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

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

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), οπού 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 μνήμες χωρίς απαιτήσεις ισχύος.

1.3 Διάρθρωση της Διπλωματικής Εργασίας

Στο Κεφάλαιο 2 παρουσιάζεται το ψηφιακό βίντεο, με ποίες μορφές 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 ?? παρουσιάζονται τα βήματα που έγιναν για την παραγωγή των codebooks και την επιπλέον κατηγοριοποίηση τους με στόχο την μείωση της εντροπίας.

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

Στο Κεφάλαιο ?? παρουσιάζονται τα αποτελέσματα του VQ-H.264 και συγκρίνονται με τις επιδόσεις του JM-H.264.

Κεφάλαιο 2

Ψηφιακό βίντεο και τεχνικές συμπίεσης

2.1 Εισαγωγή

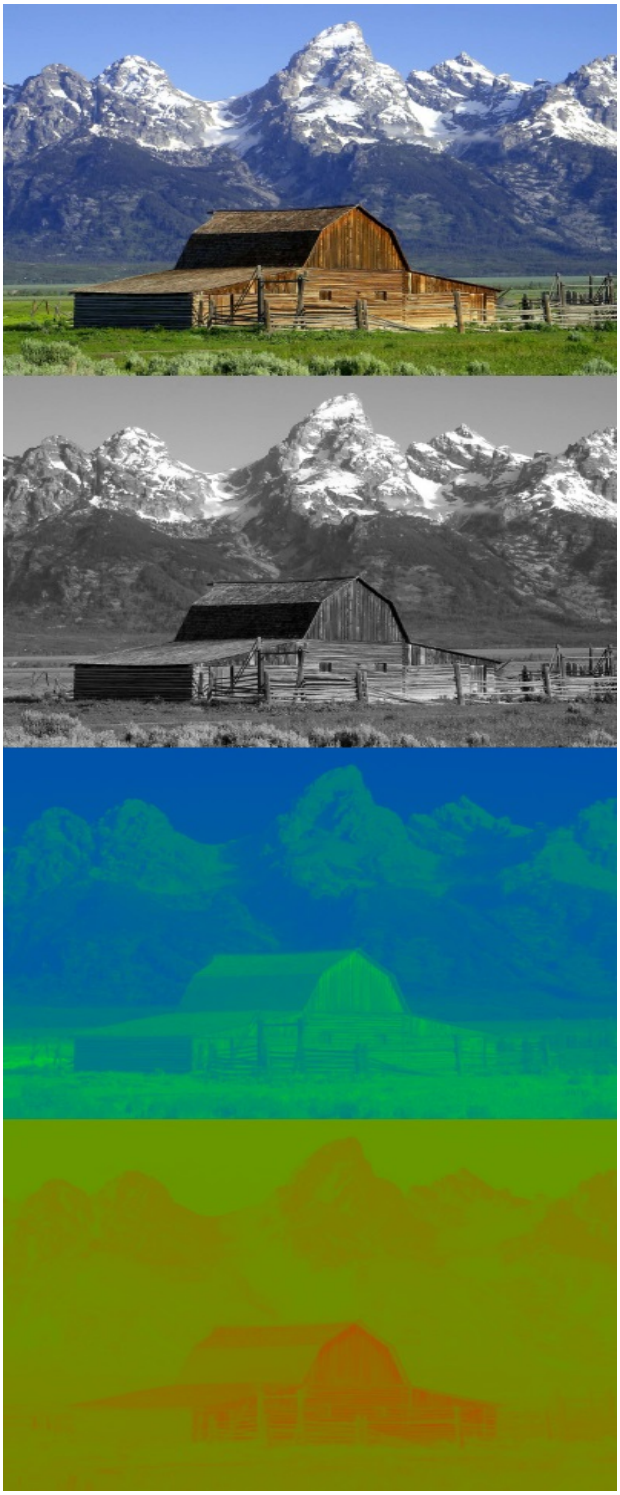
Σε αυτό το κεφάλαιο γίνεται μία εισαγωγή στους τρόπους που οι κωδικοποιητές βίντεο διαχειρίζονται τα δεδομένα και τους τρόπους που χρησιμοποιούν για να τα συμπιέσουν. Επίσης εδώ θα μας γίνει ξεκάθαρο γιατί το βήμα της κβαντοποίησης (εισαγωγή θορύβου) είναι αναγκαία για να έχουμε τόσο μεγάλους λόγους συμπίεσης

