δυο πίνακες δημιουργούνται( ). Ο πίνακας EHUFCO και ο EHUFSI.




- Η μεταβλητή lastk δείχνει την τελευταία είσοδο στον πίνακα.




- Το SLL CODE 1 σημαίνει αριστερή λογική ολίσθηση κατά ένα bit.

Οι δυο πίνακες HUFFCODE και HUFFSIZE έχουν δημιουργηθεί. Οι καταχωρήσεις στους πίνακες διατάσσονται σύμφωνα με αύξουσα αριθμητική τιμή και μέγεθος κώδικα Huffman.Οι πίνακες κωδικων της διαδικασίας κωδικοποίησης, EHUFCO και EHUFSI δημιουργούνται αναδιατάσσοντας τους κώδικες που έχουν καθοριστεί στους πίνακες HUFFCODE και HUFFSIZE σύμφωνα με τις τιμές συμβόλων που έχουν ανατεθεί σε κάθε κώδικα στον πίνακα HUFFVAL.

# Κεφάλαιο 6ο

## Τεχνικές βελτιστοποίησης Lossless JPEG κωδικοποιητή

### Περιεχόμενα

## 1. Εισαγωγή

Η μεταπτυχιακή αυτή διατριβή έχει ως στόχο να βελτιώσει την απόδοση του Lossless JPEG κωδικοποιητή. Δηλαδή, του κωδικοποιητή που δεν στηρίζεται στον διακριτό μετασχηματισμό συνημίτονου αλλά στην πρόβλεψη. Ο πηγαίος κώδικας πάνω στον οποίο βασίστηκε η διπλωματική βρίσκεται στο url: http://compression-links.info/JPEGLS?search_score=%3E%3D5 με το όνομα

## 2. Ανάλυση πηγαίου κώδικα

### 2.1 Δομή

Ο πηγαίος κώδικας αποτελείται από τα δέκα αρχεία γραμμένα σε γλώσσα προγραμματισμού C ( ) και από 6 αρχεία επικεφαλίδας( ).

| ***Αρχεία .c*** | ***περιγραφή*** |
|---|---|
| write.c | Ρουτίνες για εγγραφή των Jpeg αρχείων |
| util.c | Ρουτίνες που χρησιμοποιούνται στον κωδικ / αποκωδικ. |
| read.c | Ρουτίνες για διάβασμα Jpeg αρχείων |
| predictor.c | Ρουτίνες για υπολογισμό πρόβλεψης |
| pnmtoljpg.c | Κύρια ρουτίνα για τη διαδικασία κωδικοποίησης Jpeg |
| pmread.c | Ρουτίνες για διάβασμα εικόνων |
| mcu.c | Ρουτίνες για διαχείριση mcu |
| ljpgtopnm.c | Κύρια ρουτίνα για τη διαδικασία αποκωδικοποίησης Jpeg |
| huffc.c | Ρουτίνες για κωδικοποίηση Jpeg |
| huffd.c | Ρουτίνες για αποκωδικοποίηση Jpeg |

| ***Αρχεία .h*** | ***περιγραφή*** |
|---|---|
| proto.h | Δηλώσεις συναρτήσεων |
| predictor.h | Υπολογισμοί πρόβλεψης |
| pnmtoljpg.h | Καθολικές μεταβλητές |
| mcu.h | δηλώσεις |
| jpeg.h | Βασικές δομές |
| io.h | Εγγραφή αρχείων |

## 2.2 Λειτουργία

Τα αρχεία του πηγαίου κώδικα χρησιμοποιούνται σε δυο ξεχωριστές διαδικασίες. Ορισμένα από αυτά χρησιμοποιούνται στην διαδικασία κωδικοποίησης ενώ κάποια άλλα στην διαδικασία αποκωδικοποίησης. Κάποια χρησιμοποιούνται και στις δυο διαδικασίες.

### 2.2.1 Διαδικασία κωδικοποίησης

Η κωδικοποίηση ενός lossless JPEG κωδικοποιητή πραγματοποιείται σε πέντε βήματα.

1. Διάβασμα της εικόνας από το αρχείο εισόδου και αποθήκευση σε κάποιον buffer.
2. Δημιουργία είτε καθορισμένων Huffman πινάκων είτε προσαρμοσμένων ύστερα από επιλογή καλύτερου τρόπου πρόβλεψης σε όλο το αρχείο.
3. Εγγραφή επικεφαλίδας καρέ.
4. Εγγραφή επικεφαλίδας σάρωσης και άλλων marker κομματιών.
5. Κωδικοποίηση εικόνας και εγγραφή κωδικοποιημένης πληροφορίας στο νέο συμπιεσμένο αρχείο σύμφωνα με την επιλογή πρόβλεψης

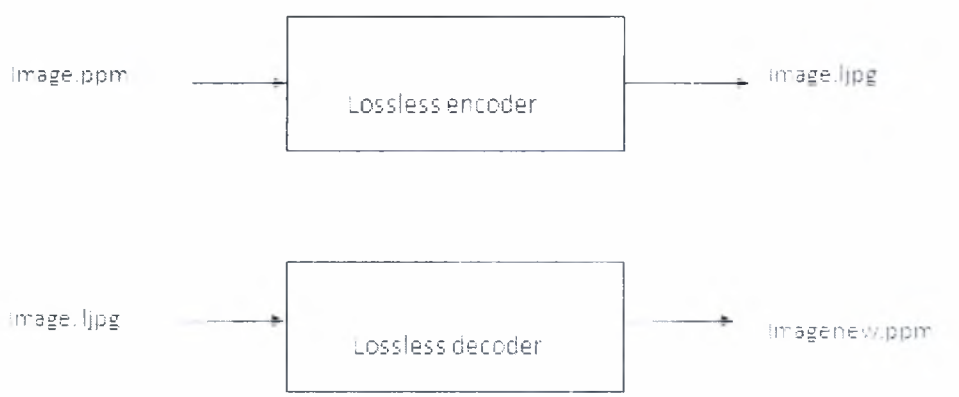
### 2.2.2 Διαδικασία αποκωδικοποίησης

Η αποκωδικοποίηση ενός lossless JPEG κωδικοποιητή πραγματοποιείται σε τέσσερα βήματα.

1. Διάβασμα επικεφαλίδας καρέ.
2. Διάβασμα επικεφαλίδας σάρωσης .
3. Εγγραφή επικεφαλίδας αναδημιουργημένης εικόνας.
4. Αποκωδικοποίηση και εγγραφή αποκωδικοποιημένης πληροφορίας στο νέο αρχείο

### 2.2.3 Αρχεία εισόδου και εξόδου

Το αρχείο εισόδου στην διαδικασία κωδικοποίησης θα είναι μια εικόνα με κατάληξη της μορφής .ppm και το αρχείο εξόδου θα είναι της μορφής .ljpg. Το αρχείο εισόδου της διαδικασίας αποκωδικοποίησης θα είναι το αρχείο .ljpg που προέκυψε από την διαδικασία κωδικοποίησης και το αρχείο εξόδου θα είναι μια νέα εικόνα της μορφής .ppm (εικόνα 6.1).




Οι εικόνες .ppm που χρησιμοποιήθηκαν σε αυτή την εργασία είναι οι :


### 2.2.4 *Εικόνες ppm(Portable PixMap)*

Το συγκεκριμένο format είναι ένας απλός τρόπος για την κωδικοποίηση μιας εικόνας. Αποτελείται από μια επικεφαλίδα και από το κύριο σώμα της εικόνας. Χρησιμοποιούμε το format για να αναπαραστήσουμε έγχρωμες εικόνες. Κάθε δείγμα αποτελείται από 3 byte (RGB) τα οποία αναπαρίστανται ως μη προσημασμένοι χαρακτήρες. Η τιμή (255,255,255 ) είναι το απόλυτο λευκό.

