

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,  
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ  
ΔΙΚΤΥΩΝ

Κβαντοποίηση Διανυσμάτων σε Κωδικοποιητές Βίντεο

Vector Quantization in Video Encoding

Διπλωματική Εργασία

Καλός Πέτρος

Επιβλέποντες Καθηγητές : Κατσαβουνίδης Ιωάννης  
Αναπληρωτής Καθηγητής

Ποταμιάνος Γεράσιμος  
Αναπληρωτής Καθηγητής

Βόλος, Ιούνιος 2013





ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ  
ΠΟΛΥΤΕΧΝΙΚΗ ΣΧΟΛΗ  
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ Η/Υ,  
ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ ΚΑΙ ΔΙΚΤΥΩΝ

Κβαντοποίηση Διανυσμάτων σε Κωδικοποιητές Βίντεο

Διπλωματική Εργασία

Καλός Πέτρος

**Επιβλέποντες :** Κατσαβουνίδης Ιωάννης  
Αναπληρωτής Καθηγητής

Ποταμιάνος Γεράσιμος  
Αναπληρωτής Καθηγητής

Εγκρίθηκε από την διμελή εξεταστική επιτροπή την 28<sup>η</sup> Ιουνίου 2013

.....  
Ι.Κατσαβουνίδης  
Αναπληρωτής Καθηγητής

.....  
Γ.Ποταμιάνος  
Αναπληρωτής Καθηγητής



Διπλωματική Εργασία για την απόκτηση του Διπλώματος του Μηχανικού Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας, στα πλαίσια του Προγράμματος Προπτυχιακών Σπουδών του Τμήματος Μηχανικών Η/Υ, Τηλεπικοινωνιών και Δικτύων του Πανεπιστημίου Θεσσαλίας.

.....

Καλός Πέτρος

Διπλωματούχος Μηχανικός Ηλεκτρονικών Υπολογιστών, Τηλεπικοινωνιών και Δικτύων Πανεπιστημίου Θεσσαλίας

Copyright © Petros Kalos, 2013  
Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό οκοπό.  
Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για οκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα.  
Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό οκοπό πρέπει να απευθύνονται προς τον συγγραφέα.



*Στην οικογένειά μου και στους φίλους μου*

# Περιεχόμενα

Κατάλογος πινάκων	iv
Κατάλογος σχημάτων	v
Περίληψη	vi
Abstract	vii
<b>1 Εισαγωγή</b>	<b>1</b>
1.1 Περιγραφή του Προβλήματος . . . . .	1
1.2 Συμβολή της Εργασίας . . . . .	2
1.3 Διάρθρωση της Διπλωματικής Εργασίας . . . . .	2
<b>2 Ψηφιακό βίντεο και τεχνικές συμπίεσης</b>	<b>3</b>
2.1 Εισαγωγή . . . . .	3
2.2 Συστατικά στοιχεία του βίντεο . . . . .	3
2.3 Οργάνωση των pixels από τους κωδικοποιητές . . . . .	5
2.4 Μετασχηματισμός . . . . .	8
2.5 Κβαντοποίηση . . . . .	9
2.6 Αποκωδικοποίηση βίντεο . . . . .	10
2.7 Ποιότητα Βίντεο . . . . .	11
2.8 Δομή του H.264 . . . . .	12
<b>3 Θεωρία Πληροφοριών</b>	<b>13</b>
3.1 Εισαγωγή . . . . .	13
3.2 Εντροπία . . . . .	13
3.3 Κωδικοποιητές Εντροπίας . . . . .	14
3.4 Αλγόριθμος clustering k-means . . . . .	16
<b>4 Παραγωγή Codebooks</b>	<b>21</b>
4.1 Εισαγωγή . . . . .	21
4.2 Επιλογή του training set . . . . .	21
4.3 Εξαγωγή training set από τον H.264 . . . . .	24
4.4 K-means . . . . .	25

4.5 Υπό συνθήκη εντροπία . . . . .	25
5 VQ H.264 . . . . .	29
5.1 Εισαγωγή . . . . .	29
5.2 Encoding . . . . .	29
5.3 Decoding . . . . .	31
6 Μελέτη της Απόδοσης του VQ H.264 . . . . .	33
6.1 Εισαγωγή . . . . .	33
6.2 Encoding με VQ H.264 . . . . .	33
6.3 Encoding με JM H.264 . . . . .	35
6.4 Αποτελέσματα των VQ H.264,JM H.264 . . . . .	36
7 Συμπεράσματα . . . . .	39
7.1 VQ H.264 vs JM H.264 . . . . .	39
7.2 Πιθανές προσθίκες . . . . .	40
Βιβλιογραφία . . . . .	41

# Κατάλογος πινάκων

1.1	Κυριότεροι κωδικοποιητές βίντεο. [18]	1
2.1	Πίνακας Κβαντοποίησης. [19]	9
2.2	Παράδειγμα συντελεστών DCT 8x8. [19]	10
2.3	Κβαντοποιημένοι συντελεστές DCT. [19]	10
3.1	Διάρκεια αρχικοποίησης (seconds) για KKZ,Random με 1,2,4,6 threads.	17
3.2	Συμπεριφορά του k-means για Random,KKZ αρχικοποίηση με $k = 65536, d = 16, n = 100000$ . Φαίνονται το MSE στην επανάληψη 5 και στο σημείο τερματισμού.	17
3.3	Διάρκεια μίας επανάληψης (seconds) για Full Search,FastNN με 1,2,4,6 threads με $k = 65536, d = 16, n = 100000$	18
4.1	Πληροφορίες για τα codebooks.	25
4.2	Υπό συνθήκη εντροπία με την χρήση contexts.	26
6.1	PSNR των Test videos με κωδικοποίηση στον VQ H.264	33
6.2	PSNR των Test videos με κωδικοποίηση στον JM H.264	35
6.3	Διαφορές του PSNR JM-VQ. Στα θετικά πρόσημα ο JM είναι καλύτερος από τον VQ ενώ στα αρνητικά πρόσημα το ανάποδο.	35

# Κατάλογος σχημάτων

2.1	Στην πρώτη εικόνα φαίνεται ένα καρέ με όλες τις συνιστώσες ενώ παρακάτω ξεχωριστά το YUV για το ίδιο καρέ [20]. . . . .	4
2.2	Διαμερισμός του macroblock στον H.264.[4] . . . . .	5
2.3	Τρόποι που χρησιμοποιούνται στον H.264 για Intra prediction. [1] . . . . .	6
2.4	Inter prediction στον mpeg και απεικόνιση ενός ενδεικτικού GOP. [12] . . . . .	7
2.5	Παραδείγμα χρήσης RLE . . . . .	8
2.6	Σάρωση ZigZag. [11] . . . . .	8
2.7	H.264 encoder structure. [15] . . . . .	12
3.1	Δέντρο Huffman. [13] . . . . .	14
3.2	Βήματα αριθμητικής κωδικοποίησης. [17] . . . . .	15
3.3	k-means με $k = 2, d = 2, n = 100, MSE = 284.671$ . [10] . . . . .	16
4.1	Training βίντεο. [5] . . . . .	23
4.2	Με καφέ είναι το τρέχον μπλοκ για κάθε περίπτωση και με κίτρινο από ποια γειτονικά βγαίνει ο μέσος όρος . . . . .	26
4.3	Οι κόκκινες γραμμές δείχνουν τις περιοχές ενέργειας που αντιστοιχούν σε κάθε κατηγορία ενώ με μπλε απεικονίζεται η συχνότητα του κάθε cluster. . . . .	27
5.1	VQ H.264 Encoder Structure . . . . .	31
5.2	VQ H.264 Decoder Structure . . . . .	32
6.1	Βίντεο που δοκιμάστηκαν να γίνουν encoding με τον VQ H.264. . . . .	34
6.2	Σύγκριση των bits του JM H.264 και VQ H.264 με context entropy και χωρίς context. . . . .	36
6.3	Σύγκριση του χρόνου εκτέλεσης των encoder JM H.264 και VQ H.264. . . . .	37
6.4	Σύγκριση του χρόνου εκτέλεσης των decoder JM H.264 και VQ H.264. . . . .	38

# Περίληψη

Λέξεις Κλειδιά:

Κραντοποίηση Διανυσμάτων, Εντροπία, Θεωρία Πληροφοριών, Κωδικοποιητές Βίντεο, Κωδικοποιητές Εντροπίας, H.264, k-means

# Abstract

## Keywords:

Vector Quantization, Entropy, Information Theory, Video Codec, Entropy Encoding, H.264, k-means



# Κεφάλαιο 1

## Εισαγωγή

### 1.1 Περιγραφή του Προβλήματος

Οι κωδικοποιητές βίντεο είναι αναγκαίοι για να μειώσουν τον τεράστιο αριθμό δεδομένων που έχει ένα βίντεο. Σε οπτικά σήματα με ισχύ μεγαλύτερη των 35dB δεν είναι εύκολο να παρατηρήθει διαφορά από το ανθρώπινο μάτι. Συνεπώς, επιλέγεται να εισαχθεί ομοιόμορφος θόρυβος για να επιτευχθεί μεγαλύτερος λόγος συμπίεσης θυσιάζοντας την ποιότητα. Η κωδικοποίηση βίντεο υλοποιείται τόσο σε επίπεδο λογισμικού αλλά και υλικού. Ενώ οι προσωπικοί υπολογιστές συνήθιζαν να χρησιμοποιούν λογισμικό για την αναπαραγωγή βίντεο, τα τελευταία χρόνια η αύξηση των αναλύσεων του βίντεο (1080p, 4K) καθιστά αδύνατη ή πολύ δαπανηρή την αποκωδικοποίηση ενός συμπιεσμένου βίντεο από έναν επεξεργαστή γενικής χρήσης. Έτσι συναντάμε υλικό ειδικά σχεδιασμένο (hardware accelerators) για κωδικοποίηση/αποκωδικοποίηση βίντεο που βοηθά τον κύριο επεξεργαστή.

Έτος	Κωδικοποιητής	Διάφορες Χρήσεις	Bitrate(Mbps) (720x480)
1993	MPEG-1	VCD	7
1995	MPEG-2	DVD	6
1999	MPEG-4	DivX,XViD	5
2003	H.264	BluRay,DVB-TS	4
2013	H.265	next generation H.264	2

Πίνακας 1.1: Κυριότεροι κωδικοποιητές βίντεο. [18]

Σήμερα όπου υπάρχει βίντεο υπάρχει και κωδικοποίηση. Όπως φαίνεται στον πίνακα 1.1 με το πέρασμα του χρόνου οι κωδικοποιητές γίνονταν ολοένα και πιο αποτελεσματικοί αυξάνοντας όμως κατά πολύ την πολυπλοκότητα τους. Πλέον, υπάρχει η δυνατότητα με λίγα δεδομένα να έχουμε βίντεο καλής ποιότητας σε πολύ μεγάλες αναλύσεις. Ενδεικτικά για ένα ασυμπίεστο βίντεο ανάλυσης DVD

## 1. Εισαγωγή

---

(720x480) με διάρκεια 20sec και ρυθμό ανανέωσης 30Hz απαιτείται χώρος 296MB. Κάνοντας συμπίεση με τον H.264 προκύπτει ένα αρχείο 0.8MB.

### 1.2 Συμβολή της Εργασίας

Στην εργασία αυτή γίνεται χρήση της κβαντοποίησης διανυσμάτων στον κωδικοποιητή H.264. Είναι μια τεχνική συμπίεσης δεδομένων με απώλειες (lossy compression), η οποία έχει δοκιμαστεί ελάχιστα παλαιότερα στο βίντεο και φαίνεται πως μπορεί να δώσει λύσεις στον λόγο συμπίεσης καθώς επίσης και στην μείωση της πολυπλοκότητας του αποκωδικοποιητή. Ο τρόπος που μειώνεται η πολυπλοκότητα μπορεί να μετατρέψει ακριβά ενεργοβόρα κυκλώματα σε φθηνές μηνύμες χωρίς απαιτήσεις ισχύος.

### 1.3 Διάρθρωση της Διπλωματικής Εργασίας

Στο Κεφάλαιο 2 παρουσιάζεται το ψηφιακό βίντεο, τις μορφές αποθηκεύσης του, πώς οι κωδικοποιητές το αντιμετωπίζουν και ποιές είναι οι κύριες τεχνικές που χρησιμοποιούνται για την συμπίεση του.

Στο Κεφάλαιο 3 γίνεται μία σύντομη εισαγωγή στον τομέα της Θεωρίας Πληροφοριών. Είναι αναγκαίο να εξηγηθεί τι είναι η πληροφορία καθώς και η εντροπία αυτής. Επίσης θα εξηγηθεί αναλυτικά ο κύριος αλγόριθμος clustering k-means καθώς και οι όποιες βελτιστοποίησες προέκυψαν στην πορεία της παρούσας διπλωματίκης.

Στο Κεφάλαιο 4 παρουσιάζονται τα βήματα που έγιναν για την παραγωγή των codebooks και την επιτρόποθετη κατηγοριοποίηση τους με στόχο την μείωση της εντροπίας.

Στο Κεφάλαιο 5 παρουσιάζεται η επέμβαση που έγινε στον H.264 έτσι ώστε να χρησιμοποιεί τα codebooks για να κάνει την κωδικοποίηση/αποκωδικοποίηση.

Στο Κεφάλαιο 6 παρουσιάζονται τα αποτελέσματα του VQ-H.264.

Στο Κεφάλαιο 7 συγκρίνονται τα αποτελέσματα του VQ-H.264 με τον JM-H.264.

# Κεφάλαιο 2

## Ψηφιακό βίντεο και τεχνικές συμπίεσης

### 2.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μία εισαγωγή στους τρόπους που οι κωδικοποιητές βίντεο διαχειρίζονται τα δεδομένα και τους τρόπους που χρησιμοποιούν για να τα συμπιέσουν. Ακόμη, σε αυτό το κεφάλαιο θα αποσαφηνιστεί γιατί το βίντα πης κραντοποίησης (εισαγωγή σφάλματος) είναι αναγκαίο για να έχουμε μεγάλους λόγους συμπίεσης.