2.2 Συστατικά στοιχεία του βίντεο

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

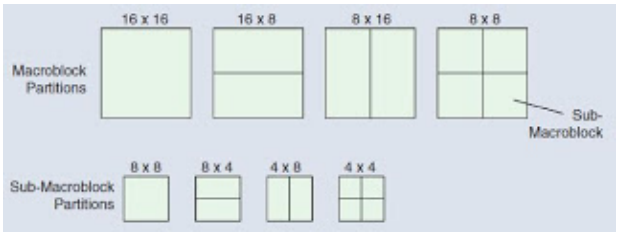
Κάθε pixel έχει και ένα βάθος (bitdepth), δηλαδή ποιο είναι το εύρος τιμών που παίρνει. Οι σύγχρονοι κωδικοποιητές υποστηρίζουν 8-14bits έτσι συνεχίζοντας το παράδειγμα μας από την προηγούμενη παράγραφο και έχοντας συνολικά 518400 στοιχεία για ένα καρέ και υποθέτοντας ότι το bitdepth 8 τότε συνολικά χρειαζόμαστε 518400*8bits $\approx$ 0.5MB. Το πιο κοινό bitdepth που χρησιμοποιείται από τους σημερινούς κωδικοποιητές είναι αυτό τον 8bits.



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

2.3 Οργάνωση των pixels από τους κωδικοποιητές

Η πλειοψηφία των σημερινών κωδικοποιητών ομαδοποιούν τα pixels. Η πιο συνηθισμένη ομαδοποίηση είναι αυτή του macroblock, οπού 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.

Μέτα τον χωρισμό σε macroblocks οι κωδικοποιητές ομαδοποιούν τα καρέ σε Intra και Inter οπού τα τελευταία χωρίζονται σε P και B. Όλες οι παραπάνω κατηγοριοποιήσεις έχουν να κάνουν με τον μηχανισμό του Motion Estimation οπού ο encoder δημιουργεί διαφορές pixels (residuals). Τα residuals έχουν μικρότερη ενεργεία οπότε είναι οι διάφοροι μετασχηματισμοί που εφαρμόζονται αργότερα στην αλυσίδα μας δίνουν μικρότερη ενεργεία και επιπλέον περισσότερα μηδενικά.

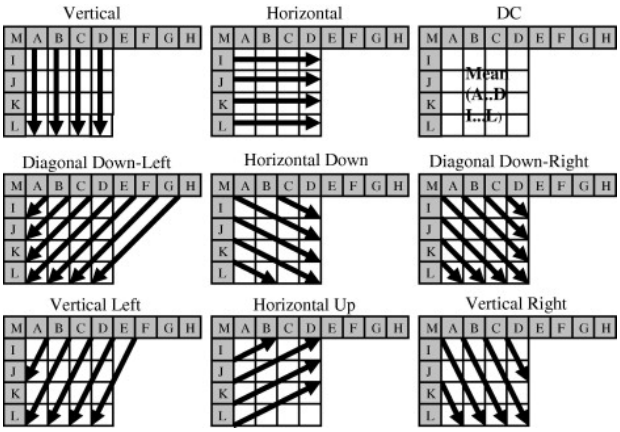
- Intra η αλλιώς I είναι τα καρέ που χρησιμοποιούν πληροφορία μόνο από το τρέχων καρέ για να βγάλουν τα residuals. Αυτό γίνετε απλά παίρνοντας τις τιμές των γειτονικών block και εφαρμόζοντας μια πράξη π.χ μέσος όρος οπού το αποτέλεσμα αυτής αφαιρείται από τις τιμές των pixel του τρέχων block. Υπάρχουν διάφορες πράξεις που μπορούν να γίνουν όπως φαίνεται και στο Σχήμα 2.3 και δοκιμάζονται κάθε φόρα όλες μέχρι να καταλήξουμε στην καλύτερη. 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