Η δομή της επικεφαλίδας ενός .ppm αρχείου είναι η ακόλουθη :

- αποτελείται από έναν μαγικό αριθμό –P6 αν η εικόνα έχει byte format δηλαδή ένα byte ανά συστατικό εικόνας ανά δείγμα είτε P3 αν έχει γραφτεί σαν κείμενο με ascii χαρακτήρες.
- Από σχόλια τα οποία αρχίζουν με '#'.
- Από έναν αριθμό που είναι το πλάτος της εικόνας.
- Από έναν αριθμό που είναι το ύψος της εικόνας.
- Από έναν αριθμό που είναι η μεγαλύτερη τιμή ενός δείγματος και υποδηλώνει την ακρίβεια του δείγματος.
- Από έναν χαρακτήρα whitespace. Συνήθως χαρακτήρας νέας γραμμής.

Το κύριο σώμα της εικόνας αποτελείται από μια ακολουθία από γραμμές, όσε και το ύψος της εικόνας. Κάθε γραμμή αποτελείται από δείγματα , όσα και το πλάτος της εικόνας. Κάθε δείγμα αποτελείται από 3 συστατικά εικόνας (R,G,B) με αυτή τη σειρά. Για παράδειγμα,

```
P3
# example from the man page
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

## 2.3 Επιλογές κατά την διαδικασία κωδικοποίησης

Κατά την διαδικασία κωδικοποίησης, έχουμε την ακόλουθες επιλογές.

1. Επιλογή καθορισμένων πινάκων Huffman (-d)
2. Επιλογή παραμέτρου point transform (- p)
3. Επιλογή προσαρμοσμένων πινάκων Huffman
4. Επιλογή restart interval (-r)

Οι πρώτες τρείς επιλογές επηρεάζουν το μέγεθος του συμπιεσμένου αρχείου ενώ η τέταρτη όχι.

## 2.4 Εργαλεία που χρησιμοποιήθηκαν

Για την ολοκλήρωση της εργασίας χρησιμοποιήθηκαν ορισμένα λογισμικά σχετικά με την ανάπτυξη πηγαίου κώδικα, image viewers , λογισμικά για την σύγκριση δυαδικών αρχείων καθώς και batch files για την αυτοματοποίηση της διαδικασίας. Συγκεκριμένα

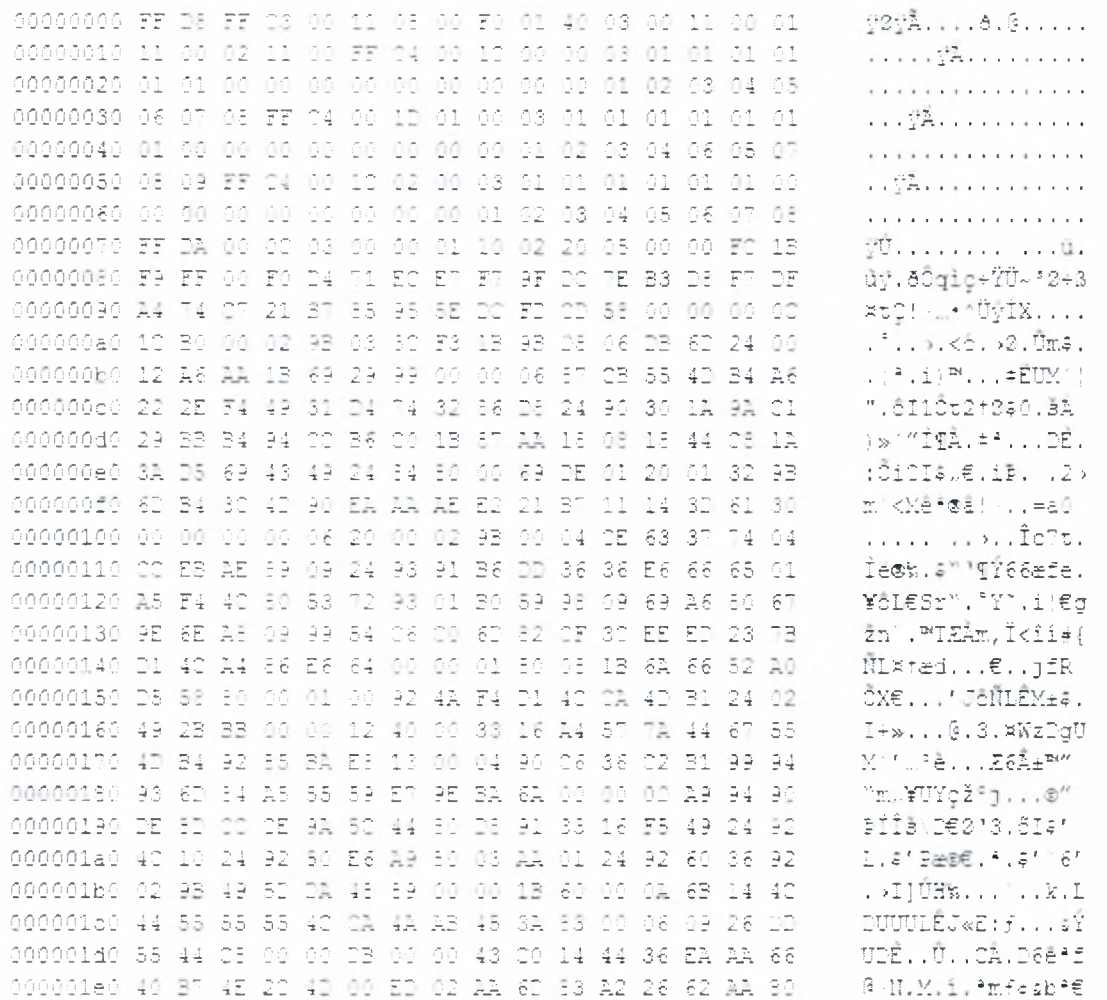
- για την ανάπτυξη πηγαίου κώδικα χρησιμοποιήθηκε το «Microsoft Visual Studio 2005»
- Image viewer ο «IrfanView».
- για την σύγκριση δυαδικών αρχείων το λογισμικό «Ultra Compare Professional – Binary Compare»

## 2.5 Παράδειγμα κωδικοποίησης και αποκωδικοποίησης

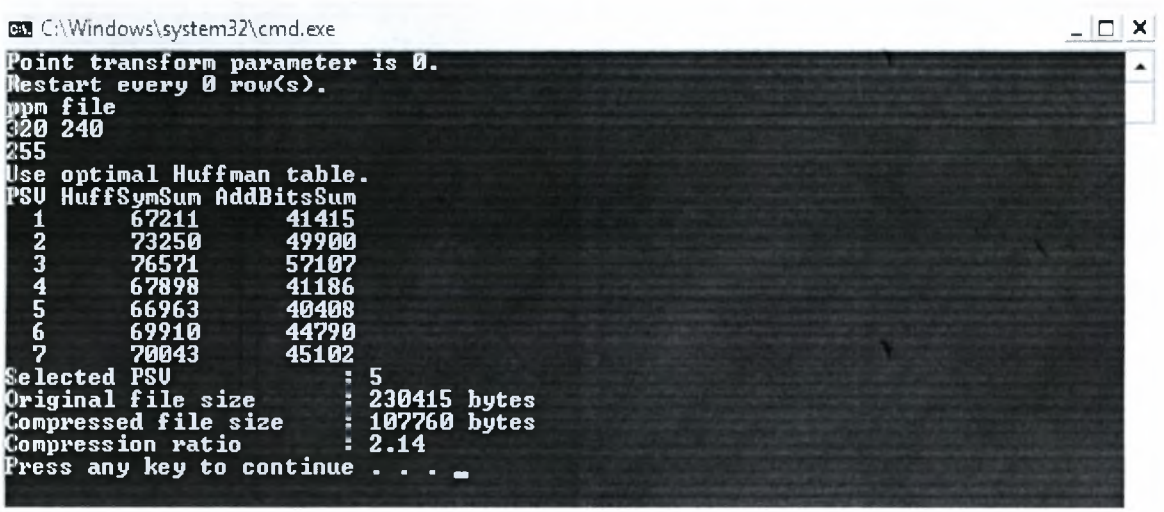
Στην φαίνεται ένα μέρος της δομής της εικόνας F-18.ppm .


