ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Επιτάχυνση της διαδικασίας συμπίεσης video με χρήση δεδομένων αισθητήρων και τεχνικές παραλληλοποίησης

Παναγιώτης Μπελεμένης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέποντες:

Αθανάσιος Λουκόπουλος

Επίκουρος Καθηγητής

Λαμία, 2018

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

Επιτάχυνση της διαδικασίας συμπίεσης video με χρήση δεδομένων αισθητήρων και τεχνικές παραλληλοποίησης

Παναγιώτης Μπελεμένης

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέποντες

Αθανάσιος Λουκόπουλος

Επίκουρος Καθηγητής

Λαμία, 2018

Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. *Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί* ***χωρίς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. *Δέχομαι ότι η αυτολεξεί* ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. *Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

4. *Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: ……/…../20……

Ο – Η Δηλ.

(Υπογραφή)

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.

Επιτάχυνση της διαδικασίας συμπίεσης video με χρήση δεδομένων αισθητήρων και τεχνικές παραλληλοποίησης

Παναγιώτης Μπελεμένης

Τριμελής Επιτροπή:

ΛΟΥΚΟΠΟΥΛΟΣ ΑΘΑΝΑΣΙΟΣ, Επίκουρος Καθηγητής

ΔΗΜΗΤΡΙΟΣ ΙΑΚΩΒΙΔΗΣ, Αναπληρωτής Καθηγητής

ΙΩΑΝΝΗΣ ΑΝΑΓΝΩΣΤΟΠΟΥΛΟΣ, Αναπληρωτής Καθηγητής

ΠΕΡΙΛΗΨΗ

Στόχος αυτής της εργασίας είναι η επιτάχυνση της διαδικασίας κωδικοποίησης ενός βίντεο, χρησιμοποιώντας τόσο τα στοιχεία της κίνησης μιας συσκευής εγγραφής βίντεο όσο και τεχνικές παραλληλοποίησης wavefront.

Στο πρώτο κεφάλαιο γίνεται μια ιστορική αναδρομή της συμπίεσης βίντεο καθώς και μια σύντομη περιγραφή της διαδικασίας κωδικοποίησης.

Στο δεύτερο κεφάλαιο γίνεται μια εκτενής περιγραφή του coding standard του HEVC το οποίο και θα χρησιμοποιήσουμε.

Στο τρίτο κεφάλαιο, αναλύεται η κατασκευή μιας εφαρμογής android η οποία θα βιντεοσκοπεί και παράλληλα θα συλλέγει τα δεδομένα για την κίνηση του βίντεο κατά τη διάρκεια της βιντεοσκόπησης.

Στο τέταρτο κεφάλαιο, θα περιγραφεί ο τρόπος με τον οποίο αυτά τα δεδομένα μετατρέπονται σε μορφή που θα είναι ικανός ο κωδικοποιητής HM -ο κωδικοποιητής του HEVC- να τα χρησιμοποιήσει. Επίσης αναπτύσσονται διάφοροι μέθοδοι για την επιτάχυνση της διαδικασίας της κωδικοποίησης μέσω των δεδομένων για την κίνηση.

Στο πέμπτο κεφάλαιο, γίνεται η παραλληλοποίηση του γραμμικού κωδικοποιητή HM.

Στο έκτο κεφάλαιο, λαμβάνει μέρος η πειραματική αξιολόγηση των μεθόδων που αναπτύχθηκαν στο τέταρτο και το πέμπτο κεφάλαιο.

ΠΕΡΙΕΧΟΜΕΝΑ

6. Πειραματική Αξιολόγηση

ΚΕΦΑΛΑΙΟ 1

Εισαγωγή

Με την έλευση των κοινωνικών δικτύων και των έξυπνων συσκευών στην καθημερινότητά μας, ο όγκος των δεδομένων βίντεο που διακινούνται στο Διαδίκτυο έχει εκτιναχθεί. Σύμφωνα με την μελέτη προβλέψεων της Cisco το 2019[1] τα δεδομένα βίντεο θα αντιστοιχούσαν στο 80% του φόρτου κινητής τηλεφωνίας. Συγχρόνως, η ανάγκη για ολοένα μεγαλύτερη ανάλυση με την πρόσφατη έλευση των 4K καμερών και 8K τηλεοράσεων σημαίνει ότι το ποσοστό των δεδομένων βίντεο στο Διαδίκτυο αναμένεται να αυξηθεί.

Η μετάδοση ασυμπίεστου (raw) βίντεο, συνεπάγεται τεράστιες ανάγκες σε εύρος ζώνης. Ως εκ τούτου από τη δεκαετία του ’80 η συμπίεση βίντεο έχει τύχει μεγάλης ερευνητικής δραστηριότητας[2]. Κατά τη φάση της συμπίεσης, το αρχικό βίντεο κωδικοποιείται και μετατρέπεται σε μια αναπαράσταση της πληροφορίας (bitstream) μικρότερη σε μέγεθος. Ύστερα το bitstream αποκωδικοποιείται, δηλαδή μεταφράζεται έτσι ώστε να παράξει το αρχικό βίντεο. Ανακύπτει, λοιπόν η ανάγκη για την ύπαρξη ενός κοινού τρόπου με τον οποίο κωδικοποιείται και αποκωδικοποιείται ένα βίντεο. Αυτή η ανάγκη καλύπτεται με τα πρότυπα κωδικοποίησης βίντεο (video coding standards).

Ένα coding standard περιγράφει τη δομή του bitstream καθώς και τον τρόπο αποκωδικοποίησης του. Παρέχει ένα λεξιλόγιο δηλαδή με το οποίο το αρχικό βίντεο συμπιέζεται και με το οποίο ένα συμπιεσμένο βίντεο αποσυμπιέζεται. Ένα coding standard μπορεί να παρέχει lossless compression, όταν η αποκωδικοποίηση ενός βίντεο παράγει ένα βίντεο που είναι ίδιο με το αρχικό, ή lossy compression, όταν έχουμε απώλειες στην ποιότητα, δηλαδή το αποσυμπιεσμένο βίντεο δεν είναι πιστό αντίγραφο του αρχικού. Το lossless compression αν και μας δίνει το βέλτιστο αποτέλεσμα όσον αφορά την ποιότητα, ενέχει μικρό βαθμό συμπίεσης σε αντίθεση με το lossy compression. Ανακύπτει, λοιπόν, ένας συμβιβασμός μεταξύ ποιότητας και μεγέθους της πληροφορίας.

Το πρώτο standard δημιουργήθηκε το H.120 το 1984 από την ITU. Η ποιότητα του bitstream ήταν τόσο χαμηλή που το standard θεωρήθηκε μη πρακτικό. Το πρώτο πρακτικό standard ήρθε το 1988 από την ITU, ονομαζόμενο H.261.Το 1993 δημιουργήθηκε το MPEG-1 με σκοπό την αποθήκευση βίντεο και ήχου σε CD-ROMs. Ακολούθησε το MPEG-2 το 1995 το οποίο χρησιμοποιείται για τη μετάδοση όσο και για την αποθήκευση βίντεο. Το 1999 δημιουργήθηκε το MPEG-4 το οποίο αποτελεί

βελτίωση των προηγουμένων. Το 2003 ήρθε το H.264 το οποίο είναι το κυρίαρχο standard μέχρι σήμερα.

Πρότυπο	Έτος	Κύριο Application	Ανάλυση
H.120	1984	Μη πρακτικό	-
H.261	1988	Μετάδοση Βίντεο	352×288 176×144
MPEG-1	1993	Αποθήκευση Βίντεο Ήχου σε CD-ROMs	352 x 240,352 x 288
MPEG-2	1995	Μέταδοση και Αποθήκευση Βίντεο	128×96 176×144 352×288 704×576 1408×1152
MPEG-4	1999	Βίντεο στο Ίντερνετ	
H.264	2003	Αποθήκευση βίντεο σε υλικό και Μετάδοση	128×96 εώς 4,096×2,304
H.265	2012	Αποθήκευση βίντεο σε υλικό και Μετάδοση	Μέχρι 8192×4320

Σχήμα 1.1: Σύγκριση χρήσεων και υποστηριζόμενων αναλύσεων των codecs.

Δύο είναι τα κύρια χαρακτηριστικά του βίντεο που προσπαθούν να εκμεταλλευτούν τα πρότυπα με στόχο την επίτευξη μεγάλου βαθμού συμπίεσης με μικρή συνακόλουθη μείωση ποιότητας. Το πρώτο είναι η εκμετάλλευση των χωρικών συσχετίσεων ανάμεσα στα εικονοστοιχεία μιας εικόνας. Κοντινά εικονοστοιχεία είναι πολύ πιθανό να παρουσιάζουν ομοιότητες στην ένταση και στο χρώμα οι οποίες μπορεί κατάλληλα κωδικοποιούμενες (Intra coding) ώστε να οδηγήσουν σε μείωση των bits που απαιτούνται για την αναπαράσταση της πληροφορίας.

Το δεύτερο χαρακτηριστικό αφορά στις χρονικές συσχετίσεις. Με άλλα λόγια μεταξύ διαδοχικών εικόνων, τα εικονοστοιχεία μιας εικόνας συνήθως παρουσιάζουν ομοιότητες με τα αντίστοιχα ή κοντινά εικονοστοιχεία μιας επόμενης ή προηγούμενης εικόνας. Ως εκ τούτου μπορεί να αναπαρασταθεί η πληροφορία που περιέχεται σε διαδοχικές εικόνες αθροιστικά με λιγότερα bits (Inter coding).

Όλα τα κυριότερα πρότυπα των τελευταίων δεκαετιών (από το H.261[3] και μετά) υλοποιούν τεχνικές για Intra και Inter coding. Στην τρέχουσα φάση το πιο

διαδεδομένο πρότυπο, με ευρεία χρήση σε έξυπνες κινητές συσκευές και κάμερες, είναι το H.264/AVC[4] που οριστικοποιήθηκε το 2007. Το H.264/AVC εισήγαγε καινοτομίες όπως πολύ περισσότερα Intra coding modes, περισσότερα modesγια VBSME (Variable Block Size Motion Estimation), κωδικοποίηση εντροπίας με CABAC (Context-Adaptive Binary Arithmetic Coding), slices για μετάδοση μέρους εικόνας και παραλληλοποίηση κλπ. Αν και οι καινοτομίες αυτές αρκούσαν για την εποχή των HD(High Definition) και Full HD αναλύσεων, στην εποχή του 4K το πρότυπο δείχνει τις αδυναμίες του. Για παράδειγμα ο προτεινόμενος ρυθμός μετάδοσης 4K βίντεο κωδικοποιημένου με H.264/AVC από το YouTube αγγίζει ή/και ξεπερνά τα 40 Mbps[5].

Για το λόγο αυτό εισήχθησαν νέα και πιο αποδοτικά πρότυπα κωδικοποίησης με κυριότερα το HEVC (High Efficiency Video Coding) από το MPEG group το 2012 (επονομαζόμενο και σαν H.265), και το VP9 από την Googleτο 2013. Τα δύο αυτά πρότυπα πετυχαίνουν μεγαλύτερο βαθμό συμπίεσης σε σχέση με το H.264/AVC για την ίδια ποιότητα βίντεο. Ιδιαιτέρως το HEVC το οποίο σε σειρά συγκρίσεων πχ. [6], [7] έχει αποδειχτεί καλύτερο από το VP9, πετυχαίνει σημαντική αύξηση της συμπίεσης σε σχέση με το H.264/AVC [8].

Θα πρέπει να σημειωθεί πως αν και η απόδοση του HEVC το καθιστά πιθανώς τον καλύτερο υποψήφιο για αντικατάσταση του H.264/AVC εντούτοις, για την ώρα έχει αποτύχει στο να εκπληρώσει το σκοπό αυτό κυρίως λόγω του δύσκαμπτου συστήματος εξασφάλισης πνευματικών δικαιωμάτων που το συνόδευε [9]. Αν και έγιναν αλλαγές στο σχήμα χορήγησης με στόχο την ευρεία είσοδο του προτύπου στην αγορά, η κατάσταση ακόμα είναι ρευστή, κυρίως με την έλευση του προτύπου AV1 το Μάρτιο του 2018 ως μετεξέλιξη του VP9, από ένα συνασπισμό εταιρειών την AOMedia [10], στον οποίο συμμετέχουν εταιρείες όπως η Google, η Netflixκλπ.

Παρόλα αυτά μελέτες όπως [11] και [12] έχουν καταδείξει ότι όσον αφορά τα υπάρχοντα λογισμικά κωδικοποίησης το HEVC φαίνεται να πετυχαίνει καλύτερη απόδοση σε σχέση με το βασικό λογισμικό του AV1, τουλάχιστον όσον αφορά σε κωδικοποίηση με σταθερό QP (Quantization Parameter), φτάνοντας βελτίωση του bitrate της τάξης του 30%. Επιπλέον είναι σε εξέλιξη η δημιουργία του διάδοχου προτύπου του HEVC με την ονομασία VVC (Versatile Video Coding) που υπόσχεται 50% μεγαλύτερη συμπίεση σε σχέση με το HEVC. Συνακόλουθα η τρέχουσα κατάσταση του λογισμικού για AV1 ενέχει κωδικοποιητή που είναι πολλές φορές 2 τάξεις μεγέθους πιο αργός (άρα όχι ακόμα πρακτικός) σε σχέση με το codec x265 [13] που είναι αρκετά διαδεδομένο εμπορικά για HEVC.

Για τους παραπάνω λόγους, στα πλαίσια της πτυχιακής εργασίας επιλεχτεί το HEVC ως βάση για συγκριτική ανάλυση των τεχνικών βελτιστοποίησης χρόνου που αναπτύχθηκαν. Η ανάγκη για τέτοιες τεχνικές, προκύπτει από το γεγονός ότι το υπάρχον λογισμικό κωδικοποίησης HEVC, πχ. HM [14], x265 κλπ., δεν πετυχαίνει

realtime απόδοση, ή αν επιτυχαίνει αυτό γίνεται μόνο κατόπιν θυσιών στην ποιότητα που προκύπτουν από παραλήψεις/συμβιβασμούς ως προς τη χρήση ολόκληρων μεθόδων και τεχνικών που περιγράφονται ως ενδεδειγμένα βήματα από το πρότυπο, πχ., half και quarter pixel motion estimation.

Στόχος της παρούσας πτυχιακής είναι να μελετήσει τεχνικές που θα μειώνουν το χρόνο κωδικοποίησης χωρίς σημαντικές μειώσεις στην ποιότητα και το βαθμό συμπίεσης. Μεγάλο μέρος της κωδικοποίησης λαμβάνει χώρα στο επίπεδο του hardware μιας κάμερας ή έξυπνης κινητής συσκευής. Ένα άλλο ολοένα αυξανόμενο πεδίο εφαρμογής είναι σε συστήματα Υπολογιστικού Νέφους όπου η ανάγκη είναι για λογισμικό κωδικοποίησης. Λαμβάνοντας υπόψη και τα παραπάνω σενάρια, στην παρούσα πτυχιακή εργασία οι μέθοδοι που εξετάζονται αφορούν δυνητικά και στα δύο. Πιο συγκεκριμένα, εξετάζεται κατά πόσο η χρήση των αισθητήρων μιας έξυπνης συσκευής μπορεί να καθοδηγήσει τη διαδικασία του motion estimation στο Inter coding, καθώς επίσης και ο συνδυασμός της με παραλληλοποίηση τύπου wavefront.

Το υπόλοιπο της πτυχιακής εργασίας χωρίζεται σε 5 κεφάλαια. Στο Κεφάλαιο 2 παρουσιάζεται μια γενική περιγραφή του HEVC στάνταρ. Το τρίτο κεφάλαιο ασχολείται με την ανάπτυξη μιας εφαρμογής σε περιβάλλον android που να τραβάει βίντεο και ταυτόχρονα να χρησιμοποιεί τα αισθητήρια όργανα ώστε να μας ενημερώνει για την κίνηση της κινητής συσκευής σε κάθε frame του βίντεο, καθώς ανακύπτει η ανάγκη για παραγωγή dataset που θα περιέχουν τόσο το βίντεο όσο και τα στοιχεία για την κίνηση της συσκευής εγγραφής κατά τη διάρκεια του βίντεο. Στο τέταρτο κεφάλαιο παρουσιάζονται τρόποι με τους οποίους μπορεί να εισαχθούν τα δεδομένα για την κίνηση της κινητής συσκευής στον κωδικοποιητή HM του HEVC, καθώς και με τους τρόπους όπου ο κωδικοποιητής είναι ικανός να εκμεταλλευτεί αυτήν την πληροφορία. Το πέμπτο κεφάλαιο έχει να κάνει με την παραλληλοποίηση με χρήση wavefront του κωδικοποιητή HM πού κατά βάση είναι ένας σειριακός κωδικοποιητής. Τέλος, στο κεφάλαιο 6 θα πραγματοποιηθεί μια αξιολόγηση των μεθόδων που αναπτύχθηκαν τόσο για την παραλληλοποίηση όσο και για την εκμετάλλευση της κίνησης. Η αξιολόγηση θα γίνει με βάση την ποιότητα των παραγόμενων βίντεο καθώς και το μέγεθος των παραγόμενων αρχείων.

Κεφάλαιο 2

Γενική Περιγραφή του HEVC Standard

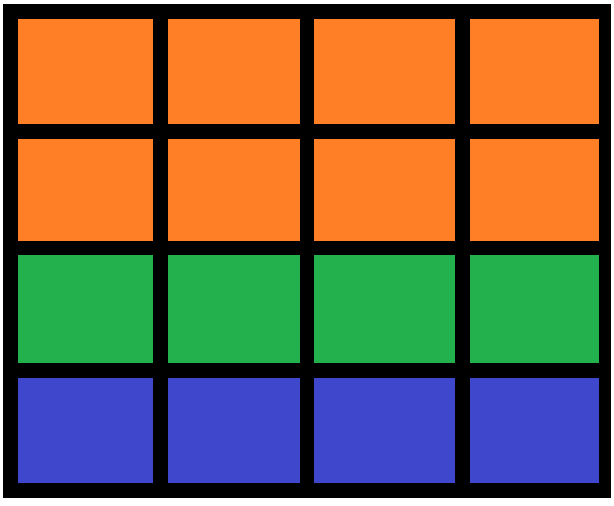
2.1 Γενικά

Πριν την εκκίνηση της διαδικασίας της κωδικοποίησης είναι αναγκαία η μετατροπή του βίντεο σε YUV format. Το YUV format ορίζεται ως κατάλληλο γιατί λαμβάνει υπόψιν το πώς αντιλαμβάνεται ο άνθρωπος τα χρώματα.

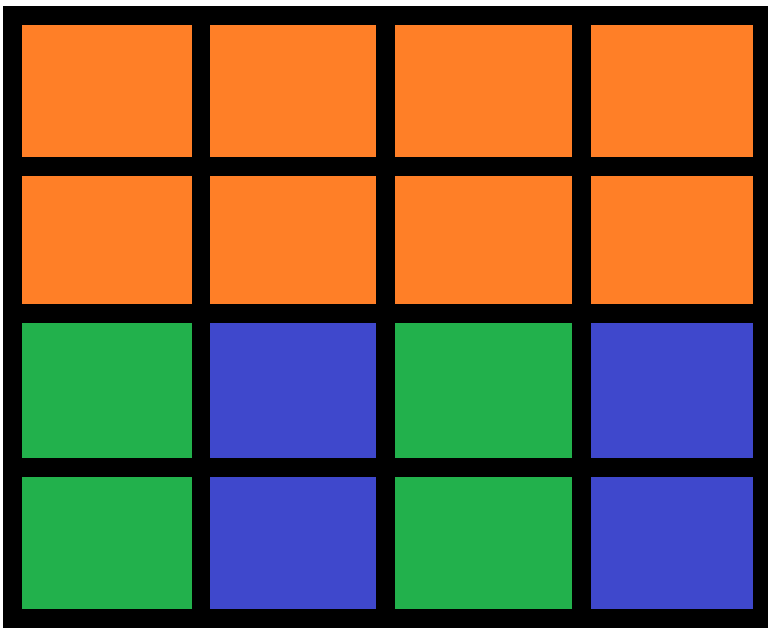
Ένα pixel YUV αποτελείται από το δείκτη φωτεινότητας(Y) και τους δείκτες χρωμάτων(UV). Χρησιμοποιούνται μόνο δυο δείκτες χρωμάτων, πιο συγκεκριμένα για το κόκκινο και το μπλε, καθώς είναι συμπληρωματικοί του τρίτου, του πράσινου.

Εφόσον ο άνθρωπος αντιλαμβάνεται πιο έντονα τη φωτεινότητα, μπορούμε να χρησιμοποιήσουμε τη φωτεινότητα για κάθε pixel και να χρησιμοποιήσουμε τους ίδιους δείκτες χρώματος για γειτονικά pixels, κερδίζοντας έτσι σε μέγεθος της πληροφορίας χάνοντας αμελητέα σε ποιότητα. Έτσι προκύπτουν διάφορα formats για την αναπαράσταση αυτή.

Οι κωδικοποιητές υποστηρίζουν διάφορα τέτοια formats με πιο διαδεδομένο να είναι το 4:2:0.