### 2.2 Συστατικά στοιχεία του βίντεο

Το ψηφιακό βίντεο αποτελείται από μία σειρά καρέ που αναπαράγονται με σταθερό ρυθμό (συνήθως 25Hz ή 30Hz). Το κάθε καρέ απεικονίζεται σε ένα χώρο χρωμάτων που ονομάζεται YUV, οπού το Y είναι η φωτεινότητα και το U,V η χρωματικότητα. Η κάθε συνιστώσα στο ανθρώπινο μάτι φαίνεται όπως στο Σχήμα 2.1.

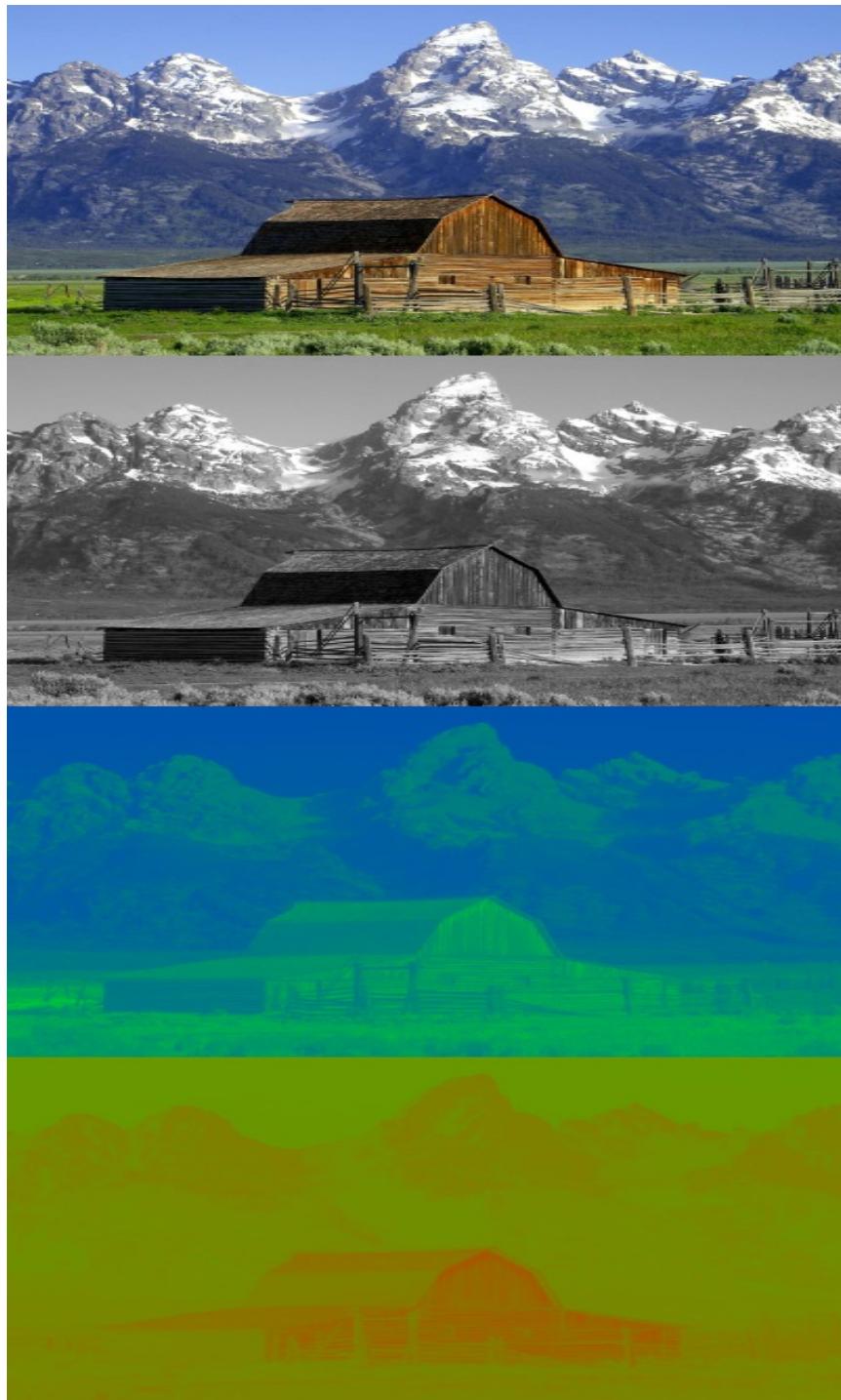
Σε κάθε σημείο του βίντεο (pixel) αντιστοιχίζεται μια τέτοια τιμή YUV. Υπάρχουν διάφορες μέθοδοι που γίνεται αυτή η αντιστοίχιση με κυριότερες αυτές του YUV420 και YUV444. Η πρώτη υποδεικνύει πως κάθε για κάθε τέσσερα pixel υπάρχουν τέσσερις τιμές Y, μία τιμή U και μία τιμή V. Η δεύτερη δείχνει πως για κάθε τέσσερα pixel υπάρχουν τέσσερις τιμές για την κάθε συνιστώσα π.χ για ένα καρέ ανάλυσης  $720 \times 480$  τύπου YUV420 έχουμε  $720 \times 480 = 345600$  στοιχεία Y,  $720 \times 480/4$  στοιχεία U και  $720 \times 480/4$  στοιχεία V αρά σύνολο 518400. Ο τρόπος αποθήκευσης του βίντεο γίνετε πάντα ανά καρέ και υπάρχουν δύο τρόποι, ο packed και ο planar. Στον πρώτο το YUV για κάθε pixel αποθηκεύεται με την σειρά ενώ στον δεύτερο και επικρατέστερο αποθηκεύεται πρώτα όλο το Y και μετά ακολουθούν τα U,V.

Κάθε pixel έχει και ένα βάθος (bitdepth), δηλαδή το εύρος τιμών που παίρνει. Οι σύγχρονοι κωδικοποιητές υποστηρίζουν 8-14bits έτσι συνεχίζοντας το παραδειγμα μας από την προηγούμενη παράγραφο και έχοντας συνολικά 518400 στοιχεία για ένα καρέ και υποθέτοντας ότι το bitdepth 8, καταλήγουμε στο συμπέρασμα ότι

## 2. Ψηφιακό βίντεο και τεχνικές συμπίεσης

---

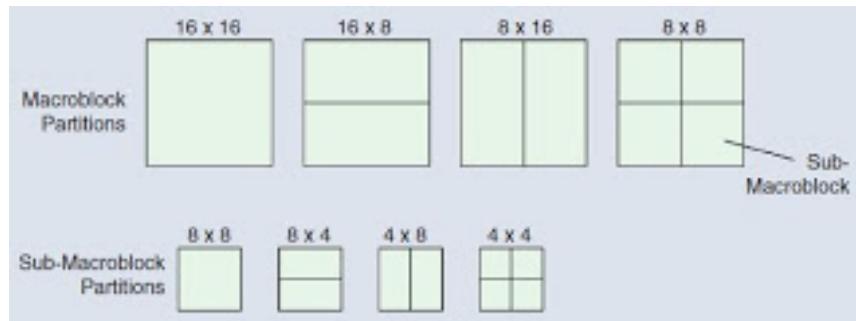
συνολικά χρειαζόμαστε  $518400 * 8\text{bits} \approx 0.5\text{MB}$ . Το πιο κοινό bitdepth που χρησιμοποιείται από τους σημερινούς κωδικοποιητές είναι αυτό των 8bits.



Σχήμα 2.1: Στην πρώτη εικόνα φαίνεται ένα καρέ με όλες τις συνιστώσες ενώ παρακάτω ξεχωριστά το YUV για το ίδιο καρέ [20].

## 2.3 Οργάνωση των pixels από τους κωδικοποιητές

Η πλειοψηφία των σημερινών κωδικοποιητών ομαδοποιούν τα pixels. Η πιο συνηθισμένη ομαδοποίηση είναι αυτή του macroblock, το κάθε καρέ διαιρείται σε μικρά πλακίδια σταθερής διάστασης NxN pixels όπου συνήθως N=16. Συνεπώς, το καρέ του παραδείγματος μας αποτελείται από  $\frac{345600}{16 \times 16} = 1350$  macroblocks που για YUV420 το καθένα περιέχει 1 Y macroblock 16x16 και 8 UV 8x8. Το κάθε macroblock μπορεί να σπάσει σε blocks και το κάθε block σε subblocks όπως φαίνεται στο Σχήμα 2.2.



Σχήμα 2.2: Διαιρερισμός του macroblock στον H.264.[4]

Μετά τον χωρισμό σε macroblocks οι κωδικοποιητές ομαδοποιούν τα καρέ σε Intra και Inter (Temporal ή Special) με τα τελευταία να χωρίζονται σε P (predictive) και B (bidirectional). Όλες οι παραπάνω κατηγοριοποιήσεις έχουν να κάνουν με τον υπχανισμό του Motion Estimation ο οποίος δημιουργεί διαφορές pixels (residuals). Κύριο χαρακτηριστικό των residuals είναι η μικρή τους ενέργεια (οι τιμές τους είναι κοντά στο 0).

- Intra ή αλλιώς I είναι τα καρέ που χρησιμοποιούν πληροφορία μόνο από το τρέχον καρέ για να βγάλουν τα residuals. Αυτό γίνεται απλά παίρνοντας τις τιμές των γειτονικών block και εφαρμόζοντας ένα mode π.χ μέσος όρος όπου το αποτέλεσμα αυτού του mode αφαιρείται από τις τιμές των pixel του τρέχοντος block. Υπάρχουν διάφορα modes που μπορούν να γίνουν όπως φαίνεται και στο Σχήμα 2.3 και δοκιμάζονται κάθε φόρα όλα μέχρι να καταλήξουν στο καλύτερο, δηλαδή αυτό που έχει το μικρότερο σφάλμα ή αυτό που χρειάζεται τον μικρότερο αριθμό από bits για να κωδικοποιηθεί. Η απόδοση της συμπίεσης είναι συγκριτικά η χειρότερη σε σχέση με τα άλλα είδη καρέ αλλά χωρίς αυτά το βίντεο δε θα μπορούσε να ξεκινήσει γιατί δεν θα υπήρχε σημείο εκκίνησης για να αποκωδικοποιηθεί η πληροφορία.
- P καρέ είναι αυτά τα οποία ψάχνουν το καλύτερο block για να κάνουν διαφορές από κάποιο αριθμό προηγούμενων καρέ όπως φαίνεται στο Σχήμα 2.4. Επιπρόσθετα κάθε macroblock έχει ως pixel αναφοράς pixel που προέρχονται από ένα συγκεκριμένο καρέ.

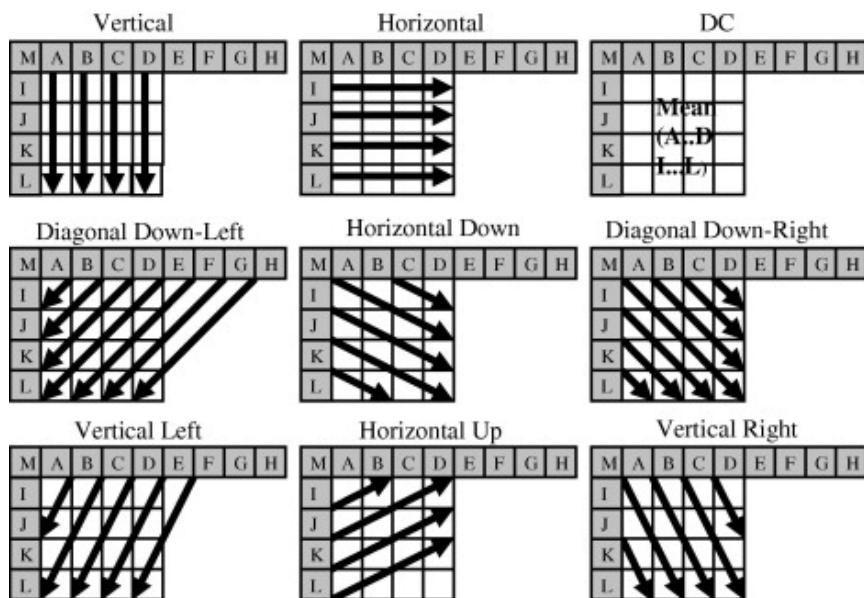
## 2. Ψηφιακό βίντεο και τεχνικές συμπίεσης

---

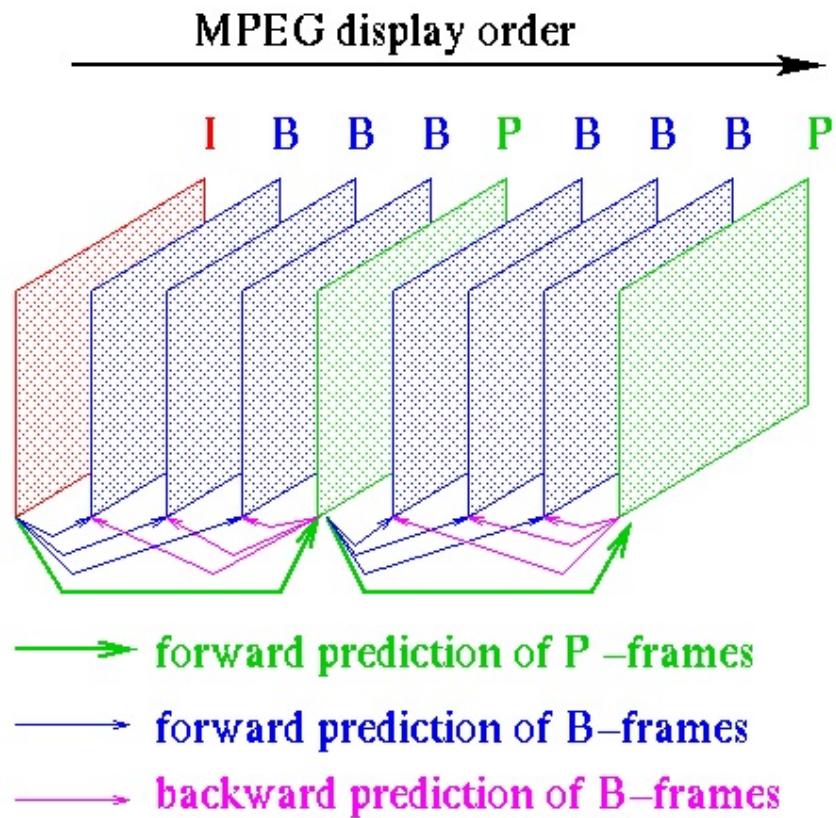
- Β καρέ είναι αυτά τα οποία ψάχνουν το καλύτερο block για να κάνουν διαφορές από κάποιο αριθμό προηγούμενων αλλά και επόμενων καρέ όπως φαίνεται στο Σχήμα 2.4. Σε αντίθεση με τα P τα macroblock των B μπορούν να προβλέπονται από τον μέσο όρο pixel που προέρχονται από δύο καρέ. Αυτά έχουν την καλύτερη απόδοση συμπίεσης αλλά η πολυπλοκότητα αποκωδικοποίησης είναι η μεγαλύτερη.

