

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^η 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
1 Εισαγωγή	1
1.1 Περιγραφή του Προβλήματος	1
1.2 Συμβολή της Εργασίας	2
1.3 Διάρθρωση της Διπλωματικής Εργασίας	2
2 Ψηφιακό βίντεο και τεχνικές συμπίεσης	3
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	10
2.8 Δομή του H.264	11
3 Θεωρία πληροφοριών	13
3.1 Εισαγωγή	13
3.2 Εντροπία	13
3.3 Κωδικοποιητές Εντροπίας	14
3.4 Αλγόριθμος clustering kmeans	16
4 Παραγωγή Codebooks	19
4.1 Εισαγωγή	19
4.2 Επιλογή του training set	19
4.3 Εξαγωγή training set από τον H.264	22
4.4 Kmeans	23

4.5 Εντροπία υπό συνθήκη	23
5 VQ H.264	27
5.1 Εισαγωγή	27
5.2 Encoding	27
5.3 Decoding	29
6 Μελέτη της Απόδοσης του VQ H.264	31
6.1 Εισαγωγή	31
6.2 Encoding με VQ H.264	31
6.3 Encoding με JM H.264	33
6.4 VQ H.264 vs JM H.264	34
7 Συμπεράσματα	37
7.1 VQ H.264 vs JM H.264	37
7.2 Πιθανές προσθίκες	38
Βιβλιογραφία	39

Κατάλογος πινάκων

1.1	Κυριότεροι κωδικοποιητές βίντεο. [15]	1
2.1	Πίνακας Κβαντοποίησης. [16]	9
2.2	Παράδειγμα συντελεστών DCT 8x8. [16]	9
2.3	Κβαντοποιημένοι συντελεστές DCT. [16]	10
3.1	Διάρκεια αρχικοποίησης (seconds) για KKZ,Random με 1,2,4,6 threads.	17
3.2	Συμπεριφορά του kmeans 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 Simple, FastNN με 1,2,4,6 threads με $k = 65536, d = 16, n = 100000$	18
4.1	Πληροφορίες για τα codebooks.	23
4.2	Εντροπία μετά την χρήση contexts	24
6.1	PSNR των Test videos με κωδικοποίηση στον VQ H.264	31
6.2	PSNR των Test videos με κωδικοποίηση στον JM H.264	33
6.3	Διαφορές του PSNR JM-VQ. Στα θετικά ο JM είναι καλύτερος από τον VQ ενώ στα αρνητικά το ανάποδο.	33

Κατάλογος σχημάτων

2.1	Στην πρώτη εικόνα φαίνεται ένα καρέ με όλες τις συνιστώσες ενώ παρακάτω ξεχωριστά το YUV για το ίδιο καρέ [17].	4
2.2	Διαμερισμός του macroblock στον H.264.[3]	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. [9]	7
2.5	Παραδείγμα χρήσης RLE	8
2.6	Σάρωση ZigZag. [8]	8
2.7	H.264 encoder structure. [12]	11
3.1	Δέντρο Huffman. [10]	14
3.2	Βήματα αριθμητικής κωδικοποίησης. [14]	15
3.3	kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$. [7]	16
4.1	Training βίντεο. [4]	21
4.2	Με καφέ είναι το τρέχων μπλοκ για κάθε περίπτωση και με κίτρινο από ποιά γειτονικά βγαίνει ο μέσος όρος	24
4.3	Οι κόκκινες γραμμές δείχνουν το σημείο που σταματάει μια κατηγορία ενώ οι μπλε το πλήθος των vectors που αντιστοιχούν στο cluster	25
6.1	Βίντεο που δοκιμάστηκαν να γίνουν encoding με τον VQ H.264. [13] . . .	32
6.2	Σύγκριση της απόδοσης PSNR του JM H.264 και VQ H.264 με context entropy και χωρίς context.	34
6.3	Σύγκριση του χρόνου εκτέλεσης των encoder JM H.264 και VQ H.264. . .	35
6.4	Σύγκριση του χρόνου εκτέλεσης των decoder JM H.264 και VQ H.264. . .	35

Περίληψη

Λέξεις Κλειδιά:

Κραντοποίηση Διανυσμάτων, Εντροπία, Θεωρία Πληροφοριών, Κωδικοποιητές Βίντεο, Κωδικοποιητές Εντροπίας, H.264, kmeans

Abstract

Keywords:

Vector Quantization, Entropy, Information Theory, Video Codec, Entropy Encoding, H.264, kmeans

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. Έτσι επιλέγεται να εισαχθεί ομοιόμορφος θόρυβος για να επιτευχθεί μεγαλύτερος λόγος συμπίεσης θυσιάζοντας την ποιότητα. Η κωδικοποίηση βίντεο υλοποιείται τόσο σε επίπεδο λογισμικού αλλά και υλικό. Οι προσωπικοί υπολογιστές συνήθιζαν να χρησιμοποιούν λογισμικό για την αναπαραγωγή βίντεο αλλά 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 βλέπουμε υλικό ειδικά σχεδιασμένο (hardware accelerators) για κωδικοποίηση/αποκωδικοποίηση βίντεο που βοηθά τον κύριο επεξεργαστή.

Χρονιά	Κωδικοποιητής	Διάφορες Χρήσεις	Λόγος Συμπίεσης
1993	MPEG-1	VCD	26:1
1995	MPEG-2	DVD	31:1
1999	MPEG-4	DivX,XViD	200:1
2003	H.264	BluRay,DVB-TS	400:1
2013	H.265	next generation H.264	550:1

Πίνακας 1.1: Κυριότεροι κωδικοποιητές βίντεο. [15]

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 ούμως κατά πολύ την πολυπλοκότητα τους. Πλέον μπορούμε με λίγα δεδομένα να έχουμε βίντεο καλής ποιότητας σε πολύ μεγάλες αναλύσεις. Ενδεικτικά για ένα ασυμπίεστο βίντεο ανάλυσης DVD (720x480) με

1. Εισαγωγή

διάρκεια 20sec και ρυθμό ανανέωσης 30Hz χρειαζόμαστε χώρο 296MB. Κάνοντας συμπίεση με τον H.264 θα έχουμε 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, με ποίες μιօρφές αυτό αποθηκεύεται, πώς οι κωδικοποιητές το αντιμετωπίζουν και ποίες είναι οι κύριες τεχνικές που χρησιμοποιούνται για την συμπίεση του.

Στο Κεφάλαιο 3 γίνετε 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 διπλωματικής.

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

Στο Κεφάλαιο 5 παρουσιάζεται η επέμβαση που έγινε στον H.264 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 αυτοί χρησιμοποιούν για να τα συμπιέσουν. Επίσης σε αυτό το κεφάλαιο θα αποσαφνιστεί γιατί το βίντα της κβαντοποίησης (εισαγωγή θορύβου) είναι αναγκαία για να έχουμε τόσο μεγάλους λόγους συμπίεσης.

2.2 Συστατικά στοιχεία του βίντεο

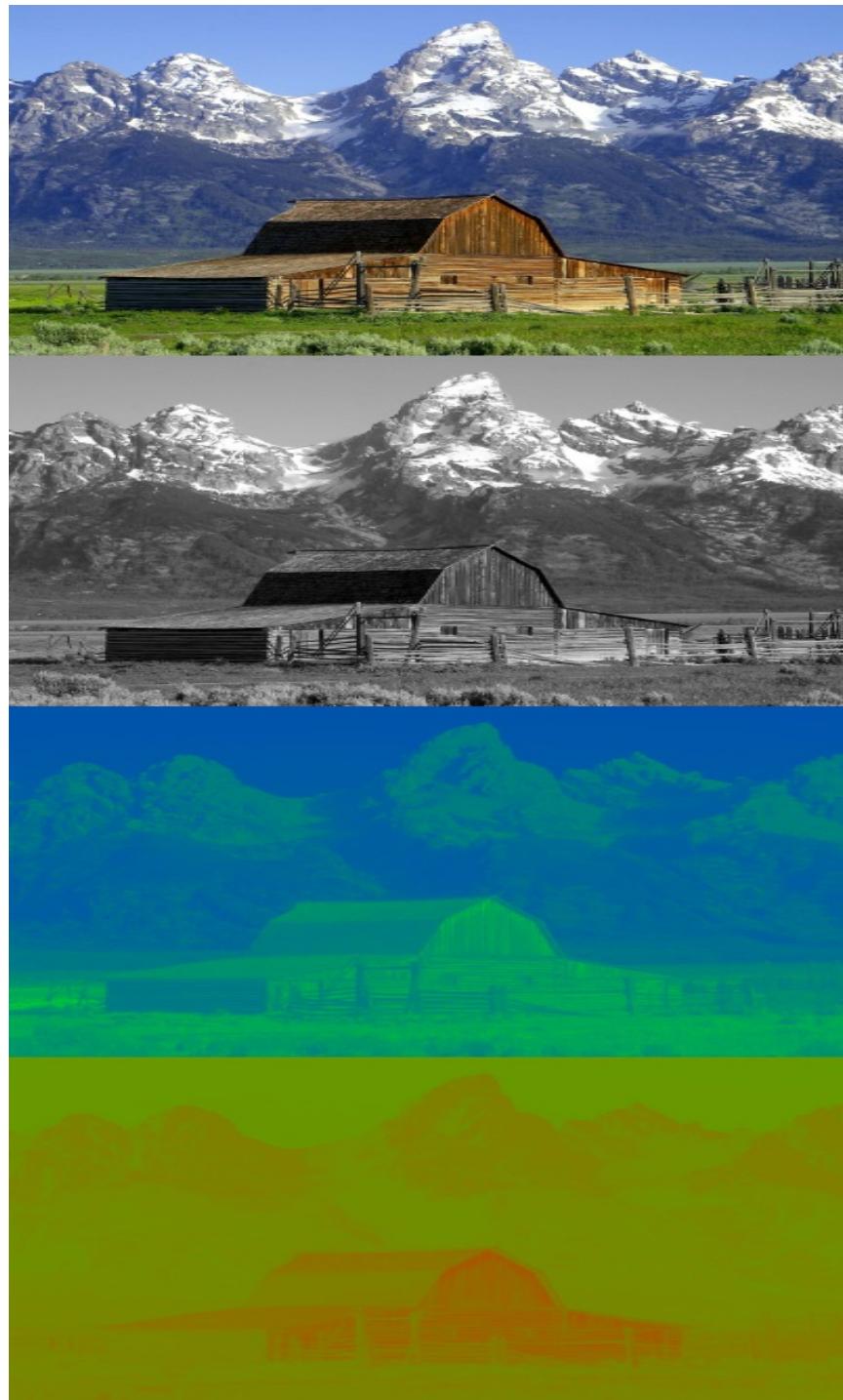
Το ψηφιακό βίντεο αποτελείται από μία σειρά καρέ που αναπαράγονται με σταθερό ρυθμό (συνήθως 25Hz ή 30Hz). Το κάθε καρέ απεικονίζεται σε ένα χώρο χρωμάτων που ονομάζεται YUV, οπού το Y είναι η φωτεινότητα και το U,V η χρωματικότητα. Η κάθε συνιστώσα στο ανθρώπινο μάτι φαίνεται όπως στο Σχήμα 2.1.