Σχήμα 2.1: Format YUV 420. Πορτοκαλί: Y. Πράσινο: U. Μπλε: V



Σχήμα 2.2: Format NV21

Η διαδικασία του encoding εκμεταλλεύεται το χωρικό και το χρονικό πλεονασμό. Χωρικός πλεονασμός υπάρχει όταν γειτονικά pixels μέσα σε ένα frame έχουν την ίδια τιμή ή εμφανίζουν έντονη ομοιότητα. Ο χρονικός πλεονασμός γίνεται αντιληπτός από την ύπαρξη ομοιοτήτων μεταξύ γειτονικών frames όταν δηλαδή διάφορες περιοχές του βίντεο μένουν στατικές ή είναι μετατοπισμένες από frame σε frame.

2.2 Διαχώριση Εικόνας σε Ομάδες (Μπλοκ) Εικονοστοιχείων

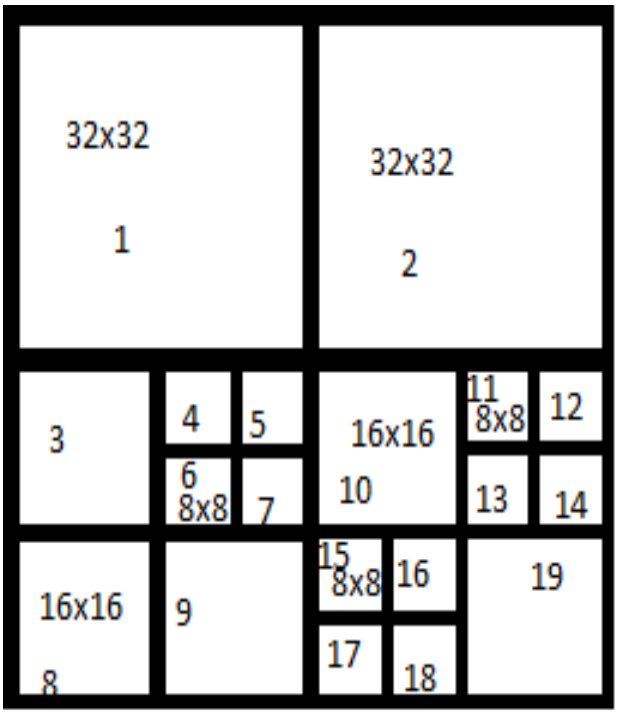
Για να κωδικοποιηθεί ένα frame χωρίζεται σε blocks από pixels. Στα προηγούμενα standards το μέγεθος αυτών των blocks ήταν σταθερό με αποτέλεσμα μεγάλο bitrate αφού δεν αξιοποιούνταν αρκετά ο χωρικός και ο χρονικός πλεονασμός για την παραγωγή μικρότερων αρχείων. Στο HEVC χρησιμοποιούνται τα CTUs(Coding Tree Units) ώστε να αξιοποιηθεί ο πλεονασμός της πληροφορίας χωρίς την περεταίρω μείωση της ποιότητας.

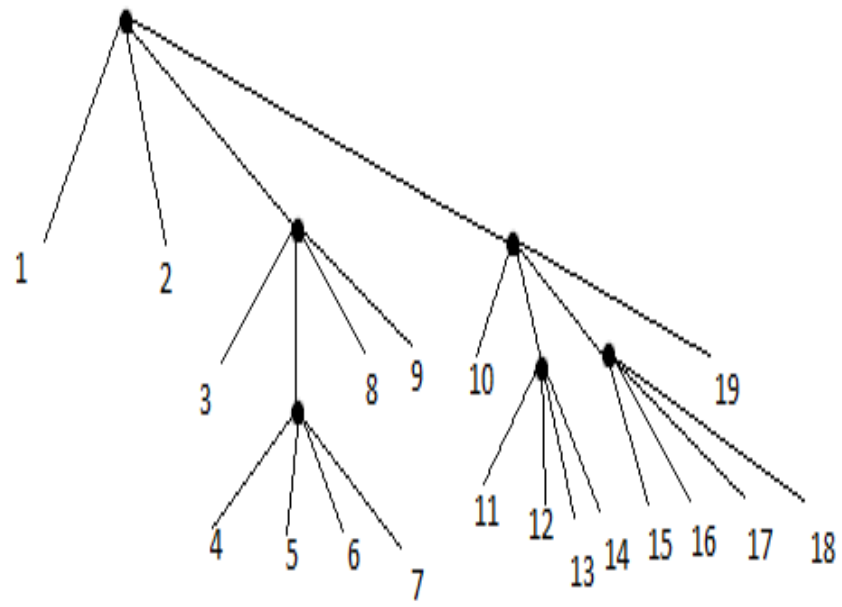
Τα CTUs είναι blocks διαστάσεων 64x64, το καθένα από τα οποία κωδικοποιείται ξεχωριστά από τα άλλα. Ένα CTUs έχει δομή δέντρου και αποτελείται από CUs(Coding Units). Ύστερα, στο επίπεδο CU γίνεται η επιλογή μεταξύ inter και intra prediction[15].

Τα CUs μπορούν να είναι διαστάσεων 64x64,32x32,16x16,8x8. Ένα CU των 64x64 μπορεί να χωριστεί σε τέσσερα CUs των 32x32, ένα CU των 32x32 μπορεί να χωριστεί σε τέσσερα CUs των 16x16 και ένα CU των 16x16 μπορεί να χωριστεί σε τέσσερα CUs των 8x8.

Η δομή CTU με τα μεταβαλλόμενα μεγέθη των CUs στοχεύει σε ένα συμβιβασμό μεταξύ της μείωσης του μεγέθους της κωδικοποιημένης πληροφορίας(bits) και της ελαχιστοποίησης της παραμόρφωσης σε σχέση με το αρχικό βίντεο, καλύτερο από τα προηγούμενα standards όπου το μέγεθος των blocks ήταν σταθερό. Έτσι, ένα block κωδικοποιείται ανάλογα με την πολυπλοκότητα του, στοχεύοντας έτσι στην καλύτερη αποδοτικότητα της διαδικασίας.

Έτσι, ένας κωδικοποιητής δοκιμάζει αναδρομικά όλα τα partitions για να βρει τον καλύτερο καταμερισμό του CTU σε CUs. Αυτή η διαδικασία αν και δίνει καλύτερα αποτελέσματα είναι υπολογιστικά πολύπλοκη και χρονοβόρα.

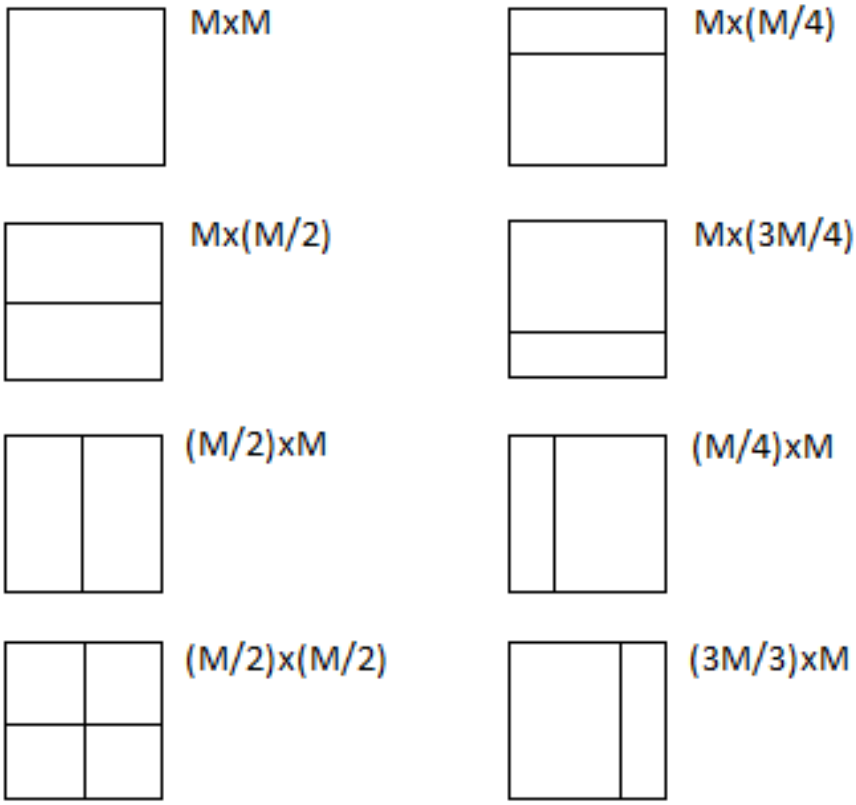




Σχήμα 2.3: Παράδειγμα δομής CTU. Αριστερά: διαίρεση του CTU σε CTUS. Δεξιά: Αναπαράσταση της διαίρεσης σε μορφή δέντρου.

CTUs μεγαλύτερων διαστάσεων ενώ παρέχουν καλύτερη αποδοτικότητα όσον αφορά τη διαδικασία κωδικοποίησης αυξάνουν την υπολογιστική πολυπλοκότητα, αφού εισάγουν περισσότερα βάθη και κατ' επέκταση περισσότερα partitions τα οποία θα πρέπει να εξεταστούν.

Για κάθε Coding Unit που χωρίζεται πρέπει να γίνει επιλογή ανάμεσα σε intra-picture και inter-frame prediction. Αυτό επιτυγχάνεται με την περαιτέρω διαίρεση των CUs σε PUs(prediction Units) για ακόμα καλύτερη αποδοτικότητα. Για το κάθε PU επιλέγεται είτε intra prediction είτε inter prediction. Επίσης χρησιμοποιείται για το Transform, το οποίο εξηγείται παρακάτω. Όσον αφορά το μέγεθος τους μπορεί να είναι από 64x64 μέχρι 4x4. Στο HEVC υπάρχουν 8 τρόποι με τους οποίους μπορεί να χωριστεί ένα CU. Έτσι το CU μπορεί να κωδικοποιηθεί είτε ως ένα σύνολο PU.

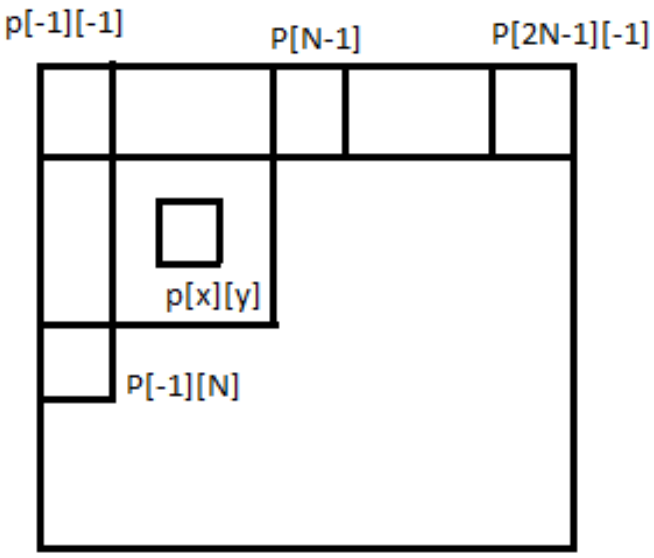


Σχήμα 2.4: Modes διαίρεσης του Coding Unit σε Prediction Unit όπου Μ το μέγεθος του CU.

2.3 Τύποι Πρόβλεψης

Intra-picture Prediction

Όπως αναφέρθηκε και παραπάνω η διαδικασία αυτή αφορά την αξιοποίηση του χωρικού πλεονασμού. Σκοπός της είναι να προβλεφθούν οι τιμές ενός block διαστάσεων N x N βάσει ενός πίνακα αναφοράς δισδιάστατο πίνακαρ 2Nx 2N. Με αυτόν τον τρόπο θα χρειαστούν 2N x 2 pixels για να ανακατασκευαστεί το block N x N, Θα αναφερόμαστε στα reference pixels ως τις περιοχές p[-1][-1..2N-1] και p[-1..2N-1] [-1] και στο block ως την περιοχή p[0..N-1][0..N-1],όπως φαίνεται στο παρακάτω σχήμα.



Σχήμα 2.6: Ο πίνακας p που χρησιμοποιείται για τη διαδικασία intra –picture prediction

Με αυτόν τον τρόπο χρειαζόμαστε απλά τα 2N reference pixels για να ανακατασκευάσουμε το N x N block, χρησιμοποιώντας έτσι λιγότερα bits. Χρησιμοποιώντας τα reference pixels στο HEVC υπάρχουν 35 τρόποι με τους οποίους μπορεί να ανακατασκευαστεί το block. Σκοπός είναι η επιλογή του τρόπου με τον οποίο το ανακατασκευασμένο block που προκύπτει έχει παραμορφωθεί το λιγότερο σε σχέση με το αρχικό [16].

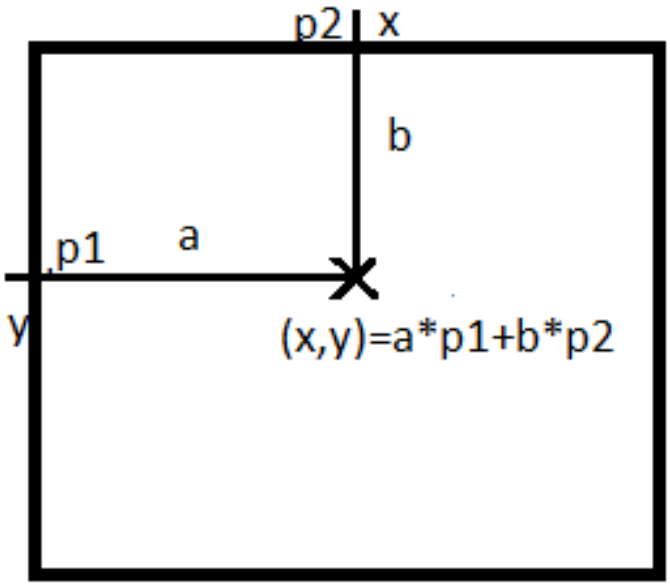
Η μέθοδος 0 ονομαζόμενη Planar, η είναι ικανή να παράγει ομαλές επιφάνειες χωρίς την ύπαρξη ασυνεχειών στο block. Ορίζεται ως:

$$p[x][y] = (ph[x][y] + pv[x][y] + N) >> (log2(N) + 1), \text{ όπου}$$

$$ph[x][y] = (N - 1 - x \cdot p[-1][y] + (x + 1) \cdot p[N][-1] \text{και}$$

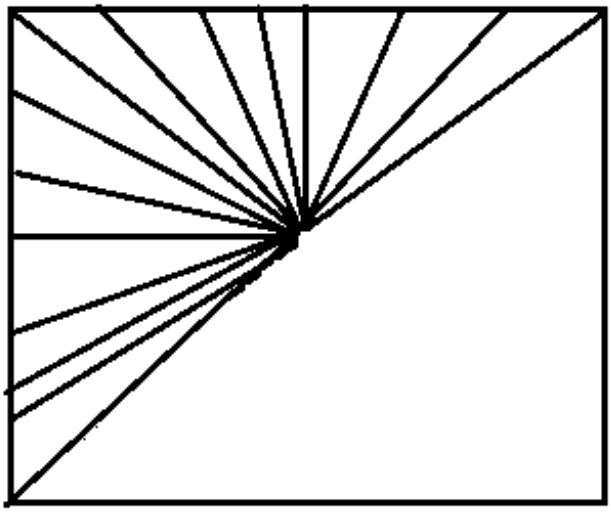
$$ph[x][y] = (N - 1 - v) \cdot p[x][-1] + (y + 1) \cdot p[-1][N] \qquad 2.1$$

Η μέθοδος 1 ονομαζόμενη DC mode, η οποία για την πρόβλεψη του στοιχείου p[x][y] χρησιμοποιεί έναν γραμμικό συνδυασμό του p[x][-1] και του p[-1][y].Είναι η πιο απλή των μεθόδων υπολογιστικά.



Σχήμα 2.7: DC mode

Οι μέθοδοι 2-34 είναι angular,δηλαδή επιλέγεται μία κατεύθυνση βάσει με την οποία το p[x][y] θα επιλέξει τα pixels αναφοράς p[1...2N[-1] και p[-1][1…2N]. Είναι αποτελεσματικές για τα blocks τα οποία παρουσιάζουν πλήθος ακμών.



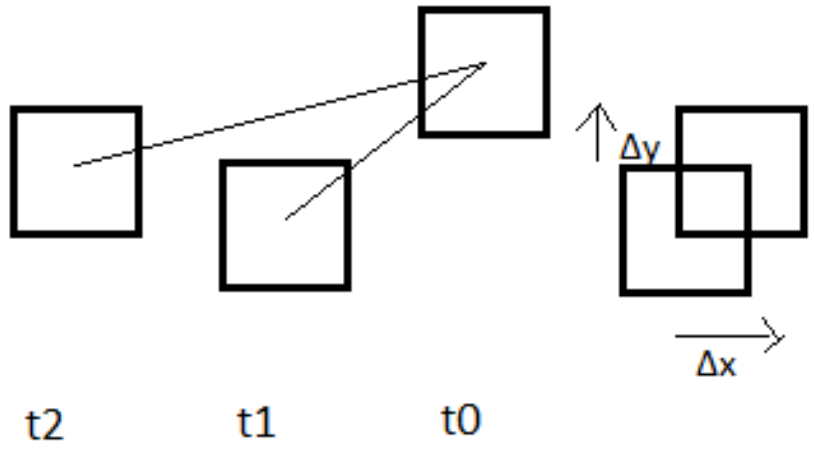
Σχήμα 2.8: Angular modes

Μετά την πρόβλεψη ακολουθεί μια διαδικασία post processing. Η λειτουργία της ορίζεται από την επεξεργασία των reference pixels με τη χρήση διάφορων φίλτρων με σκοπό τη διόρθωση όποιων ατελειών.

Με αυτόν τον τρόπο, η μετάδοση κάποιων reference pixels και της μεθόδου πρόβλεψης είναι αρκετή για την ανακατασκευή του αρχικού block στο συμπιεσμένο βίντεο.

Inter-picture prediction

Με αυτή τη διαδικασία πραγματοποιείται η αξιοποίηση του χρονικού πλεονασμού. Θεωρώντας ότι ένα αντικείμενο κινείται ανάμεσα στα frames ενός βίντεο, αρκεί να περιγράψουμε την κίνηση των blocks του αντικειμένου αυτού και έτσι να μη χρειαστεί να ορίσουμε εκ νέου αυτά τα blocks για τα επόμενα frames.



Σχήμα2.9: Inter-picture Prediction, όπου (Δx,Δy) το moving vector του block τη χρονική στιγμή t0 και t2, t1 οι χρονικές στιγμές που αντιστοιχούν σε προηγούμενα frames.

Για την επίτευξη αυτού θα χρειαστεί να οριστεί το block, το reference frame και η μετατόπιση (Δx,Δy) στους άξονες x,y αντίστοιχα, η οποία καλείται motion Vector. Έτσι για ένα block προκύπτει η δομή δεδομένων (Δx,Δy,Δt). Ο encoder στην ουσία μετατοπίζει το block σε ένα επόμενο frame κατά διάφορες κατευθύνσεις μέχρι να βρει το καλύτερο fit, δηλαδή το block του επομένου frame που ταυτίζεται με το

αρχικό. Τα (Δx,Δy) μπορεί να είναι δεκαδικοί αριθμοί για ακόμα μεγαλύτερο βαθμό ακρίβειας[17].

Υπάρχουν δύο είδη inter-picture prediction:

1)Uni-prediction: Προκύπτει ένα διάνυσμα (Δx,Δy,Δt)

2)Bi-prediction: Προκύπτουν δύο διανύσματα (Δx,Δy,Δt) , πολλές φορές από διαφορετικά frames, τα οποία συνδυάζονται ώστε να παράγουν το τελικό διάνυσμα. Χρησιμοποιούνται διάφοροι τρόποι για το συνδυασμό τους, όπως ο μέσος όρος τους ή κάποιος άλλος γραμμικός συνδυασμός.

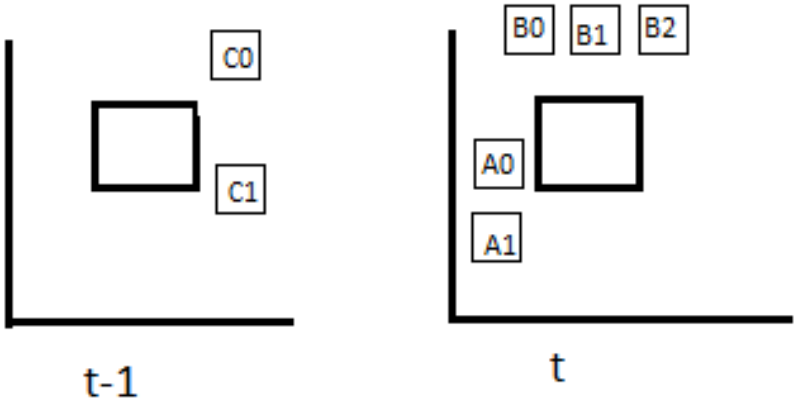
Στο HEVC, η μετατόπιση των blocks κωδικοποιείται σχετικά με συγκεκριμένα motion vectors, ονομαζόμενα motion vector predictors, τα οποία είναι ορισμένα από το ίδιο το HEVC standard. Έτσι αρκεί να υπολογίσουμε τη διαφορά μεταξύ της μετατόπισης Δx,Δy και του predictor. Στο επίπεδο του encoder, η χρήση των predictors περιορίζει την αναζήτηση στις πιο πιθανές περιοχές καθιστώντας τη διαδικασία λιγότερο χρονοβόρα.

$$MVDx = \Delta x - MVPx$$

$$MVDy = \Delta y - MVPy \qquad 2.2$$

Το HEVC χρησιμοποιεί predictors οι οποίοι προκύπτουν από την κίνηση γειτονικών block στο ίδιο frame(spatial), predictors που προκύπτουν βάσει της κίνησης του ίδιου του block στα προηγούμενα frame(temporal) καθώς και predictors που υποθέτουν ότι η κίνηση του block είναι μηδενική.