Ένα επιπλέον στοιχείο των encoder είναι το group of pictures (GOP) το οποίο χαρακτηρίζει την σειρά με την οποία τα είδη των καρέ τοποθετούνται. Το GOP ξεκινάει από ένα I-frame συνεχίζει με P,B και περιοδικά έρχεται ένα I που σημαδοτεί το τέλος του τρέχοντος GOP. Το GOP εξαρτάται από την εφαρμογή και μπορεί να έχει μεγάλες αποκλίσεις. Ένα τυπικό φαίνεται στο Σχήμα 2.4

Έτσι λοιπόν για να ανακατασκευαστεί ένα καρέ χρειάζεται την πληροφορία πρόβλεψης. Για τα Intra καρέ, αυτή η πληροφορία είναι το mode που έγινε η κωδικοποίηση Σχήμα 2.3. Για τα Inter είναι ένα διάνυσμα τριών διαστάσεων (motion vector)  $mv = [framenum, x, y]$ , οπού framenum είναι ο αριθμός του frame που βρίσκεται η πληροφορία και  $x, y$  οι συντεταγμένες του. Εννοείται, πως το framenum, πρέπει ήδη να έχει αποκωδικοποιηθεί πλήρως είτε επειδή είναι Intra είτε επειδή έχει όλη την πληροφορία από προηγούμενα/επόμενα P,B καρέ.



Σχήμα 2.3: Τρόποι που χρησιμοποιούνται στον H.264 για Intra prediction. [1]



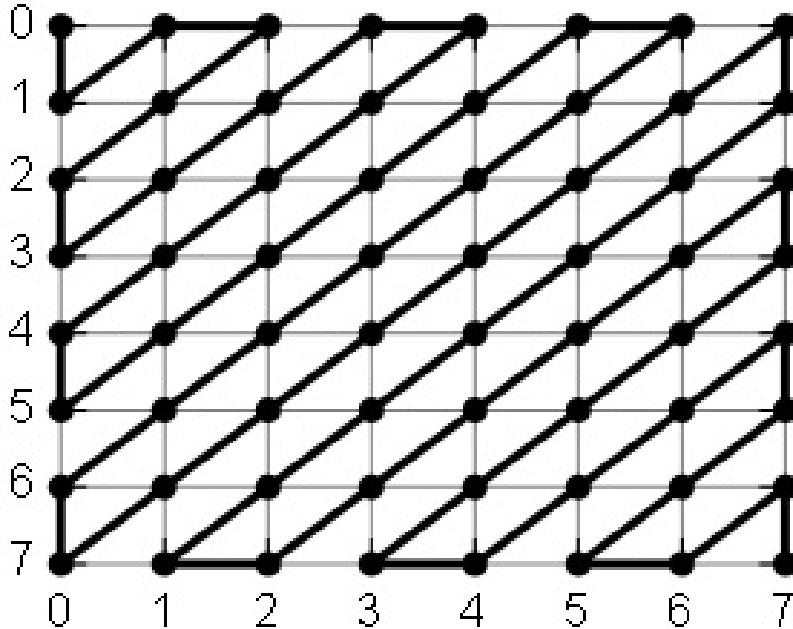
Σχήμα 2.4: Inter prediction στον mpeg και απεικόνιση ενός ενδεικτικού GOP. [12]

## 2.4 Μετασχηματισμός

Έχοντας ο encoder κατασκευάσει τα residuals για το macroblock, το επόμενο βήμα είναι ένας μετασχηματισμός, που συνήθως είναι ο διακριτός μετασχηματισμός συνημιτόνου DCT(Discrete Cosine Transform). Στους κωδικοποιητές μpeg1,2,4 εφαρμόζεται με διάσταση 8x8, στον H.264 και για 4x4 ενώ στον H.265 και για 16x16. Αυτό συμβαίνει γιατί ο μετασχηματισμός επιστρέφει πολλά 0 αν το σήμα μας είναι χαμηλής ενέργειας, όπως συμβαίνει με τα residuals. Με αυτόν τον τρόπο, χρησιμοποιώντας κάποιο ειδικό σύμβολο μπορούν να απεικονιστούν πολλά μηδενικά του μετασχηματισμού και να μειωθεί το πλήθος των αριθμών προς κωδικοποίηση όπως φαίνεται στο Σχήμα 2.5 μέσω της εφαρμογής του αλγορίθμου RLE (Run Length Encoding). Η σάρωση του macroblock δεν γίνεται γραμμικά αλλά με την μέθοδο του zigzag όπως στο Σχήμα 2.6 γιατί έτσι έχει παρατηρηθεί πως έχουμε παραπάνω μηδενικά στην σειρά άρα και καλύτερη απόδοση του RLE.

17	8	54	0	0	0	97	5	0	16
$\xrightarrow{RLE}$									
(0, 17)	(0, 8)	(0, 54)	(3, 97)	(0, 5)	(1, 16)				

Σχήμα 2.5: Παραδείγμα χρήσης RLE



Σχήμα 2.6: Σάρωση ZigZag. [11]

## 2.5 Κβαντοποίηση

Η κβαντοποίηση είναι το σημείο που σε κάθε encoder εισάγεται το σφάλμα. Μετά τον μεταχρηματισμό τα residuals έχουν αντικατασταθεί με συντελεστές οι οποίοι είναι πραγματικοί αριθμοί. Η τακτική που εφαρμόζεται είναι να γίνει ακέραια διαίρεση του συντελεστή της κάθε συχνότητας με έναν σταθερό ακέραιο αριθμό, πιθανόν διαφορετικό για κάθε συχνότητα όπως φαίνεται στον Πίνακα 2.1. Επομένως για τους δεδομένους πίνακες συντελεστών του Πίνακα 2.2 έχουμε τα αποτελέσματα στον Πίνακα 2.3 τα οποία είναι αυτά που θα δοθούν στο RLE και εν συνεχείᾳ σε κάποιον entropy encoder για να συμπιεστούν.

Ο Πίνακας 2.1 μεταβάλλεται με βάση μια μεταβλητή που καθορίζει την ποιότητα του βίντεο η οποία ονομάζεται Quantization Parameter (QP). Στον H.264, το  $QP \in [0, 51]$ . Στο περίπτωση του QP 0 εμφανίζεται η μέγιστη δυνατή ποιότητα η οποία μπορεί να θεωρηθεί σχεδόν χωρίς απώλειες (lossless) και όσο το QP αυξάνει τόσο η ποιότητα πέφτει. Αυτό συμβαίνει γιατί το QP έχει αναλογική σχέση με τους πολλαπλασιαστές του Πίνακα 2.1 του οποίου οι τιμές μεγαλώνουν. Επομένως, στην ακέραια διαίρεση χάνεται περισσότερο πληροφορία καθώς γίνεται διαίρεση με μεγαλύτερο αριθμό. Υπάρχει κέρδος όμως σε μέγεθος γιατί έτσι προκύπτουν περισσότερα μηδενικά και μικρότερες τιμές κατά απόλυτη τιμή οι οποίοι απαιτούν λιγότερα bits για την κωδικοποίηση τους.

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	14	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Πίνακας 2.1: Πίνακας Κβαντοποίησης. [19]

## 2. Ψηφιακό βίντεο και τεχνικές συμπίεσης

---

-415.38	-30.19	-61.20	27.24	56.13	-20.10	-2.39	0.46
4.47	-21.86	-60.76	10.25	13.15	-7.09	-8.54	4.88
-46.83	7.37	77.13	-24.56	-28.91	9.93	5.42	-5.65
-48.53	12.07	34.10	-14.76	-10.24	6.30	1.83	1.95
12.12	-6.55	-13.20	-3.95	-1.88	1.75	-2.79	3.14
-7.73	2.91	2.38	-5.94	-2.38	0.94	4.30	1.84
-1.03	0.18	0.42	-2.42	-0.88	-3.02	4.12	-0.66
-0.17	0.14	-1.07	-4.19	-1.17	-0.10	0.70	1.68

Πίνακας 2.2: Παραδειγμα συντελεστών DCT 8x8. [19]

-26	-3	-6	2	2	-1	0	0
0	-2	-4	1	1	0	0	0
-3	1	5	-1	-1	0	0	0
-3	1	2	-1	0	0	0	0
1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

Πίνακας 2.3: Κβαντοποιημένοι συντελεστές DCT. [19]

## 2.6 Αποκωδικοποίηση βίντεο

Για την αποκωδικοποίηση εκτελούνται όλα τα βίματα που κάνει ο encoder με την ανάποδη σειρά.

- Μετά την entropy decoding τα νούμερα πολλαπλασιάζονται με τα στοιχεία του πίνακα κβαντοποίησης. Έτσι ανακτώνται οι συντελεστές του μετασχηματισμού, έχοντας πλέον εισάγει το σφάλμα που προέκυψε λόγω της ακέραιας διαίρεσης και τοποθετούνται στη σωστή θέση του πίνακα με την αντίστροφη σάρωση zigzag.
- Γίνεται αντίστροφος μετασχηματισμός και ανακτώνται τα residuals.
- Τέλος το Motion Compensation βρίσκει τα ένα ή δύο blocks που χρησιμοποιούνται για να παραχθούν τα residuals (μέσω των motion vectors) και τα προσθέτει με τα ανακατασκευασμένα residuals για να παραχθούν τα ανακατασκευασμένα pixels.

## 2.7 Ποιότητα Βίντεο

Με τον όρο ποιότητα βίντεο ορίζουμε το μέσο όρο των αθροίσματος των διαφόρων στο τετράγωνο MSE (Mean Squared Error) όλων των pixels μεταξύ του ασυμπτέστου βίντεο και του συμπιεσμένου.

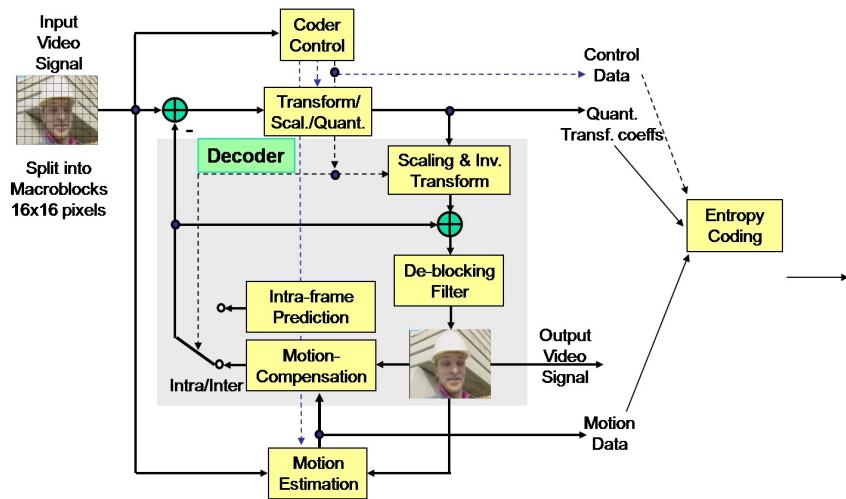
$$\text{MSE} = \frac{\sum_{i=1}^{X*Y} (\text{Source}_{pixel_i} - \text{Reconstructed}_{pixel_i})^2}{X * Y} \quad (2.1)$$

Συνήθως γίνεται μετατροπή του MSE σε PSNR (Peak Signal to Noise Ratio) με μονάδα μέτρησης το dB με τον τύπο  $PSNR = 10 * \log_{10} \frac{MAX_i^2}{MSE}$  οπού το  $MAX_i$  είναι η μέγιστη τιμή που μπορεί να πάρει ένα pixel. Αυτό εξαρτάται από το bitdepth, π.χ αν έχουμε bitdepth 8bits τότε  $MAX_i = 255$ .

Ο μέσος όρος του PSNR όλων των καρέ μας δίνει την ποιότητα του βίντεο, το PSNR υπολογίζεται ξεχωριστά για τις συνιστώσες Y,UV και επειδή το ανθρώπινο μάτι είναι πιο ευαίσθητο στην φωτεινότητα, το Y λαμβάνεται ως μετρική για όλο το βίντεο.

## 2.8 Δομή του H.264

Στο Σχήμα 2.7, παρουσιάζεται η δομή του H.264 encoder. Το υποσύνολο με γκρι χρώμα είναι το κομμάτι του αποκωδικοποιητή (decoder). Κάθε encoder υλοποιεί στο εσωτερικό του και έναν decoder για να έχει τα pixel αναφοράς που χρησιμοποιούνται για το Motion Estimation και για το Motion Compensation συγχρονισμένα με τα αντίστοιχα pixel του αποκωδικοποιητή. Υπάρχουν και άλλα στοιχεία, όπως το φίλτρο απομάκρυνσης γραμμών (deblocking filter), ο ελεγκτής ρυθμού συμπίεσης (rate controller) που σε αυτή την διπλωματική δεν κρίνεται απαραίτητο να εξηγήθουν περαιτέρω διότι δεν βοηθούν στην κατανόση της.



Σχήμα 2.7: H.264 encoder structure. [15]

# Κεφάλαιο 3

## Θεωρία Πληροφοριών

### 3.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια εισαγωγή σε βασικά στοιχεία της θεωρίας πληροφοριών και η επεξήγηση της έννοιας εντροπίας της πληροφορίας. Επίσης θα αναφερθούν αλγόριθμοι συμπίεσης (entropy encoding) καθώς και ο αλγόριθμος k-means που χρησιμοποιήθηκε για την παραγωγή των codebooks.