Σε κάθε σημείο του βίντεο (pixel) αντιστοιχίζεται μια τέτοια τιμή YUV. Υπάρχουν διάφοροι τρόποι που γίνεται αυτή η αντιστοίχηση με κυριότερες αυτές του YUV420 και YUV444. Η υποδεικνύει πως κάθε για κάθε τέσσερα pixel υπάρχουν τέσσερις τιμές Y μία τιμή U και μια τιμή V ενώ η δεύτερη πως για κάθε τέσσερα pixel υπάρχουν τέσσερις τιμές για την κάθε συνιστώσα π.χ για ένα καρέ ανάλυσης 720*480 τύπου YUV420 έχουμε $720 * 480 = 345600$ στοιχεία Y, $720 * 480 / 4$ στοιχεία U και $720 * 480 / 4$ στοιχεία V αρά σύνολο 518400. Ο τρόπος αποθήκευσης του βίντεο γίνεται 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

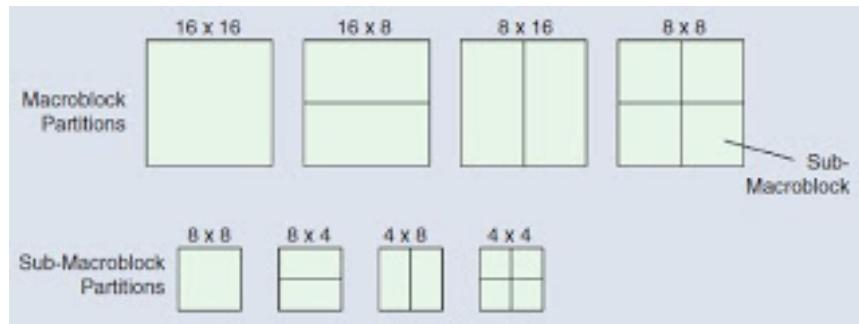
συνολικά χρειαζόμαστε $518400 * 8\text{bits} \approx 0.7\text{MB}$. Το πιο κοινό bitdepth που χρησιμοποιείται από τους σημερινούς κωδικοποιητές είναι αυτό των 8bits.



Σχήμα 2.1: Στην πρώτη εικόνα φαίνεται ένα καρέ με όλες τις συνιστώσες ενώ παρακάτω ξεχωριστά το YUV για το ίδιο καρέ [17].

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 οπού συνήθως $N=16$. Έτσι το καρέ του παραδείγματος μας αποτελείται από $518400/(16*16) = 2025$ macroblocks. Το κάθε macroblock μπορεί να σπάσει σε blocks και το κάθε block σε subblocks όπως φαίνεται στο Σχήμα 2.2.



Σχήμα 2.2: Διαμερισμός του macroblock στον H.264.[3]

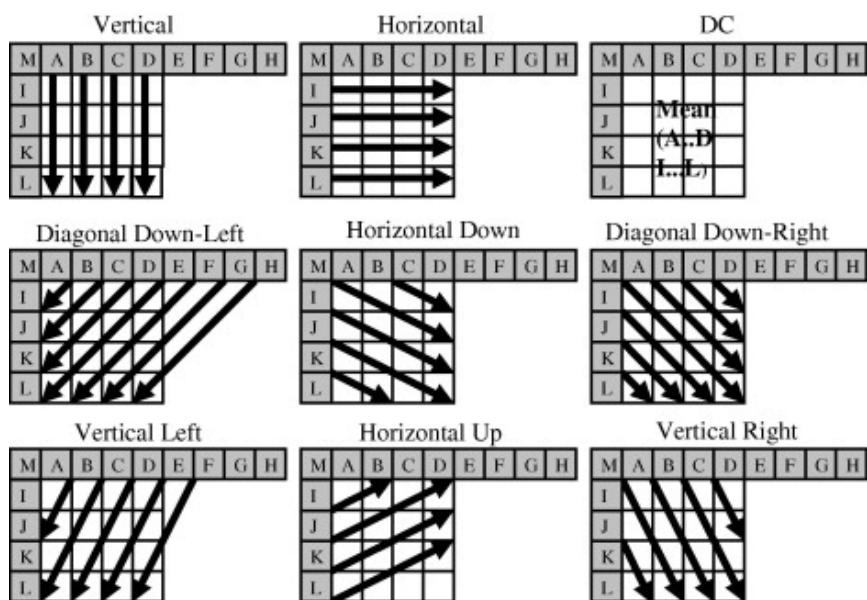
Μέτα τον χωρισμό σε macroblocks οι κωδικοποιητές ομαδοποιούν τα καρέ σε Intra και Inter με τα τελευταία να χωρίζονται σε P και B. Όλες οι παραπάνω κατηγοριοποίησεις έχουν να κάνουν με τον υπχανισμό του Motion Estimation ο οποίος δημιουργεί διαφορές pixels (residuals). Κύριο χαρακτηριστικό των residuals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

- Intra η αλλιώς I είναι τα καρέ που χρησιμοποιούν πληροφορία μόνο από το τρέχων καρέ για να βγάλουν τα residuals. Αυτό γίνεται απλά παίρνοντας τις τιμές των γειτονικών block και εφαρμόζοντας ένα mode π.χ μέσος όρος οπού το αποτέλεσμα αυτής 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 καταλήξουμε στο καλύτερο. Η απόδοση της συμπίεσης είναι η χειρότερη σε αυτήν την κατηγορία αλλά χωρίς αυτά το βίντεο δε θα μπορούσε να ξεκινήσει γιατί δεν θα είχαμε όλη την 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.
- B 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. Αυτά έχουν την καλύτερη απόδοση συμπίεσης αλλά η πολυπλοκότητα αποκωδικοποίησης είναι η μεγαλύτερη.

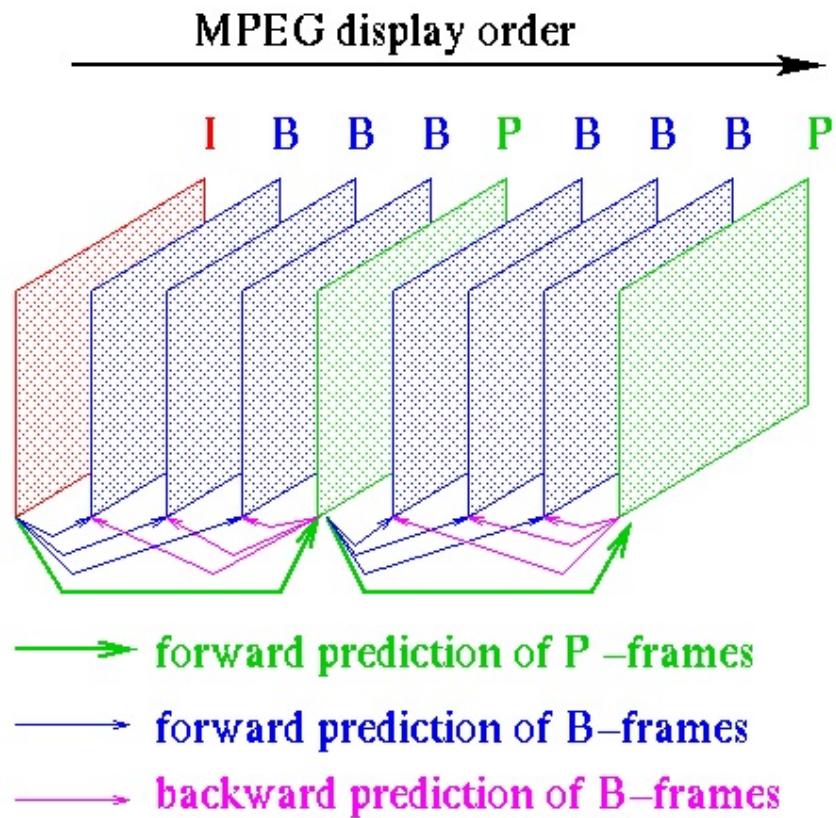
2. Ψηφιακό βίντεο και τεχνικές συμπίεσης