Η εικόνα F-18.ppm είναι το αρχείο εισόδου στον κωδικοποιητή. Το συμπιεσμένο αρχείο που παράγεται έχει όνομα F-18new.ljpg με επιλογές στην κωδικοποίηση την επιλογή προσαρμοσμένων πινάκων Huffman. Στην παρακάτω εικόνα παρουσιάζεται ένα μέρος της συμπιεσμένης εικόνας ( ).

D:\DIPLOMATIKI\REFERENCE_CODE\standardencoder\images\F-18new.ljpg

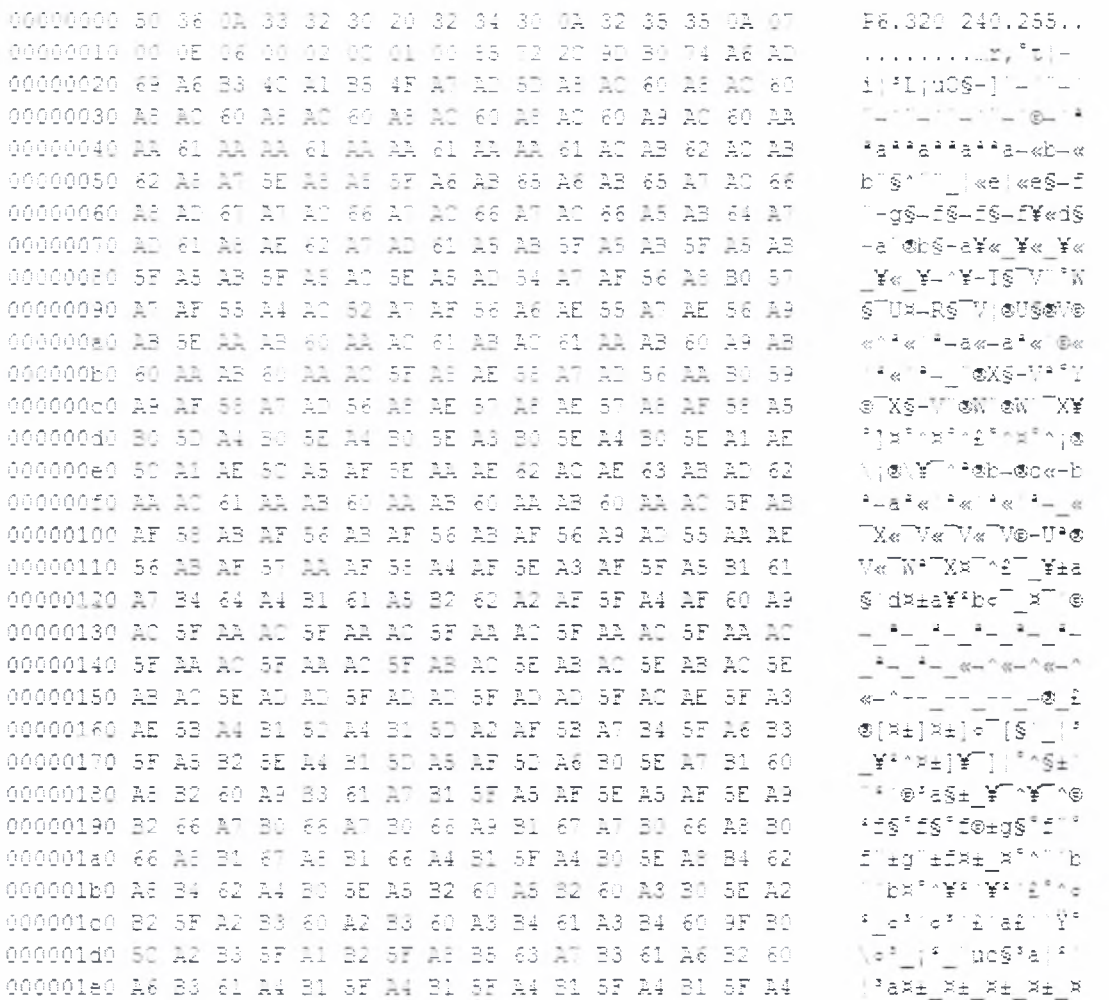


Μόλις ολοκληρώθηκε η διαδικασία κωδικοποίησης. Τα αποτελέσματα παρουσιάζονται στην

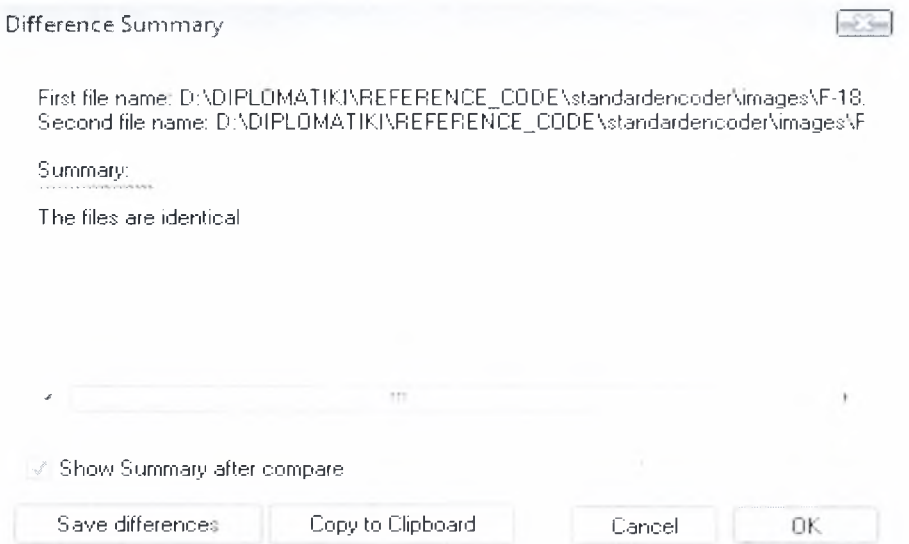



Κατά την αποκωδικοποίηση, ο lossless αποκωδικοποιητής θα δεχθεί ως αρχείο εισόδου το F-18new.ljpg και θα παράγει ένα νέο αρχείο F-18new.ppm. Ένα μέρος της δομής του φαίνεται στην παρακάτω εικόνα ( ).

D:\DIPLOMATIKI\REFERENCE_CODE\standardencoder\images\F-18new.ppm

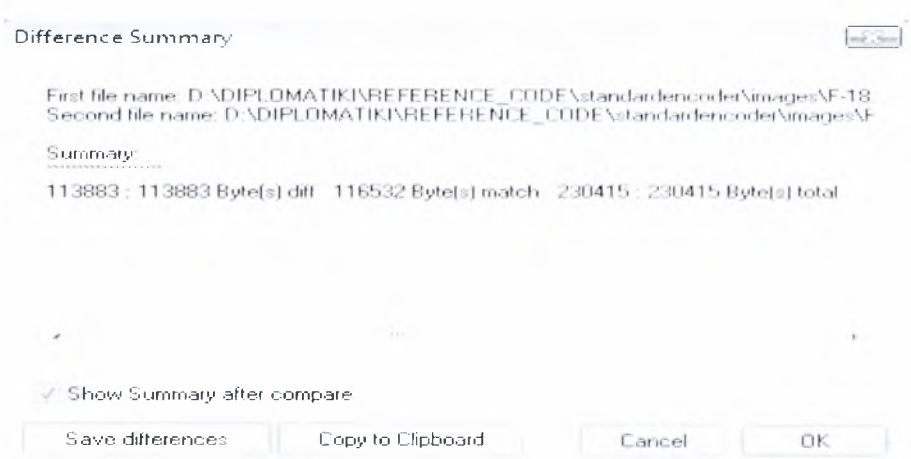


Η φαίνεται αρκετά ίδια με την . Το λογισμικό UltraCompare το επιβεβαιώνει κατά την σύγκριση( )




Αυτό που πετύχαμε είναι να επιβεβαιώσουμε ότι όντως ο κωδικοποιητής και αποκωδικοποιητής είναι lossless, δηλαδή δεν έχουμε απώλεια πληροφορίας, το οποίο είναι και το ζητούμενο.

Εάν επιλέξουμε την παράμετρο point transform και επιλέξουμε να ελέγξουμε εάν όντως έχουμε κωδικοποίηση και αποκωδικοποίηση χωρίς απώλειες δεδομένων θα διαπιστώσουμε ότι κατά την σύγκριση τα αρχεία δεν είναι πανομοιότυπα ( ).



Αυτό οφείλεται στο γεγονός ότι όταν η παράμετρος point transform είναι ενεργοποιημένη έχουμε μια μεταβολή στην ποσότητα των δειγμάτων. Παρόλα αυτά αν αποδείξουμε ότι η διαφορά κάθε byte μεταξύ της αρχικής εικόνας .ppm και της νέας εικόνας που δημιουργείται είναι μικρότερη από $2^{Pt}$ -1 , τότε έχουμε μεταβολή στην τιμή ενός δείγματος στα λιγότερα σημαντικά bits χωρίς όμως να επηρεάζεται ολοκληρωτικά η τιμή και κατ' επέκταση το χρώμα της. Για παράδειγμα,

Έστω ο αριθμός δέκα ο οποίος γράφεται με 1010 και $P_t$=3 . Τότε $2^{Pt}$ -1=7.

(10) 1010 << 3 => 0001 >> 3 => 1000 (8)

Η διαφορά 10-8=2 <7 άρα είμαστε σωστοί. Όσο η διαφορά πλησιάζει στην επιτρεπόμενη τιμή τόσο χάνεται πληροφορία ( ) αλλά και τόσο μειώνεται η ποσότητα της συμπιεσμένης πληροφορίας

F-18.ppm

F-18newp1.ppm

F-18newp4.ppm

F-18newp6.ppm

Ο παρακάτω κώδικας εκτελεί τον έλεγχο.