Πριν κωδικοποιηθεί το διάνυσμα κίνησης θα πρέπει να ληφθεί υπόψιν η απόσταση μεταξύ του τρέχοντος frame και του reference frame. Έτσι ανακύπτει η ανάγκη για ένα παράγοντα αντιπροσωπευτικό της χρονικής απόστασης, το λεγόμενο scale factor. Έτσι το διάνυσμα που προκύπτει θα είναι ανάλογο του scale factor.



Σχήμα2.10: (a) Πως προκύπτουν οι temporal predictors C0,C1(PRED C),(b) Πως προκύπτουν οι spatial predictors A0,A1(PRED A) καιB0,B1,B2(PRED B).

Το HEVC υποστηρίζει επίσης και τη συγχώνευση blocks για την καλύτερη αναπαράσταση της κίνησης ενός αντικειμένου και κατόπιν την πιο αποτελεσματική κωδικοποίηση της πληροφορίας.

Έτσι, η αξιοποίηση της κίνησης των αντικειμένων του βίντεο στη διαδικασία της κωδικοποίηση έχει ως αποτέλεσμα την καλύτερη αποδοτικότητα, κυρίως στο μέγεθος της πληροφορίας που μεταδίδεται.

Η διαδικασία της συγχώνευσης χρησιμοποιεί μια δομή παρόμοια με το AMVP, τη λίστα υποψηφίων για συγχώνευση. Η λίστα αυτή αποτελείται από:

1)Τέσσερεις υποψηφίους από τα γειτονικά blocks.

2)Δύο υποψηφίους που προκύπτουν από κάποιο block που προέρχεται από κάποιο προηγούμενο frame.

3)Δύο πρόσθετους υποψηφίους που λαμβάνουν υπόψιν τη μηδενική κίνηση καθώς και το συνδυασμό των υποψηφίων της κίνησης.

Interpolation filters

Στην περίπτωση που τα Δx,Δy είναι δεκαδικοί αριθμοί θα πρέπει να γίνει μετατροπή τους σε ακέραιους. Αυτό πραγματοποιείται με τη χρήση interpolation filters.

Αυτά συνδυάζουν τα διανύσματα (Δx,Δy,Δt) που προκύπτουν είτε από το bi-prediction, είτε από δύο διανύσματα που προκύπτουν από uni-prediction. Ύστερα, αφού ο συνδυασμός των Δx,Δy είναι δεκαδικός θα αναπαριστάται με περισσότερα bits από ότι ένας ακέραιος, οπότε εφαρμόζεται ένα bit-shifting για τη μετατροπή σε ακέραιο.

TZSearch

Προφανώς η αναζήτηση του κατάλληλου motion vector ανάμεσα σε όλα τα pixels θα ήταν χρονοβόρα. Για αυτό ο κωδικοποιητής χρησιμοποιεί μια μορφή αναζήτησης η οποία αποσκοπεί στην αναζήτηση λιγότερων σημείων, για την ελάφρυνση των υπολογισμών, με την όσο δυνατόν καλύτερη επιλογή του κατάλληλου motion vector,για τη μικρότερη μείωση της ποιότητας. Η διαδικασία αυτή ονομάζεται tzsearch.

Ο Κωδικοποιητής HM διαλέγει τον κατάλληλο τρόπο αναζήτησης ως εξής:

```
If( isFullSearch || bBi){
    fullSearch();
}

if(isDiamondEnabled){
        tzSearch();
}
else if(isSelectiveSearchEnbled){
        tzSearchSelective();
}
else if{isEncancedSearchEnabled}{
        tzSearchEnhanced();
}
```

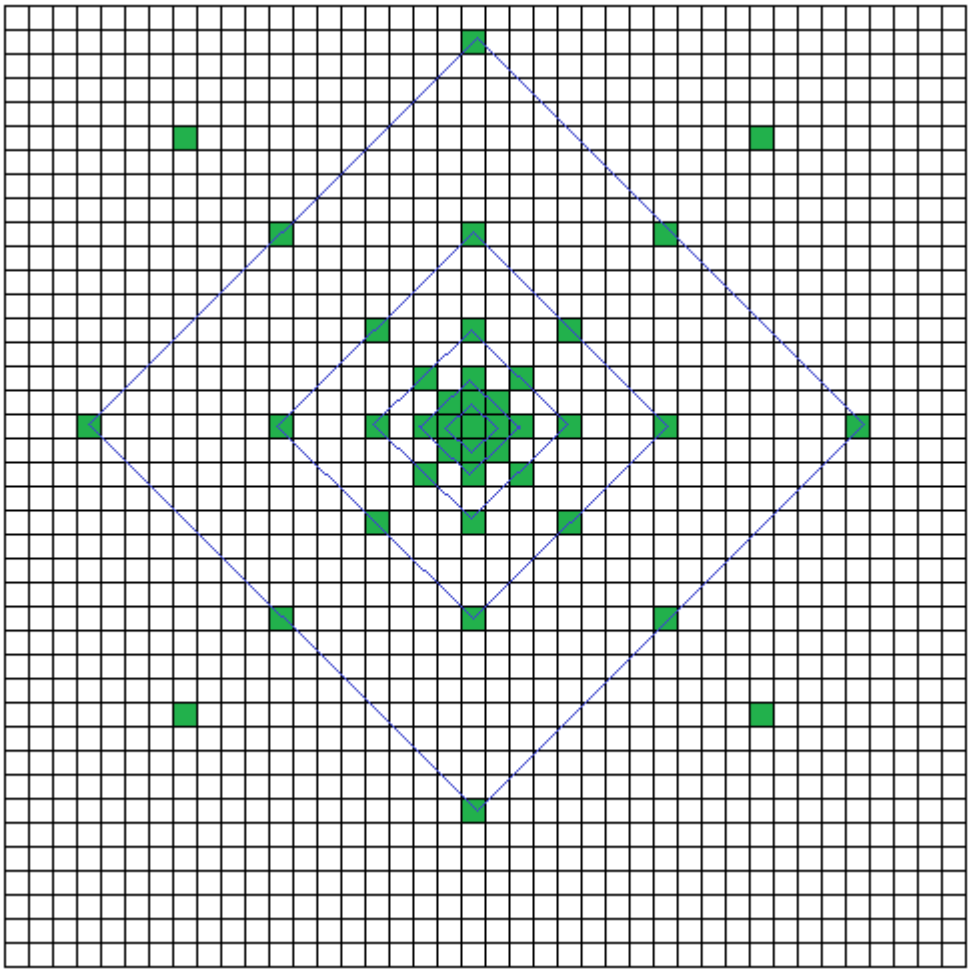
Όπου isFullSearch=false, καθώς και όπου isDiamondSearchEnabled=true, isSelectiveSearchEnabled=false και isEnhancedSearchEnabled=false από τις default ρυθμίσεις του κωδικοποιητή. bBi παίρνει την τιμή true αν πρόκειται για biPredictive prediction και false αν όχι.

Η TZSearch αποτελείται από τρία μέρη:

Επιλογή του κατάλληλου αρχικού σημείου αναζήτησης: Σε αυτό το στάδιο εξετάζονται ορισμένα σημεία: ο median predictor, τα σημεία PREDA, PREDB, PREDC καθώς και το μηδενικό σημείο (0,0) ,δηλαδή ο predictor για το οποίο γίνεται το motion estimation. Το σημείο που επιλέγεται ορίζεται ως το βέλτιστο και ως το αρχικό της αναζήτησης

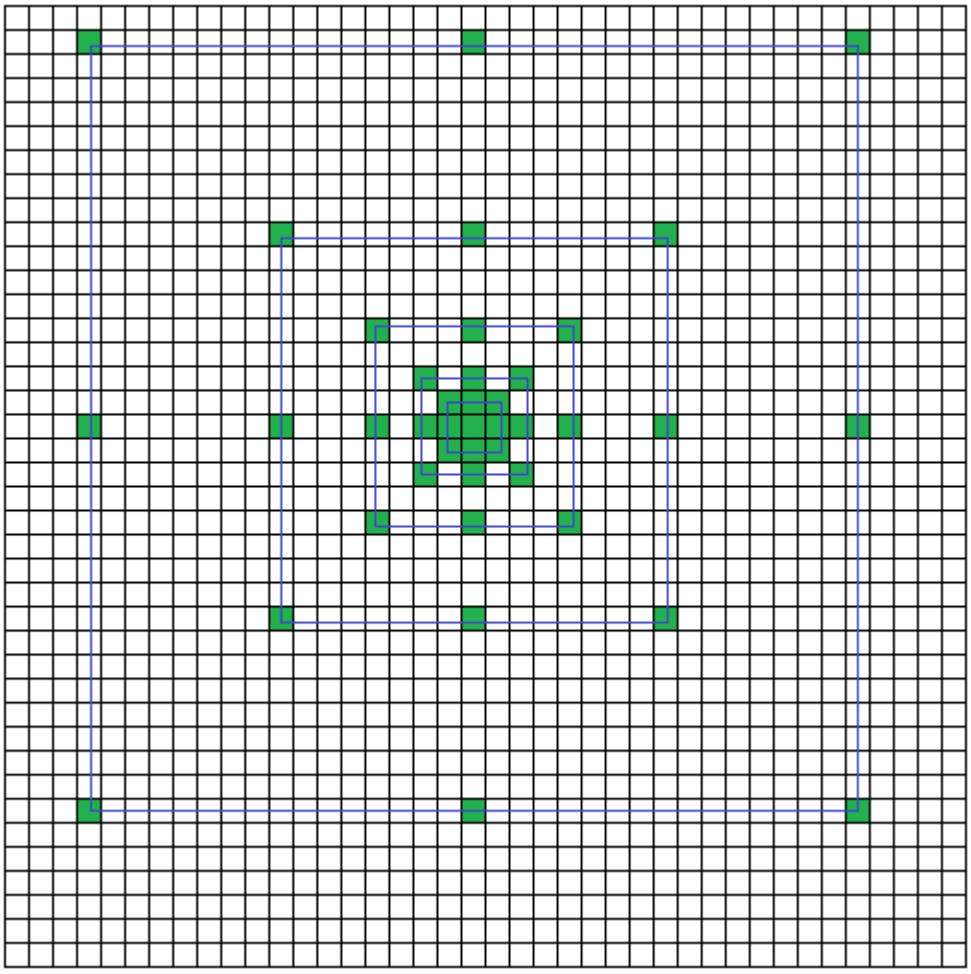
Μπορεί να γίνει εύκολα αντιληπτό ότι η αναζήτηση όλων των σημείων εντός του Search Range θα ήταν διαδικασία χρονοβόρα. Για το λόγο αυτό η αναζήτηση περιορίζεται σε ορισμένα σημεία. Έτσι, η διαδικασία γίνεται πιο γρήγορη με μια μικρή απώλεια ποιότητας. Ο κωδικοποιητής χρησιμοποιεί 2 τέτοια μοτίβα επονομαζόμενα από το σχήμα τους:

1)DiamondSearch: Κάθε βήμα του μοτίβου εκτυπώνει έναν ρόμβο. Εκτελούνται τέσσερα τέτοια βήματα.



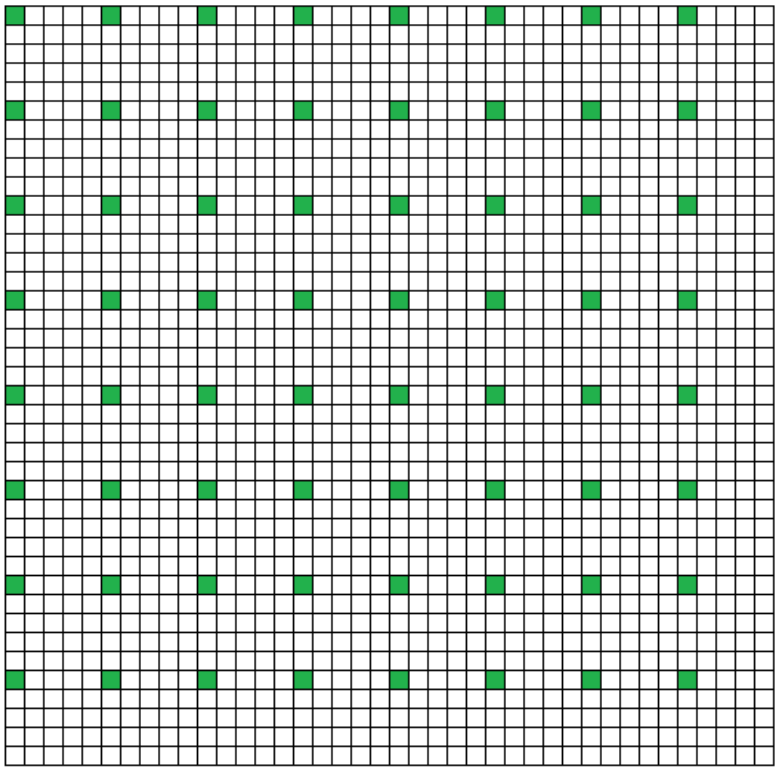
Σχήμα 2.11: Μοτίβο Diamond Search, με μπλε χρώμα αναπαριστάται το κάθε βήμα

2)SquareSearch:Κάθε βήμα του μοτίβου εκτυπώνει ένα τετράγωνο. Εκτελούνται τέσσερα τέτοια βήματα.



Σχήμα 2.12: Μοτίβο SquareSearch, με μπλε αναπαριστάται το κάθε βήμα

Περαιτέρω αναζήτηση Raster: Σε κάθε βήμα τα παραπάνω μοτίβα εξετάζουν 8 σημεία. Είναι προφανές λοιπόν ότι για μεγάλες αποστάσεις δε θα είναι αποτελεσματικά. Γι αυτό το λόγο χρησιμοποιείται το μοτίβο Raster το οποίο είναι ομοιόμορφο στο χώρο.



Σχήμα 2.13 Μοτίβο Raster Search

Η διαδικασία αυτή πραγματοποιείται για όλους τους AMVP candidates.

Στον κωδικοποιητή γίνεται ένας συμβιβασμός μεταξύ full search και TZSearch για τη βελτιστοποίηση των αποτελεσμάτων.

Γενικά, το HEVC δεν εισάγει τόσο κάτι νέο όσον αφορά το inter-picture prediction όσο αποτελεί μια βελτίωση των όσων μεθόδων έχουν υλοποιηθεί παλιότερα στα προηγούμενα standards.

2.4 Παραλληλοποίηση σε Επίπεδο Ομάδων CTUs

Το HEVC επίσης υποστηρίζει διάφορα partitions τα οποία επιτρέπουν την παράλληλη επεξεργασία σε πολυπυρηνικά συστήματα. Αυτές είναι:

1)Slices: Πρόκειται για ομάδες CTUs οι οποίες κωδικοποιούνται με ένα βαθμό ανεξαρτησίας μεταξύ τους. Έτσι μια εικόνα μπορεί να χωριστεί σε ένα σύνολο slices . Υπάρχουν τρία είδη slices [18][19]:

1)I-Slices: Slicesγια τα οποία γίνεται αποκλειστικά intra picture prediction

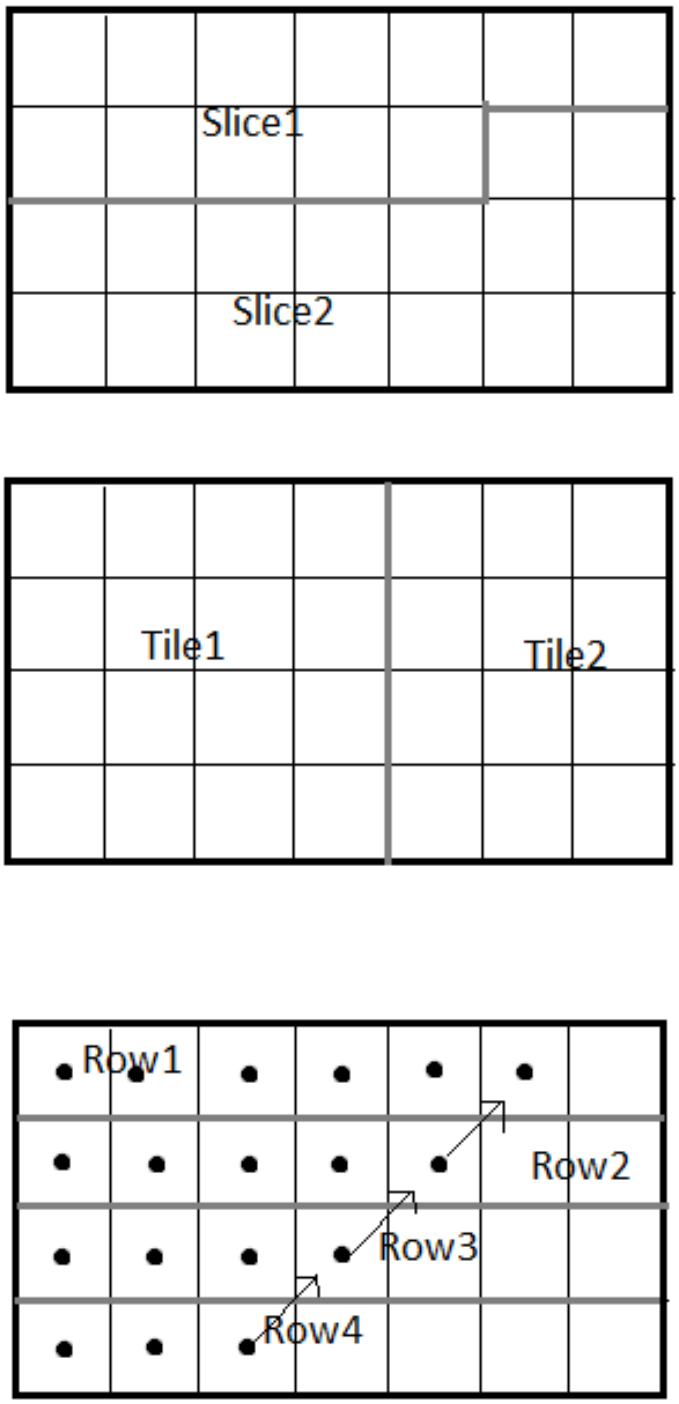
2)P-Slices: Slices για τα οποία γίνεται εκτός από intra picture prediction και inter picture prediction σε P-mode, δηλαδή χρησιμοποιούνται μόνο προηγούμενα frames ως reference.

3)B-Slices: Slices για τα οποία γίνεται εκτός από intra picture prediction και inter picture prediction σε B-mode, δηλαδή χρησιμοποιούνται προηγούμενα καθώς και μελλοντικά frames ως reference.

Ο βαθμός ανεξαρτησίας τους μπορεί να διαφέρει. Όταν είναι εντελώς ανεξάρτητα υπάρχει καλύτερη αξιοποίηση ενός παράλληλου συστήματος αλλά απώλεια στην ποιότητα και τη συμπίεση αφού δεν αξιοποιείται η σχέση μεταξύ τους. Όταν ο βαθμός ανεξαρτησίας τους είναι μεγάλος υπάρχει overhead αφού απαιτείται επικοινωνία μεταξύ slices αλλά η απώλεια σε ποιότητα και συμπίεση κυμαίνεται σε χαμηλότερα επίπεδα. Αυτό καθιστά αυτή τη μορφή slices κατάλληλη σε περιπτώσεις που η αποδοτικότητα της κωδικοποίησης είναι πρωτίστης σημασίας.

2) Tiles: Αυτή η δομή χωρίζει την εικόνα σε ανεξάρτητες ορθογώνιες περιοχές που ανήκουν στο ίδιο slice header, σε αντίθεση με τα slices όπου το κάθε slice έχει το δικό του slice header. Με αυτή τη δομή επιτυγχάνεται η μέγιστη αξιοποίηση ενός παράλληλου συστήματος.

3)Wavefronts: Κάθε γραμμή από CTUs επεξεργάζεται ανεξάρτητα από τις άλλες με προϋπόθεση ότι για την επεξεργασία ενός CTU πρέπει να έχει ολοκληρωθεί η επεξεργασία του προηγούμενου CTU στην ίδια γραμμή καθώς και η επεξεργασία των πάνω και πάνω δεξιά CTUs τα οποία βρίσκονται στην προηγούμενη γραμμή. Αν και ο βαθμός ανεξαρτησίας είναι μικρότερος εδώ, η απώλεια σε ποιότητα και βαθμό συμπίεση είναι σχεδόν αμελητέα.



Σχήμα 2.5: (a) Υλοποίηση Slices στο χώρο της εικόνας,(b) Υλοποίηση tiles στο χώρο της εικόνας,(c) Υλοποίηση wavefront στο χώρο της εικόνας, οι κουκίδες αναπαριστούν τις εξαρτήσεις του κάθε CTUμε το CTU πάνω δεξιά.