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. 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. Εννοείτε 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]



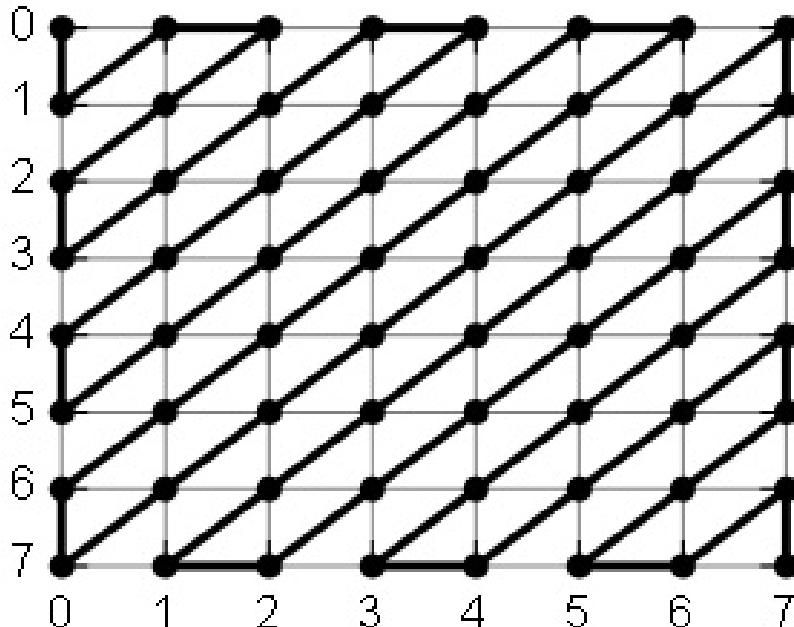
Σχήμα 2.4: Inter prediction στον mpeg και απεικόνιση ενός ενδεικτικού GOP. [9]

2.4 Μετασχηματισμός

Έχοντας ο encoder κατασκευάσει τα residuals για το macroblock τότε το επόμενο βήμα είναι ένας μετασχηματισμός οπού συνήθως είναι ο DCT. Αυτό συμβαίνει γιατί ο μετασχηματισμός μας επιστρέφει πολλά 0 αν το σήμα μας είναι χαμηλής ενέργειας, όπως συμβαίνει με τα residuals. Έτσι χρησιμοποιώντας κάποιο ειδικό σύμβολο μπορούν να απεικονιστούν πολλά μηδενικά του μετασχηματισμού και να μειώσουμε το data rate μας όπως φαίνεται στο Σχήμα 2.5 μέσω της εφαρμογής του αλγορίθμου RLE (Run Length Encoding). Η σάρωση του macroblock δεν γίνετε 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}									

Σχήμα 2.5: Παραδείγμα χρήσης RLE



Σχήμα 2.6: Σάρωση ZigZag. [8]

2.5 Κβαντοποίηση

Η κβαντοποίηση είναι το σημείο που σε κάθε encoder εισάγεται ο θόρυβος, θα μπορούσε να θεωρηθεί και ο DCT,iDCT λόγο ακρίβειας αλλά εκεί το σφάλμα είναι παρά πολύ μικρό. Μετά τον μετασχηματισμό τα residuals έχουν αντικατασταθεί με συντελεστές οι οποίοι είναι πραγματικοί αριθμοί. Η τακτική που εφαρμόζεται είναι να γίνει ακέραια διαιρέση ο συντελεστής της κάθε συχνότητας με έναν σταθερό αριθμό διαφορετικό για κάθε συχνότητα Πίνακας 2.1. Επομένως για τον δεδομένο πίνακα κβαντοποίησης με τους συντελεστές του Πίνακα 2.2 έχουμε τα αποτελέσματα στον Πίνακα 2.3 τα οποία είναι αυτά που θα δοθούν σε κάποιον entropy encoder για να συμπιεστούν.

Ο Πίνακας 2.1 μεταβάλετε με βάση μια μεταβλητή που καθορίζει την ποιότητα του βίντεο. Στον H.264 το $QP \in [0, 51]$. Στο 0 δεν γίνετε καθόλου κβαντοποίηση και όσο το QP αυξάνει τόσο η ποιότητα πέφτει. Αυτό συμβαίνει γιατί το QP πολλαπλασιάζεται με τον Πίνακα 2.1 και οι τιμές του μεγαλώνουν. Επομένως στην ακέραια διαιρέση χάνουμε περισσότερο πληροφορία διότι διαιρούμε με μεγαλύτερο αριθμό. Κερδίζουμε όμως σε μέγεθος γιατί έτσι προκύπτουν περισσότερα μπδενικά.

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. [16]

-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: Παράδειγμα συντελεστών DCT 8x8. [16]

2. Ψηφιακό βίντεο και τεχνικές συμπίεσης

-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. [16]

2.6 Αποκωδικοποίηση βίντεο

Για την αποκωδικοποίηση κάνουμε 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 ανακτούνται οι συντελεστές του μετασχηματισμού με έχοντας πλέον εισαχθεί το σφάλμα που προέκυψε λόγο της ακέραιας διαίρεσης.
- Γίνετε αντίστροφος μετασχηματισμός και ανακτούνται τα residuals.
- Τέλος το Motion Decomposition βρίσκει τα δύο block που χρησιμοποιήθηκαν για να παραχθούν τα residuals (μέσω των motion vectors) και τα προσθέτουμε για να πάρουμε τα ανακατασκευασμένα pixels.

2.7 Ποιότητα Βίντεο

Με τον όρο ποιότητα βίντεο ορίζουμε τον μέσω όρο του άθροισματος των διαφόρων στο τετράγωνο MSE (Mean Square Error) όλων των pixels μεταξύ του ασυμπίεστου βίντεο και του αυτού που αποσυμπιέσαμε.

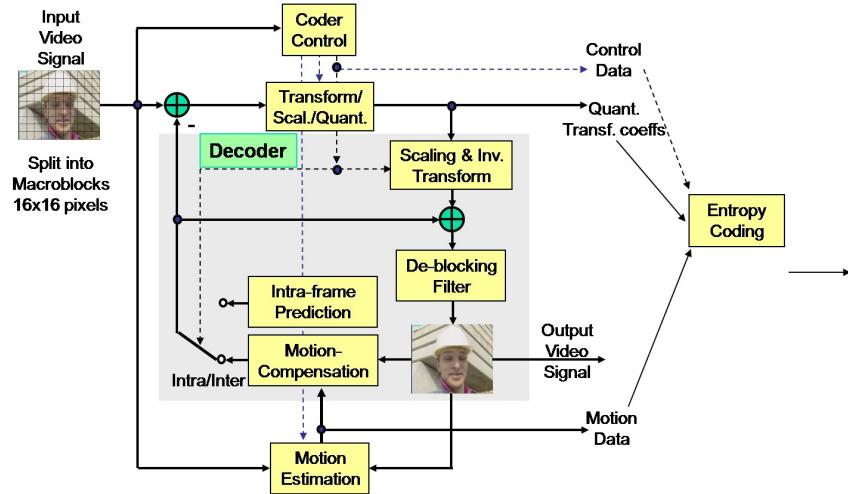
$$MSE = \frac{\sum_{i=1}^{X*Y} (Source_{pixel_i} - Reconstructed_{pixel_i})^2}{X * Y} \quad (2.1)$$

Συνήθως το μετατρέπουμε σε PSNR 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.

2.8 Δομή του H.264

Παρακάτω βλέπουμε την δομή του H.264 encoder Σχήμα 2.7. Το σημείο με το γκρι είναι το κομμάτι του decoder. Κάθε encoder υλοποιεί στο εσωτερικό του και έναν decoder αλλά σε αυτή την διπλωματική δεν θα εξηγήσουμε κάτι παραπάνω γιατί δεν βοηθάει στην κατανόηση της.



Σχήμα 2.7: H.264 encoder structure. [12]

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 kmeans που χρησιμοποιήθηκε για την ταραγωγή 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 [11]. Η κύρια πληροφορία που παρέχεται από την εντροπία και είναι χρήσιμη σε εμάς είναι το απόλυτο κάτω όριο μέχρι το οποίο η πληροφορία μπορεί να συμπιεστεί. Η εντροπία ορίζεται ως $H(X) = -\sum_{i=1}^n p(x_i) * \log_b p(x_i)$ οπού $p(x_i)$ είναι η πιθανότητα του ενδεχομένου x_i και η βάση b ορίζει την μονάδα μέτρησης της εντροπίας. Για την συμπίεση δεδομένων συνήθως ισχύει $b = 2$, ώστε να έχουμε την εντροπία σε bits.