```
if(formata==formatb)
{
    if (rowsa==rowsb && colsa==colsb)
    {
        if (*valuea==*valueb)
        {
            maxval=*valuea=*valueb;
            do{
                cha=getc(infile1);
                chb=getc(infile2);
                if(cha-chb>maxval)
                {
                    printf("mistake\n");
                    exit(-1);
                }
            }while(cha!=EOF || chb!=EOF);
        }
        else
        {
            exit(-1);
        }
    }
    else
    {
        exit(-1);
    }
}
```

Τα αποτελέσματα στον απεικονίζουν το μέγεθος του συμπιεσμένου αρχείου για τις διάφορες επιλογές κωδικοποίησης.

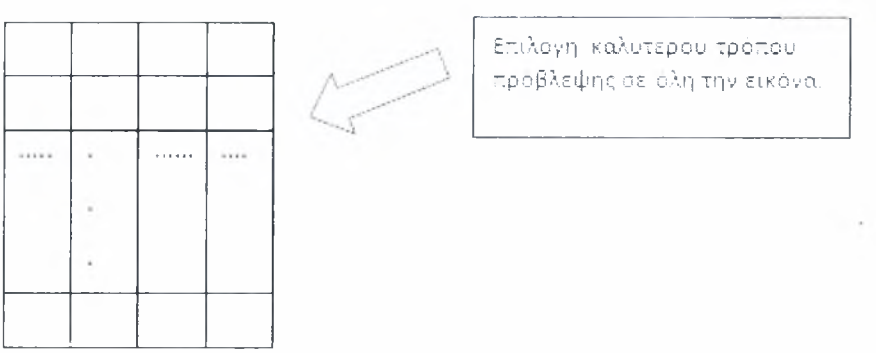
| | STANDARDENCODER | STANDARDENCODER |
|---|---|---|
| **IMAGES** | **F-18.ppm** | **pepers.ppm** |
| | SIZE(bytes) | SIZE(bytes) |
| **ORIGINAL** | 230415 | 786447 |
| **DEFAULT HUFFMAN TABLES WITHOUT HEADERS** | 118256 | 490820 |
| **DEFAULT HUFFMAN TABLES WITH HEADERS *** | 118324 | 490888 |
| **ORIGINAL** | 230415 | 786447 |
| **CUSTOMIZED HUFFMAN TABLES WITHOUT HEADERS** | 107634 | 476222 |
| **CUSTOMIZED HUFFMAN TABLES WITH HEADERS *** | 107760 | 476347 |
| **ORIGINAL** | 230415 | 786447 |
| **P1** | 83067 | 405197 |
| **P2** | 61849 | 332655 |
| **P4** | 40138 | 189275 |

## 3. Τεχνικές βελτιστοποίησης

Όπως περιγράφτηκε και στην παράγραφο 2.2.1 του παρόντος κεφαλαίου, η διαδικασία κωδικοποίησης ενός lossless JPEG κωδικοποιητή πραγματοποιείται σε πέντε βήματα.

1. Διάβασμα της εικόνας από το αρχείο εισόδου και αποθήκευση σε κάποιον buffer.
2. Δημιουργία είτε καθορισμένων Huffman πινάκων είτε προσαρμοσμένων ύστερα από επιλογή καλύτερου τρόπου πρόβλεψης σε όλο το αρχείο.
3. Εγγραφή επικεφαλίδας καρέ.
4. Εγγραφή επικεφαλίδας σάρωσης και άλλων marker κομματιών.
5. Κωδικοποίηση εικόνας και εγγραφή κωδικοποιημένης πληροφορίας στο νέο συμπιεσμένο αρχείο σύμφωνα με την επιλογή πρόβλεψης

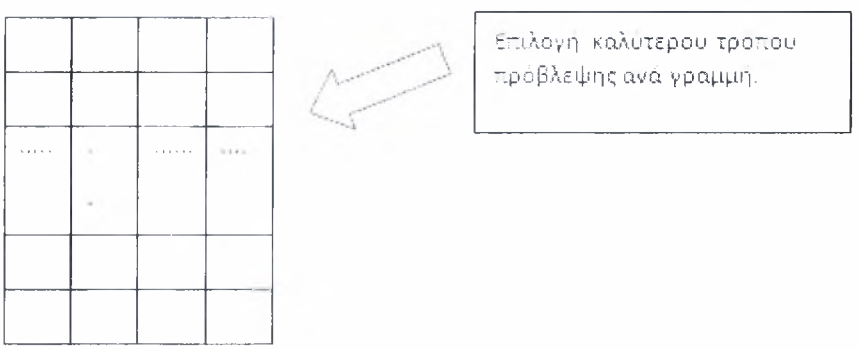
Για την επιλογή του καλύτερου τρόπου πρόβλεψης, ακολουθείται η εξής διαδικασία: Αρχικά βρίσκονται οι συχνότητες των συμβόλων προς κωδικοποίηση σε όλη την εικόνα. Στη συνέχεια, για κάθε τρόπο πρόβλεψης από τους εφτά δημιουργούνται οι πίνακες Huffman. Καλύτερος τρόπος πρόβλεψης θεωρείται αυτός που χρειάζεται την λιγότερη πληροφορία για την αναπαράσταση των συμβόλων ( ). Τέλος η κωδικοποίηση γίνεται με τους πίνακες Huffman που παράχθηκαν σύμφωνα με τον επιλεγμένο τρόπο πρόβλεψης για όλο το αρχείο.



## 3.1 Βελτιστοποίηση 1η (newecoder1.0)

*Περιγραφή:*

Η πρώτη βελτιστοποίηση που πραγματοποιήθηκε στην διατριβή αυτή είναι η κωδικοποίηση ανά γραμμή και όχι κωδικοποίηση σε όλο το αρχείο. Για να γίνει κωδικοποίηση ανά γραμμή , πρέπει να βρεθεί ο καλύτερος τρόπος πρόβλεψης για κάθε γραμμή ( ), να παραχθούν οι πίνακες Huffman για κάθε γραμμή και η κωδικοποίηση για κάθε γραμμής να γίνει σύμφωνα με τον επιλεγμένο τρόπο κωδικοποίηση και το αντίστοιχο πίνακα Huffman.




*Συναρτήσεις:*

Για τον σκοπό αυτό γράφτηκαν οι παρακάτω συναρτήσεις :

1.void HuffOptimize_line (CompressInfo *cPtr,int k);

// CompressInfo *cPtr : pointer to line CompressInfo

// int k: row number

Η συνάρτηση αυτή βρίσκει την καλύτερη πρόβλεψη για κάθε γραμμή και δημιουργεί τους κατάλληλους προσαρμοσμένους πίνακες Huffman για κάθε γραμμή.

2.void FreqCountAllSelValue_first_row(CompressInfo *cPtr)

//CompressInfo *cPtr : pointer to line CompressInfo

Η συνάρτηση αυτή βρίσκει την συχνότητα των συμβόλων μόνο της πτώτης γραμμής.

3.void FreqCountAllSelValue_each_row(CompressInfo *cPtr,int k)

// CompressInfo *cPtr : pointer to line CompressInfo

// int k:row number

Η συνάρτηση αυτή βρίσκει την συχνότητα των συμβόλων μόνο κάθε γραμμής εκτός από την πρώτη.

4. void StdPickSelValue_rows (CompressInfo *cPtr);

//CompressInfo *cPtr : pointer to line CompressInfo

Η συνάρτηση αυτή επιλέγει την καλύτερη πρόβλεψη για τους καθορισμένους πίνακες Huffman

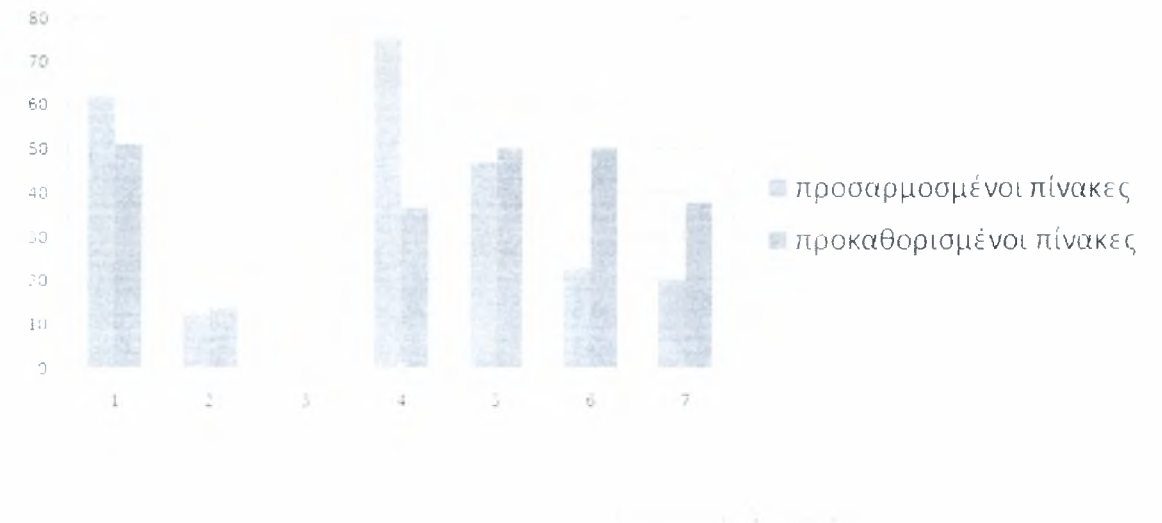
```
5.void HuffEncode_each_row(CompressInfo * cPtr, MCU *curRowBuf, MCU
*prevRowBuf)