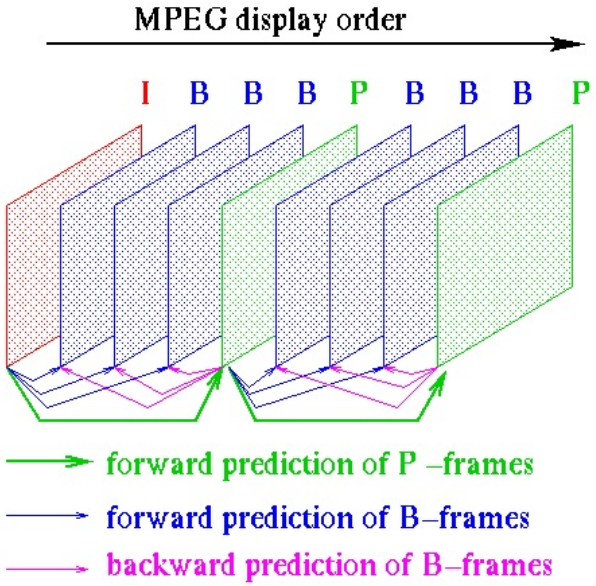
- B καρέ είναι αυτά τα οποία ψάχνουν το καλύτερο block για να κάνουν διαφορές από κάποιο αριθμό προηγούμενων αλλά και επόμενων καρέ όπως φαίνεται στο Σχήμα 2.4. Αυτά έχουν την καλύτερη απόδοση συμπίεσης αλλά η πολυπλοκότητα αποκωδικοποίησης είναι η μεγαλύτερη.

Ένα επιπλέον στοιχείο των encoder είναι το group of pictures (GOP) οπού αυτό 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 διαφέρει κατά πολύ. Ένα τυπικό φαίνεται στο Σχήμα 2.4



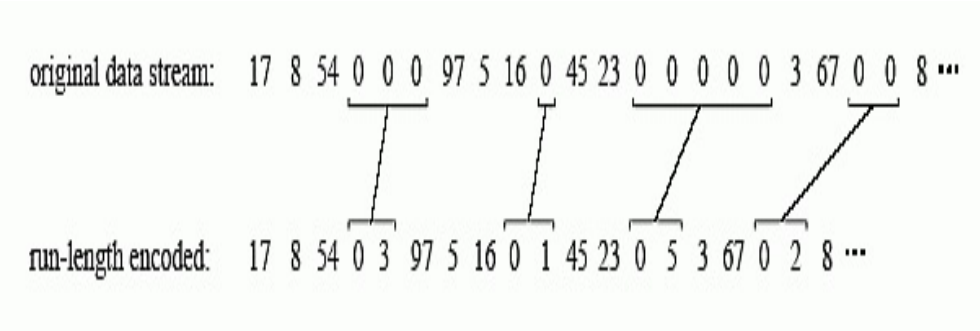
Σχήμα 2.3: Τρόποι που χρησιμοποιούνται στον H.264 για Intra prediction



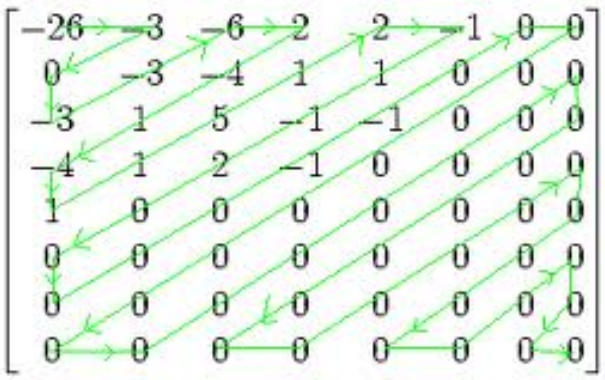
Σχήμα 2.4: Inter prediction στον mpeg και απεικόνιση ενός ενδεικτικοί GOP

2.4 Μετασχηματισμός

Αφού ο encoder έχει 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. 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.