### 3.2 Εντροπία

Η εντροπία κατά Shannon που εισήχθηκε από τον Claude E. Shannon το 1948 είναι η ποσοτικοποίηση της αβεβαιότητας μιας τυχαίας μεταβλητής και συνήθως μετριέται σε bits ή nats [14]. Η κύρια ποσότητα που παρέχεται από την εντροπία και είναι χρήσιμη σε εμάς είναι το απόλυτο κάτω όριο μέχρι το οποίο η πληροφορία μιας πηγής μπορεί να συμπιεστεί. Η εντροπία ορίζεται ως  $H(X) = -\sum_{i=1}^n p(x_i) * \log_b p(x_i)$  όπου  $p(x_i)$  είναι η πιθανότητα του ενδεχομένου  $x_i$  και η βάση  $b$  ορίζει την μονάδα μέτρησης της εντροπίας. Για την συμπίεση δεδομένων συνήθως χρησιμοποιούμε  $b = 2$ , ώστε να έχουμε την εντροπία εκφρασμένη σε bits.

Για να γίνει κατανοητή η έννοια της εντροπίας θα δοθεί ένα παραδειγμα. Έστω ότι πρέπει να αναπαρασταθεί μια ακολουθία από αριθμούς  $x_i \in [0, 3]$  σε δυαδικό σύστημα. Επομένως υπάρχουν  $n = 4$  διαφορετικά ενδεχόμενα που χρειάζονται 2

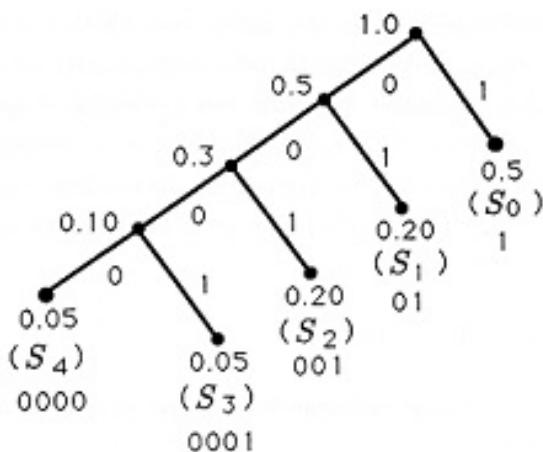
bits το καθένα για να αναπαρασταθούν τα  $x_i = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$ . Αν τα ενδεχόμενα είναι

ισοπίθανα ισχύει ότι  $\forall x_i, p(x_i) = 0.25$  και προκύπτει ότι  $H(X) = 2 bits$ . Συνεπώς, δε μπορεί να γίνει περαιτέρω συμπίεση. Αν όμως ισχύει ότι  $p(x_1) = 0.7, p(x_2) = p(x_3) = p(x_4) = 0.1$  τότε έχουμε  $H(X) = 1.35678 bits$  ανά σύμβολο κατά μέσο όρο. Επομένως τα δεδομένα μπορούν ιδανικά να συμπιεστούν κατά  $\frac{(2-1.35678)}{2} \% = 32\%$ .

### 3.3 Κωδικοποιητές Εντροπίας

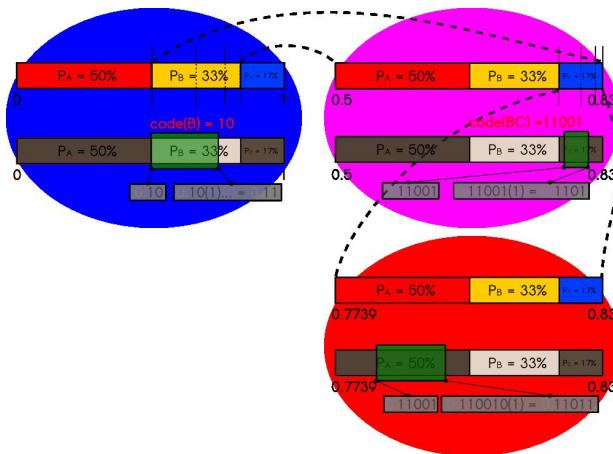
Οι μέθοδοι που σήμερα υπάρχουν για συμπίεση καταφέρνουν να έρθουν πολύ κοντά στο όριο που δίνει η εντροπία αλλά δε το φτάνουν. Δύο είναι οι κυρίαρχες μέθοδοι, η μέθοδος Huffman και αυτή της Αριθμητικής Κωδικοποίησης. Η δεύτερη είναι δημοφιλής στο βίντεο με την παραλλαγή της που λέγεται CABAC (Context Adaptive Binary Arithmetic Coding).

- Η μέθοδος Huffman έχει την μικρότερη πολυπλοκότητα μεταξύ των δύο, συγκεκριμένα έχει σχεδόν μηδενική πολυπλοκότητα κωδικοποίησης (look-up tables) ενώ η πολυπλοκότητα αποκωδικοποίησης είναι μια πράξη ανά *bit*. Ο αλγόριθμος έχει δύο στάδια, το στάδιο κατασκευής ενός δυαδικού δέντρου το οποίο γίνεται μια φορά για κάθε κατανομή πιθανότητας, και σαν επόμενο στάδιο είναι αυτό της κωδικοποίησης όπου τα σύμβολα χρησιμοποιούνται σαν διεύθυνση ενός look up table. Τα περιεχόμενα του είναι οι δυαδικές ακολουθίες που έχουν αντιστοιχηθεί στα φύλλα του δέντρου Huffman. Ο αλγόριθμος κατασκευής του δέντρου βάζει τα ενδεχόμενα σε ένα δυαδικό δέντρο και τους αναθέτει λέξεις με διαφορετικό μήκος. Στόχος είναι το ενδεχόμενο με την μεγαλύτερη πιθανότητα να έχει το μικρότερο μήκος και αυτό με την μικρότερη πιθανότητα το μεγαλύτερο μήκος. Έστω λοιπόν ότι υπάρχουν 5 ενδεχόμενα  $x_i$  με πιθανότητες  $p(S_0) = 0.5, p(S_1) = p(S_2) = 0.2, p(S_3) = p(S_4) = 0.05$ . Το αντίστοιχο δέντρο Huffman για αυτό το παράδειγμα φαίνεται στο Σχήμα 3.1. Η αποδοτικότητα του αλγόριθμου Huffman είναι  $H(X) \leq L_c \leq H(X) + 1\text{bit}$  όπου  $L_c$  είναι το μέσο μήκος ανά σύμβολο.



Σχήμα 3.1: Δέντρο Huffman. [13]

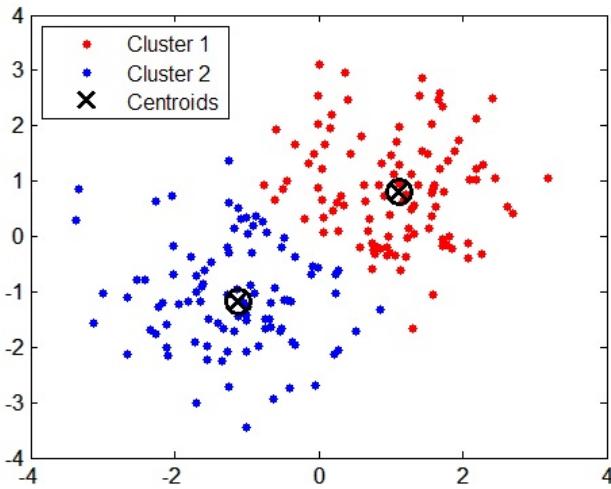
- Η μέθοδος της Αριθμητικής Κωδικοποίησης έχει την μεγαλύτερη πολυπλοκότητα μεταξύ των δύο αλλά μας εξασφαλίζει μια εν γένει καλύτερη συμπίεση από αυτή τού Huffman και αρκετά "κοντά" στο όριο εντροπίας. Ο αλγόριθμος όπως και στον Huffman έχει στόχο να δώσει μεγάλο μήκος στα ενδεχόμενα με την μικρότερη πιθανότητα και μικρό σε αυτά με την μεγαλύτερη. Η διαφορά του με τον Huffman είναι πως δεν κωδικοποιεί ανά σύμβολο αλλά όλο την σειρά συμβόλων σε έναν μοναδικό αριθμό  $n \in R$ . Για παράδειγμα έστω μια κατανομή με 3 σύμβολα A,B,C και τις πιθανότητες τους  $p(A) = 0.5, p(B) = 0.33, p(C) = 0.17$  και έστω το μήνυμα που κωδικοποιείται είναι το "BCA". Στο Σχήμα 3.2 φαίνονται τα βήματα της κωδικοποίησης. Στο πρώτο βήμα έρχεται το B και κωδικοποιείται με τον αριθμό 0b01(x) γιατί έχει το μικρότερο μήκος και βρίσκεται στο [0.5, 0.83) οπού x σημαίνει αυθαίρετη ακολουθία από bits. Με αυτόν τον τρόπο συνεχίζουν και τα υπόλοιπα βήματα μέχρι τον αριθμό που αντιστοιχεί στην ακολουθία. Η τυπική υλοποίηση του αλγορίθμου CABAC απαιτεί 50 ως 100 αριθμητικές και λογικές πράξεις για την κωδικοποίηση ή την αποκωδικοποίηση ενός δυαδικού συμβόλου.



Σχήμα 3.2: Βήματα αριθμητικής κωδικοποίησης. [17]

### 3.4 Αλγόριθμος clustering k-means

Άλλο ένα στοιχείο που χρησιμοποιήθηκε σε αυτή την διπλωματική και πηγάζει από την Θεωρία Πληροφοριών είναι ο αλγόριθμος clustering k-means. Είναι ένας επαναληπτικός αλγόριθμος όπου στόχος του είναι να χωρίσει με το ελάχιστο σφάλμα  $n$  σημεία σε διάσταση χώρου  $R^d$  σε  $k$  περιοχές  $k \leq n$  όπως φαίνεται στο Σχήμα 3.3. Ο αλγόριθμος k-means έχει πολύ μεγάλη υπολογιστική πολυπλοκότητα και ανάγεται στα NP-hard προβλήματα.



Σχήμα 3.3: k-means με  $k = 2, d = 2, n = 100, MSE = 284.671$ . [10]

---

#### Algorithm 1 K-Means pseudo code

---

```

1: Choose initial centers for clusters K;
2: while the clusters are changing do
3:   Reassign the data points and set  $Ktemp = 0$ ;
4:   for all Data points  $n$  do
5:     Assign data point  $n_i$  to the cluster  $k_j$  whose center is closest;
6:      $Ktemp_j += n_i$ 
7:   end for
8:   Update the cluster centers;
9:   for j: 1 to k step 1 do
10:     $r_j$  number of points in  $Ktemp_j$ ;
11:     $K_j = \frac{Ktemp_j}{r_j}$ ;
12:   end for
13: end while

```

---

Στο Βίμα 1 του Αλγορίθμου 1 γίνεται η αρχικοποίηση είτε με τυχαίο τρόπο (κάθε cluster παίρνει τιμές από ένα τυχαίο σημείο) είτε με κάποια στρατηγική. Στην παρούσα διπλωματική επιλέχθηκε η στρατηγική KKZ [8] η οποία έχει μεγαλύτερη πολυπλοκότητα από την τυχαία αλλά οδηγεί τον αλγόριθμο k-means σε μικρότερο σφάλμα. Ο αλγόριθμος αρχικοποίησης παραλληλοποιήθηκε με OpenMP και επιτεύχθηκαν οι επιδόσεις του Πίνακα 3.1. Πρέπει να σημειωθεί πως ο KKZ εκτός από τα αρχικά δεδομένα δίνει και τα αποτελέσματα της πρώτης επανάληψης του k-means άρα στον χρόνο της Random προστίθεται και το πρώτο iteration. Στον Πίνακα 3.2 φαίνεται η συμπεριφορά του k-means με τυχαία και KKZ αρχικοποίηση και παρατηρείται η κυριαρχία του KKZ όσο αναφορά το σφάλμα.

Αρχικοποίηση	1	2	4	6
KKZ	90.5	63.7	42.9	36.7
Random	61.5	31.2	16.6	12.3

Πίνακας 3.1: Διάρκεια αρχικοποίησης (seconds) για KKZ,Random με 1,2,4,6 threads.

Τύπος	$d$	$n$	$k$	KKZ 5	Random 5	KKZ	End	Random	End
IntraY	16	100000	65536	0.437	2.710	0.434	29	2.709	9
IntraUV	16	100000	65536	0.182	0.945	0.181	33	0.944	7
InterY	16	100000	65536	0.168	1.095	0.167	32	1.094	10
InterUV	16	100000	65536	0.071	0.464	0.071	27	0.463	7

Πίνακας 3.2: Συμπεριφορά του k-means για Random,KKZ αρχικοποίηση με  $k = 65536$ ,  $d = 16$ ,  $n = 100000$ . Φαίνονται το MSE στην επανάληψη 5 και στο σημείο τερματισμού.

Η πιο σημαντική βελτιστοποίηση επιτεύχθηκε στο σημείο της αναζήτησης του κοντινότερου cluster στο Βίμα 5 του Αλγορίθμου 1. Στην απλή εκδοχή του αλγορίθμου για κάθε data point σαρώνονται όλα τα clusters για να βρεθεί το κοντινότερο (full search). Στην παρούσα διπλωματική, χρησιμοποιήθηκε ο αλγόριθμος FastNN (Fast Nearest Neighbor) [9] ο οποίος οργανώνει τις αποστάσεις σε ένα δυαδικό δέντρο. Ο FastNN επιτρέπει να γίνουν περίπου  $\log_2 k$  αναζητήσεις (πολυπλοκότητα αναζήτησης σε δυαδικό δέντρο), διαφορετικά θα έπρεπε να γίνουν  $k$  αναζητήσεις. Αυτό οδηγεί σε μεγάλη επιτάχυνση του προβλήματος της αναζήτησης που είναι το κύριο σημείο συμφόρησης του αλγορίθμου. Σε συνδυασμό με την παραλληλοποίηση στο Βίμα 4 με OpenMP επέτρεψε το τρέξιμο μεγάλων πειράματων σε εύλογο χρονικό διάστημα. Παραδοσιακά ο αλγόριθμος k-means ο οποίος είναι το απαραίτητο στάδιο για τη χρήση VQ, απαιτούσε τόσο πολύ χρόνο καθιστώντας το ανεφάρμοστο. Στην παρούσα διπλωματική προσπαθήθηκε και σε μεγάλο βαθμό επιτεύχθηκε

### 3. Θεωρία Πληροφοριών

---

η χρήση k-means με τον συνδυασμό αλγορίθμων (FastNN, KKZ) και υπολογιστικών τεχνικών (OpenMP). Οι επιταχύνσεις που επιτεύχθηκαν με την χρήση του FastNN και της OpenMP παρουσιάζονται στον Πίνακα 3.3.