2.5 Transform, Κβαντισμός και κώδικας εντροπίας

Εκτός από την πρόβλεψη για μια εικόνα, υπολογίζεται και το σφάλμα της πρόβλεψης αυτής σε σχέση με την αρχική εικόνα. Αυτό το σφάλμα κωδικοποιείται μαζί με την πρόβλεψη για τη βελτιστοποίηση της πιστότητας με το αρχικό ασυμπίεστο βίντεο βάσει της αρχής:

$$B = B' + error \qquad 2.3$$

Όπου B η αρχική εικόνα, B’ η προβλεπόμενη εικόνα και errorτο σφάλμα της πρόβλεψης.

Το σφάλμα αυτό περνάει από ένα DCT φίλτρο, αφού οι σημαντικές αποκλίσεις από την αρχική εικόνα θα υπάρχουν στις χαμηλές συχνότητες. Ακολουθεί κβαντισμός του για περαιτέρω συμπίεση και lossless συμπίεση με έναν κωδικοποιητή εντροπίας.

Η διαδικασία αυτή χρησιμοποιεί τη δομή pixels ονομαζόμενη Transform Block(TB). Το HEVC υποστηρίζει τέσσερεις διαστάσεις: 4x4, 8x8,16x16 και 32x32.

Για την αποκωδικοποίηση θα ακολουθηθεί η αντίστροφη διαδικασία, δηλαδή αποκωδικοποίηση εντροπίας, ψευδοαντριστοφή του κβαντισμού και αντιστροφή του DCT φίλτρου.

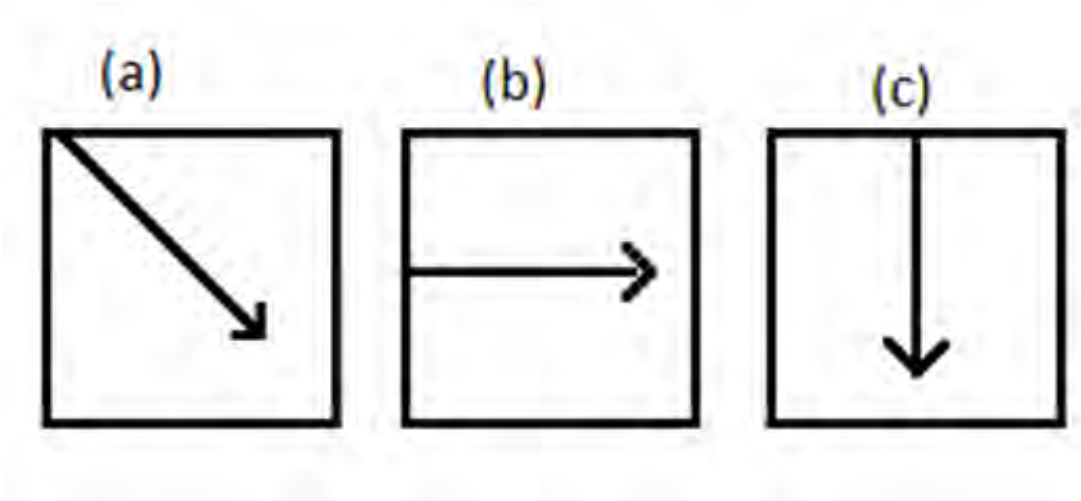
Ο κβαντισμός ανάλογα με το βήμα που θα επιλεγεί καθιστά και τόσες τιμές DCT μηδενικές. Αυτό οδηγεί σε μεγαλύτερη συμπίεση. Να σημειωθεί ότι αφού το στάδιο του κβαντισμού δε μπορεί να αντιστραφεί πλήρως το ανακατασκευασμένο σφάλμα δε θα είναι ακριβές.

Πολλά από τα παλιότερα standards χρησιμοποιούσαν αριθμητικούς κωδικοποιητές εντροπίας, παραδείγματος χάριν Huffman coding, οι οποίοι δε λάμβαναν υπόψιν τη σχέση μεταξύ γειτονικών pixels.

Το HEVC χρησιμοποιεί το Context Adaptive binary arithmetic Coding(CABAC). Το CABAC χρησιμοποιεί για τις όσο πιο συχνά εμφανιζόμενες τιμές τόσα λιγότερα bits, ενώ εκμεταλλεύεται τη σχέση μεταξύ γειτονικών τιμών στην εικόνα δεδομένου του

ότι οι γειτονικές τιμές συσχετίζονται μεταξύ τους, προσφέροντας έτσι καλύτερη συμπίεση από τους απλούς αριθμητικούς κωδικοποιητές εντροπίας.

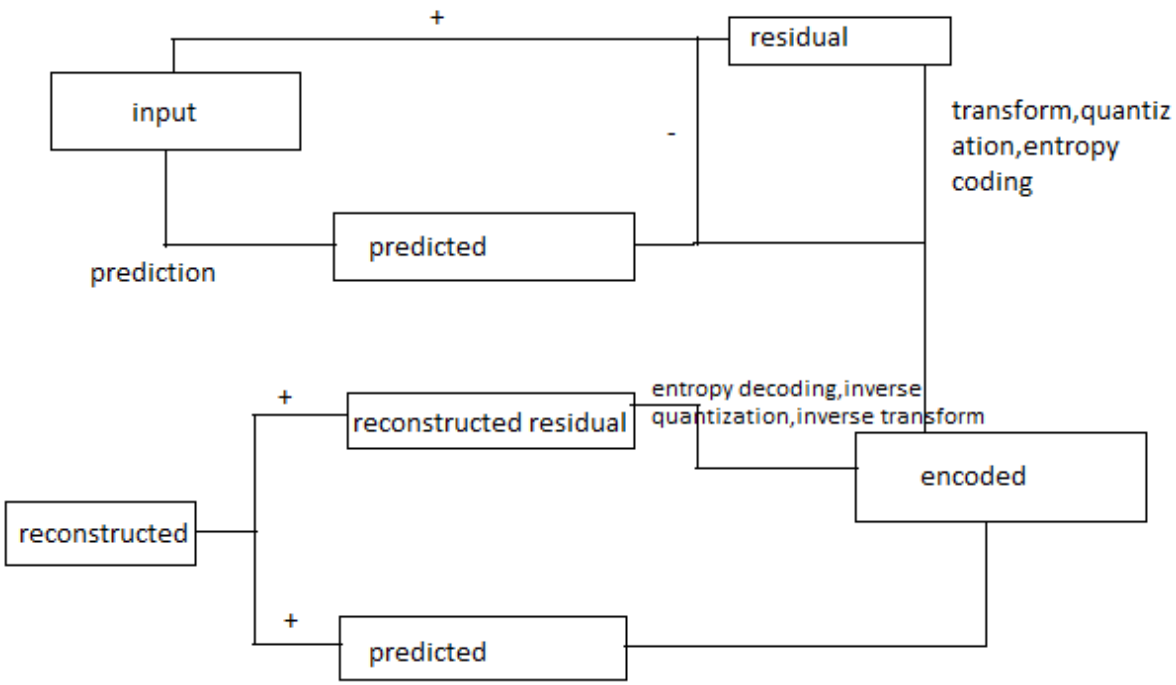
Ο κωδικοποιητής εντροπίας επιλέγει μια κατεύθυνση βάσει με την οποία θα σκανάρει το block. Αφού η κατεύθυνση έχει επιλεγεί απομένει η έκφραση των τιμών DCT. Επειδή μεγάλο πλήθος των τιμών DCT θα είναι μηδενικές αρκεί να εκφραστούν μόνο οι μη μηδενικές τιμές και πόσες μηδενικές τιμές βρίσκονται μεταξύ τους.



Σχήμα 2.14: (a)Άνω διαγώνια κατεύθυνση,(b)Οριζόντια κατεύθυνση,(c)Κάθετη κατεύθυνση

Αρκεί λοιπόν ένα διάνυσμα (run, level, last), όπου run το πόσες μηδενικές τιμές υπάρχουν μεταξύ της προηγούμενης μη μηδενικής τιμής και της συγκεκριμένης, level τιμή της κβαντισμένης τιμής και last η boolean τιμή για το αν το block έφτασε στο τέλος του ή όχι. Αυτά τα διανύσματα βάσει ενός λεξικού μετατρέπονται σε binary(δυαδική) μορφή, της οποίας το μέγεθος εξαρτάται από τη συχνότητα

εμφάνισης των τιμών αυτών.



Σχήμα 2.15: Αναπαράσταση της διαδικασίας encodingκαι decoding

Στο HEVC επίσης χρησιμοποιούνται διάφορα flags τα οποία δημιουργούνται κατά την κωδικοποίηση ενός CU για την περεταίρω αύξηση της αποδοτικότητας της κωδικοποίησης εντροπίας.

Το HEVC υποστηρίζει τρεις κατευθύνσεις scan: την οριζόντια, την κάθετη και την άνω διαγώνια. Αν και χρησιμοποιεί λιγότερες από άλλα standards όπως το AVC, μαζί με τα παραπάνω flags τα οποία παρέχουν παραπάνω context τα αποτελέσματα τόσο στη συμπίεση όσο και στην ποιότητα του ανακατασκευασμένου βίντεο είναι καλύτερα[19].

2.6 Σύνοψη

Το HEVC αποτελεί βελτίωση των προηγούμενων standards, με την καινοτομία των CTUs και τις τροποποιήσεις τόσο των προβλέψεων σε intra και inter επίπεδο όσο και των κωδικοποιητών εντροπίας. Σαφώς αυτές οι βελτιώσεις καθιστούν τη διαδικασία της κωδικοποίησης χρονικά πιο πολύπλοκη από τα προηγούμενα στάνταρ. Η built-in υποστήριξη της παραλληλοποίησης βοηθάει στη βελτίωση της πολυπλοκότητας σημαντικά.

ΚΕΦΑΛΑΙΟ 3

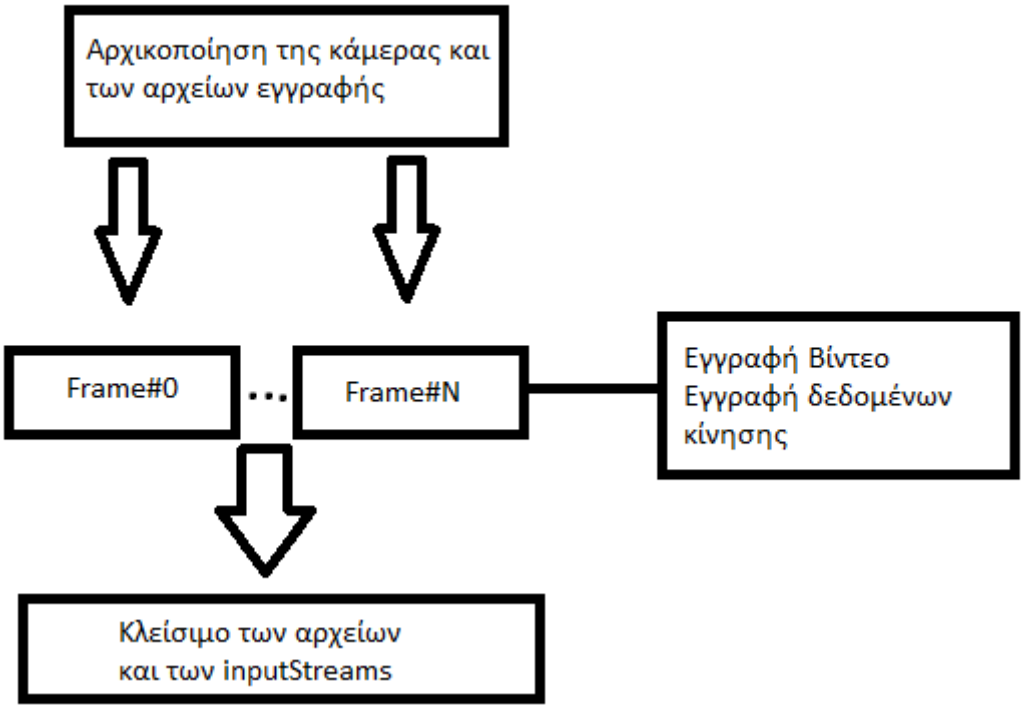
Ανάπτυξη εφαρμογής Android

3.1 Γενική περιγραφή και Layoutεφαρμογής

Κατά την κωδικοποίηση, η κίνηση της κάμερας με την οποία τραβήχτηκε το βίντεο θα μπορούσε να μας δώσει πληροφορίες που θα περιορίσουν σημαντικά το εύρος της αναζήτησης της TZSearch μειώνοντας έτσι το χρόνο.

Με τα όργανα που υπάρχουν στις κινητές συσκευές που ανιχνεύουν την κίνηση της συσκευής, πιο συγκεκριμένα το γυροσκόπιο και το επιταχυνσιόμετρο, καθώς και με το API του λειτουργικού συστήματος του android που μας δίνει πρόσβαση σε αυτά τα όργανα μπορεί αυτή η κίνηση να καταγραφεί.

Αναπτύχθηκε λοιπόν μια εφαρμογή η οποία τραβάει βίντεο και παράλληλα για κάθε frame του βίντεο αυτού καταγράφει και την κίνηση της συσκευής τη χρονική αυτή στιγμή.

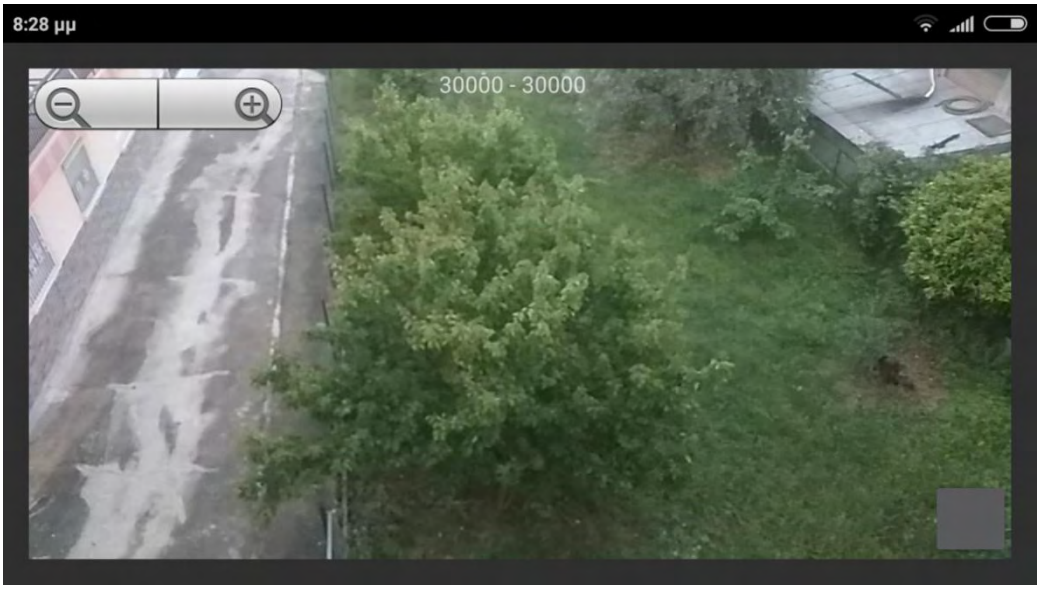


Σχήμα 3.1:Γενική αρχιτεκτονική της εφαρμογής

Το layout της εφαρμογής αποτελείται από ένα surfaceView το οποίο χρησιμοποιείται για την προβολή της εικόνας που καταγράφεται από την κάμερα, ένα button για την έναρξη/παύση της βιντεογράφησης, ένα πεδίο κειμένου το

οποίο περιγράφει την κατάσταση της εφαρμογής καθώς και χειριστές της τιμής για το zoom. Για τη στοίχιση των αντικειμένων αυτών στην οθόνη χρησιμοποιείται η επιλογή frameLayout η οποία εισάγει την έννοια του βάθους.

```
<FrameLayout
android:layout_width="match_parent"
android:layout_height="match_parent"
android:paddingBottom="@dimen/activity_vertical_margin"
android:paddingLeft="@dimen/activity_horizontal_margin"
android:paddingRight="@dimen/activity_horizontal_margin"
android:id="@+id/frame"
android:paddingTop="@dimen/activity_vertical_margin"
>

<Button
android:id="@+id/button"
android:layout_width="100px"
android:layout_height="100px"
android:layout_gravity="right|bottom"

/>

<SurfaceView
android:id="@+id/surfaceview"
android:layout_width="wrap_content"

android:layout_height="wrap_content" />

<TextView
android:id="@+id/textt"
android:layout_height="100px"
android:layout_width="200px"
android:text="not started yet"
android:layout_gravity="center_horizontal|top"

/>

</FrameLayout>
```



Σχήμα 3.2:Layout της εφαρμογής

Το surfaceView χειρίζεται από το object SurfaceHolder. Μέσω του SurfaceHolder πραγματοποιείται η προβολή της εικόνας που λαμβάνεται από την κάμερα καθώς και η πρόσβαση στο byte array της εικόνας. Αποκτούμε πρόσβαση στο SurfaceHolder του surfaceView ως εξής[20]:

```
preview=(SurfaceView)findViewById(R.id.surfaceview);
holder=preview.getHolder();
```

Ο previewholder αποκτά reference της κάμερας ως εξής:

```
camera.setPreviewDisplay(holder);
```

Για να αποκτηθεί πρόσβαση στο byte array της εικόνας θα πρέπει να οριστεί ένα callback όπου μια συνάρτηση θα αναλαμβάνει το χειρισμό του byte array(onPreviewFrame). Η συχνότητα με την οποία θα καλείται αυτή η συνάρτηση είναι αυτή της συχνότητας της κάμερας η οποία ορίζεται από το object Camera.Parameters. Για τη δημιουργία του dataset επιλέχθηκε η τιμή 30 fps (frames per second), καθώς ανάλογα με τις δυνατότητες της συσκευής μπορεί να μην ανταποκρίνεται σωστά η εφαρμογή σε υψηλότερες τιμές. Το callback ορίζεται ως εξής:

```
holder.addCallback(this);
```

Αρκεί ο ορισμός της συνάρτησης onPreviewFrame, μέσω της οποίας αποκτάται πρόσβαση στο byte array του frame, το οποίο θα εγγραφεί στο αρχείο. Από εκεί

πυροδοτείται το νήμα το οποίο αναλαμβάνει την εγγραφή του byte array. Εκτός από την πυροδότηση του νήματος η συνάρτηση αναλαμβάνει και την εγγραφή των δεδομένων που αφορούν την κίνηση της συσκευής. Επίσης εγγράφονται σε αρχείο τα διάφορα specs της κάμερας τα οποία θα χρησιμοποιηθούν αργότερα για τη χρησιμοποίηση της κίνησης στη διαδικασία της κωδικοποίησης. Παρακάτω αναφέρεται επιγραμματικά ο ορισμός και η λειτουργία της onPreviewFrame[21].

```
public void onPreviewFrame(byte[] data, final Camera
camera) {
if(timestamp<500 &&boom) {
if(timestamp==0){
                fText.setText("recording");
                Camera.Parameters p=camera.getParameters();
                horizontalViewAngle=Math.toDegrees(p.getHori
                zontalViewAngle());
                verticalViewAngle=Math.toDegrees(p.getVertic
                alViewAngle());
                double focalLength=p.getFocalLength();
                height=(int)(Math.tan(verticalViewAngle/2)*2
                *focalLength);
                width=(int)(Math.tan(horizontalViewAngle/2)*
                2*focalLength);

          }
          OneShotTask t=new OneShotTask(timestamp,data);
new Thread(t).start();

        αποθήκευση των δεδομένων κίνησης σε JSONObjects
        }
if(timestamp==500){

fText.setText("Wait for 10s then exit app");
          Runnable runnable=new Runnable() {
@Override
public void run() {

        εγγραφή των δεδομένων κίνησης στο δίσκο
        }
        };
new Thread(runnable).start();


    }
if(boom){
timestamp++;
    }
}
```

3.2CameraConfiguration

Το android API δίνει πρόσβαση στην κάμερα μέσω του object Camera. Ο περαιτέρω χειρισμός της πραγματοποιείται μέσω του object Camera.Parameters. Μέσω αυτού του object είναι δυνατή η τροποποίηση διάφορων πεδίων όπως οι διαστάσεις της εικόνας, η τιμή του zoom, το εστιακό βάθος κτλ. Η αναφορά σε αυτά τα objects πραγματοποιείται ως εξής:

```
camera=Camera.open();
Camera.Parameters p = camera.getParameters();
```

Με τη χρήση του object Camera.parameters, του μηχανισμού eventListener καθώς και το API του GUI εισάγεται μηχανισμός ρύθμισης του zoom

```
zoomControls.setOnZoomInClickListener(new
View.OnClickListener() {
@Override
public void onClick(View v) {
        Camera.Parameters p = camera.getParameters();
int maxZoom = p.getMaxZoom();
if (p.isZoomSupported()) {
int zoom = p.getZoom();
            zoom += 1;
if (zoom > maxZoom) {
                zoom -= 1;
            }
            p.setZoom(zoom);
        }
camera.setParameters(p);
    }
});
zoomControls.setOnZoomOutClickListener(new
View.OnClickListener() {
@Override
public void onClick(View v) {
        Camera.Parameters p = camera.getParameters();

if (p.isZoomSupported()) {
int zoom = p.getZoom();
zoom -= 1;
if (zoom<0) {
zoom += 1;
            }
            p.setZoom(zoom);
        }
camera.setParameters(p);
    }
});
```

Το βίντεο είναι 30fps. Επιλέγεται ένα fps range το οποίο να περιλαμβάνει τα 30fps ως εξής:

```
for(int
i=0;i<parameters.getSupportedPreviewFpsRange().size();i++
){
if(parameters.getSupportedPreviewFpsRange().get(i)[0]>=15
000 &&
parameters.getSupportedPreviewFpsRange().get(i)[1]>=30000
){
        index=i;

    }
}
```

όπου γίνεται iteration ανάμεσα στα supported Fps Ranges. Στη μεταβλητή index αποθηκεύεται το πρώτο index που θα πληρεί τις παραπάνω προϋποθέσεις.