Σχήμα 2.5: Παράδειγμα ένωσης μηδενικών



Σχήμα 2.6: Σάρωση ZigZag

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 αριθμό. Η τακτική που εφαρμόζεται είναι να γίνει ακέραια διαίρεση ο συντελεστής της κάθε συχνότητας με έναν σταθερό αριθμό διαφορετικό για κάθε συχνότητα Σχήμα 2.7. Επομένως για τον δεδομένο πίνακα κβαντοποίησης με τους συντελεστές του Σχήματος 2.8 έχουμε τα αποτελέσματα του Σχήματος 2.9 έτοιμα προς κωδικοποίηση με κάποιον entropy encoder και εγγραφή.

$$Q = \begin{bmatrix} 16 & 11 & 10 & 16 & 24 & 40 & 51 & 61 \\ 12 & 12 & 14 & 19 & 26 & 58 & 60 & 55 \\ 14 & 13 & 16 & 24 & 40 & 57 & 69 & 56 \\ 14 & 17 & 22 & 29 & 51 & 87 & 80 & 62 \\ 18 & 22 & 37 & 56 & 68 & 109 & 103 & 77 \\ 24 & 35 & 55 & 64 & 81 & 104 & 113 & 92 \\ 49 & 64 & 78 & 87 & 103 & 121 & 120 & 101 \\ 72 & 92 & 95 & 98 & 112 & 100 & 103 & 99 \end{bmatrix}.$$

Σχήμα 2.7: Παράδειγμα πίνακα κβαντοποίησης για μετασχηματισμό 8x8

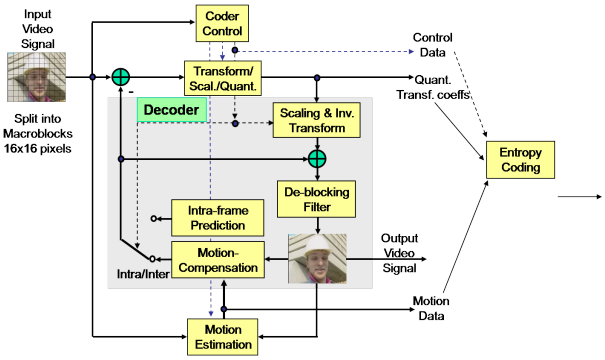
$$G = \begin{bmatrix} & & & & \overset{u}{\longrightarrow} & & & \\ -415.38 & -30.19 & -61.20 & 27.24 & 56.13 & -20.10 & -2.39 & 0.46 \\ 4.47 & -21.86 & -60.76 & 10.25 & 13.15 & -7.09 & -8.54 & 4.88 \\ -46.83 & 7.37 & 77.13 & -24.56 & -28.91 & 9.93 & 5.42 & -5.65 \\ -48.53 & 12.07 & 34.10 & -14.76 & -10.24 & 6.30 & 1.83 & 1.95 \\ 12.12 & -6.55 & -13.20 & -3.95 & -1.88 & 1.75 & -2.79 & 3.14 \\ -7.73 & 2.91 & 2.38 & -5.94 & -2.38 & 0.94 & 4.30 & 1.85 \\ -1.03 & 0.18 & 0.42 & -2.42 & -0.88 & -3.02 & 4.12 & -0.66 \\ -0.17 & 0.14 & -1.07 & -4.19 & -1.17 & -0.10 & 0.50 & 1.68 \end{bmatrix} \downarrow v.$$