Αναζήτηση	1	2	4	6
FastNN	9.7	6.1	4.4	3.2
Simple	61.4	31.2	16.6	12.3

Πίνακας 3.3: Διάρκεια μίας επανάληψης (seconds) για Full Search, FastNN με 1,2,4,6 threads με  $k = 65536, d = 16, n = 100000$

Για τα πειράματα της διπλωματικής χρονικοποιήθηκε ο εξοπλισμός με τα ακόλουθα χαρακτηριστικά:

- HP Blade Server
- OS Microsoft Windows Server 2008 R2 Datacenter.
- CPU 2xIntel Xeon E5-2600 @ 2.30Ghz οπού παρείχαν συνολικά 12 Threads + 12 με HyperThreading.
- RAM 32GB.



# Κεφάλαιο 4

## Παραγωγή Codebooks

### 4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστεί ο τρόπος με τον οποίο εκπαιδεύτηκε (training) ο αλγόριθμος k-means και με ποια δεδομένα (training set). Επίσης θα δειχθεί ο τρόπος εξαγωγής των δεδομένων από τον κωδικοποιητή H.264.

### 4.2 Επιλογή του training set

Όπως εξηγήθηκε στο Κεφάλαιο 2, τα δεδομένα που ένας κωδικοποιητής συμπιέζει είναι τα residuals και όχι τα pixels. Πρέπει να είναι ξεκάθαρο πως τα residuals μαζί με τα motion vectors μας δίνουν όλη την πληροφορία για να ανακατασκευάσουμε ένα καρέ. Αν δεν έχει παρεμβληθεί η κβαντοποίηση τότε το ανακατασκευασμένο με το αυθεντικό καρέ θα είναι πανομοιότυπα. Η χρήση του μετασχηματισμού δημιουργεί ανισοτροπία μεταξύ των συντελεστών του μετασχηματισμού, αυτήν την ανισοτροπία εκμεταλλεύεται η βαθμωτή κβαντοποίηση. Στην περίπτωση της κβαντοποίησης διανυσμάτων έχει δειχθεί ότι ο μετασχηματισμός που είναι ισοδύναμος με στροφή σε χώρο  $d$  διαστάσεων δεν επηρεάζει το MSE που προκύπτει κατά την διαδικασία του clustering [3], συνεπώς κάποιος θα μπορούσε να επιλέξει την κβαντοποίηση residuals ή την κβαντοποίηση συντελεστών μετασχηματισμού με ίδια αποτελέσματα. Καθότι όμως η πράξη του μετασχηματισμού και του αντιστρόφου του έχουν υπολογιστική πολυπλοκότητα επιλέχθηκαν τα residuals να αποτελούν το training set του k-means και όχι οι συντελεστές του μετασχηματισμού.

Μέχρι τώρα έχει γίνει σαφές πως για κάθε καρέ διάστασης  $W*H*1.5$  pixels υπάρχουν  $W*H*1.5$  residuals. Επομένως είναι εφικτό να τεμαχίσουμε το καρέ σε  $m*m$  blocks, το  $m$  έχει τον μοναδικό περιορισμό πως πρέπει να είναι μικρότερο από την διάσταση του block του encoder. Στον H.264 στον οποίο βασίστηκε η παρούσα διπλωματική το  $blocksize \in [2 * 2, 16 * 16]$  και μπορεί να ρυθμιστεί ανάλογα. Η διάσταση που επιλέχθηκε είναι το  $m = 4$  που πληροί το κριτήριο "block size" και είναι η προεπιλεγμένη τιμή block size στον H.264. Έστω ένα καρέ ανάλυσης  $720*480$  με

#### 4. Παραγωγή Codebooks

---

YUV420 τότε συνολικά έχουμε  $720 * 480 = 345.600$  pixels και  $\frac{345.600}{4*4} = 21.600$  training vectors για το Y και ακόμα 10.800 για το UV.

Τα βίντεο από τα οποία θα προέκυψε το training set έπρεπε να επιλεχθούν προσεκτικά γιατί χρειάζεται να πληρούν κάποιες προϋποθέσεις.

- Θα πρέπει να υπάρχει ένας ικανοποιητικός αριθμός από vectors  $n$  για να μπορέσουμε να έχουμε αντιπροσωπευτικά στατιστικά. Στην διπλωματική χρονικούτοιθηκαν 2600 καρέ από 10 διαφορετικά βίντεο ανάλυσης και το πλήθος των vectors φαίνεται στον Πίνακα 4.1. Κρίνονται αρκετά για κάθε codebook γιατί κατα μέσο όρο κάθε cluster έχει  $\frac{n}{k}$  vectors που του αντιστοιχούν.
- Το περιεχόμενο των βίντεο παιζει καθοριστικό ρόλο για το PSNR του VQ όταν δοκιμάζεται σε βίντεο εκτός του training set. Για παράδειγμα εάν το training set μας δε περιλαμβάνει σκηνές που απεικονίζουν βουνά τότε αν κωδικοποιηθεί ένα βίντεο που περιέχει σκηνές από βουνά θα είχαμε υψηλό σφάλμα κβαντοποίησης το οποίο συνεπάγεται χαμπλή μετρική PSNR (φαινόμενο mismatch). Τα 10 βίντεο που χρονικούτοιθηκαν είχαν διαφορετικές σκηνές και μπορούμε να δούμε κάποια στιγμιότυπα στο Σχήμα 4.1.

Θεωρητικά το καλύτερο training set θα ήταν αντιπροσωπευτικά στιγμιότυπα που κυκλοφορούν στον πλανήτη (talk shows, ταινίες, αθλήματα κ.τ.λ) αλλά τότε η πολυπλοκότητα του προβλήματος αυξάνεται σε επίπεδα που οι σημερινοί υπολογιστές δεν μπορούν να λύσουν σε εύλογο χρονικό διάστημα. Τα 10 βίντεο προέρχονται από το VQEG Group [?].



Σχήμα 4.1: Training βίντεο. [5]

### 4.3 Εξαγωγή training set από τον H.264

Η εξαγωγή των residuals από τον H.264 έγινε από τον decoder και όχι από τον encoder. Αυτή η επιλογή έγινε γιατί ο κώδικας του encoder είναι πιο περίπλοκος από αυτόν του decoder. Επίσης αναφέρθηκε στο Κεφάλαιο 2 ότι ο encoder κάνει δοκιμές για να βρει το καλύτερο Intra Prediction Mode αν το καρέ είναι I, ενώ ψάχνει να βρει σε προηγούμενα/επομένα καρέ το καλύτερο match ώστε να βγάλει το Motion Vector για τα P,B. Η παραπάνω διαδικασία δυσκόλεψε τον εντοπισμό του σημείου που βρίσκονται τα residuals του καλύτερου mode.

Στον decoder η διαδικασία ήταν αρκετά απλή. Απλά έπρεπε να βρεθεί το σημείο που ο decoder αποκωδικοποιεί τα macroblocks, και ειδικότερα την ανακατασκευή των residuals σε κάθε macroblock το οποίο αποτελεί το εύκολα προσβάσιμο σημείο για επεξεργασία. Στην συνέχεια το macroblock διάστασης 16x16 με τα residuals χωρίζονται σε block διάστασης dxz και γράφονται στο αρχείο. Το επόμενο που έγινε είναι να μπορεί να χωρίζει τα vectors σε I,P,B και να τα γράφει σε ξεχωριστά αρχεία καθώς επίσης και να ξεχωρίζει τα Y,UV. Ακόμα ένα στοιχείο που προστέθηκε είναι να υπάρχει η δυνατότητα να εξαχθεί ολόκληρο καρέ διάστασης όσο το βίντεο εισόδου. Τέλος το configuration file του JM H.264 [5] τροποποιήθηκε έτσι ώστε να μπορούμε να δώσουμε τις παρακάτω επιλογές.

- ResidualsFileY το αρχείο εξόδου των residuals του Y.
- ResidualsFileUV το αρχείο εξόδου των residuals του UV.
- ResidualsDims η διάσταση των residuals.
- KeepI,B,P διακόπτες που ενεργοποιούν την εξαγωγή μόνο τον I,P,B αντίστοιχα.
- ResidualsMode αν είναι 0 τότε δεν γίνετε καθόλου εξαγωγή. Αν είναι 1 τότε γίνετε εξαγωγή για είσοδο στον k-means. Αν είναι 2 γίνετε εξαγωγή ολόκληρων καρέ.

Αφού η διαδικασία γίνεται στον decoder θα έπρεπε αρχικά να συμπιέσουμε το βίντεο. Η συμπίεση που έγινε ήταν σε lossless mode με την λειτουργία FReXT του H.264 γιατί στον decoder θέλαμε τα residuals χωρίς σφάλμα. Η συμπίεση έγινε δύο φορές με δύο διαφορετικά GOP.

- Για να παραχθεί το Intra training set συμπιέσαμε τα 2600 καρέ με GOP I-I-I-.... και αποσυμπιέσαμε με KeepI 1 και ResidualsMode 1.
- Για να παραχθεί το Inter training set συμπιέσαμε τα 2600 καρέ με GOP I-B-B-P-B-P-... και αποσυμπιέσαμε με KeepP 1 και ResidualsMode 1.

Συνεπώς παράχθηκαν 4 training set τα Intra Y,Intra UV,Inter Y,Inter UV διάστασης 16 και πλήθους αρκετά μεγάλου ώστε να μπορούν να εξαχθούν καλά codebooks.

## 4.4 K-means

Ο k-means εφαρμόστηκε στα training set που φαίνονται στον Πίνακα 4.1 και παρατηρούμε πως τα πειράματα έτρεχαν για 23 μέρες. Το  $k = 65536$  επιλέχθηκε για να έχουμε  $VQ_{indices}$  στο διάστημα  $[0,65.535]$  έτσι ώστε να χρειαζόμαστε ακριβώς 2 bytes (ασυμπίεστο) για να αποθηκεύσουμε ένα  $VQ_{index}$ , και γιατί αυτό αντιστοιχεί σε μέγιστη εντροπία  $1bit/pixel$  δηλαδή δίνει ένα PSNR υψηλής ποιότητας. Η εντροπία είναι διαιρεμένη με την διάσταση οπότε αντιπροσωπεύει το κάθε residual. Τόσο η εντροπία όσο και η ποιότητα είναι υπολογισμένα στο training set.

Τύπος	$d$	$n$	$k$	$\frac{n}{k}$	Εντροπία	PSNR(dB)	Επαναλήψεις	Διάρκεια
IntraY	16	56.160.000	65.536	856,9	0,712229	33.6	3249	12154
IntraUV	16	28.080.000	65.536	428,5	0,743071	42.1	2697	3119
InterY	16	42.117.616	65.536	642,7	0,692577	40.5	3270	9120
InterUV	16	21.058.808	65.536	321,3	0,707785	48.1	4221	8509

Πίνακας 4.1: Πληροφορίες για τα codebooks.

## 4.5 Υπό συνθήκη εντροπία

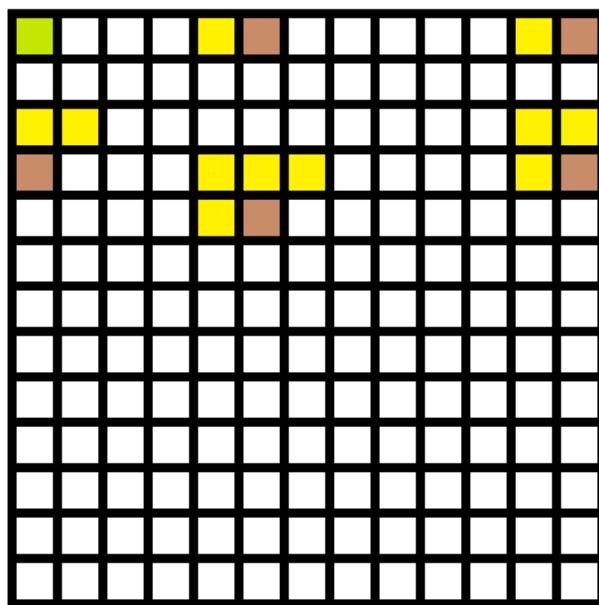
Είναι γνωστό από την Θεωρία Πληροφοριών (Θεώρημα 2.6.5 του βιβλίου [2]) ότι αν υπάρχουν δύο τυχαίες μεταβλητές  $X, Y$  τότε η πληροφορία που παρέχεται από την  $Y$  μπορεί μόνο να μειώσει την εντροπία της  $X$ , ισχύει δηλαδή πως  $H(X|Y) \leq H(X)$  με την ισότητα να ισχύει μόνο όταν οι  $X, Y$  είναι ανεξάρτητες. Αυτό συμβαίνει μόνο όταν διατίθεται η πληροφορία για το  $Y$  χωρίς κάπιο επιπλέον κόστος δηλαδή που τα  $X, Y$  είναι κομμάτι στοχαστικής διαδικασίας.

Για να δοκιμαστεί και να χρησιμοποιηθεί αυτό το θεώρημα το οποίο συνεπάγεται την μείωση της εντροπίας όπως υπολογίστηκε στον Πίνακα 4.1 έγιναν τα παρακάτω βήματα. Παράχθηκε το κάθε training set με την μορφή καρέ τρέχοντας τον decoder με *ResidualsMode = 2*. Για το συγκεκριμένο πείραμα τον ρόλο της μεταβλητής  $Y$  (συνθήκη) παίζει ο μέσος όρος της ενέργειας των ίδιων κβαντισμένων block όπως φαίνεται στο Σχήμα 4.2. Καθότι τα ίδια κβαντισμένα block είναι γνωστά τόσο στον encoder όσο και στον decoder η τιμή της  $Y$  υπολογίζεται χωρίς την αποστολή επιπλέον πληροφορίας. Τα  $y_i$  είναι περιοχές ενέργειας όπως φαίνεται στο Σχήμα 4.3 οι οποίες παράχθηκαν ταξινομώντας με βάση την ενέργεια και χωρίζοντας σε 8 ισοπίθανες περιοχές. Εεκινώντας από πάνω αριστερά, κβαντοποιούνται τα block και κρατιούνται στατιστικά για το ποια είναι η κατηγορία του τρέχοντος block ( $S_1 \dots S_8$ ) με βάση την ενέργεια των γειτόνων του. Έτσι δημιουργούνται οχτώ νέες υπό συνθήκη κατανομές  $P_1 \dots P_8$  πιθανότητας μεγέθους 65536, καθεμία μια από αυτές έχει δικιά της πιθανότητα και την δική της εντροπία.