Το range ρυθμίζεται παρακάτω:

```
parameters.setPreviewFpsRange(parameters.getSupportedPrev
iewFpsRange().get(index)[0],parameters.getSupportedPrevie
wFpsRange().get(index)[1]);
```

Παρομοίως καθορίζονται και οι διαστάσεις της εικόνας. Επιλέγονται οι διαστάσεις που καθορίζουν το μεγαλύτερο εμβαδόν ως εξής:

```
Camera.Size result=null;

for (Camera.Size size :
parameters.getSupportedPreviewSizes()) {


if(size!=null) {
if(result==null){
                result=size;
            }

int resultArea = result.width * result.height;
int newArea = size.width * size.height;

if (newArea > resultArea) {
                result = size;
            }
        }

}
```

3.3Καταγραφή τιμών

Η εφαρμογή τραβάει ένα βίντεο όπου τα frame του είναι σε YUV 420 format. Εκτυπώνει σε format τα εξής αρχεία που έχουν να κάνουν με την κίνηση της συσκευής σε μορφή json object. Ως attribute αυτό το object έχει τον αριθμό του frame για το οποίο περιγράφεται η κίνηση. Τα json objects είναι τα εξής:

-rot.txt. Δίνει το τρισδιάστατο διάνυσμα της περιστροφής στους άξονες x,y,z για το frame.Μετριέται σε rad/s

-a.txt. Δίνει το τρισδιάστατο διάνυσμα της γραμμικής επιτάχυνσης στους άξονες x,y,z για το frame χρησιμοποιώντας χαμηλοπερατό φίλτρο. Μετριέται σε m/s2

-agrav. Δίνει το τρισδιάστατο διάνυσμα της επιτάχυνσης λόγω της βαρύτητας στους άξονες x,y,z για το frame χρησιμοποιώντας χαμηλοπερατό φιλτρο. Μετριέται σε m/s2

-u.txt. Δίνει μια προσέγγιση του τρισδιάστατου διανύσματος της γραμμικής ταχύτητας ως εξής u[frame]=u[frame-1]+a[frame],υλοποιώντας δηλαδή μια πρόχειρη ολοκλήρωση. Μετριέται σε m/s.

-zoom.txt. Δίνει την ποσότητα zoom για κάθε frame.

-focal.txt. Δίνει την εστιακή ποσότητα του φακού για κάθε frame.Μετριέται σε mm.

-sensorsize.txt . Δίνει το μήκος και το πλάτος του φακού σε mm.

Η μορφή ενός json αρχείου είναι η παρακάτω:

{attribute1:value1,attribute2:value2,..,attributeN:valueN}

Η χρησιμοποίηση των αισθητήρων στο android API αναλαμβάνεται από το object sensorManager. Από ‘κει είναι δυνατή η απευθείας αναφορά στους αισθητήρες μέσω του object sensor ως εξής[22]:

```
sensorManager=(SensorManager)getSystemService(SENSOR_SERVICE);
sensor=sensorManager.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
otherSensor=sensorManager.getDefaultSensor(Sensor.TYPE_GYROSCOPE);
```

Με τη χρήση των listeners και των events γίνεται η καταγραφή και ο έλεγχος των τιμών:

```
sensorManager.registerListener(this,otherSensor,sensorManager.SENSOR_DELAY_GAME);
```

```
sensorManager.registerListener(this,sensor,sensorManager.
                    SENSOR_DELAY_GAME);
```

Όπου με τα Sensor.TYPE_GYROSCOPE και Sensor.TYPE_ACCELEROMETER δηλώνεται ο τύπος του αισθητήρα για τον οποίο γίνεται το reference και SENSOR_DELAY_GAME είναι η συχνότητα με την οποία τσεκάρεται η τιμή του αισθητήρα στα 50Hz.

Ο χειρισμός των τιμών που δίνουν οι αισθητήρες πραγματοποιείται από τη συνάρτηση onSensorChanged ως εξής:

```
public void onSensorChanged(SensorEvent event) {
//330<y 0
    // y<330 && y>-330 && x<0 90
    // y<330 && y>-330 && x>0 270
    //y<-330

if(event.sensor.getType()==Sensor.TYPE_ACCELEROMETER) {
float alpha = 0.9f;
if (timestamp == 0) {
gravity[0] = event.values[0];
gravity[1] = event.values[1];
gravity[2] = event.values[2];
gy=(int)gravity[1]*100;
gx=(int)gravity[0]*100;
if(gy>330){
orientation=0;
            }
else if(gy<330 &&gy>-330 &&gx<0){
orientation=1;
            }
else if(gy<330 &&gy>-330 &&gx>0){
orientation=3;
            }
else if(gy<-330){
orientation=2;
            }
        } else {
gravity[0] = alpha * gravity[0] + (1 - alpha) *
event.values[0];
gravity[1] = alpha * gravity[1] + (1 - alpha) *
event.values[1];
gravity[2] = alpha * gravity[2] + (1 - alpha) *
event.values[2];
        }

temp[0] = event.values[0] - gravity[0] >0.1f ||
```

```
event.values[0] - gravity[0] < -0.1f ? event.values[0] -
gravity[0] : 0;
temp[1] = event.values[1] - gravity[1] >0.1f ||
event.values[1] - gravity[1] < -0.1f ? event.values[1] -
gravity[1] : 0;
linAcc[2] = event.values[2] - gravity[2] >0.1f ||
event.values[2] - gravity[2] < -0.1f ? event.values[2] -
gravity[2] : 0;
linAcc[0]=(float)axisSwap[orientation][0]*temp[axisSwap[o
rientation][2]];
linAcc[1]=(float)axisSwap[orientation][1]*temp[axisSwap[o
rientation][3]];
speed[0] += linAcc[0];
speed[1] += linAcc[1];
speed[2] += linAcc[2];
    }
if(event.sensor.getType()==Sensor.TYPE_GYROSCOPE){
rotation[0]=event.values[0];
rotation[1]=event.values[1];
rotation[2]=event.values[2];
    }
}
```

Αν ο αισθητήρας για τον οποίο καλείται η συνάρτηση είναι αυτός του γυροσκοπίου τότε σε έναν πίνακα 3x1 εν ονόματι rotation αποθηκεύονται οι τιμές του γυροσκοπίου.

Αν ο αισθητήρας για τον οποίο καλείται η συνάρτηση είναι αυτός του επιταχυνσιόμετρου τότε σε πίνακα gravity αποθηκεύεται η επιτάχυνση λόγω της βαρύτητας και σε πίνακα linAcc αποθηκεύεται η επιτάχυνση του λόγω της γραμμικής κίνησης που πραγματοποιεί. Επίσης σε πίνακα speed αποθηκεύεται η γραμμική ταχύτητα με μία πρόχειρη προσέγγιση της ολοκλήρωσης της linAcc.

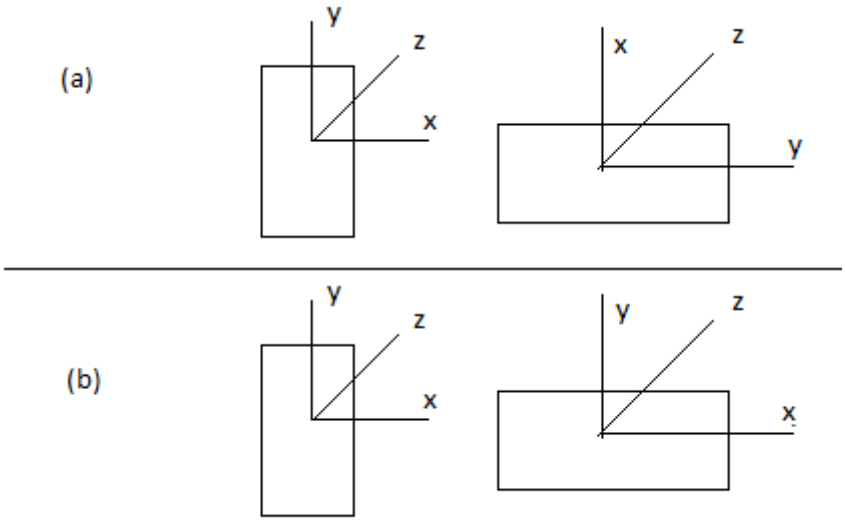
Το επιταχυνσιόμετρο καταγράφει και την επιτάχυνση της βαρύτητας και τη γραμμική επιτάχυνση μαζί. Ανακύπτει λοιπόν η ανάγκη για την ύπαρξη ενός χαμηλοπερατού φίλτρου το οποίο θα διαχωρίζει αυτές τις δύο τιμές. Θεωρώντας ότι το μεγαλύτερο μέρος της τιμής του επιταχυνσιόμετρου(περίπου το 90%,alpha=0.9) προέρχεται από τη επιτάχυνση της βαρύτητας προκύπτει:

$$gravity = alpha * gravity + (1 - alpha)\, value$$

$$linAcc = value - gravity \qquad 3.1$$

Επίσης λόγω στατικού θορύβου το linAcc πολλές φορές δίνει τιμές μεγαλύτερες του μηδενός όταν η συσκευή είναι ακίνητη. Για αυτό το λόγο όταν οι τιμές της linAcc που είναι μικρότερες από 0.1m/s^2 μηδενίζονται.

Ένα ακόμα πρόβλημα που ανακύπτει με τη linAcc είναι ότι οι άξονες του x,y,z είναι σταθεροί ως προς την ανατομία της συσκευής και όχι ως προς της συντεταγμένες του πραγματικού κόσμου. Αυτό αντιμετωπίζεται με τη χρήση της τιμής της επιτάχυνσης της βαρύτητας για το orientation της συσκευής έτσι ώστε να παραμένουν σταθεροί οι άξονες κατά την κίνηση της συσκευής.



Σχήμα 3.3: Ορισμός συστήματος συντεταγμένων.(a)Default,(b)Οι άξονες που ορίζει η εφαρμογή

Η εγγραφή αυτών των τιμών σε μορφή json γίνεται μέσω του object JSONObject. Η μορφή των object αυτών είναι η παρακάτω:

{Frame0:{ x:xvalue,y:value, z:zvalue },…,FrameN:{ x:xvalue, y:value, z:zvalue}}

3.4Outputs

Πρώτα γίνεται η δημιουργία των frame json objects με τη μορφή (x,y,z) τα οποία έπειται εισάγονται με attribute το δείκτη του frame στο main json object της τιμής που καταγράφεται. Η διαδικασία αυτή δίνεται παρακάτω:

```
JSONObjectobj     = newJSONObject();
JSONObjectobjj    = newJSONObject();
JSONObjectobjjj   = newJSONObject();
JSONObjectobjjjj  = newJSONObject();
obj.put("x",linAcc[0]);
obj.put("y",linAcc[1]);
obj.put("z",linAcc[2]);
obj1.put(""+timestamp,obj);
objj.put("x",speed[0]);
objj.put("y",speed[1]);
objj.put("z",speed[2]);
objjj.put("x",gravity[0]);
objjj.put("y",gravity[1]);
objjj.put("z",gravity[2]);
objjjj.put("x",rotation[0]);
objjjj.put("y",rotation[1]);
objjjj.put("z",rotation[2]);
obj2.put(""+timestamp,objj);
obj3.put(""+timestamp,objjj);
obj4.put(""+timestamp,objjjj);
obj5.put(""+timestamp,camera.getParameters().getZoom());
obj6.put(""+timestamp,camera.getParameters().getFocalLeng
th());
```

Απομένει η καταγραφή των object αυτών σε ξεχωριστά αρχεία. Ως directory επιλέγεται ένας φάκελος στο main directory που χρησιμοποιεί το λειτουργικό σύστημα για την εσωτερική αποθήκευση. Για τη μοναδική ονομασία τόσο των αρχείων που περιγράφων την κίνηση όσο και του αρχείου του βίντεο χρησιμοποιείται η ημερομηνία και η ώρα του με τον παρακάτω τρόπο:

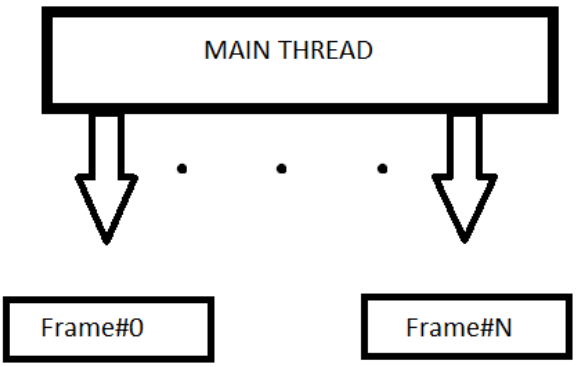
```
Date d=Calendar.getInstance().getTime();
name=d.getYear()+"_"+d.getMonth()+"_"+d.getDay()+"_"+d.ge
tHours()+"_"+d.getMinutes()+"_"+d.getSeconds();
```

Χρησιμοποιούνται οι I/O βιβλιοθήκες της java για την εγγραφή των αρχείων στο δίσκο ως εξής(Βλ Π8):

```
File f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+ "rawvideo/" +
name+"_"+"<όνομα_αρχείου>"+camera.getParameters().getPrev
iewSize().width+"_"+camera.getParameters().getPreviewSize
().height+".txt");
```

3.5 Εγγραφή βίντεο στο δίσκο

Η εγγραφή του byte array του κάθε frame αναλαμβάνεται από ένα ξεχωριστό νήμα έτσι ώστε να μην υπάρχουν καθυστερήσεις στο κύριο νήμα το οποίο αναλαμβάνει το GUI τμήμα της εφαρμογής



Σχήμα 3.4: Ανατίθεται ένα νήμα για κάθε ένα frame. Ο συντονισμός τους πραγματοποιείται από το main thread.

Το νήμα για το frame i θα πρέπει να βρει την ακριβή θέση στο αρχείο στην οποία θα πάει να γράψει το byte array του frame. Αφού το format του βίντεο είναι NV21 και για κάθε pixel θα υπάρχει ένα byte για το Y-plane και για κάθε 4 θα υπάρχει ένα byte για το U-plane και ένα byte για το V-plane αυτό σημαίνει ότι ο αριθμός των bit του κάθε frame θα είναι:

$$Bits = pixels + pixels/4 + pixels/4 \quad (3.2)$$

Επομένως η θέση στην οποία θα πάει το νήμα να γράψει θα είναι:

$$Θέση = i \cdot Bits \quad (3.3)$$

Για την απευθείας μετατόπιση του δείκτη εγγραφής η εφαρμογή χρησιμοποιεί το object RandomAccesFile της I/O βιβλιοθήκης της java.

```
File f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+ "rawvideo/" + name + ".yuv");
if(!f.exists()){
    f.createNewFile();
```

```
}
RandomAccessFile rfa=new RandomAccessFile(f,"rw");
rfa.seek(timestamp*data.length);
rfa.write(data);
rfa.close();
```

Το byte array που δίνει ο preview holder είναι σε format NV21. Πριν γράφει στον αποθηκευτικό χώρο κάποιο frame γίνεται μετατροπή των byte που δίνει ο previewholder από NV21 σε YUV 420 ως εξής:

```
        int k=0;
        int m=size;
        int n=size+size/4;
        for(int i=size;i<size+size/2;i++){
      if(k==0){
              k++;
              newdata[n]=data[i];
              n++;
         }
      else{
              k--;
              newdata[m]=data[i];
              m++;
         }
      }
```

Στην ουσία γίνεται μια μετατόπιση των Y,U,V στοιχείων των pixel ώστε να έρθουν σε μορφή YUV 420, έτσι να είναι αποδεκτά από τον κωδικοποιητή.

3.6 Σύνοψη

Με την ανάπτυξη της παραπάνω εφαρμογής αποκτούμε τις απαραίτητες πληροφορίες για την κίνηση του κινητού κατά τη βιντεοσκόπηση. Το επόμενο βήμα είναι η αξιοποίηση των πληροφοριών αυτών κατά τη διαδικασία της κωδικοποίησης.

ΚΕΦΑΛΑΙΟ 4

Μετατροπές στον κωδικοποιητή

4.1 Mapping τιμών γυροσκοπίου σε pixels

Το επόμενο βήμα είναι η μετατροπή των μετρήσεων του γυροσκοπίου στη μετατόπιση των pixels στον κωδικοποιητή. Λόγω της υπεροχής του γυροσκοπίου έναντι του επιταχυνσιόμετρου, θα περιγραφεί η κίνηση της συσκευής αποκλειστικά από τις μετρήσεις του γυροσκοπίου.

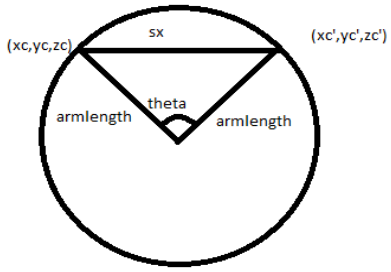
Έστω A(xc,yc,zc) σημείο στο σύστημα συντεταγμένων κάμερας τη χρονική στιγμή t1 και σημείο A'(xc',yc',zc') το σημείο A τη χρονική στιγμή t2. Λόγω της κίνησης της συσκευής έχουμε A'=(xc+sx,yc+sy,zc+sz).

Έστω ότι (rotx,roty,rotz) οι τιμές του γυροσκοπίου σε rad/s για το frame μεταξύ t1 και t2.

Θεωρούμε ότι η συσκευή πραγματοποιεί κυκλική κίνηση στον άξονα χ γύρω από το χειριστή της με ακτίνα την απόσταση μεταξύ τους, δηλαδή την προέκταση των χεριών του χεριστή (armlength).

Από το σχήμα προκύπτει:

$$sx = 2 \cdot armlength \cdot \sin(\left(\frac{rotx}{\Delta t}\right)/2) \qquad 4.1$$



4.1 Αναπαράσταση της κίνησης του χρήστη στον άξονα χ.

Παρομοίως θεωρώντας ότι η συσκευή στον άξονα y πραγματοποιεί περιστροφή γύρω από το κέντρο βάρους της με ακτίνα το πλάτος της (phone_width) και θεωρώντας ότι η συσκευή βρίσκεται σε landscape mode προκύπτει:

$$\mathrm{sy} = 2 \cdot \mathrm{phone\_height} \cdot \sin((\mathrm{roty}/\Delta t)/2) \quad (4.2)$$

Στο σύστημα συντεταγμένων του φακού ισχύουν οι σχέσεις

$$xs = f \cdot \frac{xc}{zc} \quad (4.3)$$

και

$$ys = f \cdot yc/zc$$

, όπου f η εστιακή απόσταση του φακού. Αν τώρα xccd, yccd οι διαστάσεις του φακού και rows,columns οι διαστάσεις της εικόνας τη χρονική στιγμή t1[23]:

$$i = ((columns - 1)/yccd) \cdot xs + (rows + 1)/2 \quad (4.4)$$

$$j = ((rows - 1)/xccd) \cdot xs + (columns + 1)/2, \quad (4.5)$$

Τη χρονική στιγμή t2:

$$i' = ((columns - 1)/yccd) \cdot x'sensor + (rows + 1)/2 \quad (4.6)$$

$$j' = ((rows - 1)/xccd) \cdot y'sensor + (columns + 1)/2, \quad (4.7)$$

Από τις (4.4),(4.6) προκύπτει:

$$\Delta i = i' - i = ((rows - 1)/yccd) \cdot x'sensor + (rows + 1)/2 - ((rows - 1)/yccd) * xsensor - (rows + 1)/2$$

$$= ((rows - 1)/yccd) \cdot (x'sensor - xsensor)$$

$$= ((rows - 1)/yccd) \cdot f \cdot (xc + sx - xc)/zc$$

$$\Delta i = ((rows - 1)/yccd) \cdot f \cdot 2 \cdot armlength \cdot (sin(rotx/\Delta t)/2)/zc, \quad (4.8)$$

Παρομοίως από τις (4.5),(4.7) προκύπτει:

$$\Delta j = ((columns - 1)/xccd) \cdot f \cdot phone\_width \cdot sin((roty/\Delta t)/2)/zc. \quad (4.9)$$

Εύκολα διαπιστώνουμε ότι Δi,Δj θα είναι τα αντίθετα των moving vectors του encoder.

Για να βρεθεί το κατάλληλο βάθος zc θα πρέπει να δοκιμαστούν διαφορετικές τιμές για αυτό.