Σχήμα 2.8: Παράδειγμα συντελεστών DCT 8x8

$$B = \begin{bmatrix} -26 & -3 & -6 & 2 & 2 & -1 & 0 & 0 \\ 0 & -2 & -4 & 1 & 1 & 0 & 0 & 0 \\ -3 & 1 & 5 & -1 & -1 & 0 & 0 & 0 \\ -3 & 1 & 2 & -1 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}.$$

Σχήμα 2.9: Παράδειγμα συντελεστών DCT 8x8

Παρακάτω βλέπουμε την δομή του H.264 οπού επίτηδες έχω παραλείψει να αναφερθώ σε κάποια κομμάτια τους αφού δεν κρίνονται αναγκαία για την κατανόηση αυτής της διπλωματικής.



Σχήμα 2.10: H.264 structure

2.6 Αποκωδικοποίηση βίντεο

Για την αποκωδικοποίηση κάνουμε όλα τα βήματα προς τα πίσω.

- Μετά την entropy decoding πολλαπλασιάζουμε τα νούμερα με τα στοιχεία του πίνακα κβαντοποίησης. Έτσι παίρνουμε τους DCT συντελεστές αλλά λανθασμένους λόγο της ακέραιας διαίρεσης.
- Έπειτα κάνουμε iDCT και παίρνουμε τα residuals.
- Τελευταίο βήμα είναι να κάνουμε το Motion Decomposition βρίσκοντας τα δύο block που χρησιμοποιήθηκαν για να παραχθούν τα residuals και τα προσθέτουμε για να πάρουμε τα ανακατασκευασμένα pixels.

Κεφάλαιο 3

Θεωρία πληροφοριών

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο γίνεται μια εισαγωγή σε βασικά στοιχεία της Θεωρίας πληροφοριών και κυρίως στην επεξήγηση της έννοιας εντροπία της πληροφορίας. Επίσης θα αναφερθούν αλγόριθμοι συμπίεσης (entropy encoding) καθώς και ο αλγόριθμος kmeans που χρησιμοποιήθηκε για την παραγωγή των codebooks.

3.2 Εντροπία

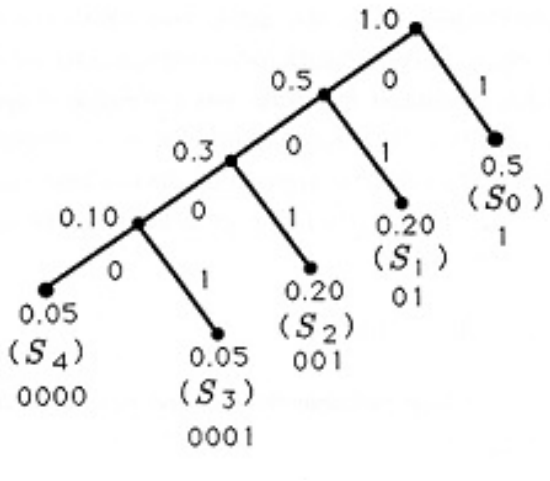
Η εντροπία κατά Shannon που εισήχθηκε από τον Claude E. Shannon το 1948 είναι η ποσοτικοποίηση της αβεβαιότητας μιας τυχαίας μεταβλητής και συνήθως μετριέται σε bits η nats. Η κύρια πληροφορία που παρέχεται από την εντροπία και είναι χρήσιμη σε εμάς είναι το απόλυτο κάτω όριο οπού η πληροφορία μπορεί 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 b είναι οι μονάδα μέτρησής της εντροπίας.Για συμπίεση δεδομένων συνήθως ισχύει $b = 2$ έτσι ώστε να έχουμε την εντροπία σε bits.