#### 4. Παραγωγή Codebooks

---

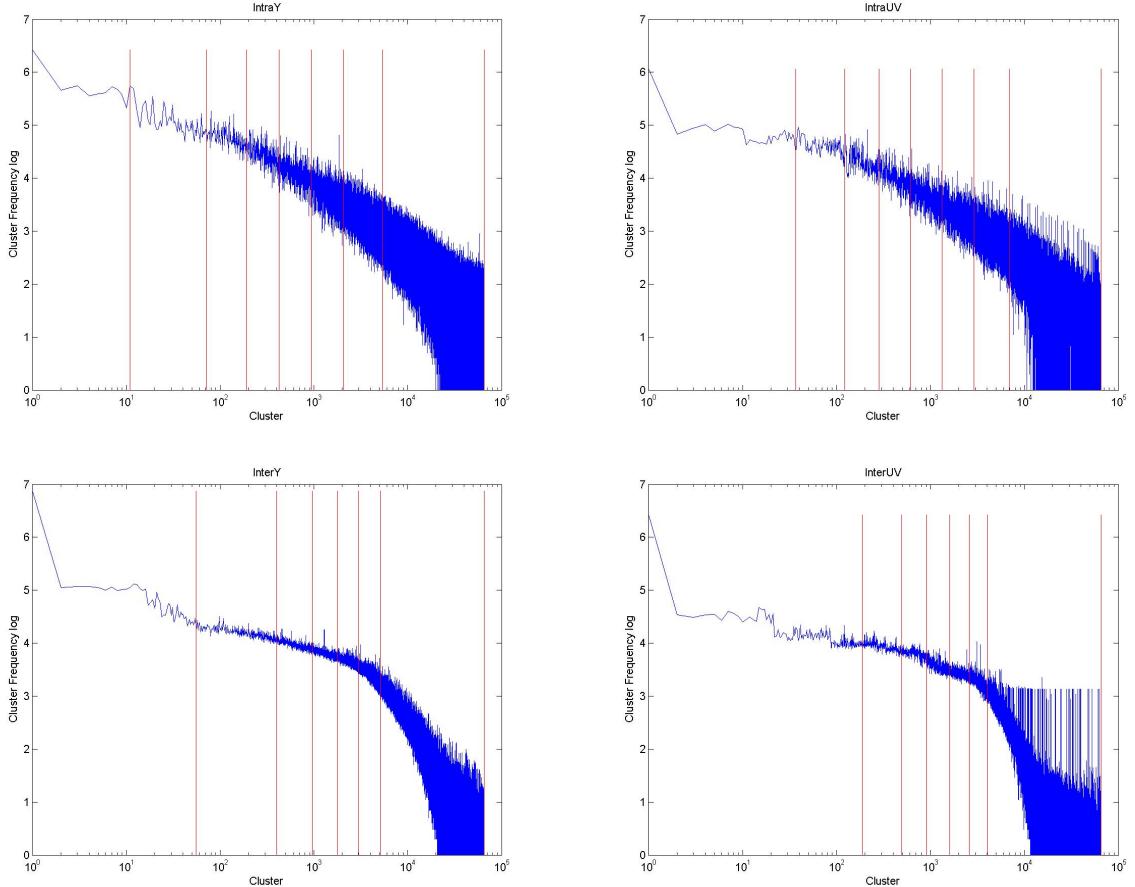
Αυτό το πείραμα με λίγα λογία λέει ότι εφόσον είναι γνωστή η ενέργεια των κβαντοποιημένων block τότε είναι γνωστή και η κατηγορία που ανήκει το τρέχον block κάτι το οποίο μειώνει την αβεβαιότητα για το τρέχον block, συνεπώς αποτείται λιγότερη πληροφορία. Έτσι γίνεται κωδικοποίηση (entropy encoding) με βάση την πιθανότητα του συγκεκριμένου cluster στην συγκεκριμένη κατηγορία ενέργειας. Το πείραμα πέτυχε καλύτερη εντροπία όπως φαίνεται στον Πίνακα 4.2 σε σχέση με αυτήν του Πίνακα 4.1 και αυτό συνέβη γιατί τα γειτονικά block μοιάζουν μεταξύ τους, άρα έχουν κοντινές ενέργειες. Επομένως έχουν μεγάλη εξάρτηση μεταξύ τους με κριτήριο την ενέργεια. Η χρήση γειτονικής πληροφορίας σε μια στοχαστική διαδικασία ονομάζεται context.



Σχήμα 4.2: Με καφέ είναι το τρέχον μπλοκ για κάθε περίπτωση και με κίτρινο από ποια γειτονικά βγαίνει ο μέσος όρος

Τύπος	Υπό Συνθήκη Εντροπία
IntraY	0.5411
IntraUV	0.6434
InterY	0.5443
InterUV	0.6314

Πίνακας 4.2: Υπό συνθήκη εντροπία με την χρήση contexts.



Σχήμα 4.3: Οι κόκκινες γραμμές δείχνουν τις περιοχές ενέργειας που αντιστοιχούν σε κάθε κατηγορία ενώ με μπλε απεικονίζεται η συχνότητα του κάθε cluster.

Αξίζει να παρατηρήσουμε ότι υπάρχουν λίγα στο πλήθος αλλά με μεγάλη πιθανότητα clusters μικρής ενέργειας (αριστερό μέρος), ενώ υπάρχουν πολλά με μικρή πιθανότητα clusters μεγάλης ενέργειας (δεξί μέρος).



# Κεφάλαιο 5

## VQ H.264

### 5.1 Εισαγωγή

Στο τελευταίο κομμάτι αυτής της διπλωματικής έγινε η ενσωμάτωση του VQ (Vector Quantization) στον H.264 ώστε να μπορούμε να κάνουμε encoding και decoding με βάση τα VQ codebooks. Το να αντικατασταθεί όλος ο μηχανισμός κβαντοποίησης-μετασχηματισμού του encoder H.264 και ο αντίστοιχος μετασχηματισμός αντίστροφου μετασχηματισμού, κβαντοποίησης του decoder αποδείχθηκε αρκετά περίπλοκο και η ιδέα εγκαταλείφθηκε γρήγορα. Αντί για αυτό επιλέχθηκε να γίνει μια άλλη τροποποίηση η οποία αφήνει τον H.264 να κάνει όλες τις λειτουργίες του αλλά κάπου στο ενδιάμεσο γίνεται παρέμβαση για να τροποποιηθούν τα residuals με το VQ.

### 5.2 Encoding

Αφού πραγματοποιείται η Prediction ο H.264 συνεχίζει με μετασχηματισμό και κβαντοποίηση των residuals. Έπειτα κάνει αντίστροφο μετασχηματισμό και αντίστροφη κβαντοποίηση (την διαδικασία που κάνει ο decoder) για να μπορεί να ανακατασκευάσει τα pixels με το σφάλμα που εισήχθη από την κβαντοποίηση. Αυτό είναι αναγκαίο διότι ο decoder γνωρίζει μόνο τα κβαντοποιημένα residuals αρά στον encoder θα πρέπει να παίρνουμε αποφάσεις μόνο με τα κβαντοποιημένα residuals. Με τον όρο αποφάσεις εννοείται ότι η επιλογή του καλύτερου Motion Vector στον encoder θα γίνει με βάση τα ανακατασκευασμένα pixel. Επομένως κάπου στον encoder υπάρχει ένα σημείο που ανακατασκευάζει τα pixels, το οποίο βρίσκεται στο Σχήμα 2.7 στο σημείο μεταξύ Scaling & Inv. Transform και Deblocking Filter. Επίσης αυτό το Σχήμα φανερώνει ότι τα reconstructed residuals χρησιμοποιούνται για να γίνεται το Motion Estimation.

Η συνάρτηση που ο encoder H.264 κάνει την πράξη  $Pixel_{reconstructed_i} = Residual_{reconstructed_i} + Pixel_{reference_i}$  βρίσκεται στο αρχείο blk\_prediction.c και ονομάζεται sample\_reconstruction. Δέχεται σαν ορίσματα έναν pointer στην ανακατασκευασμένη εικόνα, το μέγεθος του

## 5. VQ H.264

---

block καθώς και την θέση του στην εικόνα, τα Reference Pixels και τα Reconstructed Residuals.

Η τακτική που ακολουθήθηκε ήταν να κωδικοποιηθεί το βίντεο με  $QP = 0$  (σχεδόν καθόλου απώλεια λόγω βαθμωτής κβαντοποίηση) άρα τα  $ReconstructedResiduals = OriginalResiduals$ . Αρά στην ουσία επιτεύχθηκε ο στόχος της αφαίρεσης του μηχανισμού μετασχηματισμού-κβαντοποίησης. Μετά από αυτό το βήμα, τροποποιήθηκε η συνάρτηση `sample_reconstruction` έτσι ώστε προτού την πράξη ανακατασκευής των pixels να εισάγονται τα residuals του VQ. Αυτό επιτεύχθηκε πολύ απλά με τον Αλγόριθμο 2 όπου για την αναζήτηση του minimum distance χρησιμοποιήθηκε και εδώ ο αλγόριθμος FastNN για να επιταχύνει την διαδικασία του encoding. Μετά από αυτό το βήμα το καρέ ανακατασκευάζεται με βάση τα  $VQ_{residuals}$ , επομένως όλες οι αποφάσεις παίρνονται με βάση αυτά.

Το επόμενο που έγινε είναι να αποθηκεύονται τα τελικά  $VQ_{indices}$  σε ένα αρχείο. Όπως αναφέρθηκε για κάθε macroblock ο H.264 encoder δοκιμάζει όλα τα επιτρεπόμενα modes μέχρι να βρεθεί το καλύτερο, αυτό σημαίνει πως η συνάρτηση `sample_reconstruction` θα κλιθεί όσες φόρες είναι και το πλήθος των επιτρεπόμενων modes ανά block. Επομένως χρειάζεται να κρατώνται τα προσωρινά  $VQ_{indices}$  που γράφει ο Αλγόριθμος 2 και κάθε φόρα που βρίσκεται ότι ένα mode είναι καλύτερο από το προηγούμενο να αντιγράφονται σε ένα global πίνακα που κρατάει τα καλύτερα VQ indices για κάθε macroblock.

Αφού τελειώσει η διαδικασία αυτή για κάθε macroblock τα  $VQ_{indices}$  γράφονται σε ένα αρχείο. Στην παρούσα διπλωματική, τα  $VQ_{indices}$  που γράφονται για κάθε Y macroblock είναι  $\frac{16*16}{4*4} = 16$  όπου  $16*16$  η διάσταση του macroblock και  $4 * 4 = 16$  η διάσταση του codebook. Επομένως γράφονται  $2 * 16 = 32$  bytes ανά macroblock για το Y και  $2 * 8$  για το UV, το 2 bytes προκύπτει επειδή τα  $VQ_{indices} \in [0, 65535]$ . Οι αλλαγές φαίνονται στο Σχήμα 5.1.

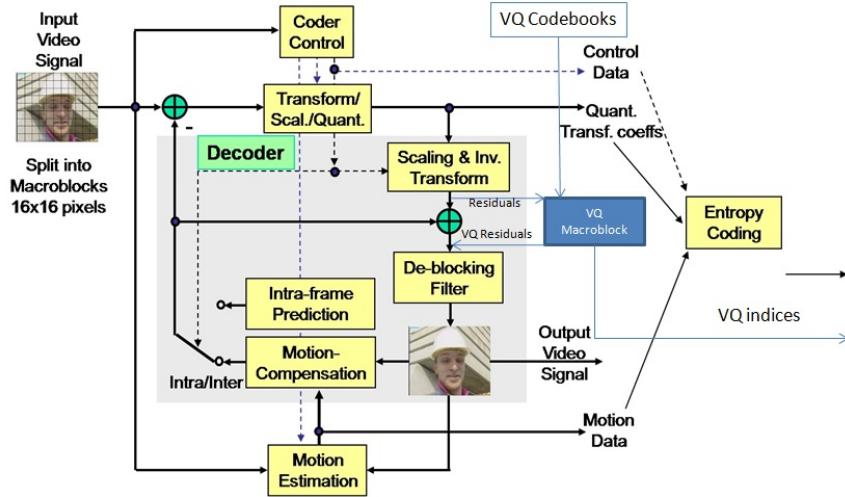
---

### Algorithm 2 VQ Algorithm

---

```
1: function quantize_mb(residuals,width,height,x,y,plane,mode)
2:   Load codebook for mode(Intra,Inter) and plane(Y,UV);
3:   for all subblocks with dimension dxd in block[x,y] do
4:     vqi = codebook index with minimum distance from subblock;
5:     Replace dxd residuals with codebook[vqi] data;
6:     Store vqi for subblock;
7:   end for
8: end function
```

---



Σχήμα 5.1: VQ H.264 Encoder Structure

Για να παραμετροποιείται ο JM H.264 encoder προστέθηκαν οι παρακάτω επιλογές στο configuration file.

- VQcodebook[YI,YB,YP,UVI,UVB,UVP] οπού δίνονται τα ονόματα των αρχείων των codebooks.
- VQindices οπού δίνεται το αρχείο των  $VQ_{indices}$ .
- VQdim οπου δίνεται η διάσταση  $d$  των codebooks. Αν είναι 0 το VQ απενεργοποιούται.
- VQcrlen οπού δίνεται το μήκος των codebooks  $k$ .
- Για να γίνει VQ πρέπει όλα τα QP να είναι 0.

Ο VQ H.264 παράγει και ένα αρχείο ίδιο με αυτό του JM H.264 με κατάλογο .264 που περιέχει κατά κύριο λόγο τα Motion Vectors και κάποια headers. Επίσης εμπεριέχονται και τα συμπιεσμένα residuals, όμως αυτά είναι άχροντα για τον decoder αφού διαβάζονται από τα  $VQ_{indices}$ .