Για να γίνει κατανοητή η έννοια της εντροπία 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

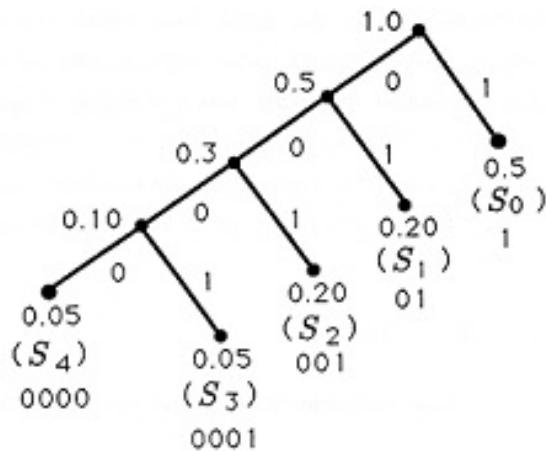
bits το καθένα για να αναπαρασταθούν τα $x_i = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$. Αν τα ενδεχόμενα είναι

ισοπίθανα ισχύει ότι $\forall x_i, p(x_i) = 0.25$ και προκύπτει ότι $H(X) = 2 bits$. Συνεπώς, δε μπορεί να γίνει 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 μας μπορούν ιδανικά να συμπιεστούν κατά $(1 - 1.35678/2)\% = 32\%$.

3.3 Κωδικοποιητές Εντροπίας

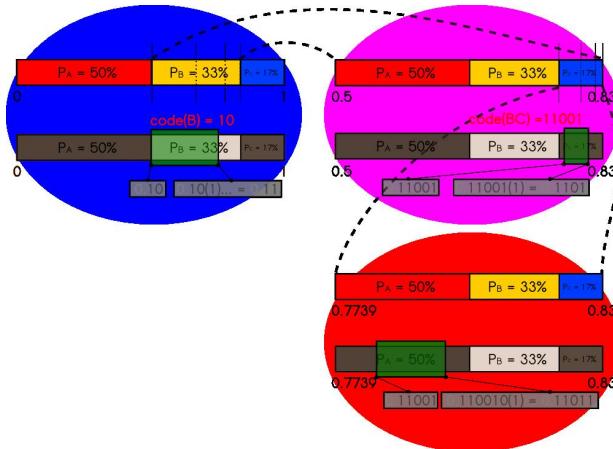
Οι μέθοδοι που σήμερα υπάρχουν για συμπίεση καταφέρνουν να έρθουν πολύ κοντά στο όριο που δίνει η εντροπία αλλά δε το φτάνουν. Δύο είναι οι κυρίαρχες, η μέθοδος Huffman και Αριθμητική κωδικοποίηση. Η δεύτερη να είναι η πιο δημοφιλής περισσότερο στο βίντεο με την παραλλαγή της που λέγεται CABAC (Context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

- Η μέθοδος Huffman έχει την μικρότερη πολυπλοκότητα $O(n \log n)$ μεταξύ των δύο και μας εξασφαλίζει πώς $H(X) \leq HC \leq H(X) + 1\text{bit}$ όπου HC είναι το μέσο μήκος ανά σύμβολο που δίνει ο Huffman. Ο αλγόριθμος 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 έχουμε 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.



Σχήμα 3.1: Δέντρο Huffman. [10]

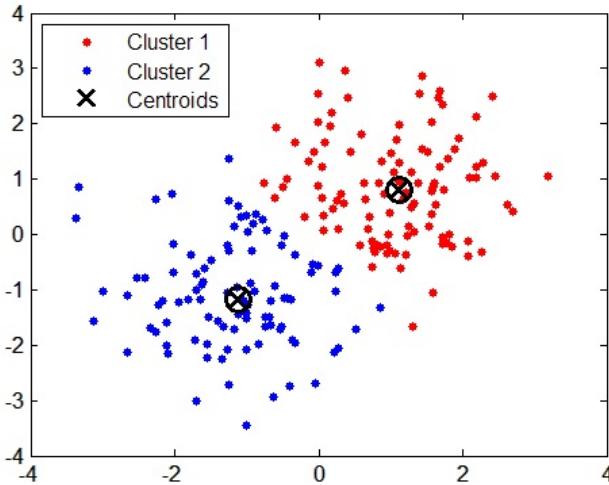
- Η μέθοδος της Αριθμητικής Κωδικοποίησης έχει την μεγαλύτερη πολυπλοκότητα μεταξύ των δύο αλλά μας εξασφαλίζει μια εν γένει καλύτερη συμπίεση από αυτή του Huffman και αρκετά "κοντά" την εντροπία. Ο αλγόριθμος όπως και στον Huffman έχει στόχο να δώσει μεγάλο μήκος στα ενδεχόμενα με την μικρότερη πιθανότητα 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 μίνυμα που κωδικοποιείται είναι το "BCA". Στο Σχήμα 3.2 φαίνονται τα βήματα της κωδικοποίησης. Στο πρώτο βήμα έρχεται το B και το κωδικοποιούμε με τον αριθμό 0b01(x) γιατί έχει το μικρότερο μήκος και βρίσκεται στο [0.5,0.83) οπού x σημαίνει αυθαίρετη ακολουθία από bits. Έτσι συνεχίζουν και τα υπόλοιπα βήματα μέχρι να έχουμε τον αριθμό που αντιστοιχεί στην ακολουθία.



Σχήμα 3.2: Βήματα αριθμητικής κωδικοποίησης. [14]

3.4 Αλγόριθμος clustering kmeans

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



Σχήμα 3.3: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$. [7]

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 να πάρνει τιμές από ένα τυχαίο point) είτε με κάποια στρατηγική. Στην παρούσα διπλωματική επιλέχθηκε η στρατηγική KKZ [5] η οποία έχει μεγαλύτερη πολυπλοκότητα από την τυχαία αλλά οδηγεί τον kmeans σε μικρότερο σφάλμα. Ο αλγόριθμος παραλληλοποιήθηκε με OpenMP και επιτεύχθηκαν οι επιδόσεις του Πίνακα 3.1. Στον Πίνακα 3.2 φαίνεται η συμπεριφορά του kmeans με τυχαία και KKZ αρχικοποίηση και παρατηρείτε η κυριαρχία του KKZ όσο αναφορά το σφάλμα.

Αρχικοποίηση	1	2	4	6
KKZ	90.5	63.7	42.9	36.7
Random	0.01	0.005	0.003	0.003

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

Tύπος	d	Πλήθος	k	KKZ 5	Random 5	KKZ/Iter	Random/Iter
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 kmeans 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. Στην παρούσα διπλωματική, χρησιμοποιήθηκε ο αλγόριθμος Fast Nearest Neighbour [6] ο οποίος οργανώνει τις αποστάσεις σε ένα δυαδικό δέντρο. Ο FastNN μας επιτρέπει να κάνουμε $\log_2 k$ 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 συμφόρωσης του αλγορίθμου. Σε συνδυασμό με την παραλληλοποίηση στο Βίμα 4 με OpenMP μας επέτρεψε να τρέχουμε μεγάλα πειράματα σε εύλογο χρονικό διάστημα. Οι επιταχύνσεις που επιτεύχθηκαν με την χρήση του FastNN και της OpenMP παρουσιάζονται στον Πίνακα 3.3.

3. Θεωρία πληροφοριών

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 Simple, FastNN με 1,2,4,6 threads με $k = 65536, d = 16, n = 100000$

Για τα πειράματα της διπλωματικής χρησιμοποιήθηκε 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 ο kmeans (training) και με ποία δεδομένα (training set). Επίσης θα δειχθεί ο τρόπος εξαγωγείς των δεδομένων από τον 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 vector 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. Έτσι λοιπόν επιλέχθηκαν τα residuals να είναι το training set του kmeans.

Μέχρι τώρα έχει γίνει σαφές πως για κάθε καρέ διάστασης W^*H pixels υπάρχουν W^*H residuals. Επομένως είναι εφικτό να τεμαχίσουμε το καρέ σε $d*d$ blocks οπού d η διάσταση του training set. Το d έχει τον μοναδικό περιορισμό ότι πρέπει να είναι μικρότερο από την διάσταση του block του encoder, στον H.264 που αυτή η διπλωματική βασίστηκε $blocksize \in [2 * 2, 16 * 16]$ και μπορεί να ρυθμιστεί ανάλογα. Η διάσταση που επιλέχθηκε είναι το $d = 4$ που πληρεί το κριτήριο "blocksize" και είναι η προεπιλεγμένη τιμή blocksize στον H.264. Έστω ένα καρέ ανάλυσης $720*480$ με YUV420 τότε συνολικά έχουμε $720 * 480 * 1.5 = 518400$ pixels και $\frac{518400}{4*4} = 32400$ training vectors.

Τα βίντεο που θα μας έδινα το training set έπρεπε να επιλεχθούν προσεκτικά γιατί χρειάζεται να πληρούν κάποιες προϋποθέσεις.