//    CompressInfo * cPtr: pointer to line CompressInfo
//    MCU *curRowBuf:pointer to line
//    MCU *prevRowBuf:pointer to previous line
```

Η συνάρτηση αυτή εκτελεί την κωδικοποίηση κάθε γραμμής.

*Αποτελέσματα:*

Η συχνότητα της κάθε πρόβλεψης για τους προσαρμοσμένους πίνακες Huffman και για τους καθορισμένους αναπαρίσταται στο παρακάτω ιστόγραμμα .




Τα αποτελέσματα της κωδικοποίησης ανά γραμμή παρουσιάζονται στον πίνακα 6.4.

| | STANDARDENCODER | NEWENCODER1.0 | |
|---|---|---|---|
| **IMAGES** | **F-18.ppm** | **F-18.ppm** | **%** |
| | SIZE(bytes) | SIZE(bytes) | |
| ORIGINAL | 230415 | 230415 | |
| DEFAULT HUFFMAN TABLES WITHOUT HEADERS | 118256 | 114080 | -3.5 |
| DEFAULT HUFFMAN TABLES WITH HEADERS * | 118324 | 122274 | 3.3 |
| ORIGINAL | 230415 | 230415 | |
| CUSTOMIZED HUFFMAN TABLES WITHOUT HEADERS | 107634 | 98053 | -8.9 |
| CUSTOMIZED HUFFMAN TABLES WITH HEADERS * | 107760 | 117976 | 9.4 |
| ORIGINAL | 230415 | 230415 | |
| P1 | 83067 | 95713 | 15.2 |
| P2 | 61849 | 77062 | 24.5 |
| P4 | 40138 | 56060 | 39.6 |

| | STANDARDENCODER | NEWENCODER1.0 | |
|---|---|---|---|
| **IMAGES** | **pepers.ppm** | **pepers.ppm** | **%** |
| | SIZE(bytes) | SIZE(bytes) | |
| **ORIGINAL** | 786447 | 786447 | |
| **DEFAULT HUFFMAN TABLES WITHOUT HEADERS** | 490820 | 480029 | -2.1 |
| **DEFAULT HUFFMAN TABLES WITH HEADERS *** | 490888 | 497471 | 1.34 |
| **ORIGINAL** | 786447 | 786447 | |
| **CUSTOMIZED HUFFMAN TABLES WITHOUT HEADERS** | 476222 | 458798 | -3.6 |
| **CUSTOMIZED HUFFMAN TABLES WITH HEADERS *** | 476347 | 503945 | 5.7 |
| **ORIGINAL** | 786447 | 786447 | |
| **P1** | 405197 | 434953 | 7.34 |
| **P2** | 332655 | 363530 | 9.28 |
| **P4** | 189275 | 224288 | 18.4 |

*Συμπεράσματα:*

1. Με αυτή τη βελτιστοποίηση πετυχαίνουμε καλύτερη συμπίεση δεδομένων, χωρίς όμως την εγγραφή των αρχείων επικεφαλίδας . Όταν όμως εγγράψουμε και τα αρχεία επικεφαλίδας τότε η χωρητικότητα του συμπιεσμένου αρχείου αυξάνεται. Αυτό εξηγείται ως εξής : Χωρίς την εγγραφή των αρχείων επικεφαλίδας, έχουμε λιγότερη χωρητικότητα διότι κάθε γραμμή έχει κωδικοποιηθεί με ξεχωριστό πίνακα Huffman. Αυτό πρακτικά σημαίνει ότι τα σύμβολα προς κωδικοποίηση θα έχουν μικρότερο μήκος αφού η συχνότητα των συμβόλων είναι μικρότερη. Όταν όμως εγγράφουμε και τα αρχεία επικεφαλίδας, πρέπει να εγγράψουμε όλους τους πίνακες Huffman κάθε γραμμής. Στην προκειμένη περίπτωση, εγγράφουμε στο συμπιεσμένο αρχείο 240 πίνακες.
2. Ένα άλλο αρνητικό στοιχείο είναι ότι χρειαζόμαστε αρκετούς κύκλους μηχανής. Για κάθε γραμμή, ελέγχουμε και τους εφτά τρόπους πρόβλεψης και δημιουργούνται πίνακες Huffman για κάθε πρόβλεψη μέχρι να αποφασίσουμε ποιά πρόβλεψη είναι καλύτερη.

3. Τέλος, παρατηρούμε ότι με την ενεργοποίηση του point transform parameter, μειώνεται η χωρητικότητα του αρχείου.

## 3.2 Βελτιστοποίηση 2ⁿ (newencoder2.0)

*Περιγραφή:*

Η δεύτερη βελτιστοποίηση την οποία προτείνουμε προκύπτει από το δεύτερο συμπέρασμα της πρώτης βελτιστοποίησης. Σύμφωνα λοιπόν με αυτό το συμπέρασμα, ο αρνητικός παράγοντας είναι ότι ενώ η καθαρή πληροφορία της εικόνας είναι μικρότερη από την αρχική, αναγκαζόμαστε να γράψουμε τους πίνακες Huffman για κάθε γραμμή με αποτέλεσμα η χωρητικότητα του τελικού συμπιεσμένου αρχείου να αυξάνεται σημαντικά. Αυτό που προτείνουμε είναι κωδικοποίηση ανά γραμμή αλλά συνολικά θα έχουμε 7 διαφορετικούς πίνακες Huffman, έναν για κάθε τρόπο πρόβλεψης. Για να γίνει αυτό, κάθε φορά που θα βρίσκουμε την καλύτερη πρόβλεψη ανά γραμμή θα προσθέτουμε τα ιστογράμματα. Στο τέλος, ο extended πίνακας με τα σύμβολα για την πρόβλεψη 1 για παράδειγμα, θα περιέχει όλα τα ιστογράμματα των γραμμών που κωδικοποιήθηκαν με την πρόβλεψη 1. Τέλος, θα δημιουργηθεί ο πίνακας Huffman με βάση τα αθροισμένα ιστογράμματα.

Ψευδοκώδικας για υπολογισμό Extended Huffman Table :