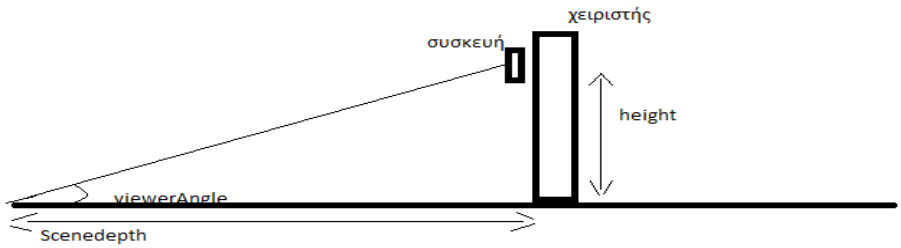
Το βάθος της σκηνής είναι:

$$SceneDepth = height \cdot tan(viewerAngle) \quad 4.10$$

,όπου height το ύψος του χειριστή και viewerAngle η γωνία της συσκευής η οποία δίνεται από τον τύπο

$$ViewerAngle = acos(gravz/10) \quad 4.11$$

, όπου gravz η επιτάχυνση βαρύτητας της συσκευής.



Σχήμα 4.2: Τι συμβαίνει κατά την καταγραφή ενός βίντεο

Έτσι το sceneDepth θα χρησιμοποιηθεί για την εύρεση του κατάλληλου scale Factor. Στα πειράματα έχει επιλεχτεί iteration μεταξύ του sceneDepth και του sceneDepth/5 με βήμα -sceneDepth/5.

Έτσι προκύπτουν πέντε υποψήφια σημεία μέσω αυτής της διαδικασίας. Παραδείγματος χάριν για τις τιμές rotx=-0.521682739257813 roty=0.0787353515625000 και gravz=3.0816040039062500 προκύπτουν τα ακόλουθα σημεία (di ,dj) για τα βάθη zc.

zc:5.093781 di:18 dj:18

zc:4.075025 di:23 dj:22

zc:3.056269 di:30 dj:30

zc:2.037512 di:46 dj:45

zc:1.018756 di:92 dj:90

4.2 Πέρασμα τιμών στον κωδικοποιητή

Στο επίπεδο TEncSlice, ανάλογα με τον αριθμό του frame, κάθε slice τιμοδοτείται με τις προβλεπόμενες τιμές[24].

```
int i = rpcSlice->getPOC();
   fstream file;
   string s1, s2;
   double rotx = -1;

   file.open("rotx.txt", fstream::in);
   file.seekg(i * 20, file.beg);
   getline(file, s1);
   rotx = ::atof(s1.c_str());
   file.flush();
   file.close();
   rpcSlice->rotx = rotx;
   double roty = -1;

   file.open("roty.txt", fstream::in);
   file.seekg(i * 20, file.beg);
   getline(file, s2);
   roty = ::atof(s2.c_str());
   file.flush();
   file.close();
   rpcSlice->roty = roty;
   double gravz = -1;
   file.open("gravz.txt", fstream::in);
   file.seekg(i * 20, file.beg);
   getline(file, s2);
   gravz = ::atof(s2.c_str());
   gravz = acos((gravz) / 10.0f);

   gravz = 1.65f*tan(gravz);
   if (gravz < 0){
      gravz = -gravz;
   }
   file.flush();
   file.close();
   float rows = 1280.0f, columns = 720.0f;

   float xccd = 3.0f, yccd = 6.0f;
```

```
float f = 3.789f;
float dt = 1 / (1.0f / 22.0f);
float armlength = 0.20f;
float cellphone_width = 0.0375f;
float si1, si2;
float di, dj;
si1 = sin((rotx * dt / 2));
di = armlength*((rows - 1) / xccd)*f * 2 * si1;

si2 = sin((roty * dt / 2));
dj = cellphone_width*((columns - 1) / yccd)*f * 2 * si2;
int mvi, mvj;
float step = gravz / 5.0f;
float dpth = gravz;
for (i = 0; i < 5; i++){
   mvi = di / dpth;
   mvj = dj / dpth;
   //cout << mvi << " " << mvj<<" ";
   rpcSlice->board[i][0] = -mvi;
   rpcSlice->board[i][1] = -mvj;
   dpth = dpth - step;
}
```

Έπειτα στο επίπεδο του TEncSearch, για το κάθε CU οι τιμές αυτές είναι προσβάσιμες με τις εντολές :

```
pcCU->getSlice()->rotx;
```

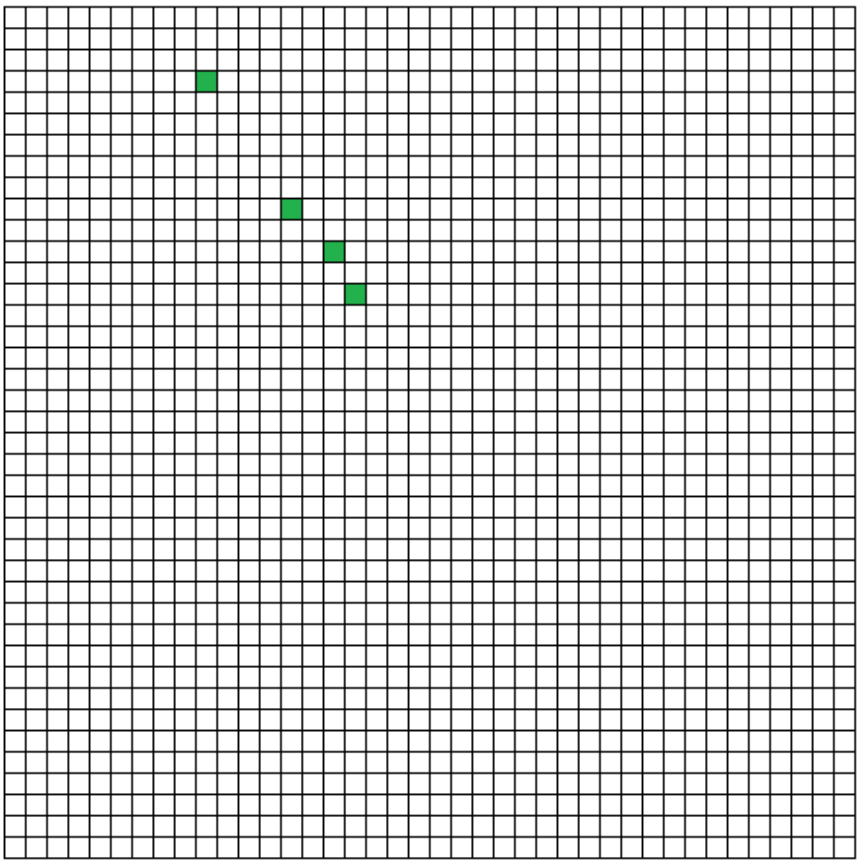
```
pcCU->getSlice()->roty;
```

```
pcCU->getSlice()->board;
```

4.3 Παραλλαγές της TZSearch

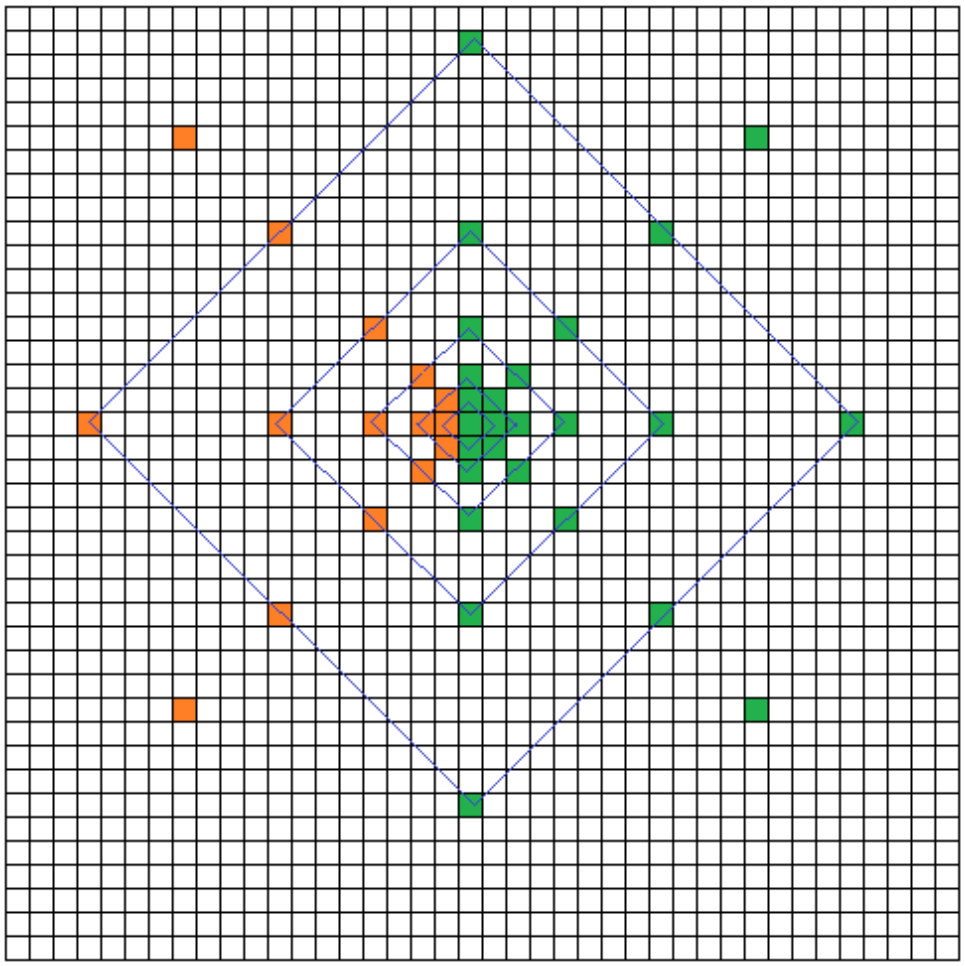
Από εκεί και πέρα οι μέθοδοι TZSearch, DiamondSearch και SquareSearch του κωδικοποιητή τροποποιούνται με τους παρακάτω τρόπους(κώδικας στο παράρτημα).

1)Ψάχνοντας μόνο στα 5 σημεία τα οποία προβλέπονται από τη mapping διαδικασία που αναλύθηκε παραπάνω(Μέθοδος 5Point)(βλ Π4).



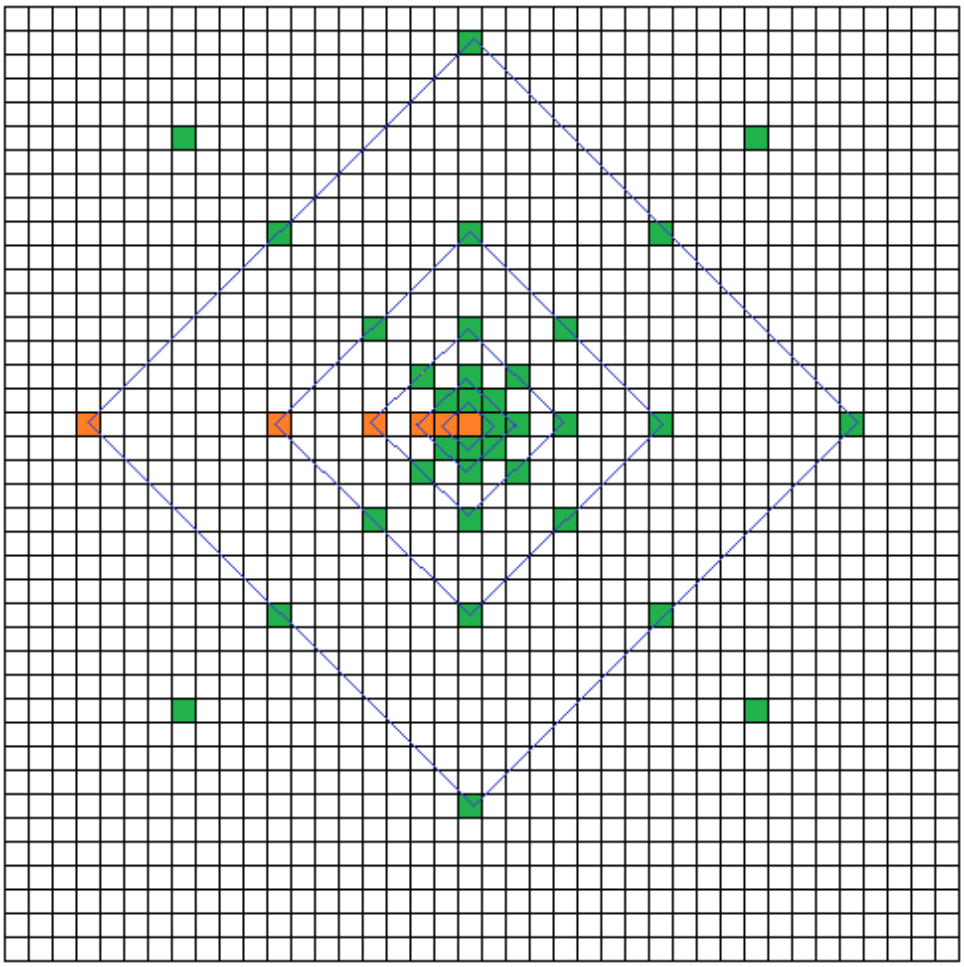
Σχήμα 4.2:Αναπαράσταση tzSearch μεθόδου 1,όπου πράσινο χρωματίζονται τα σημεία που εξετάζονται

2)Αντί για κανονική diamond Search, εξετάζοουμε μόνο τα σημεία V(vx,vy) της diamond search για τα οποία ισχύει rx-5<vx<rx+5, όπου R(rx,ry) το πρώτο σημείο που προβλέπεται από τη διαδικασία mapping(Μέθοδος Slider)(Βλ Π3).



Σχήμα 4.2:Αναπαράσταση tzSearch μεθόδου 2, με πράσινο χρωματίζονται τα σημεία της diamond Search και πορτοκαλί χρωματίζονται τα σημεία που εξετάζονται

3)Αντί για κανονική diamond, εξετάζονται μόνο τα σημεία που βρίσκονται στο οκταμόριο στο οποίο βρίσκεται και το διανύσμα (rotx, roty) (Μέθοδος Angle)(Βλ Π2).



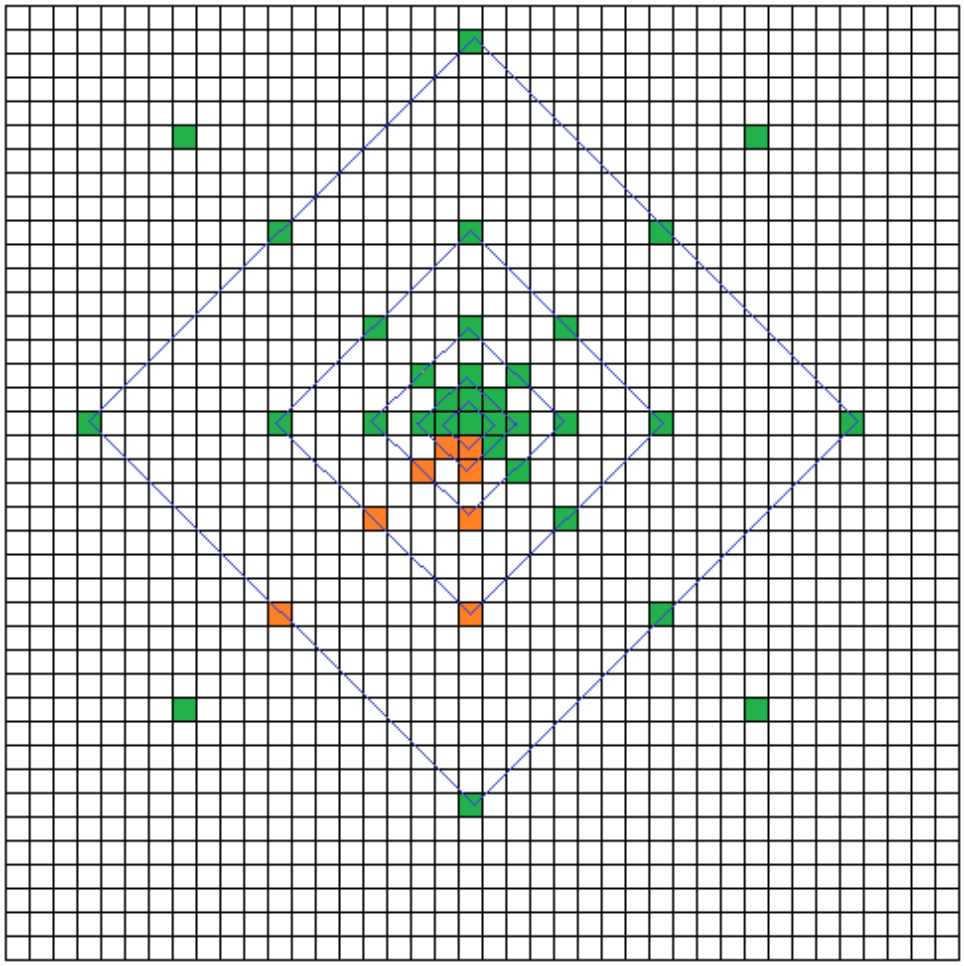
Σχήμα 4.4:Αναπαράσταση tzSearch μεθόδου 3, με πράσινο χρωματίζονται τα σημεία της diamond search και πορτοκαλί χρωματίζονται τα σημεία που εξετάζονται

Τα σημεία αυτά βρίσκονται ως εξής:

```
double rads = atan2(-roty, -rotx);
double theta = rads*(M_PI / 180.0);
int point = 0;

if (theta >= 0 && theta < 30){
   point = 5;
}
elseif (theta >= 30 && theta < 60){
   point = 3;
}
elseif (theta >= 60 && theta < 90){
   point = 2;
}
elseif (theta >= 90 && theta < 120){
   point = 2;
```

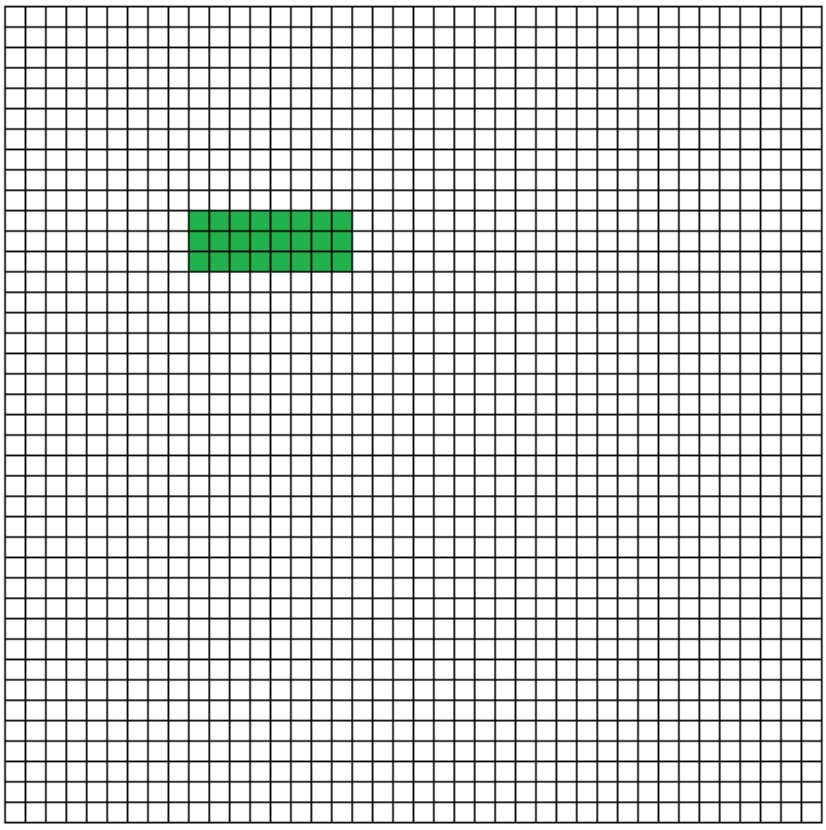
```
}
elseif (theta >= 120 && theta < 150){
   point = 1;
}
elseif (theta >= 150 && theta < 180){
   point = 4;
}
elseif (theta < 0 && theta >= -30){
   point = 5;
}
elseif (theta < -30 && theta >= -60){
   point = 8;
}
elseif (theta < -60 && theta >= -90){
   point = 7;
}
elseif (theta < -90 && theta >= -120){
   point = 7;
}
elseif (theta < -120 && theta >= -150){
   point = 6;
}
elseif (theta < -150 && theta >= -180){
   point = 4;
}
rpcSlice->point = point;
```

4)Αντί για κανονική diamond Search, εξετάζοντας μόνο τα σημεία V(vx,vy) της diamond search για τα οποία ισχύει rx-5<vx<rx+5 και ry-5<vy<ry+5 όπου R(rx,ry) το πρώτο σημείο που προβλέπεται από τη διαδικασία mapping(Μέθοδος Rect)(Βλ Π1).