Για να κατανοήσουμε την εντροπία θα δοθεί ένα παράδειγμα. Έστω ότι πρέπει να αναπαρασταθεί σε δυαδικό σύστημα μια ακολουθία από αριθμούς $x_i \in [0, 3]$. Επομένως υπάρχουν $n = 4$ διαφορετικά ενδεχόμενα που χρειάζονται 2 bits το καθένα για να αναπαρασταθούν $x_i = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 1 & 1 \end{bmatrix}$. Αν τα ενδεχόμενα είναι ισοπίθανα $p(x_i) = 0.25$ τότε έχουμε $H(X) = 2bits$ άρα δε μπορούμε να κάνουμε κάτι για να τα συμπιέσουμε. Αν όμως είναι $p(x_1) = 0.7, p(x_2) = p(x_3) = p(x_4) = 0.1$ τότε έχουμε $H(X) = 1.35678bits$ ανά σύμβολο κατά μέσο όρο. Επομένως τα δεδομένα μας μπορούν ιδανικά να συμπιεστούν κατά $(1 - 1.35678/2)\% = 32\%$.

3.3 Κωδικοποιητές Εντροπίας

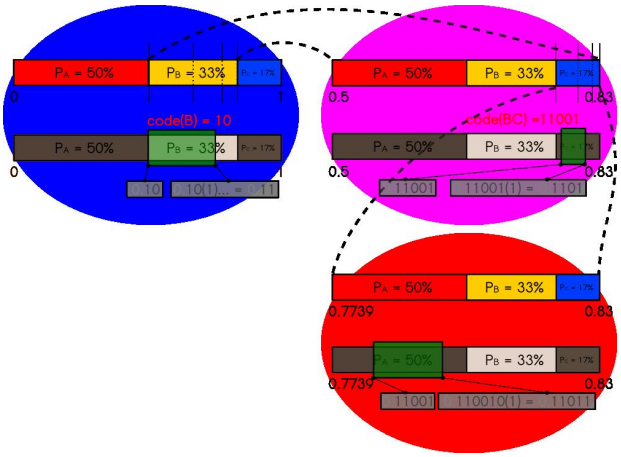
Οι μέθοδοι που σήμερα υπάρχουν για συμπίεση καταφέρνουν να έρθουν πολύ κοντά στο όριο που δίνει η εντροπία αλλά δε το φτάνουν. Δύο είναι οι κυρίαρχες, η μέθοδος Huffman και Αριθμητική κωδικοποίηση με την δεύτερη να είναι αυτή που χρησιμοποιείται περισσότερο στο βίντεο με την παραλλαγή της που λέγεται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

- Η μέθοδος Huffman έχει την μικρότερη πολυπλοκότητα O(nlogn) και μας εξασφαλίζει πώς $H(X) \leq HC \leq H(X) + 1bit$ 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 με στοχο το ενδεχόμενο με την μεγαλύτερη πιθανότητα θα 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 δέντρο Huffman για αυτό το παράδειγμα φαίνεται στο Σχήμα 3.1.



Σχήμα 3.1: Δέντρο Huffman.

- Η μέθοδος της Αριθμητικής Κωδικοποίησης έχει την μεγαλύτερη πολυπλοκότητα αλλά μας εξασφαλίζει μια συμπίεση πού τις περισσότερες φορές είναι αρκετά καλύτερή από αυτή του Huffman και πλησιάζει αρκετά 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 $n \in R$. 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 κωδικοποιείται το μήνυμα "BCA" . Στο Σχήμα 3.2 φαίνονται τα βήματα της κωδικοποίησης οπού 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 σύνεχίζουν και τα υπόλοιπα βήματα μέχρι να έχουμε τον αριθμό που αντιστοιχεί στην ακολουθία.