```
For (i=0; i<imagelines; i++){

Psv=Find_line_prediction ();

Add_frequencies (psv);

}
```

*Συναρτήσεις:*

Για το σκοπό αυτό, χρησιμοποιήθηκαν οι συναρτήσεις της προηγούμενης βελτιστοποίησης και ακόμη:

1.`void FreqCountSumInit (void);`

Η συνάρτηση αυτή αρχικοποιεί τον πίνακα με τις αθροισμένα ιστογράμματα

2. `void GenExtendedHuffTable ( CompressInfo *cPtr,int k);`

// CompressInfo *cPtr : pointer to line CompressInfo

// int k:row number

Η συνάρτηση αυτή δημιουργεί τους νέους extended Huffman πίνακες.

*Αποτελέσματα*:

Τα αποτελέσματα της κωδικοποίησης ανά γραμμή και δημιουργίας Extended Huffman Tables παρουσιάζονται στον πίνακα 6.5.

| | STANDARDENCODER | NEWENCODER1.0 | | NEWENCODER2.0 | |
|---|---|---|---|---|---|
| **IMAGES** | **F-18.ppm** | **F-18.ppm** | **%** | **F-18.ppm** | **%** |
| | SIZE(bytes) | SIZE(bytes) | | SIZE(bytes) | |
| **ORIGINAL** | 230415 | 230415 | | 230415 | |
| **DEFAULT HUFFMAN TABLES WITHOUT HEADERS** | 118256 | 114080 | -3.5 | 101231 | -14.3 |
| **DEFAULT HUFFMAN TABLES WITH HEADERS *** | 118324 | 122274 | 3.3 | 101682 | -14.06 |
| **ORIGINAL** | 230415 | 230415 | | 230415 | |
| **CUSTOMIZED HUFFMAN TABLES WITHOUT HEADERS** | 107634 | 98053 | -8.9 | 100818 | -6.3 |
| **CUSTOMIZED HUFFMAN TABLES WITH HEADERS *** | 107760 | 117976 | 9.4 | 101621 | -5.69 |
| **ORIGINAL** | 230415 | 230415 | | 230415 | |
| **P1** | 83067 | 95713 | 15.2 | 79010 | -4.8 |
| **P2** | 61849 | 77062 | 24.5 | 59638 | -3.57 |
| **P4** | 40138 | 56060 | 39.6 | 39432 | -1.75 |

| | STANDARDENCODER | NEWENCODER1.0 | | NEWENCODER2.0 | |
|---|---|---|---|---|---|
| **IMAGES** | **pepers.ppm** | **pepers.ppm** | **%** | **pepers.ppm** | **%** |
| | SIZE(bytes) | SIZE(bytes) | | SIZE(bytes) | |
| ORIGINAL | 786447 | 786447 | | 786447 | |
| DEFAULT HUFFMAN TABLES WITHOUT HEADERS | 490820 | 480029 | -2.1 | 467563 | -4.7 |
| DEFAULT HUFFMAN TABLES WITH HEADERS * | 490888 | 497471 | 1.34 | 468199 | -4.62 |
| ORIGINAL | 786447 | 786447 | | 786447 | |
| CUSTOMIZED HUFFMAN TABLES WITHOUT HEADERS | 476222 | 458798 | -3.6 | 463519 | -2.66 |
| CUSTOMIZED HUFFMAN TABLES WITH HEADERS * | 476347 | 503945 | 5.7 | 464335 | -2.52 |
| ORIGINAL | 786447 | 786447 | | 786447 | |
| P1 | 405197 | 434953 | 7.34 | 394662 | -2.5 |
| P2 | 332655 | 363530 | 9.28 | 324227 | -2.5 |
| P4 | 189275 | 224288 | 18.4 | 186490 | -1.4 |

*Συμπεράσματα:*

1. Όσον αφορά το μέγεθος του αρχείου, παρατηρούμε ότι πετυχαίνουμε καλύτερα αποτελέσματα σε όλες τις περιπτώσεις. Αυτό οφείλεται στο γεγονός ότι η κωδικοποίηση γίνεται με 7 πίνακες Huffman. Αυτό συνεπάγεται ότι τα σύμβολα κωδικοποιούνται με λιγότερα bits σε σχέση με τον standardencoder ενώ γράφονται μονό 7 πίνακες στα αρχεία επικεφαλίδας.
2. Παρόλο που κερδίζουμε σε χωρητικότητα, χάνουμε σε κύκλους μηχανής διότι ξανά για να επιλέξουμε την καλύτερη πρόβλεψη ανά γραμμή, δοκιμάζουμε όλους τους τρόπους πρόβλεψης και δημιουργούμε πίνακες Huffman για να επιλέξουμε τον καλύτερο.
3. Τέλος, παρατηρούμε ότι με την ενεργοποίηση του point transform parameter, μειώνεται η χωρητικότητα του αρχείου.

### 3.3 Βελτιστοποίηση 3η (newencoder3.0)

*Περιγραφή:*

Στις προηγούμενες βελτιστοποιήσεις παρατηρούμε ότι πάντα στα μειονεκτήματα συγκαταλέγεται οι πολλοί κύκλοι μηχανής. Ο λόγος είναι ότι για να επιλέξουμε κάθε φορά την καλύτερη πρόβλεψη ανά γραμμή, πρέπει να δημιουργήσουμε για κάθε πρόβλεψη, πίνακες Huffman και να δούμε σε ποία περίπτωση χρειαζόμαστε λιγότερα bits για την αναπαράσταση των συμβόλων. Στην δεύτερη βελτιστοποίηση δε, εφόσον βρούμε και την καλύτερη πρόβλεψη ανά γραμμή, φτιάχνουμε εκ νέου 7 νέους πίνακες Huffman, έναν για κάθε πρόβλεψη αφού πρώτα έχουμε προσθέσει τα ιστογράμματα με τα σύμβολα για εκείνες τις γραμμές που έχουν την ίδια πρόβλεψη. Το ερώτημα είναι , μπορούμε να βρούμε έναν τρόπο πρόβλεψης της γραμμής έτσι ώστε να μειώνονται οι κύκλοι μηχανής και να πετυχαίνουμε καλύτερο ή τουλάχιστον ίδιο αποτέλεσμα;