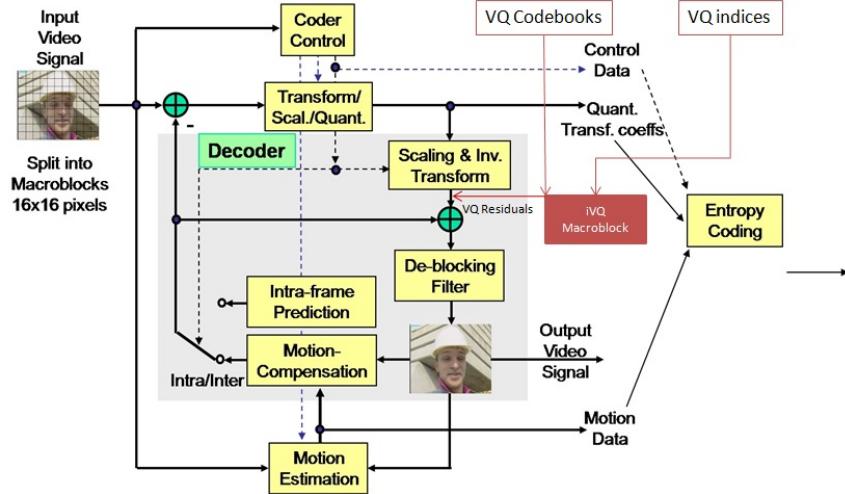
## 5.3 Decoding

Στον decoder όπως είναι αναμενόμενο τα πράγματα είναι πιο απλά. Αφού επιτραπεί στον decoder να κάνει την διαδικασία αντίστροφης κβαντοποίησης και μετασχηματισμού, στο ίδιο ακριβώς σημείο με τον encoder παρεμβάλλεται ο Αλγόριθμος 3. Η συνάρτηση αυτή αντιστοιχίζει 16  $VQ_{indices}$  που διαβάζει από το αρχείο για κάθε ένα από τα 16 subblocks του macroblock με 16  $VQ_{vectors}$  για τα Y macroblocks, ακόμα 4 για το 8x8 macroblock του U και αντιστοίχος για το V κατά

## 5. VQ H.264

---

τον Αλγόριθμο 3. Τελικά ο decoder παίρνει την κύρια πληροφορία που είναι τα residuals από τα  $VQ_{indices}$  ενώ τα headers, motion vectors από το αρχείο .264. Οι αλλαγές φαίνονται στο Σχήμα 5.2.



Σχήμα 5.2: VQ H.264 Decoder Structure

---

### Algorithm 3 Inverse VQ Algorithm

---

```

1: function quantize_mb(residuals,width,height,x,y,plane,mode)
2:   Load codebook for mode(Intra,Inter) and plane(Y,UV);
3:   for all subblocks with dimension dxd in block[x,y] do
4:     vqi = read  $VQ_{index}$  from file for this position[x,y];
5:     Replace dxd residuals with codebook[vqi] data;
6:   end for
7: end function

```

---

Για να παραμετροποιείται ο JM H.264 decoder προστέθηκαν οι παρακάτω επιλογές στο configuration file.

- VQcodebook[YI,YB,YP,UVI,UVB,UVP] οπου δίνονται τα ονόματα των αρχείων των codebooks.
- VQindices οπού δίνεται το αρχείο των  $VQ_{indices}$ .
- VQdim οπου δίνεται η διάσταση  $d$  των codebooks. Αν είναι 0 το VQ απενεγγοποιείται.
- VQcrlen οπού δίνεται το μήκος των codebooks  $k$ .

## Κεφάλαιο 6

### Μελέτη της Απόδοσης του VQ H.264

#### 6.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα παρουσιαστούν και θα αναλυθούν τα αποτελέσματα του VQ H.264, επίσης θα γίνει σύγκριση με τις επιδόσεις του JM H.264. Επίσης θα δειχθεί ότι με την χρήση VQ μπορεί να βελτιωθεί η πολυπλοκότητα του decoder σε σχέση με αυτή του JM H.264.

#### 6.2 Encoding με VQ H.264

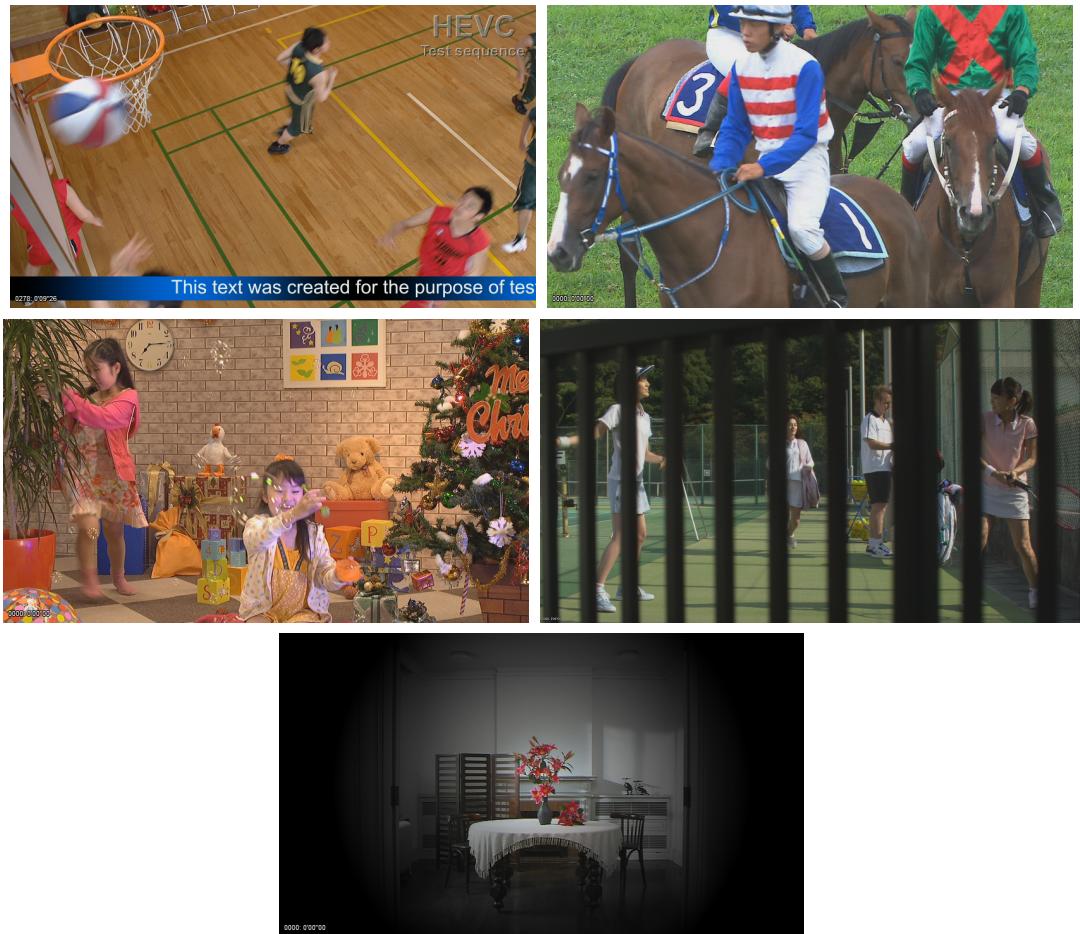
Για τις δοκιμές χρησιμοποιήθηκαν 5 βίντεο που δεν είχαν συμπεριληφθεί στο training set και τα στιγμιότυπα τους φαίνονται στο Σχήμα 6.1. Το PSNR που ο VQ H.264 πέτυχε στα test video φαίνεται στον Πίνακα 6.1. Αξίζει να σημειωθεί αποτελούν υλικό δοκιμής για το H.265 standard [16].

Test Video	PSNR I (Y/U/V)dB	PSNR P (Y/U/V)dB	PSNR B (Y/U/V)dB
test1	35.15/39.72/39.67	41.16/43.91/43.87	43.00/45.14/45.22
test2	35.51/41.10/42.83	38.87/41.53/43.28	40.38/43.01/44.71
test3	30.70/39.86/41.00	36.68/42.64/43.76	38.00/43.74/44.90
test4	44.12/47.36/47.91	46.00/47.20/47.75	46.55/47.42/48.05
test5	37.01/50.12/48.70	44.00/49.90/49.65	44.78/50.46/50.29

Πίνακας 6.1: PSNR των Test videos με κωδικοποίηση στον VQ H.264

## 6. Μελέτη της Απόδοσης του VQ H.264

---



Σχήμα 6.1: Βίντεο που δοκιμάστηκαν να γίνουν encoding με τον VQ H.264.

## 6.3 Encoding με JM H.264

Για να μπορούν να συγκριθούν τα αποτελέσματα των δύο encoder θα πρέπει να συμπιεστούν τα ίδια βίντεο με τον JM H.264 με τις ίδιες παραμέτρους που δεν αφορούν το VQ και προσπαθώντας να σημειωθούν τα ίδια PSNR που πέτυχε και ο VQ H.264. Αυτό πραγματοποιήθηκε δοκιμάζοντας διάφορα QP για τα καρέ I,P,B. Επειδή στον VQ H.264 παρατηρείται ότι το U,V αποδίδει πολύ καλύτερα από το Y σε κάποια βίντεο χρειάστηκε να υπολογιστεί ο βεβαρημένος μέσος όρος  $avg = 0.66*PSNR_Y + 0.16*PSNR_U + 0.16*PSNR_V$  έτσι ώστε να ισχύει  $avg_{vq} = avg_{jm}$ . Τα αποτελέσματα φαίνονται στον Πίνακα 6.2. Η ποσοστιαία διαφορά ως προς τον JM φαίνεται στον Πίνακα 6.3 και θα μπορούσε να χαρακτηριστεί πολύ μικρή κάτι το οποίο καθιστά την μεταξύ τους σύγκριση δύκαια.

Test Video	PSNR I (Y/U/V)dB	PSNR P (Y/U/V)dB	PSNR B (Y/U/V)dB
test1	36.31/38.88/38.90	42.01/43.95/44.54	43.50/44.75/45.49
test2	37.31/38.43/40.00	40.50/41.14/42.24	40.50/41.45/42.54
test3	33.94/37.24/37.85	38.64/40.75/41.45	38.45/40.51/41.25
test4	45.18/46.83/47.86	45.75/47.13/48.18	46.10/47.29/48.21
test5	39.20/46.76/47.34	43.94/47.66/48.28	45.14/49.29/49.76

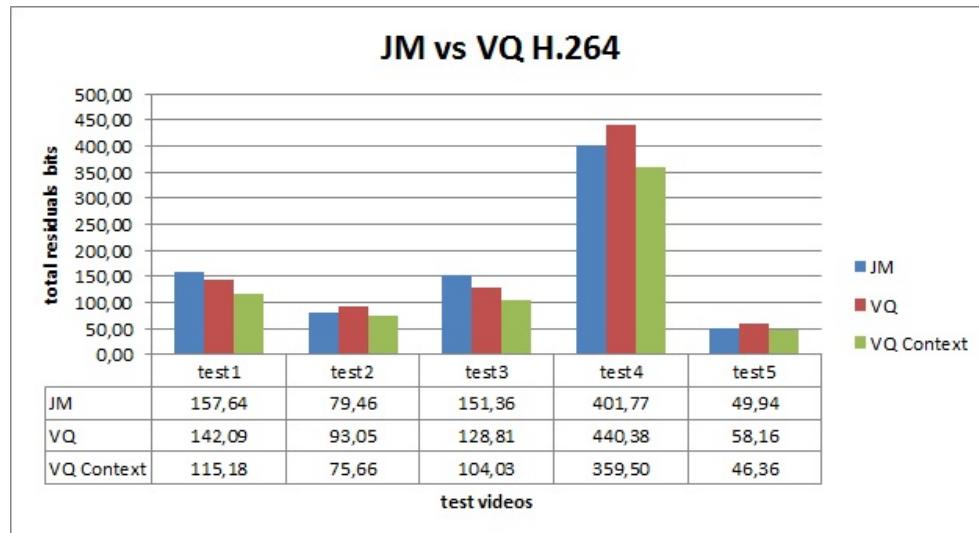
Πίνακας 6.2: PSNR των Test videos με κωδικοποίηση στον JM H.264

Test Video	% Diff I	% Diff P	% Diff B
test1	1,40	-1,40	-0,93
test2	0,83	-5,11	-4,23
test3	3,53	-5,21	-5,49
test4	1,35	-0,87	-0,18
test5	1,69	-5,38	-2,76

Πίνακας 6.3: Διαφορές του PSNR JM-VQ. Στα θετικά πρόσημα ο JM είναι καλύτερος από τον VQ ενώ στα αρνητικά πρόσημα το ανάποδο.