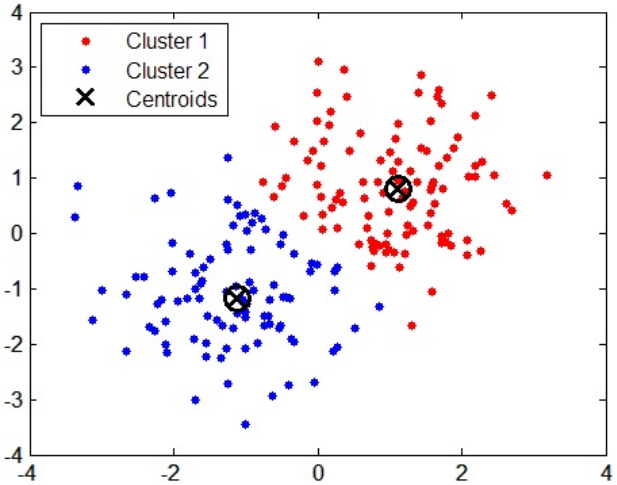


Σχήμα 3.2: Βήματα αριθμητικής κωδικοποίησης.

3.4 Αλγόριθμος clustering kmeans

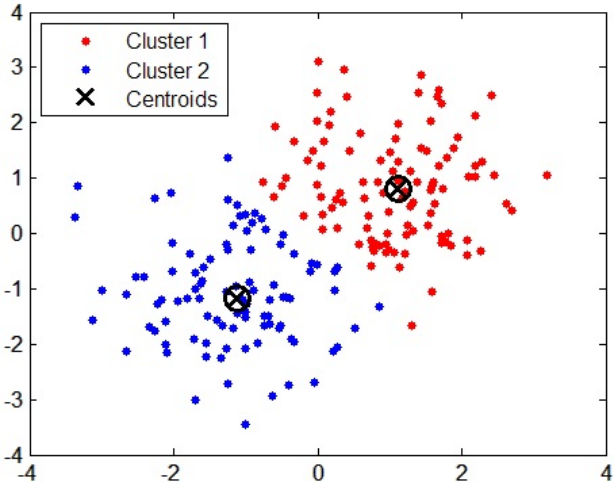
Άλλο ένα στοιχείο που χρησιμοποιήθηκε σε αυτή την διπλωματική και πηγάζει από την Θεωρία πληροφοριών είναι ο αλγόριθμος clustering kmeans. Είναι ένας επαναληπτικός αλγόριθμος όπου στόχος του είναι να χωρίσει με το ελάχιστο σφάλμα n σημεία με διάσταση d σε k περιοχές $k \leq n$ όπως φαίνεται στο Σχήμα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

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

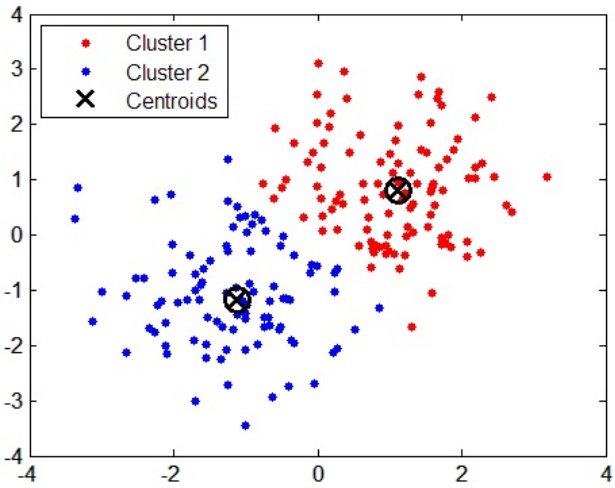


Σχήμα 3.3: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$

Στο Βημα 1 του kmeans γίνετε η αρχικοποίηση στην οποία μπορούμε να επιλέξουμε να είναι τυχαία (δηλαδή κάθε cluster να παίρνει τιμές από ένα τυχαίο point) η να έχουμε κάποια στρατηγική. Στην παρούσα διπλωματική επιλέχθηκε να η στρατηγική KKZ (Katsavounidis Kuo Zhang) η οποία είναι πιο αργή από την τυχαία αλλά ο kmeans συγκλίνει πιο γρήγορα και σε καλύτερο σφάλμα. Ο αλγόριθμος παραλληλοποιήθηκε και επιτεύχθηκε γραμμική επιτάχυνση δοκιμασμένο 1,2,4,8,12,24 νήματα. Στο Σχήμα 3.4 φαίνεται η επιτάχυνση του παράλληλου αλγορίθμου και στο Σχήμα 3.5 φαίνονται οι συνολικές επαναλήψεις που κάνει ο kmeans με τυχαία και KKZ αρχικοποίηση.