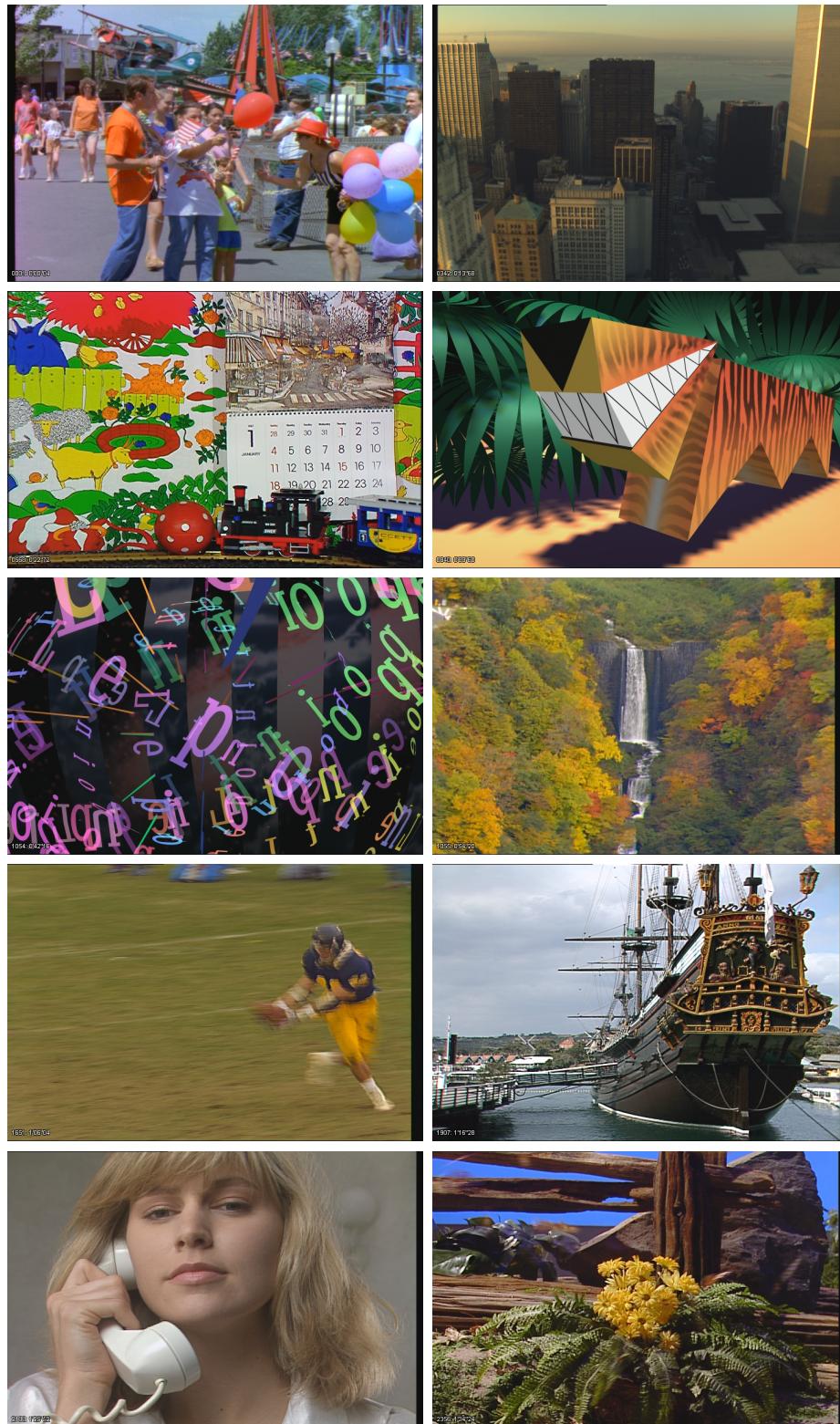
- Θα πρέπει να υπάρχει ένας ικανοποιητικός αριθμός από καρέ για να μπορέσουμε να έχουμε αντιπροσωπευτικά στατιστικά. Στην διπλωματική χρησιμοποιήθηκαν 2600 καρέ από 10 διαφορετικά βίντεο ανάλυσης $720*480$. Αρά συ-

4. Παραγωγή Codebooks

νολικά είχαμε 84240000 training vectors που θεωρείται ικανοποιητικός αριθμός.

- Το περιεχόμενο των βίντεο παίζει καθοριστικό ρόλο για το PSNR του VQ όταν δοκιμάζετε 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 έχουμε χαμηλά PSNR (φαινόμενο mismatch). Τα 10 βίντεο που χρησιμοποιήθηκαν είχαν διαφορετικές σκηνές και μπορούμε να δούμε κάποια στιγμιότυπα στο Σχήμα 4.1.

Θεωρητικά το καλύτερο training set θα ήταν όλα τα βίντεο 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.



Σχήμα 4.1: Training βίντεο. [4]

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 τα καρέ είναι I,P,B είναι ψάχνει να βρει σε προηγούμενα/επομένα καρέ το καλύτερο match ώστε να βγάλει το Motion Vector. 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, τα residuals/macroblock σε αυτό το σημείο ήταν εύκολα προσβάσιμα. Στην συνέχεια το macroblock διάστασης 16x16 με τα residuals χωρίζονταν σε block διάστασης dxd και γράφονταν 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 [4] 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 kmeans. Αν είναι 2 γίνετε εξαγωγή ολόκληρων καρέ.

Αφού η διαδικασία γίνετε 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 θόρυβο. Η συμπίεση έγινε δύο φορές με δύο διαφορετικά GOP και έγινε τρεις φόρες αποσυμπίεση μια για κάθε mode.

- Για να παραχθεί το I training set συμπιέσαμε τα 2600 καρέ με GOP I-I-I-.... και αποσυμπιέσαμε με KeepI 1 και ResidualsMode 1.
- Για να παραχθεί το P training set συμπιέσαμε τα 2600 καρέ με GOP I-P-P-B-P-P-B.... και αποσυμπιέσαμε με KeepP 1 και ResidualsMode 1.
- Για να παραχθεί το B training set συμπιέσαμε τα 2600 καρέ με GOP I-P-P-B-P-B.... και αποσυμπιέσαμε με KeepB 1 και ResidualsMode 1. Άλλα είδαμε πως

τα vectors που ήταν πραγματικά Bidirectional ήταν λίγα και δεν αρκούσαν για να εξαχθούν στατιστικά. Επομένως θέσαμε και το KeepP 1 έτσι ώστε να συμπεριληφθούν όλα τα Inter vectors.

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 Kmeans

Ο kmeans εφαρμόστηκε στα training set που φαίνονται στον Πίνακα 4.1 και παρατηρούμε πως τα πειράματα έτρεχαν για 23 μέρες. Τα k 65536 επιλέχθηκε για να έχουμε VQ indeces στο διάστημα [0,65535] έτσι ώστε να χρειαζόμαστε ακριβώς 2 bytes για να αποθηκεύσουμε ένα index, επίσης βλέπουμε πως μας δίνει ένα PSNR κοντά στο βίντεο που θέλουμε να βλέπουμε στην καθημερινότητα μας. Η εντροπία είναι διαιρεμένη με την διάσταση οπότε αντιπροσωπεύει το κάθε residual.

Τύπος	d	Πλήθος	k	Εντροπία	PSNR(dB)	Επαναλήψεις	Διάρκεια
IntraY	16	56160000	65536	0.712229	33.6	3249	12154
IntraUV	16	28080000	65536	0.743071	42.1	2697	3119
InterY	16	42117616	65536	0.692577	40.5	3270	9120
InterUV	16	21058808	65536	0.707785	48.1	4221	8509

Πίνακας 4.1: Πληροφορίες για τα codebooks.

4.5 Εντροπία υπό συνθήκη

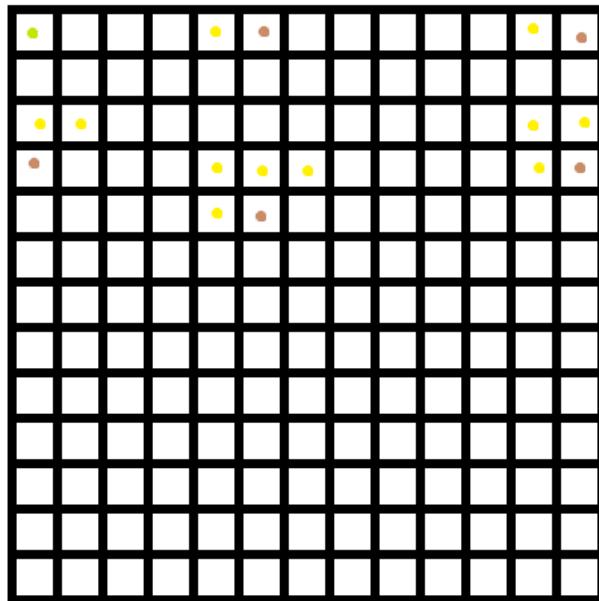
Είναι γνωστό από την θεωρία Θεώρημα 2.6.5 του βιβλίου [2] ότι αν έχουμε δύο τυχαίες μεταβλητές X, Y τότε η πληροφορία που μας δίνει η Y μπορεί μόνο να μειώσει την εντροπία της X, ισχύει δηλαδή πως $H(X|Y = y_i) \leq H(X)$ με την ισότητα να ισχύει μόνο όταν X, Y είναι ανεξάρτητες. Αυτό συμβαίνει μόνο όταν μπορούμε να έχουμε την πληροφορία για το Y χωρίς κάποιο επιπλέον κόστος.