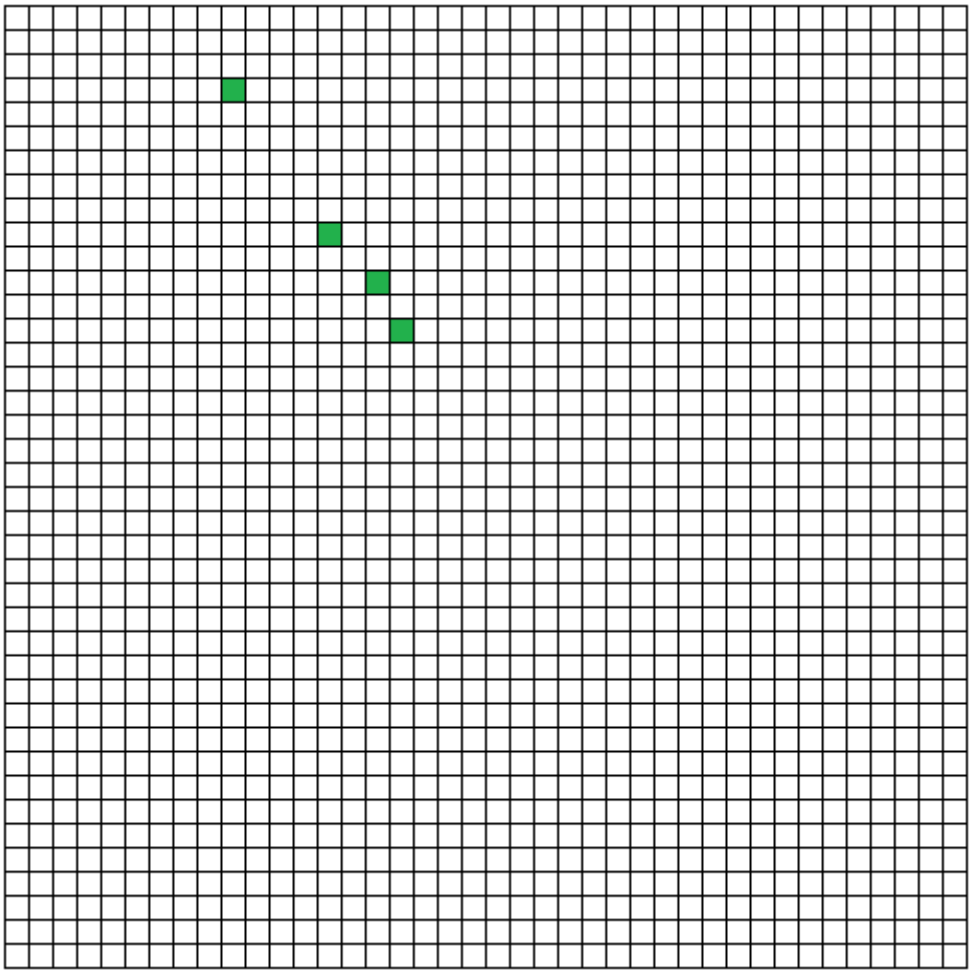
Σχήμα 4.5:Αναπαράσταση tzSearch μεθόδου 4, με πράσινο χρωματίζονται τα σημεία της diamond search και πορτοκαλί χρωματίζονται τα σημεία που εξετάζονται

5)Όπως στην 4, αλλά ψάχνοντας στο full pattern Search μόνο τα σημεία V(vx,vy) για τα οποία ισχύει rx-5<vx<rx+5 και ry-5<vy<ry+5 όπου R(rx,ry) το πρώτο σημείο που προβλέπεται από τη διαδικασία mapping (Μέθοδος fullRect)(Βλ Π5).



Σχήμα 4.6:Αναπαράσταση της fullsearch μεθόδου 5, με πράσινο χρωματίζονται τα σημεία που εξετάζονται

6)Όπως στην, αλλά ψάχνοντας στο full pattern search μόνο τα 5 σημεία που προβλέπονται από τη διαδικασία mapping(Μέθοδοςfull5)(Βλ Π6).



Σχήμα 4.7:Αναπαράσταση fullsearch μεθόδου 6, με πράσινο χρωματίζονται τα σημεία που εξετάζονται

4.4 Σύνοψη

Με αυτόν τον τρόπο αξιοποιούνται οι τιμές που δίνονται από την android εφαρμογή στον κωδικοποιητή. Είναι προφανές ότι ο σκοπός των μεθόδων που αναλύθηκαν παραπάνω είναι η εξέταση μόνο των πιο πιθανών moving vectors βάσει της κίνησης της συσκευής για τη μείωση τη χρονικής πολυπλοκότητας. Απομένει η παραγωγή ενός dataset καθώς και η πειραματική αξιολόγηση των παραπάνω μεθόδων.

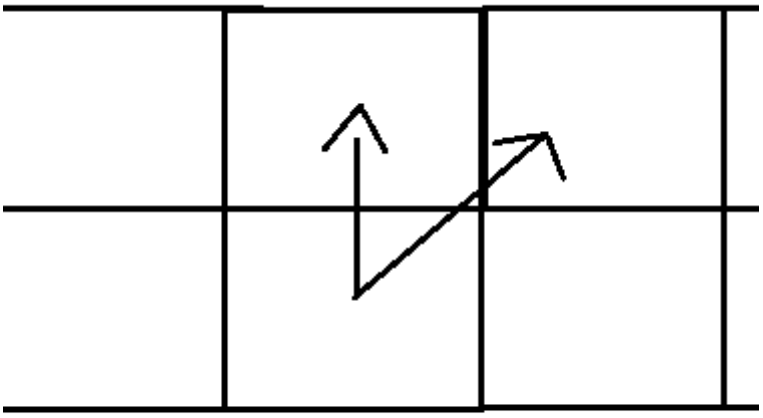
ΚΕΦΑΛΑΙΟ 5

WPPυλοποίηση του κωδικοποιητή

5.1Wavefront

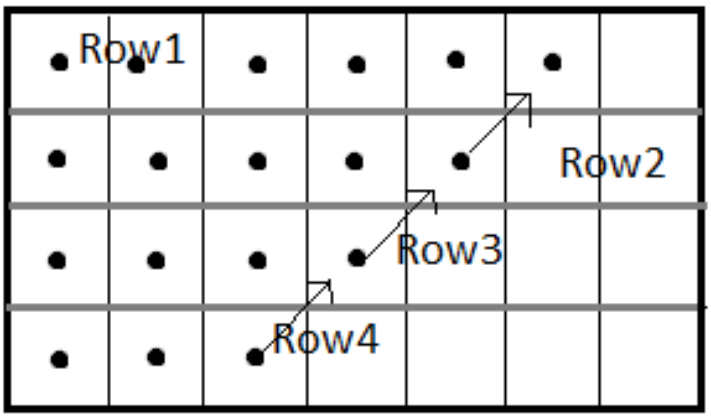
Εκτός της σειριακής υλοποίησης, θα πρέπει να εξεταστεί πως αντιδρά ένα παράλληλο σύστημα στις τροποποιήσεις της διαδικασίας της αναζήτησης των moving vectors. Επιλέχθηκε υλοποίηση με wavefronts καθώς η μείωση της ποιότητας σε σχέση με τη σειριακή υλοποίηση είναι μικρότερη σε σχέσεις με τις άλλες παράλληλες υλοποιήσεις. Το HM παρόλα αυτά είναι ένας σειριακός κωδικοποιητής/αποκωδικοποιητής. Για αυτό το λόγο, το HM software πρέπει να τροποποιηθεί για την υποστήριξη παραλληλισμού.

Στο wavefront parallel processing η εικόνα χωρίζεται σε γραμμές CTUs. Στην περίπτωση της κωδικοποίησης ενός CTU πρέπει πρώτα να κωδικοποιηθεί το πάνω, το πάνω δεξιά καθώς και το προηγούμενο CTU. Είναι απαραίτητη η κωδικοποίηση του πάνω και του πάνω δεξιά CTU πρώτα, έτσι ώστε να γίνει το intra coding.



Σχήμα 5.1:Θέματα συγχρονισμού που ανακύπτουν για το wavefront parallel processing

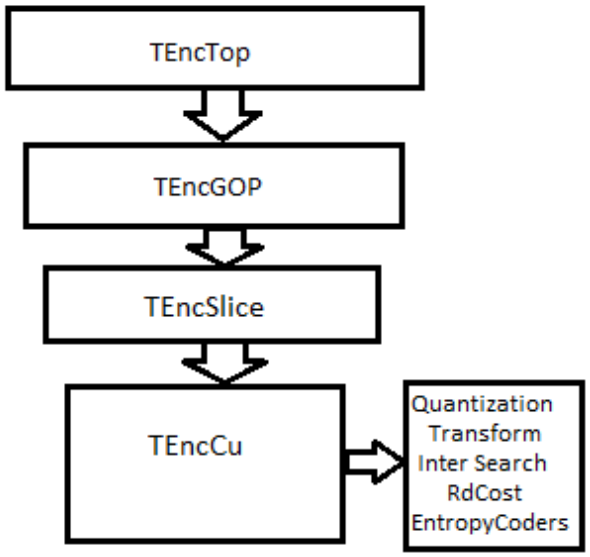
Αν και αυτή η συνθήκη απαιτεί συγχρονισμό και εισάγει overhead,η μείωση της ποιότητας σε σχέση με τη σειριακή υλοποίηση είναι αμελητέα.



Σχήμα 5.2: CTU rows

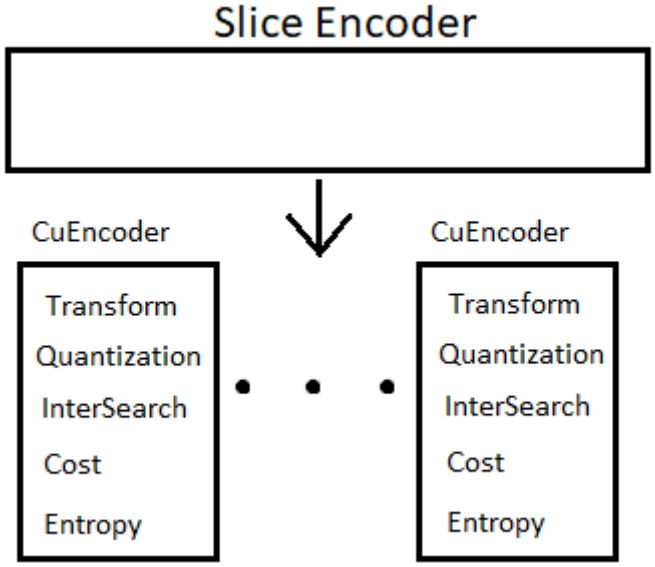
5.2 Παραλληλοποίηση του HM κωδικοποιητή

Η αρχικοποίηση της εργασίας και ο συντονισμός της στο υψηλότερο επίπεδο πραγματοποιείται από την κλάση TEncTop. Σε ένα επίπεδο πιο κάτω, η κλάση TEncGOP(Group of Pictures) αναλαμβάνει το συντονισμό της κωδικοποίησης των slices. Η κλάση TEncSlice αναλαμβάνει το συντονισμό της κωδικοποίησης κάθε CU και αναθέτει στον CU encoder το κάθε CTU το οποίο θα κωδικοποιήσει. Από εκεί και πέρα ο CU encoder με εργαλεία για transform, quantization, inter search, cost calculation και κωδικοποιητές εντροπίας κωδικοποιεί το κάθε CU.



Σχήμα 5.3: Ιεραρχία των κλάσεων του κωδικοποιητή

Η σειριακή υλοποίηση του HM χρησιμοποιεί έναν CU encoder. Στην παράλληλη υλοποίηση χρησιμοποιείται ένας CU encoder ανά γραμμή CU. Τα μέλη της κλάσης CU encoder δημιουργούνται εκ νέου για κάθε instance της κλάσης, ανάμεσα σε αυτά ο entropy coder, τα εργαλεία για transform και quantization,τα εργαλεία για τον υπολογισμό του κόστους και της αναζήτησης της κίνησης, καθώς SBAC και CABAC coders. Με αυτόν τον τρόπο περιορίζεται ο συγχρονισμός δραστικά.



Σχήμα 5.4:Αρχιτεκτονική της παράλληλης υλοποίησης του κωδικοποιητή

Στη σειριακή υλοποίηση ο μοναδικός CU encoder αναλαμβάνει την κωδικοποίηση κάθε CU και αφού τελειώσει ένα CU προχωράει στο αμέσως επόμενο στην εικόνα. Στην παράλληλη υλοποίηση κάθε CU encoder αναλαμβάνει μια γραμμή και αφού τελειώσει την κωδικοποίηση ενός CU προχωράει στο επόμενο στη γραμμή.

Ο CABAC coder για την κωδικοποίηση ενός CU χρειάζεται το context του πάνω δεξιά CU. Για αυτό το λόγο χρησιμοποιούμε έναν δισδιάστατο πίνακα από context states contexts[columns][rows], όπου στοιχείο contexts[i+1][j-1] το context state του πάνω δεξιά CU, για cu στη θέση (i,j). Έτσι συγχρονισμός απαιτείται κατά την εγγραφή των bits του CU στο αρχείο και κατά την τροποποίηση των μεταβλητών της κλάσης TencSlice, αφού είναι κοινές για όλους τους CU coders. Χρησιμοποιώντας ένα μη ολοκληρωμένο context εδώ είναι προφανές ότι θα υπάρχουν κάποιες επιπτώσεις στην ποιότητα.

Έτσι για n νήματα, ένα νήμα αφού τελειώσει την κωδικοποίηση της jης γραμμής θα αναλάβει την κωδικοποίηση της γραμμής j+n. Για να κωδικοποιηθεί το cu στη θέση

του (i,j) πρέπει να έχει κωδικοποιηθεί το CU στη θέση (i+1,j-1). Για αυτό το σκοπό χρησιμοποιείται ένας πίνακας p για τη σηματοδότηση της ολοκλήρωσης του στοιχείου (i+1,j-1.0. Αν p(i+1,j-1) είναι true τότε προχωράμε στην κωδικοποίηση αλλιώς περιμένουμε την ολοκλήρωση του (i+1,j-1). Όταν τελειώσει η κωδικοποίηση του (i,j) τότε το p(i,j) σηματοδοτείται ως true.

Για το μηχανισμό της αναμονής και της εκκίνησης ενός νήματος αρκεί μια βασική υλοποίηση του μηχανισμού των σημαφόρων. Ένας σημαφόρος χρησιμοποίει μια αριθμητική μεταβλητή. Κάθε φορά που αυτή η μεταβλητή μηδενίζεται τότε το νήμα περιμένει μέχρι αυτή η μεταβλητή να ξανατιμοδοτηθεί με μια τιμή μεγαλύτερη του μηδενός.

Στο επίπεδο TEncTop θα πρέπει να οριστούν αντικείμενα για την κωδικοποίηση του κάθε row ξεχωριστά(βλ Π10). Απομένει το πέρασμα αυτών των τιμών στο επίπεδο TEncSlice. Σημειώνεται πως εκτός αυτών των αντικειμένων χρειάζονται και τα αρχικά αντικείμενα που χρησιμοποιούνται στο επίπεδο όλου του frame για το συντονισμό της διαδικασίας της κωδικοποίησης[25].

```
VoidTEncSlice::init(TEncTop* pcEncTop)
{
  m_pcCfg = pcEncTop;
  m_pcListPic = pcEncTop->getListPic();

  m_pcGOPEncoder = pcEncTop->getGOPEncoder();
  m_pcCuEncoder = pcEncTop->getCuEncoder();
  m_pcPredSearch = pcEncTop->getPredSearch();
  cuCoders = pcEncTop->getCuCoders();
  m_pcEntropyCoder = pcEncTop->getEntropyCoder();
  entropyCoders = pcEncTop->entropyCoders;
  m_pcSbacCoder = pcEncTop->getSbacCoder();
  sbacs = pcEncTop->getSbacs();
  goOnSbacs = pcEncTop->getgoOnSbacs();
  m_pcBinCABAC = pcEncTop->getBinCABAC();
  m_pcTrQuant = pcEncTop->getTrQuant();
  cabacs =(TEncBinCABAC**) pcEncTop->goOnCabacs;
  m_pcRdCost = pcEncTop->getRdCost();
  rdCosts = pcEncTop->rdCosts;
  trQuants = pcEncTop->trQuants;
  m_pppcRDSbacCoder = pcEncTop->getRDSbacCoder();
  m_pcRDGoOnSbacCoder = pcEncTop->getRDGoOnSbacCoder();
  contexts = (TEncSbac**)malloc(12 * sizeof(TEncSbac*));
  for (int i = 0; i < 12; i++){
    contexts[i] = newTEncSbac;
  }
  // create lambda and QP arrays
```

```
  m_vdRdPicLambda.resize(m_pcCfg->getDeltaQpRD() * 2 + 1);
  m_vdRdPicQp.resize(m_pcCfg->getDeltaQpRD() * 2 + 1);
  m_viRdPicQp.resize(m_pcCfg->getDeltaQpRD() * 2 + 1);
  m_pcRateCtrl = pcEncTop->getRateCtrl();
  rateControls = pcEncTop->rateControls;
}
```

Πιο συγκεκριμένα στη σειριακή υλοποίηση στο επίπεδο TEncSlice, πραγματοποιείται ένα iteration ανάμεσα στα CTUs, μέσα στο οποίο το καθένα από αυτά κωδικοποιείται. Για να ελεγχθεί τη σειρά με τα οποία κωδικοποιούνται χρησιμοποιείται μια συνάρτηση η οποία κωδικοποιεί το καθένα CTU ξεχωριστά.

Ένα thread pool μεγέθους NUM_WPP_THREADS χρησιμοποιείται για την κωδικοποίηση κάθε γραμμής. Μόλις τελειώσει η κωδικοποίηση της I γραμμής το νήμα μεταπηδά στην κωδικοποίηση της i+NUM_WPP_THREADS γραμμής.

```
     std::thread ts[NUM_WPP_THREADS];
  for (int i = 0; i <NUM_WPP_THREADS; i++){
    ts[i]=thread(compressRow,i, frameWidthInCtus, 12, pcPic,
pcSlice, boundingCtuTsAddr, startCtuTsAddr,
bCompressEntireSlice, this, pRDSbacCoder, bitcounters);
  }
  for (int i = 0; i <NUM_WPP_THREADS; i++){
    ts[i].join();
  }
```

Η συνάρτηση compressRow αναλαμβάνει τόσο την κωδικοποίηση κάθε γραμμής όσο και το συντονισμό των νημάτων ώστε να ικανοποιηθούν οι συνθήκες συγχρονισμού που είναι απαραίτητες για τα wavefronts. Λόγω του ότι δεν είναι δυνατή η ανάθεση συνάρτησης μιας κλάσης σε νήμα, η συνάρτηση compressCu ορίζεται εκτός της κλάσσης και το αντικείμενο TEncSlice περνιέται μέσω του δείκτη this.

```
void compressRow(UIntj, UIntframeWidthInCtus, UInt
frameHeightInCtus, TComPic* pcPic, TComSlice* pcSlice, UInt
boundingCtuTsAddr, UInt startCtuTsAddr, Bool
bCompressEntireSlice, TEncSlice* sl, TEncBinCABAC**
pRDSbacCoder,TComBitCounter* bitcounters){
  thread t;
  bool k = false;
  //Sleep(1000);
  for (UInt i = 0; i <frameWidthInCtus; i++){
    if(j>0 && i<frameWidthInCtus-1){
      while (!sl->sems[i + 1][j - 1]){
        Sleep(100);
      }
```

```
    }
    sl->compressCu(i+j*20, frameWidthInCtus, startCtuTsAddr,
boundingCtuTsAddr, pcPic, pcSlice, pRDSbacCoder, bitcounters,
bCompressEntireSlice);
    sl->sems[i][j] = true;
  }
  if (j + NUM_WPP_THREADS<frameHeightInCtus){

    compressRow(j+NUM_WPP_THREADS, frameWidthInCtus, 12,
pcPic, pcSlice, boundingCtuTsAddr, startCtuTsAddr,
bCompressEntireSlice, sl, pRDSbacCoder, bitcounters);
  }

}
```

Το αντικείμενο sems του TEncSlice, το οποίο είναι ένας δισδιάστατος Boolean πίνακας χρησιμοποιείται ως μηχανισμός σημαφόρου. Μόλις το CTU στη θέση (i,j) κωδικοποιηθεί το sems[i][j] τιμοδοτείται με την τιμή true.

Αντίστοιχα το CTU στη θέση (i,j) θα περιμένει μέχρι να τελειώσει το encoding του CTUστη θέση (i+1,j-1) μέχρι να κωδικοποιηθεί.

Η συνάρτηση compressCu αναλαμβάνει την κωδικοποίηση του κάθε CU ξεχωριστά. Η συνάρτηση ορίστηκε στην κλάση TEncSlice λόγω της ανάγκης για την αναφορά στο αντικείμενο TEncSlice. Αποτελεί τροποποίηση της διαδικασίας της κωδικοποίησης ενός CTU όπως και στη σειριακή υλοποίηση με τη διαφορά ότι η compressRow χρησιμοποίει τα αντικείμενα που αφορούν την κωδικοποίηση της κάθε γραμμής κι όχι του κάθε frame. Επίσης χρησιμοποιούνται mutex κλειδιά για τη χρήση των μεταβλητών που είναι κοινές για όλες τις γραμμές CTU.