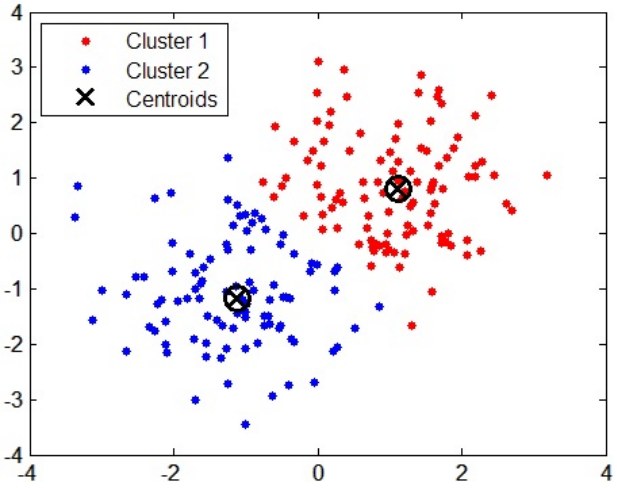


Σχήμα 3.4: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$

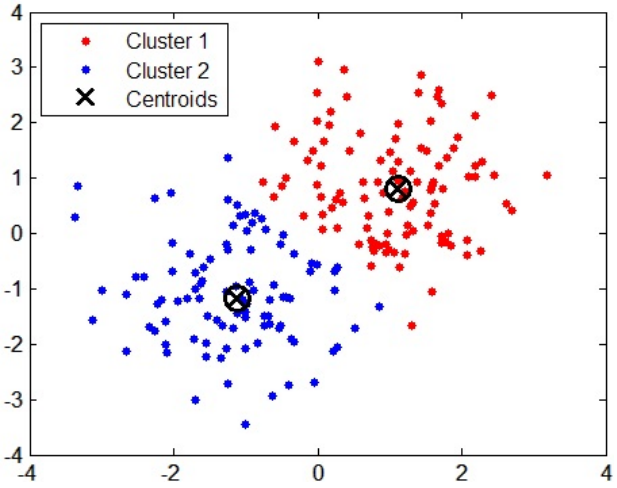


Σχήμα 3.5: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$

Η πιο σημαντική βελτιστοποίηση επιτεύχθηκε στο σημείο της αναζήτησης του κοντινότερου cluster στο Βήμα 5. Στην απλή εκδοχή του αλγορίθμου για κάθε data point σαρώνονται όλα τα clusters για να βρεθεί ποιο είναι πιο κοντά. Στην διπλωματική αυτή χρησιμοποιήθηκε ο αλγόριθμος Fast Nearest Neighbour οπού οργανώνει τις αποστάσεις σε ένα δυαδικό δέντρο και μας επιτρέπει να κάνουμε από k αναζητήσεις log(k) για κάθε data point. Αυτό δίνει μία τεράστια επιτάχυνση στο όλο πρόβλημα και σε συνδυασμό με την παραλληλοποίηση του Βήματος 4 μας επέτρεψε να τρέχουμε μεγάλα πειράματα σε εύλογο χρονικό διάστημα. Οι επιταχύνσεις που επιτεύχθηκαν με την χρήση του FastNN φαίνεται στο Σχήμα 3.6 και της παραλληλοποίησης του Βήματος 4 στο Σχήμα 3.7



Σχήμα 3.6: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$



Σχήμα 3.7: kmeans με $k = 2, d = 2, n = 100, MSE = 284.671$

Για τα πειράματα της διπλωματικής χρησιμοποιήθηκε ένας 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.