Για να δοκιμαστεί αυτό το θεώρημα έγιναν τα παρακάτω βήματα. Παράχθηκε το κάθε training set με την μορφή καρέ τρέχοντας τον decoder με ResidualsMode 2. Για το συγκεκριμένο πείραμα αυτή η επιπλέον πληροφορία που δίνετε από την Y είναι ο μέσος όρος της ενέργειας των ήδη κβαντισμένων block όπως φαίνεται στο Σχήμα 4.2. Τα y_i είναι περιοχές ενέργειας όπως φαίνεται στο Σχήμα 4.3, παράχθηκαν ταξινομώντας με βάση την πιθανότητα τους τα clusters των codebooks και χωρίζοντας την ενέργεια σε οχτώ περιοχές με συνολικά ίση πιθανότητα. Ξεκινώντας από πάνω αριστερά κβαντοποιούνται τα block και κρατιούνται στατιστικά για το ποια είναι η κατηγορία του τρέχοντος block με βάση την ενέργεια των γειτόνων του.

4. Παραγωγή Codebooks

Έτσι δημιουργούνται οχτώ νέες κατανομές πιθανότητας μεγέθους 65536, καθεμία μια από αυτές έχει δικιά της πιθανότητα.

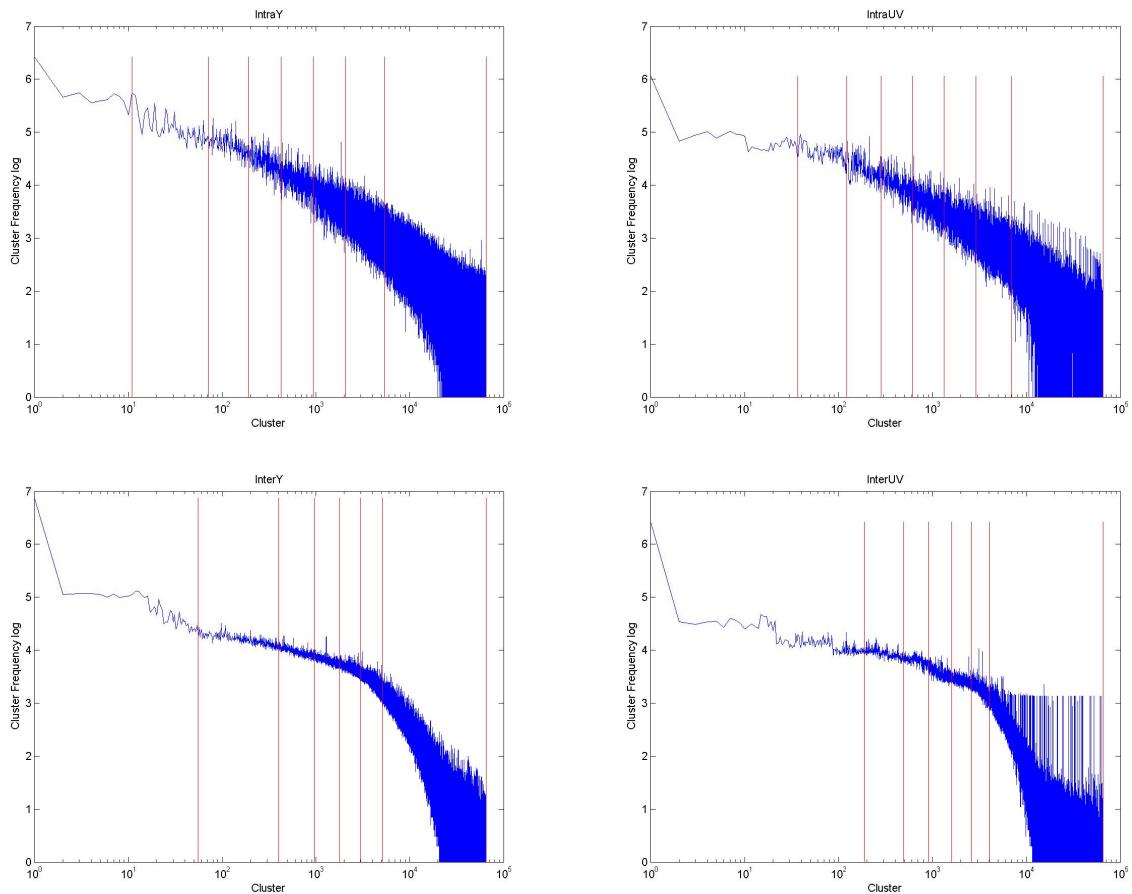
Αυτό το πείραμα με λίγα λογία λέει ότι αφού γνωρίζουμε την ενέργεια των κβαντοποιημένων block τότε ξέρουμε και την κατηγορία που ανήκει το τρέχων block. Έτσι κωδικοποιούμε (entropy encoding) με βάση την πιθανότητα του συγκεκριμένου cluster στην συγκεκριμένη κατηγορία ενέργειας. Το πείραμα πέτυχε καλύτερη 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.



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

Τύπος	Εντροπία
IntraY	0.5411
IntraUV	0.6434
InterY	0.5443
InterUV	0.6314

Πίνακας 4.2: Εντροπία μετά την χρήση contexts



Σχήμα 4.3: Οι κόκκινες γραμμές δείχνουν το σημείο που σταματάει μια κατηγορία ενώ οι μπλε το πλήθος των vectors που αντιστοιχούν στο cluster

Κεφάλαιο 5

VQ H.264

5.1 Εισαγωγή

Στο τελευταίο κομμάτι αυτής της διπλωματικής έγινε η ενσωμάτωση του VQ στον H.264 ώστε να μπορούμε να κάνουμε encoding και decoding με βάση τα codebooks. Το να αντικατασταθεί όλος ο μηχανισμός κβαντοποίησης-μετασχηματισμού του H.264 είναι 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 παρεμβαίνουμε για να εισάγουμε τα δικά μας residuals.

5.2 Encoding

Αφού γίνει το 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 τον θόρυβο που εισήχθηκε από την κβαντοποίηση. Αυτό είναι αναγκαίο διότι ο decoder γνωρίζει μόνο τα χαλασμένα residuals αρά στον encoder θα πρέπει να παίρνουμε αποφάσεις μόνο με τα χαλασμένα residuals. Με τον όρο αποφάσεις εννοώ ότι η επιλογή του καλύτερου Motion Vector στον encoder θα γίνει με βάσει 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 μας κάνει φανερό ότι τα reconstructed residuals χρησιμοποιούνται για να γίνετε το Motion Estimation.

Η συνάρτηση που ο 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 block καθώς και την θέση του στην εικόνα, τα Reference Pixels και τα Reconstructed Residuals.

5. VQ H.264

Η τακτική που ακολουθίσαμε ήταν να κωδικοποιούμε το βίντεο με QP 0 (καθόλου απώλεια στην κβαντοποίηση) άρα τα Reconstructed Residuals Original Residuals. Αρά στην ουσία πετύχαμε τον στόχο της αφαίρεσης του υπχανισμού 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 ανακατασκευάζετε 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 δοκιμάζει όλα τα επιτρέπομένα modes μέχρι να βρεθεί το καλύτερο, αυτό σημαίνει πως η συνάρτηση sample_reconstruction θα κληθεί όσες φόρες είναι και το πλήθος των modes. Επομένως πρέπει να κρατάμε τα προσωρινά $VQ_{indices}$ που γράφει η quantize_mb και κάθε φόρα που βρίσκουμε ότι ένα mode είναι καλύτερο από το προηγούμενο να τα αντιγράφουμε σε ένα global πίνακα που κρατάει τα καλύτερα VQ indices για κάθε macroblock.

Αφού τελειώσει η διαδικασία αυτή για κάθε macroblock γράφουμε τα $VQ_{indices}$ σε ένα αρχείο. Σε αυτήν την διπλωματική τα VQ indices που γράφονται για κάθε macroblock είναι $\frac{16*16}{4*4} = 16$ όπου 16*16 η διάσταση του macroblock και 4*4 η διάσταση του codebook. Επομένως γράφονται $2 * 16 = 32bytes$ ανά macroblock, το 2 bytes προκύπτει επειδή τα $VQ_{indices} \in [0, 65536]$.