## 6.4 Αποτελέσματα των VQ H.264, JM H.264

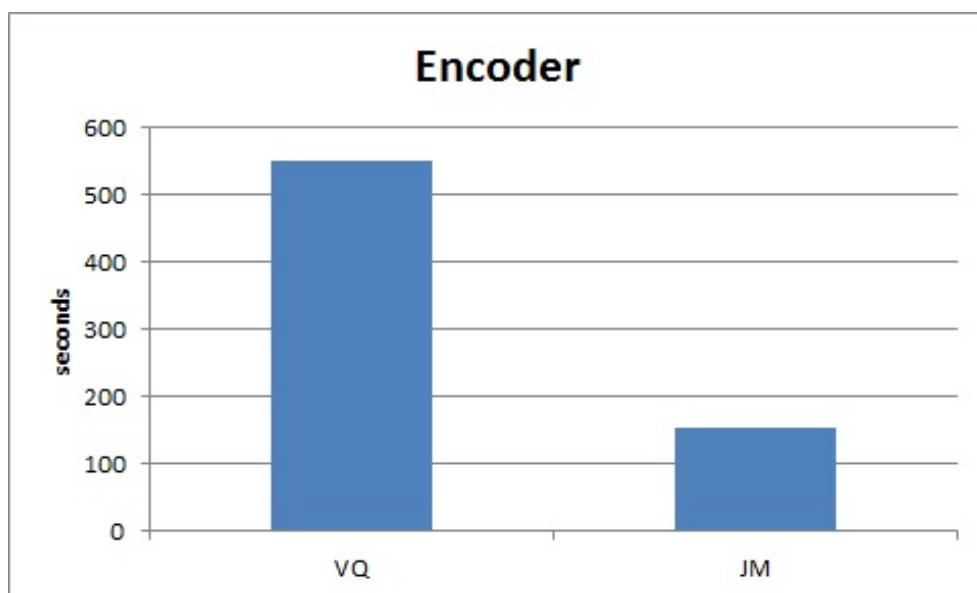
Για να γίνει η σύγκριση θα υπολογιστούν τα συνολικά bits που ο JM H.264 χρειάστηκε για να αποθηκεύσει τους κβαντοποιημένους συντελεστές του μετασχηματισμού. Αυτή η πληροφορία μας παρέχεται απευθείας από την έξοδο του encoder. Αυτό γίνεται γιατί ουσιωδώς τα  $VQ_{indices}$  αντιστοιχούν μόνο στην πληροφορία που παρέχουν τα residuals, όπως και οι συντελεστές του μετασχηματισμού. Ως γνωστόν στα standard συμπίεσης βίντεο όπως το H.264 αλλά και στα mpeg1,2,4 η διαδικασία της κβαντοποίησης οδηγεί μόνο σε μηδενικούς συντελεστές, κατά συνέπεια το συγκεκριμένο block δεν κωδικοποιείται και η παρουσία σηματοδοτείται με κάποιο συγκεκριμένο flag (CBP) στο .264 αρχείο. Στην περίπτωση που όλο το macroblock αποτελείται από μηδενικά block τότε δεν κωδικοποιείται καθόλου. Στο επόμενο βίντα υπολογίστηκε ο αριθμός των Skipped macroblock και αφαιρέθηκαν 16 Y vectors και 8 UV vectors για κάθε macroblock. Στο επόμενο βίντα πολλαπλασιάζεται η εντροπία είτε του Πίνακα 4.2 (με την χρήση context) είτε του Πίνακα 4.1 (χωρίς την χρήση context) με την διάσταση του VQ για να υπολογιστεί πόσα bits χρειάζονται για κάθε block  $dxd$ . Τέλος πολλαπλασιάζονται τα bits που χρειάζονται ανά block με τον αριθμό των non-skipped blocks και έτσι εκτιμάται το μεγέθος που θα χρειάζονται για την κωδικοποίηση των  $VQ_{indices}$  μετά από συμπίεσης. Αναμένεται μετά από χρήση είτε πινάκων Huffman είτε με την χρήση CABAC, να σημειωθούν τα αποτελέσματα τα οποία φαίνονται στο Σχήμα 6.2.



Σχήμα 6.2: Σύγκριση των bits του JM H.264 και VQ H.264 με context entropy και χωρίς context.

Εκτός από την σύγκριση της αποδοτικότητας της συμπίεσης έγινε προσπάθεια να συγκριθούν και οι χρόνοι κωδικοποίησης και αποκωδικοποίησης. Όπως αναφέρθηκε ο VQ H.264 κάνει όλη την άχροντη για αυτόν δουλειά του μετασχηματισμού,

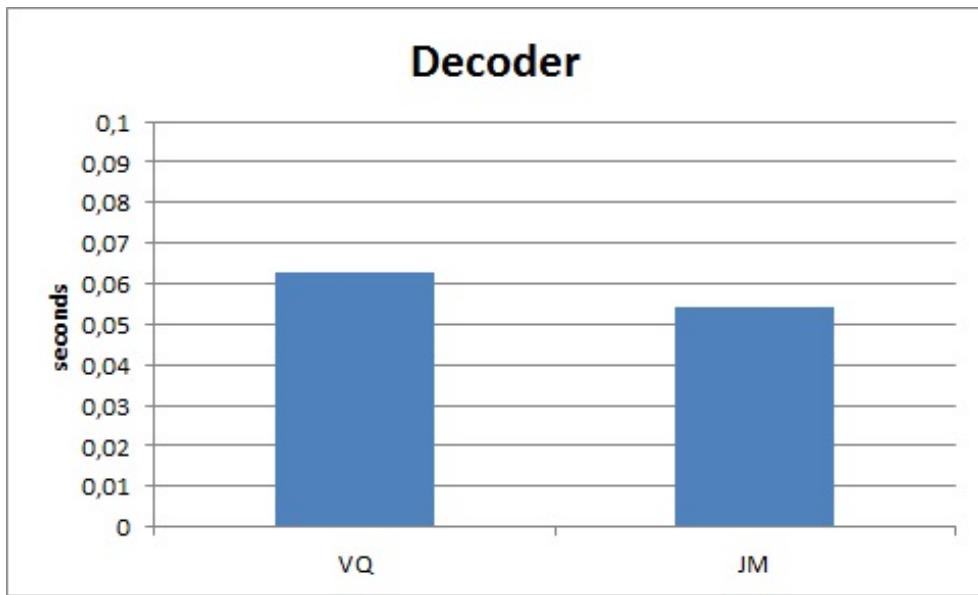
κβαντοποίησης, αντίστροφου μετασχηματισμού, αντίστροφης κβαντοποίησης. Συνεπώς, χρησιμοποιήθηκε το Intel VTune για να βρεθεί ο χρόνος αυτών των συναρτήσεων και να αφαιρεθούν από τον συνολικό. Έτσι έγινε μια καλή προσέγγιση των επιδόσεων του VQ H.264 και η σύγκρισή με τον JM H.264 φαίνεται στο Σχήμα 6.3 και στο Σχήμα 6.4. Παρατηρείται ότι στον VQ H.264 encoder υπάρχει αύξηση της πολυπλοκότητας το οποίο οφείλεται αποκλειστικά στην αναζήτηση FastNN. Στον VQ H.264 decoder παρατηρείται μια αναίσθητη διαφορά.



Σχήμα 6.3: Σύγκριση του χρόνου εκτέλεσης των encoder JM H.264 και VQ H.264.

## 6. Μελέτη της Απόδοσης του VQ H.264

---



Σχήμα 6.4: Σύγκριση του χρόνου εκτέλεσης των decoder JM H.264 και VQ H.264.

# Κεφάλαιο 7

## Συμπεράσματα

### 7.1 VQ H.264 vs JM H.264

Από το Σχήμα 6.2 βλέπουμε πως ο VQ H.264 χρησιμοποιεί κατά μέσο όρο 17% λιγότερα bits από τον JM H.264 στα βίντεο που δοκιμάστηκε. Επίσης φαίνεται να υπάρχει ακόμα μεγαλύτερο κέρδος στα βίντεο που περιέχουν γρήγορη κίνηση, πράγμα που τα κάνει δύσκολο να κωδικοποιηθούν, τέτοια βίντεο είναι τα test1,test3 με κέρδος 27% και 31% αντίστοιχα. Σε αντίθεση στα βίντεο test2,test4,test5 που δεν δυσκολεύονται τον encoder παρατηρείται ότι είναι μικρότερο.

Θα πρέπει να ληφθεί υπόψιν ότι το VQ έχει πολλά περιθώρια βελτίωσης αν μεγαλώσει το training set. Με το τρέχον training set παρατηρήθηκε πως σε κάποια βίντεο και ειδικά στα I frames δεν έχει καλή απόδοση ο VQ. Αυτό μπορεί να βελτιωθεί απλά και μόνο συμπεριλαμβάνοντας αυτά ή παραπλήσια βίντεο στο training set. Το θετικό σε αυτή την διαδικασία είναι ότι δεν χρειάζεται να γίνει τρέξιμο του training από την αρχή, αρκεί να δωθούν σαν αρχικά clusters αυτά που ήδη υπάρχουν και έτσι να μην γίνει η αρχικοποίηση αλλά και πολλά iterations του αλγορίθμου. Επομένως, ένας encoder που δουλεύει με VQ θα μπορεί να ανανεώνει τα codebooks του ανά κάποιο χρονικό διάστημα και θα επιλέγει αυτά που το βίντεο κωδικοποιήθηκε μέσω κάποιου version header μέσα στο βίντεο.

Ένα πρόβλημα που εμφανίζεται είναι η μεγάλη ανομοιομορφία του PSNR μεταξύ I,P,B αλλά και Y,UV. Δηλαδή στο βίντεο test1 το I είναι στα 35dB ενώ τα P,B στα 43dB,45dB αντίστοιχα. Αυτή η διαφορά είναι αρκετά μεγάλη και για να μειωθεί θα μπορούσαν να χρησιμοποιηθούν codebooks με διαφορετικό μέγεθος για κάθε συνιστώσα I,P,B|Y,UV. Έτσι λοιπόν αν χρησιμοποιηθούν Inter codebook μήκους 32768 για το test1 θα χρειάζονταν λιγότερα bits για να αποθηκευτούν τα  $VQ_{indices}$  αλλά ταυτόχρονα θα μειωνόταν και το PSNR. Επομένως, στο VQ το μέγεθος του codebook  $k$  λειτουργεί σαν ρυθμιστής ποιότητας παίζοντας τον ρόλο που παίζει το QP στο scalar quantization. Δεν πραγματοποιήθηκαν μετρήσεις πάνω σε αυτόν το τομέα, αλλά αφήνονται σαν μια πολύ σημαντική προσθήκη για την ολοκλήρωση του VQ H.264.

## 7. Συμπεράσματα

---

Όπως φαίνεται στο Σχήμα 6.3 ο VQ Encoder είναι 3.5 φορές πιο αργός από τον JM, κάτι που οφείλεται στο κόστος του αλγορίθμου FastNN. Αντιθέτως, ο VQ Decoder στο Σχήμα 6.4 είναι στα ίδια επίπεδα με τον JM Decoder. Αυτό που είναι σημαντικό είναι να η ύπαρξη γρήγορων decoders καθώς μπαίνουν σε φορητές συσκευές που ενδιαφέρει η ενεργειακή απόδοση. Έτσι ένας VQ Decoder μπορεί να βελτιστοποιηθεί χρησιμοποιώντας μικρές και γρήγορες μνήμες όπου θα αποθηκεύονται τα codebooks. Ένα ενεργοβόρο και ακριβό βοηθητικό κύκλωμα που επιταχύνει την διαδικασία του αντίστροφου μετασχηματισμού και της αντίστροφης κβαντοποίησης μετατρέπεται σε μια γρήγορη, χαμηλής κατανάλωσης, φθηνή μνήμη. Η μνήμη που χρειάζεται για να αποθηκευτούν τα VQ codebooks είναι αρκετά μικρή. Στην διπλωματική αυτή για ένα codebook χρειάζεται συνολικά  $k * d * 2 = 4MB$  με  $k = 65536$  και  $d = 16$  επομένως για τα 4 codebooks χρειαζόμαστε  $16MB$  τα  $2bytes$  προκύπτουν επείδη  $k = 65536 = 2^{16}$  άρα τα indices χρειάζονται  $16bits = 2bytes$  για να αποθηκευτούν.

## 7.2 Πιθανές προσθήκες

Επειδή το VQ έδειξε στην παρούσα διπλωματική εργασία ότι αξίζει παραπάνω προσοχή, παρατίθενται παρακάτω κάποιες προτάσεις που θα αυξήσουν την απόδοτικότητα του.

- Χρήση διαφορετικού μεγέθους codebooks για τις διάφορες συνιστώσες Y,UV, καθώς και παραγωγή codebooks για διάφορες ποιότητες ( $k$ ).
- Προσαρμογή του πυρήνα του JM H.264 έτσι ώστε στον encoder να μην γίνεται μετασχηματισμός και κβαντοποίηση αλλά απευθείας VQ και έπειτα να κωδικοποιούνται τα  $VQ_{indices}$  με την χρήση των contexts και την σωστή χρήση κωδικοποιητή εντροπίας. Με αυτόν τον τρόπο, ο decoder θα δέχεται μόνο ένα αρχείο με τα  $VQ_{indices}$  κωδικοποιημένα εντός του αρχείου H.264 και θα παράγει το αποκωδικοποιημένο βίντεο.
- Παραγωγή του training set από βίντεο που έχουν παραχθεί από VQ H.264, κατά αυτόν τον τρόπο αναμένεται παραγωγή καλύτερης ποιότητας codebook τα οποία μπορούν με την σειρά τους να παράγουν ακόμα καλύτερης ποιότητας training set τα οποία θα συγκλίνουν μετά από έναν συγκεκριμένο αριθμό επαναλήψεων.

Δοκιμή παραγωγής του training set από βίντεο που έχουν ήδη κωδικοποιηθεί με την χρήση VQ. Χρήση αυτών των νέων codebooks για να γίνει VQ και πάλι στο βίντεο για να δειχθεί αν έτσι βελτιώνεται η ποιότητα του VQ. Μπορεί αυτό το loop να χρειαστεί να γίνει παραπάνω από μία φόρες

# Βιβλιογραφία

- [1] Y. Adibelli, M. Parlak, and I. Hamzaoglu. A computation and power reduction technique for h.264 intra prediction. In *Digital System Design: Architectures, Methods and Tools (DSD), 2010 13th Euromicro Conference on*, pages 753–760, 2010.
- [2] Thomas M. Cover and Joy A. Thomas. *Elements of information theory*. Wiley-Interscience, New York, NY, USA, 1991.
- [3] Allen Gersho and Robert M. Gray. *Vector quantization and signal compression*. Kluwer Academic Publishers, Norwell, MA, USA, 1991.
- [4] Mrutyunjaya Hiremath. Macroblock partition mode.
- [5] iphome. JM h.264.
- [6] Petros Kalos. VQ h.264 repository. <https://github.com/petrkalos/jm>.
- [7] Petros Kalos. VQ training and statistics tools repository. <https://github.com/petrkalos/vq>.
- [8] I. Katsavounidis, C.-C. Jay Kuo, and Zhen Zhang. A new initialization technique for generalized lloyd iteration. *Signal Processing Letters, IEEE*, 1(10):144–146, 1994.
- [9] I. Katsavounidis, C.-C.J. Kuo, and Zhen Zhang. Fast tree-structured nearest neighbor encoding for vector quantization. *Image Processing, IEEE Transactions on*, 5(2):398–404, 1996.
- [10] Mathworks. K-means. <http://www.mathworks.com/help/stats/kmeans.html>.
- [11] multimedia.cx. Zig zag scan.
- [12] VSR Group of Chemnitz University of Technology. H.264 video codec - inter prediction.
- [13] Louis Scharf. Binary codes: Huffman codes for source coding.

- [14] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.
- [15] Yang Song. H.264 encoder structure.
- [16] Hannover University. Test videos.
- [17] Wikipedia. Arithmetic coding.
- [18] Wikipedia. Data compression.
- [19] Wikipedia. JPEG.
- [20] Wikipedia. Yuv.