Η απάντηση είναι ναι. Ο τρόπος προέρχεται από την αναγνώριση προτύπων (περιεχόμενο Α). Με λίγα λόγια, η αναγνώριση προτύπων είναι η περιγραφή και κατάταξη των αντικείμενων σε διάφορες κατηγορίες. Τα υπό κατάταξη αντικείμενα καλούνται πρότυπα.

Η αναγνώριση προτύπων και η εύρεση τρόπου πρόβλεψης της κάθε γραμμής σχετίζονται ως εξής: Αρχικά βρίσκουμε ορισμένα στατιστικά στοιχεία για κάθε γραμμή όπως ο μέσος όρος, η διασπορά και ο οριζόντιος, κάθετος και διαγώνιος συσχετισμός. Όποιος από τους τρείς συσχετισμούς είναι μεγαλύτερος, τότε η πρόβλεψη της γραμμής θα γίνει είτε οριζόντια (τρόπος πρόβλεψης 1) εάν ο οριζόντιος είναι μεγαλύτερος από τους άλλους δυο είτε κάθετα (τρόπος πρόβλεψης 2) εάν ο κάθετος συσχετισμός είναι μεγαλύτερος από τους άλλους είτε διαγώνια εάν ο διαγώνιος είναι μεγαλύτερος από τους άλλους 2.

*Οριζόντιος συσχετισμός:*

$$cor_{horizontal} = \frac{\sum_{i=0}^{n-2} x[i,j] * x[i,j+1] - mean[i]^2}{(n-1) * sigma[i]^2}$$

*Κάθετος συσχετισμός :*

$$\mathrm{cor}_{\mathrm{vertical}} = \frac{\sum_{i=0}^{n-1} x[i,j] * x[i+1,j] - mean[i] * mean[i+1]}{n * sigma[i] * sigma[i+1]}$$

*Οριζόντιος συσχετισμός:*

$$\mathrm{cor}_{\mathrm{diagonal}} = \frac{\sum_{i=0}^{n-2} x[i,j] * x[i+1,j+1] - mean[i] * mean[i+1]}{(n-1) * sigma[i] * sigma[i+1]}$$

*Sigma*:

$$sigma = \sqrt[2]{variation}$$

*Variation*:

$$variation = \frac{\sum_{i=0}^{n-1} sample^2}{n} - mean[i]^2$$

*Mean*:

$$variation = \frac{\sum_{i=0}^{n-1} sample}{n}$$

*Συναρτήσεις:*

Για το σκοπό αυτό, χρησιμοποίήθηκαν οι συναρτήσεις της προηγούμενης βελτιστοποίησης και ακόμη:

```
typedef struct statistical{

      double *sum;//αθροισμα δειγματων
      double *sumpower2;//αθροισμα τετραγωνων δειγματων
      double *mean;//μεσος ορος
      double *sigmapower2;//διασπορα στο τετραγωνο
      double *sigma;//ριζα διασπορας
      double *cor_h;//οριζοντιος συσχετισμος
      double *cor_v;//καθετος συσχετισμος
      double *cor_diag;//διαγωνιος συσχετισμος
}
```

1.`void find_sums(StatisticalInfo *statcPtr,int k,int columns,int components)`

// StatisticalInfo *statcPtra :pointer to line StatInfo

// StatisticalInfo *statcPtrb :pointer to next line StatInfo

//int k: number of line

//int columns :number of lines

//int components:number of image components

Η συνάρτηση αυτή υπολογίζει το αθροισμα τηων δειγματων και το άθροισμα των τετραγώνων των δειγμάτων.

2.`void find_mean(StatisticalInfo *statcPtr,int k,int columns,int components)`

Η συνάρτηση αυτή υπολογίζει το μέσο όρο τον δειγμάτων της γραμμής

3.`void find_sigmapower2(StatisticalInfo *statcPtr,int k,int columns,int components)`

Η συνάρτηση αυτή υπολογίζει το τετράγωνο της διασποράς

4.`void find_sigma(StatisticalInfo *statcPtr,int columns,int components)`

Η συνάρτηση αυτή υπολογίζει την τετραγωνική ρίζα της διασποράς .

```
5.void    find cor horizontal(StatisticalInfo    *statcPtr,int    k,int
columns,int components)
```

Η συνάρτηση αυτή υπολογίζει τον οριζόντιο συσχετισμό.

```
6.void    find cor vertical(StatisticalInfo    *statcPtra,StatisticalInfo
*statcPtrb,int k,int columns,int components)
```

Η συνάρτηση αυτή υπολογίζει τον κάθετο συσχετισμό.