Algorithm 2 Block 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
```

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 δίνετε το αρχείο των $VQ_{indices}$.
- VQdim οπού δίνετε η διάσταση των codebooks. Αν είναι 0 το VQ απενεργοποιείται.

- VQcrlen οπού δίνετε το μήκος των codebooks.
- 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 αλλά αυτά είναι άχροντα για τον decoder αφού διαβάζονται από τα codebooks.

5.3 Decoding

Στον decoder όπως περιμένει κανείς τα πράγματα είναι πιο απλά. Αφού αφήσουμε τον decoder να κάνει την διαδικασία αντιστροφής κβαντοποίησης και μετασχηματισμού, στο ίδιο ακριβώς σημείο με τον encoder παρεμβάλλουμε την συνάρτηση inv_macroblock . Η συνάρτηση αυτή αντιστοιχίζει 16 $VQ_{indices}$ που διαβάζει από το αρχείο για κάθε ένα από τα 16 subblocks του macroblock με 16 $VQ_{vectors}$. Άλγοριθμος 3. Τελικά ο decoder παίρνει την κύρια πληροφορία που είναι τα residuals από τα $VQ_{indices}$ και τα headers,motion vector από το αρχείο .264.

Algorithm 3 Block 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:      $vq_i = \text{read } VQ_{index} \text{ from file for this position}[x,y];$ 
5:     Replace dxd residuals with codebook[ $vq_i$ ] data;
6:   end for
7: end function

```

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

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

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 μπορούμε να βελτιώσουμε την πολυπλοκότητα του decoder σε σχέση με αυτή του JM H.264.

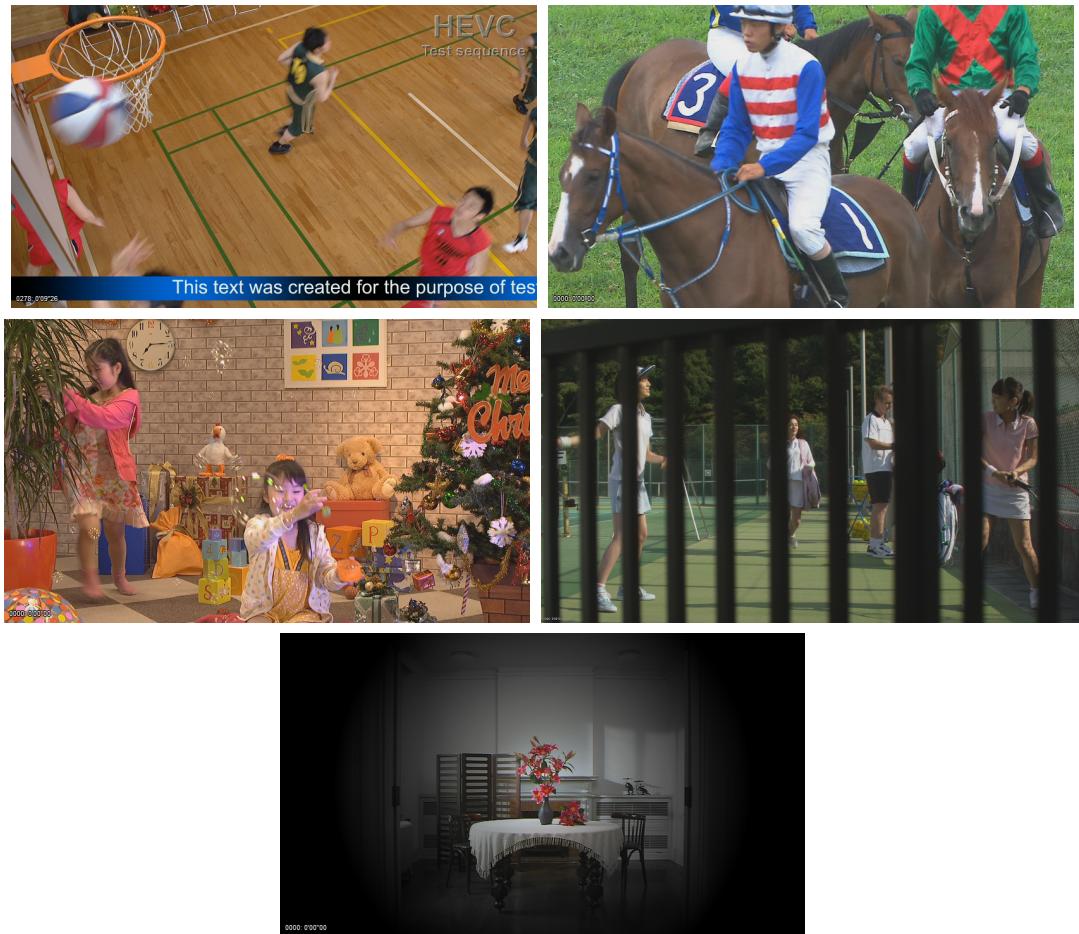
6.2 Encoding με VQ H.264

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 στα test video φαίνεται στο Σχήμα 6.1.

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. [13]

6.3 Encoding με JM H.264

Για να είμαστε σε θέση να συγκρίνουμε τα αποτελέσματα των δύο encoder θα πρέπει να συμπιέσουμε τα ίδια βίντεο με τον JM H.264 με το ίδιο configuration και προσπαθώντας να πετύχουμε τα ίδια PSNR που πέτυχε και ο VQ H.264. Αυτό επιτεύχθηκε 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 την χαρακτηρίζα πολύ μικρή έτσι ώστε να έχουμε έγκυρη σύγκριση.

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

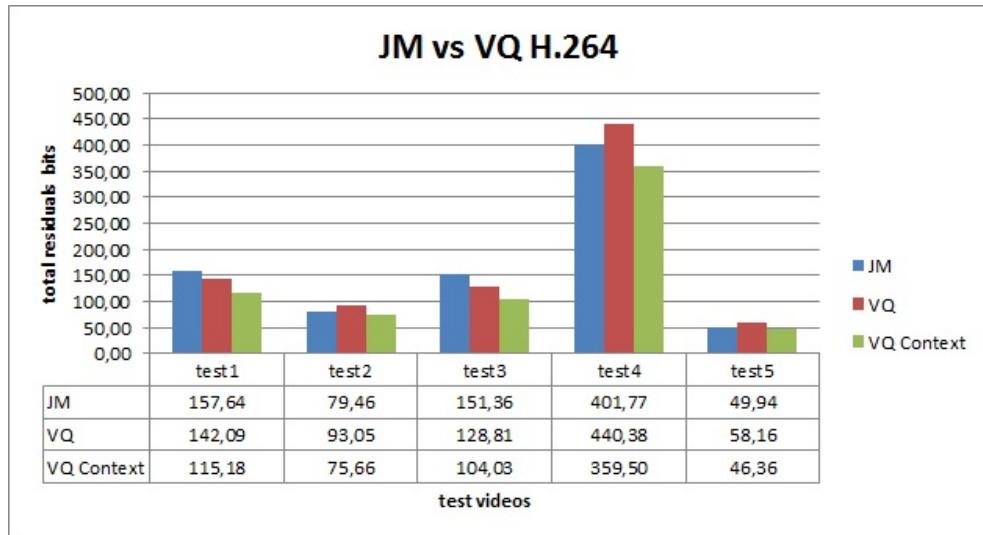
Πίνακας 6.2: PSNR των Test videos με κωδικοποίηση στον JM H.264

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 ο JM είναι καλύτερος από τον VQ ενώ στα αρνητικά το ανάποδο.

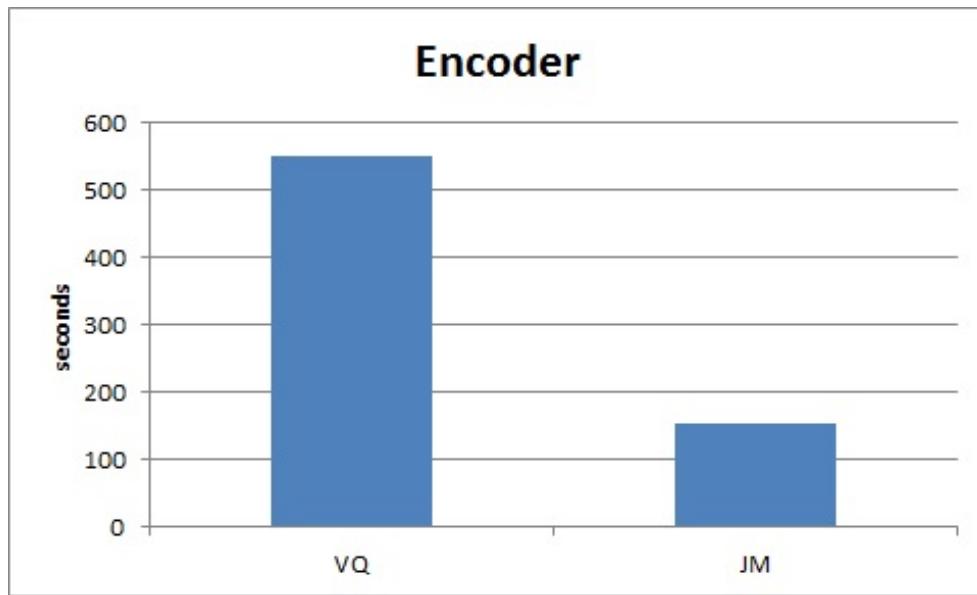
6.4 Αποτελέσματα των VQ H.264, JM H.264

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 γίνετε γιατί στην ουσία τα $VQ_{indices}$ αντιστοιχούν μόνο στην πληροφορία που μας δίνουν τα residuals, όπως και οι συντελεστές του μετασχηματισμού. Αφού αυτά βρέθηκαν υπολογίστηκε το πλήθος $len(vector_{mode})$ των vectors για κάθε βίντεο και κάθε mode (Intra,Inter) και από το πλήθος αυτών αφαιρούνται 16 vectors για κάθε macroblock που χαρακτηρίζεται από τον encoder σαν skipped. Skipped είναι ένα macroblock για το οποίο ο encoder δε γράφει συντελεστές γιατί είναι όλοι μηδέν. Στο επόμενο βήμα πολλαπλασιάζουμε την εντροπία του Πίνακα 4.2 και του Σχήματος 4.1 με την διάσταση του VQ για να βρούμε πόσα bits χρειάζονται για κάθε block dxd. Τέλος πολλαπλασιάζουμε τα bits που χρειάζονται ανά block και έχουμε το μέγεθος των $VQ_{indices}$ μετά από συμπίεση. Η αποδόσεις φαίνονται στο Σχήμα 6.2.

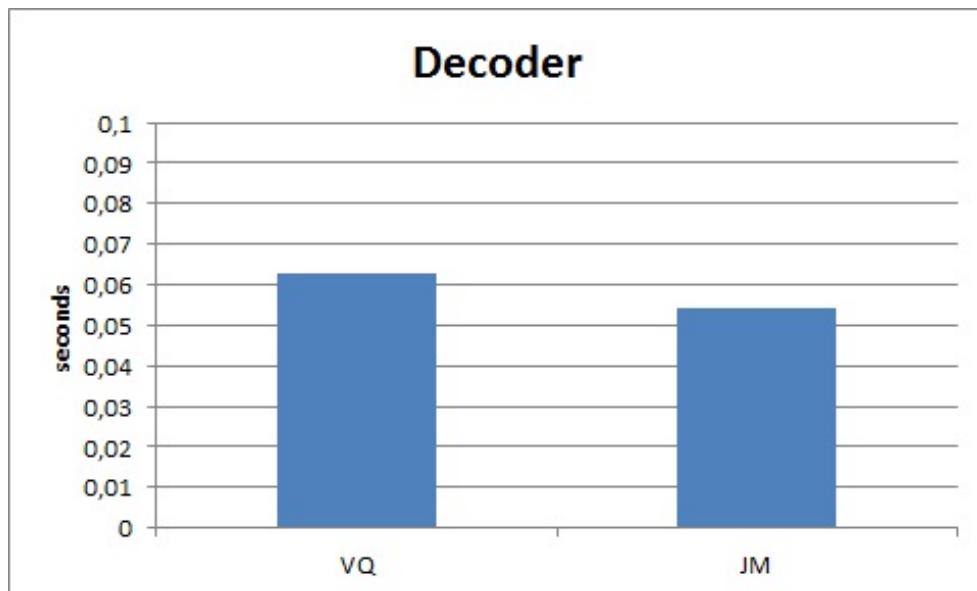


Σχήμα 6.2: Σύγκριση της απόδοσης PSNR του JM H.264 και VQ H.264 με context entropy και χωρίς context.

Εκτός από την σύγκριση στο PSNR έγινε προσπάθεια να συγκριθούν και οι χρόνοι κωδικοποίησης και αποκωδικοποίησης. Όπως είπαμε ο VQ H.264 κάνει όλη την άχροστη για αυτόν δουλειά του μετασχηματισμού, κβαντοποίησης, αντίστροφου μετασχηματισμού, αντίστροφης κβαντοποίησης επομένως χρησιμοποιήθηκε το Intel VTune για να βρεθεί ο χρόνος αυτόν των συναρτήσεων και να αφαιρεθούν από τον συνολικό. Έτσι είχαμε 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.



Σχήμα 6.3: Σύγκριση του χρόνου εκτέλεσης των encoder JM H.264 και 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 είναι κατά μέσω όρο 17% καλύτερος από τον JM H.264 στα βίντεο που δοκιμάστηκε. Επίσης φαίνεται να έχουμε 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. Αντίστοιχα στα test2,test4,test5 που δεν δυσκολεύουν ιδιαίτερα τον encoder παρατηρούμε ότι το κέρδος μειώνεται.

Θα πρέπει να λάβουμε υπόψιν μας ότι το VQ έχει πολλά περιθώρια βελτίωσης αν μεγαλώσουμε το training set. Με το τρέχων training set είδαμε πως σε κάποια βίντεο και ειδικά στα I frames δεν έχουμε καλή απόδοση του VQ. Αυτό μπορεί να βελτιωθεί απλά και μόνο συμπεριλαμβάνοντας αυτά τα βίντεο στο training set. Το θετικό σε αυτή την διαδικασία είναι ότι δεν χρειάζεται να τρέξουμε το training από την αρχή, αρκεί να δώσουμε σαν αρχικά clusters αυτά που ήδη έχουμε και έτσι να γλιτώσουμε τον KKZ αλλά και πολλά iterations του αλγορίθμου. Επομένως μπορούμε να πούμε πως 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 παρατηρούμε είναι η μεγάλη ανομοιομορφία του PSNR μεταξύ I,P,B αλλά και Y,UV. Δηλαδή βλέπουμε στο test1 ότι το I είναι στα 35dB ενώ τα P,B στα 43dB,45dB αντίστοιχα. Αυτή η διαφορά είναι αρκετά μεγάλη και για να διορθωθεί θα μπορούσαν να χρησιμοποιηθούν codebooks με διαφορετικό μέγεθος για κάθε συνιστώσα I,P,B,Y,UV. Έτσι λοιπόν αν χρησιμοποιούσαμε Intra codebook μήκους 32768 για το test1 θα χρειάζονταν 1byte για να αποθηκευτούν τα $VQ_{indices}$ αλλά ταυτόχρονα θα έπεφτε και το PSNR, αρά λοιπόν στο VQ το μήκος του codebook λειτουργεί σαν το QP στο scalar quantization. Μετρήσεις πάνω σε αυτόν το τομέα δεν έγινε αλλά αφήνεται σαν μια πολύ σημαντική προσθίκη στον VQ H.264.

Όπως βλέπουμε στο Σχήμα 6.3 βλέπουμε ότι ο VQ Encoder είναι 3.5 φορές πιο αργός από τον JM, αυτό οφείλεται στο κόστος του αλγορίθμου FastNN. Από

7. Συμπεράσματα

την άλλη μεριά βλέπουμε ότι ο VQ Decoder στο Σχήμα 6.4 είναι στα ίδια επίπεδα με τον JM Decoder. Αυτό που μας ενδιαφέρει είναι να έχουμε γρήγορους decoders γιατί μπαίνουν σε φορπτές συσκευές που μας ενδιαφέρει η ενέργεια. Έτσι ένας VQ decoder μπορεί να βελτιστοποιηθεί χρησιμοποιώντας μικρές και γρήγορες μνήμες όπου θα αποθηκεύονται τα codebooks. Έτσι ένας ενεργοβόρος και ακριβό βοηθητικό κύκλωμα που επιταχύνει την διαδικασία του αντίστροφου μετασχηματισμού και της αντίστροφης κβαντοποίησης μετατρέπετε σε μια γρήγορη, χαμηλής κατανάλωσης, φθηνή μνήμη.

7.2 Πιθανές προσθήκες

Επειδή το VQ έδειξε σε αυτή την διπλωματική ότι αξίζει παραπάνω προσοχή παρακάτω παραθέτω κάποιες ιδέες που θα το κάνουν πιο αποδοτικό.

- Χρήση διαφορετικού μεγέθους codebooks για να στις διάφορες συνιστώσες, καθώς και παραγωγή codebooks για διάφορες ποιότητες.
- Προσαρμογή του πυρήνα του JM H.264 έτσι ώστε στον encoder να μην γίνετε μετασχηματισμός και κβαντοποίηση αλλά απευθείας VQ και έπειτα να κωδικοποιούνται τα $VQ_{indices}$ με την χρήση των contexts. Έτσι ο decoder θα δέχεται μόνο ένα αρχείο με τα $VQ_{indices}$ κωδικοποιημένα και θα παράγει το βίντεο.
- Δοκιμή παραγωγής του training set από βίντεο που έχουν περάσει 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

Όλοι οι κάδικες τις διπλωματικής αυτής βρίσκονται στα παρακάτω git repositories.

- <https://github.com/petrkalos/vq>, Όλα τα εργαλεία για το training και τα contexts statistics. Περιλαμβάνει ένα Visual Studio 2010 Solution με 4 Project και κάποια αρχεία Matlab.
- <https://github.com/petrkalos/jm>, Ο VQ H.264.

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] Mrutyunjaya Hiremath. Macroblock partition mode.
- [4] iphome. Jm h.264.
- [5] 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.
- [6] 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.
- [7] Mathworks. Kmeans.
- [8] multimedia.cx. Zig zag scan.
- [9] VSR Group of Chemnitz University of Technology. H.264 video codec - inter prediction.
- [10] Louis Scharf. Binary codes: Huffman codes for source coding.
- [11] C. E. Shannon. A mathematical theory of communication. *SIGMOBILE Mob. Comput. Commun. Rev.*, 5(1):3–55, January 2001.
- [12] Yang Song. H.264 encoder structure.
- [13] Hannover University. Test videos.
- [14] Wikipedia. Arithmetic coding.

Βιβλιογραφία

[15] Wikipedia. Data compression.

[16] Wikipedia. Jpeg.

[17] Wikipedia. Yuv.