```
7.void    find cor diagonal(StatisticalInfo    *statcPtra,StatisticalInfo
*statcPtrb,int k,int columns,int components)
```

Η συνάρτηση αυτή υπολογίζει τον διαγώνιο συσχετισμό.

*Αποτελέσματα*:

Τα αποτελέσματα παρουσιάζονται στον πίνακα 6.6.

| | STANDARDENCODER | NEWENCODER1.0 | | NEWENCODER2.0 | | NEWENCODER3.0 | |
|---|---|---|---|---|---|---|---|
| **IMAGES** | **F-18.ppm** | **F-18.ppm** | **%** | **F-18.ppm** | **%** | **F-18.ppm** | **%** |
| | SIZE(bytes) | SIZE(bytes) | | SIZE(bytes) | | SIZE(bytes) | |
| ORIGINAL | 230415 | 230415 | | 230415 | | 230415 | |
| DEFAULT HUFF TABLES WITHOUT HEADERS | 118256 | 114080 | -3.5 | 101231 | -14.3 | 107410 | -9.1 |
| DEFAULT HUFF TABLES WITH HEADERS * | 118324 | 122274 | 3.3 | 101682 | -14.06 | 107773 | -8.9 |
| ORIGINAL | 230415 | 230415 | | 230415 | | 230415 | |
| CUSTOMIZED HUFF TABLES WITHOUT HEADERS | 107634 | 98053 | -8.9 | 100818 | -6.3 | 107346 | -0.2 |
| CUSTOMIZED HUFF TABLES WITH HEADERS * | 107760 | 117976 | 9.4 | 101621 | -5.69 | 107886 | 0.11 |
| ORIGINAL | 230415 | 230415 | | 230415 | | 230415 | |
| P1 | 83067 | 95713 | 15.2 | 79010 | -4.8 | 82614 | -0.5 |
| P2 | 61849 | 77062 | 24.5 | 59638 | -3.57 | 61488 | -0.58 |
| P4 | 40138 | 56060 | 39.6 | 39432 | -1.75 | 39956 | -0.4 |

| | STANDARDENCODER | NEWENCODER1.0 | | NEWENCODER2.0 | | NEWENCODER3.0 | |
|---|---|---|---|---|---|---|---|
| IMAGES | pepers.ppm | pepers.ppm | % | pepers.ppm | % | pepers.ppm | % |
| | SIZE(bytes) | SIZE(bytes) | | SIZE(bytes) | | SIZE(bytes) | |
| ORIGINAL | 786447 | 786447 | | 786447 | | 786447 | |
| DEFAULT HUFF TABLES WITHOUT HEADERS | 490820 | 480029 | -2.1 | 467563 | -4.7 | 476251 | -2.9 |
| DEFAULT HUFF TABLES WITH HEADERS * | 490888 | 497471 | 1.34 | 468199 | -4.62 | 476887 | -2.8 |
| ORIGINAL | 786447 | 786447 | | 786447 | | 786447 | |
| CUSTOMIZED HUFF TABLES WITHOUT HEADERS | 476222 | 458798 | -3.6 | 463519 | -2.66 | 471202 | -1 |
| CUSTOMIZED HUFFTABLES WITH HEADERS * | 476347 | 503945 | 5.7 | 464335 | -2.52 | 472018 | -0.9 |
| ORIGINAL | 786447 | 786447 | | 786447 | | 786447 | |
| P1 | 405197 | 434953 | 7.34 | 394662 | -2.5 | 401680 | -0.86 |
| P2 | 332655 | 363530 | 9.28 | 324227 | -2.5 | 329438 | -0.96 |
| P4 | 189275 | 224288 | 18.4 | 186490 | -1.4 | 190238 | 0.5 |

*Συμπεράσματα:*

1. Παρατηρούμε ότι ναι μεν μειώνονται οι κύκλοι μηχανής , το μέγεθος του αρχείου μειώνεται.
2. Με ενεργοποιημένη την παράμετρο point transform parameter, έχουμε μείωσει χωρητικότητας του αρχείου.

# Συμπεράσματα και προτάσεις

*Συμπεράσματα*:

Με το πέρας αυτής της διατριβής διαπιστώνουμε ότι οι τεχνικές βελτιστοποίησης που προτείνουμε όντως πετυχαίνουν καλύτερη απόδοση του lossless JPEG κωδικοποιητή. Βέβαια η κάθε μια από τις τρεις βελτιστοποιήσεις έχει πλεονεκτήματα και μειονεκτήματα. Συνοπτικά

| | Κύκλοι μηχανής | Χωρητικότητα | Χωρητικότητα με Pt |
|---|---|---|---|
| Βελτιστοποίηση 1η | αυξάνονται | Αυξάνεται με την εγγραφή των headers, μειώνεται χωρίς. | αυξάνεται |
| Βελτιστοποίηση 2η | αυξάνονται | μειώνεται | μειώνεται |
| Βελτιστοποίηση 3η | μειώνονται | μειώνεται | μειώνεται |

*Προτάσεις*:

Για περαιτέρω ενασχόληση με το συγκεκριμένο θέμα, προτείνουμε

- την επέκταση του κώδικα έτσι ώστε να αναγνωρίζονται και άλλοι τύποι εικόνων όπως bmp.
- αποκωδικοποίηση των παραγόμενων αρχείων .ljpg με τις βελτιστοποιήσεις από τον standard JPEG decoder.

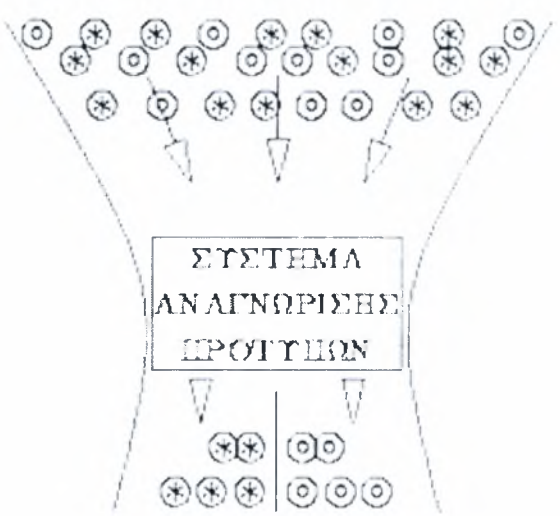
# ΠΑΡΑΡΤΗΜΑ Α

## Αναγνώριση προτύπων

*1. Ορισμός*

Η αναγνώριση προτύπων είναι ο επιστημονικός κλάδος που ασχολείται με την περιγραφή και την κατάταξη αντικειμένων σε ένα αριθμό κατηγοριών (εικόνα 1). Τα υπό κατάταξη αντικειμένων καλούνται πρότυπα. Υπάρχουν δύο κατηγορίες αναγνώρισης προτύπων:

- Σαφή πρότυπα (concrete items)
  - χαρακτήρες, εικόνες, αντικείμενα, ήχους.
- Αφηρημένα πρότυπα
  - Λύση ενός μαθηματικού προβλήματος ή ενός φιλοσοφικού επιχειρήματος.



## 2. Εφαρμογές

Η αναγνώριση προτύπων χρησιμοποιείται σε πολλές εφαρμογές όπως:

- Αναγνώριση ομιλίας.
- Αναγνώριση οπτικών χαρακτήρων
- Αναγνώριση χειρόγραφων κειμένων.
- Ταυτοποίηση ατόμων.
- Ιατρική διάγνωση.
- Γεωγραφικά συστήματα πληροφοριών.
- Βιομηχανικές εφαρμογές.
- Χρηματοοικονομικές Εφαρμογές.
- Εξόρυξη δεδομένων.

## 3. Μεθοδολογίες αναγνώρισης προτύπων

Υπάρχουν δυο μεθοδολογίες για την αναγνώριση προτύπων.

A. Στατιστική αναγνώριση προτύπων (statistical pattern recognition)
   - Εξαγωγή χαρακτηριστικών γνωρισμάτων
   - Δημιουργία διανύσματος χαρακτηριστικών στοιχείων
   - Μαθηματικές - στατιστικές μέθοδοι, γραμμική άλγεβρα, θεωρία πιθανοτήτων

B. Συντακτική ή δομημένη αναγνώριση προτύπων (syntactic or structural pattern recognition)
   - Δημιουργία πολύπλοκων ιεραρχικών περιγραφών.
   - Δέντρα αποφάσεων (decision trees), λογικοί κανόνες, γραμματικές.

# ΑΝΑΦΟΡΕΣ


1. Recommendation T.81 : Information Technology – Digital Compression and coding of continuous –tone still images – requirements and guidelines
2. Google and Wikipedia searching about
   - Huffman encoding
   - Lossless encoding
   - Jpeg
   - Pattern recognition