Η compressCu πρώτα ελέγχει αν είναι διαθέσιμο το πάνω δεξιά CU. Αν είναι τότε θα χρησιμοποιηθούν wavefronts και θα ανακτηθεί το context του ώστε να χρησιμοποιηθεί από τα εργαλεία του παρόντος CTU κατά την κωδικοποίηση. Σειρά έχει η αρχικοποίηση των εργαλείων του CTU. Ακολουθεί η πρόβλεψη και στη συνέχεια η κωδικοποίηση του CTU. Τέλος, ενημερώνεται το context της τρέχουσας γραμμής και αποθηκεύονται ο αριθμός των bit, το κόστος και ο βαθμός της παραμόρφωσης στη δομή του TEncSlice[26](βλ Π9). Ακολουθεί μια σύντομη αναπαράσταση των παραπάνω σε ψευδοκώδικα:

```
If(pcCU->isAboveAvailable()){

enableWavefronts;

}
```

sbacCoder->loadContext(contexts[row])

σετάρισμα των κωδικοποιητών εντροπίας, και του κωδικοποιητή CTU

cuEncoders[row]->compressCu(pcCU)

mutex1.lock()

cuEncoders[row]->encodeCu(pcCU)

mutex1.unlock()

contexts[row]->loadContexts(sbacCoder);

mutex2.lock();

πρόσθεση κόστους κωδικοποίησης CU στο συνολικό κόστος του Slice;

mutex2.unlock();

5.3 Σύνοψη

Με αυτόν τον τρόπο επιτυγχάνεται η παραλληλοποίηση του κωδικοποιητή με τη δομή των wavefronts. Θα μπορούσε να υλοποιηθεί παράλληλο σύστημα με τη χρήση ενός αντικειμένου κωδικοποιητή. Η επιλογή της δημιουργίας διαφορετικών κωδικοποιητών για κάθε σειρά CTU, αν και έχει ως αποτέλεσμα τη χρήση περισσότερης μνήμης, μειώνει δραστικά το χρόνο καθώς η προστασία κοινής μνήμης ανάμεσα στα νήματα θα είναι ελάχιστη. Εφόσον πρωταρχικής σημασίας είναι η χρονική απόδοση επιλέχθηκε η χρήση ενός κωδικοποιητή για κάθε νήμα. Μαζί με τις μετατροπές της TZSearch απομένει η πειραματική αξιολόγηση του συστήματος.

Κεφάλαιο 6

Πειραματική Αξιολόγηση

6.1 Περιγραφή του μηχανήματος και των ρυθμίσεων του κωδικοποιητή

Απομένει η πειραματική αξιολόγηση τόσο των τροποποιήσεων στην TZSearch όσο και της παράλληλης υλοποίησης με wavefront parallel processing. Το σύστημα πάνω στο οποίο έτρεξαν τα πειράματα είναι 64-bit Windows 7 Ultimate, με μνήμη RAM 3.99GB και επεξεργαστή Intel Core 2 Quad CPU @ 2.40,2.41 GHz. Θα αξιολογηθούν τόσο σε χρόνο ολοκλήρωσης και σε bitrate όσο και σε ποιότητα του βίντεο που παράγεται με τον δείκτη PSNR(Peak Signal to Noise Ratio).

Ο κωδικοποιητής στα πειράματα έτρεξε με τις ρυθμίσεις που περιγράφονται στο φάκελο encoder_lowdelay_main.cfg(βλ Π7). Ενδεικτικά, για μέγιστες διαστάσεις του CU έχουν επιλεχθεί 64x64, ενώ για ελάχιστες 8x8. Επίσης το QP της κωδικοποίησης έχει επιλεχθεί να είναι 32. Το εκτελέσιμο τρέχει στην κονσόλα ως εξής:

TappEncoder – i <nameOfFile>.yuv –c encoder_lowdelay_main.cfg –fr <frame rate of video> -f <frames to be encoded> -wdt <width of video> -hgt <height of video>

6.2Αξιολόγηση Wavefronts

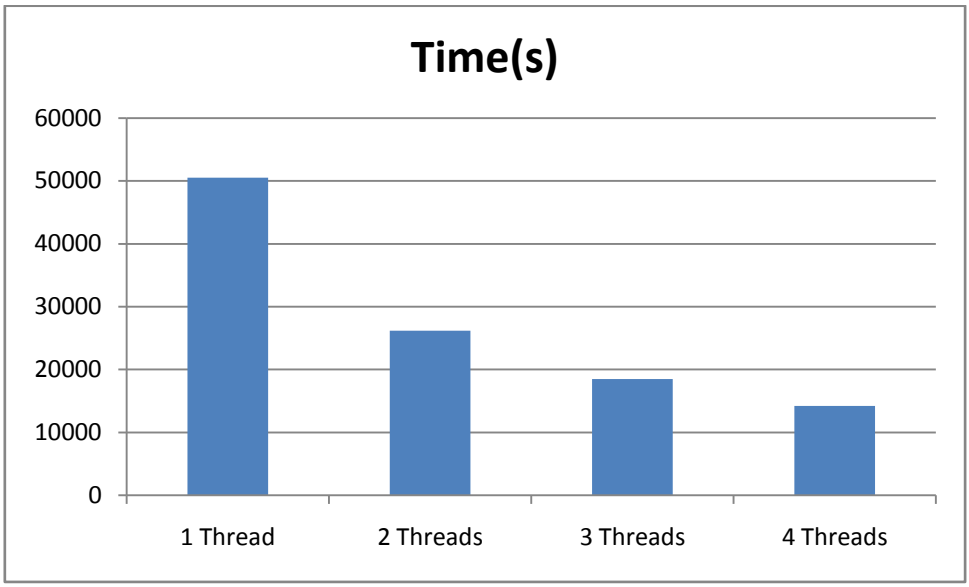
Για την πειραματική αξιολόγηση της παράλληλης υλοποίησης με wavefronts κωδικοποιήθηκαν δύο βίντεο από τη βιβλιοθήκη με τα test sequences του HEVC. Πρόκειται για stock βίντεο τα οποία χρησιμοποιούνται για αξιολογήσεις στη βιβλιογραφία. Το πρώτο είναι το Kimono 1080p 24fps 240 frames, όπου βιντεοσκοπείται μια γυναίκα με κιμονό, και το δεύτερο είναι τα πρώτα 240 frames του FourPeople(720p 24fps 500 frames), όπου βιντεοσκοπούνται τέσσερα άτομα σε ένα γραφείο ανταλλάσοντας ένα φυλλάδιο. Τα βίντεο αυτά κωδικοποιήθηκαν για ένα, δύο, τρία και τέσσερα threads.

Kimono

Σχήμα 6.1: Screenshot του video Kimono

Threads	PSNR(Y)	PSNR(U)	PSNR(V)	PSNR(YUV)	Time(s)	Bitrate(kbps)	Bytes στο δίσκο
1	39.8709	44.4745	46.0414	40.8685	50507.931	971.5803	1244890
2	39.8772	44.4449	46.1302	40.8764	26201.844	972.2583	1245760
3	39.8791	44.4691	46.1000	40.8787	18492.774	972.3402	1245842
4	39.8765	44.4839	46.1201	40.8776	14219.879	972.7700	1246408

Όπως παρατηρούμε παραπάνω, η μείωση του χρόνου όσο αυξάνονται τα threads είναι σημαντική. Φαίνεται να δημιουργείται και το φαινόμενο bottleneck, δηλαδή όσο αυξάνουν τα threads, ο ρυθμός μείωσης του χρόνου επιβραδύνεται, αφού περισσότερα threads συνεπάγονται περισσότερο χρόνο επικοινωνίας μεταξύ τους.



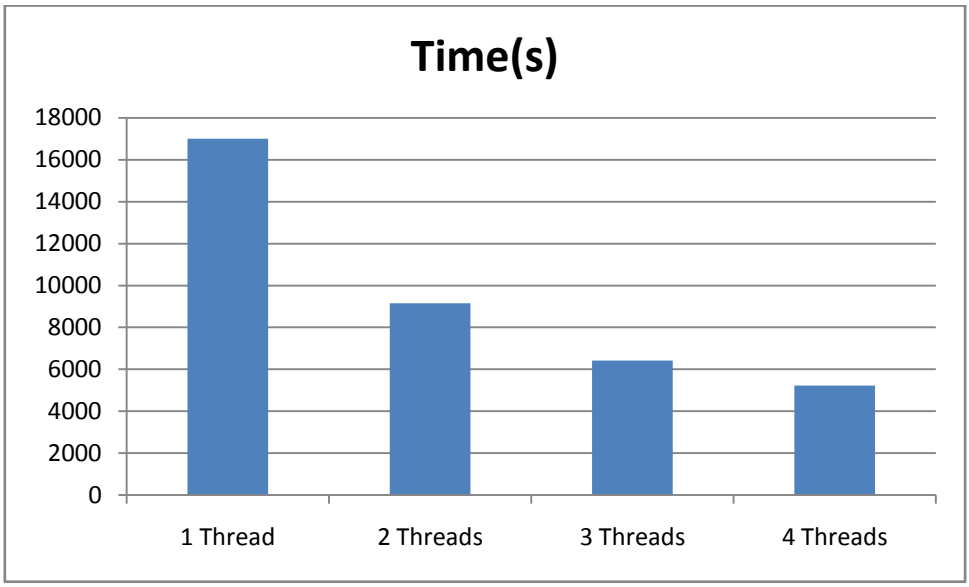
Σχήμα 6.2: Αποτελέσματα για Kimono

FourPeople

Σχήμα 6.3: Screenshot του FourPeople

Threads	PSNR(Y)	PSNR(U)	PSNR(V)	PSNR(YUV)	Time(s)	Bitrate(kbps)	Bytes στο δίσκο
1	39.6180	45.0848	45.7961	40.6766	17007.444	414.4589	231988
2	39.6417	44.9571	45.7414	40.6848	9147.578	414.6335	232554
3	39.6279	44.9933	45.7646	40.6764	6420.691	415.1518	232299
4	39.6288	44.9687	45.6394	40.6679	5227.470	414.5575	231885

Όπως παρατηρούμε παραπάνω, η μείωση του χρόνου όσο αυξάνονται τα threads είναι σημαντική. Φαίνεται να δημιουργείται και το φαινόμενο bottleneck, δηλαδή όσο αυξάνουν τα threads, ο ρυθμός μείωσης του χρόνου επιβραδύνεται, αφού περισσότερα threads συνεπάγονται περισσότερο χρόνο επικοινωνίας μεταξύ τους.



Σχήμα 6.4: Αποτελέσματα για το FourPeople

6.3Αξιολόγηση τροποποιήσεων TZSearch

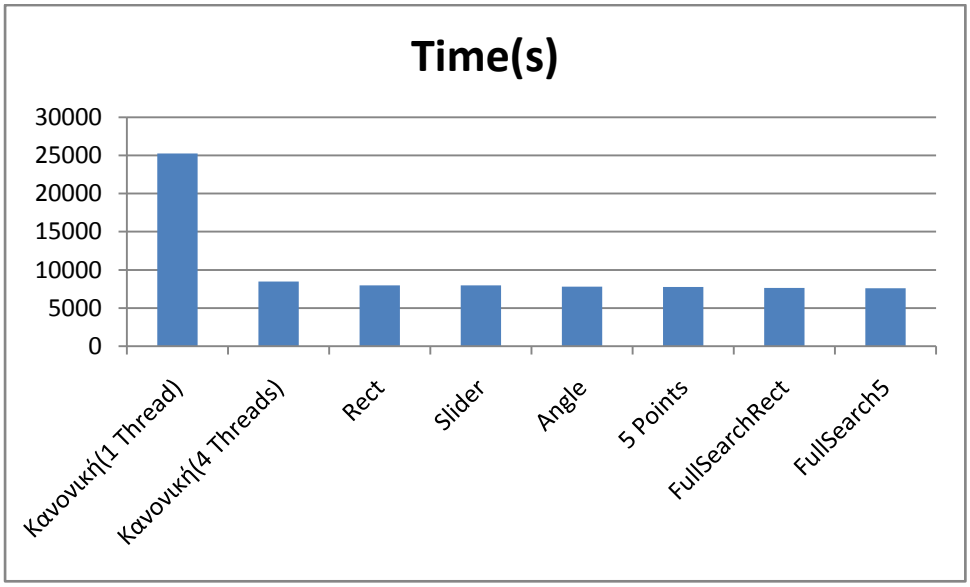
Για την πειραματική αξιολόγηση των τροποποιήσεων στη TZSearch είναι αναγκαία η δημιουργία νέου dataset που θα περιέχει το βίντεο καθώς και τις μετρήσεις από το γυροσκόπιο. Τραβήχτηκαν δύο βίντεο. Το πρώτο(118_1_5_10_29_43.yuv) είναι ένα βίντεο 720p 30fps 250 frames στο οποίο η συσκευή ακολουθάει την κίνηση μιας μπάλας μπάσκετ ενώ το δεύτερο(117_11_4_14_33_46.yuv) είναι ένα βίντεο 480p 30fps 250 frames στο οποίο η συσκευή πραγματοποιεί οριζόντια περιστροφή προς τα δεξιά καθώς τραβάει έναν εξωτερικό χώρο. Η κωδικοποίηση και των δυο πραγματοποιήθηκε για όλες τις μεθόδους που αναλύθηκαν για την tzSearch. Επίσης κωδικοποιήθηκαν και τα δύο βίντεο με την default έκδοση του κωδικοποιητή για σύγκριση του αρχικού συστήματος με τις τροποποιήσεις.

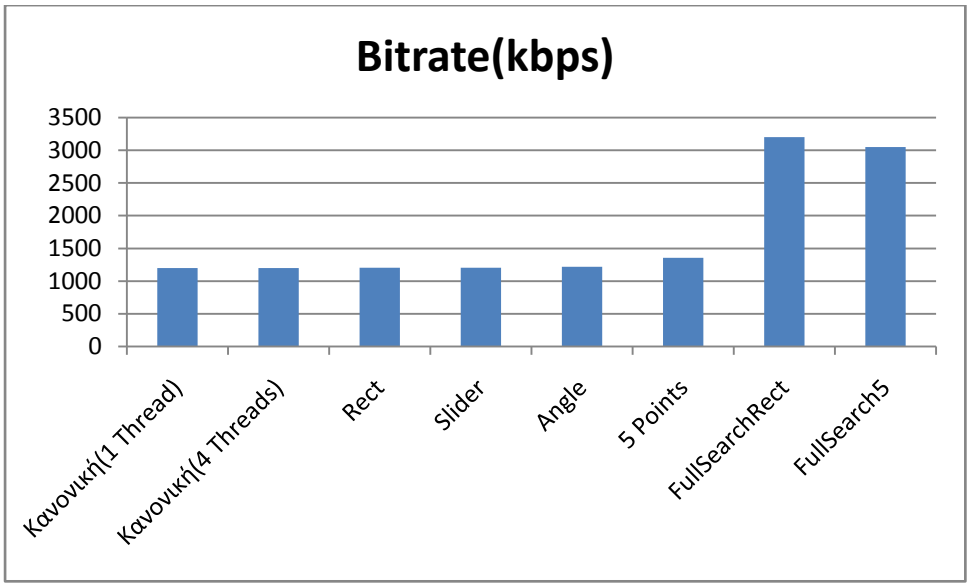
118_1_5_10_29_43.yuv

Σχήμα 6.5: Screenshot του 118_1_5_29_43.yuv

	PSNR(Y)	PSNR(U)	PSNR(V)	PSNR(YUV)	Time(s)	Bitrate(kbps)	Bytes στο δίσκο
Κανονική(1 Thread)	32.6495	46.3601	46.3594	34.2432	25243.458	1198.0023	1309414
Κανονική(4 Threads)	32.6478	46.3587	46.1050	34.2423	8480.239	1198.0665	1309969
Rect	32.6389	46.3913	46.1480	34.2341	7959.848	1203.4198	1315601
Slider	32.6436	46.3730	46.1068	34.2379	7987.291	1202.1320	1314333
Angle	32.6221	46.4150	46.0083	34.2156	7792.007	1218.1687	1330855
5 Points	32.5389	46.3836	46.0669	34.1215	7761.903	1354.5253	1472393
FullSearchRect	31.7530	46.3013	46.0253	33.1708	7630.368	3203.7725	3390937
FullSearch5	31.7671	46.2641	45.9171	33.2005	7585.415	3049.0545	3230443

Παρατητούμε από τις μετρήσεις παραπάνω ότι όσο μειώνονται τα σημεία που ψάχνονται από την tzSearch, μειώνεται και ο χρόνος. Παρατηρείται και αύξηση του bitrate όσο μειώνονται τα σημεία.





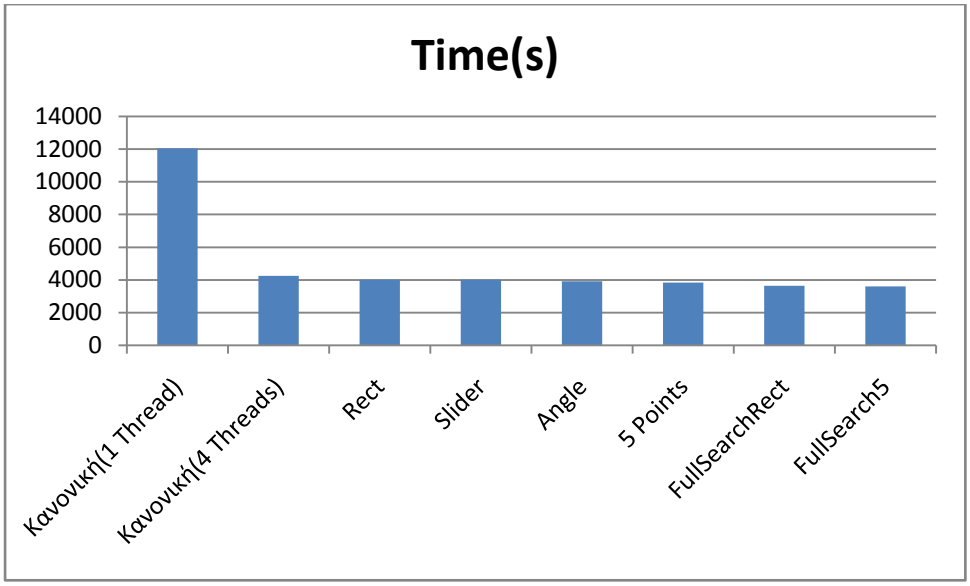
Σχήμα 6.6: Αποτελέσματα για το 118_1_5_29_43.yuv

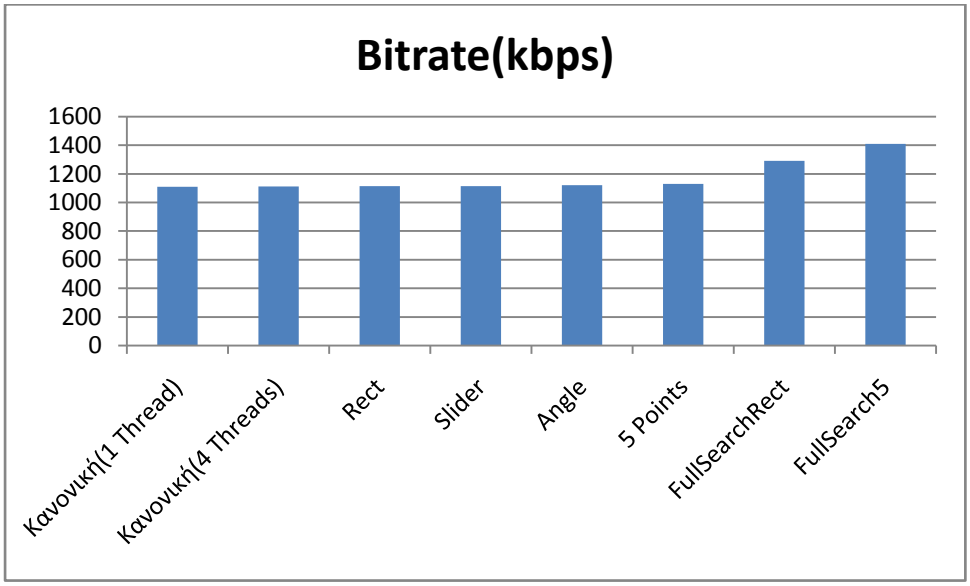
117_11_4_14_33_46.yuv

Σχήμα 6.7: Screenshot του 117_11_4_14_33_46.yuv

	PSNR(Y)	PSNR(U)	PSNR(V)	PSNR(YUV)	Time(s)	Bitrate(Kbps)	Bytes στο δίσκο
Κανονική(1 Thread)	31.8854	44.4632	44.4005	33.1084	12045.842	1109.5343	1181213
Κανονική(4 Threads)	31.8794	44.4596	44.3835	33.1023	4243.675	1112.5632	1184527
Rect	31.8785	44.4329	44.3801	33.1014	4015.846	1114.8684	1186781
Slider	31.8742	44.3693	44.3625	33.0970	4020.652	1115.7571	1187607
Angle	31.8626	44.4580	44.3726	33.0872	3915.325	1120.4357	1192523
5 Points	31.8458	44.3446	44.3989	33.0721	3838.988	1131.4487	1203932
FullSearchRect	31.6693	44.1236	44.1182	32.8987	3634.148	1291.7899	1370485
FullSearch5	31.5835	44.0031	43.9736	32.8122	3598.397	1409.8680	1492852

Παρατητούμε από τις μετρήσεις παραπάνω ότι όσο μειώνονται τα σημεία που ψάχνονται από την tzSearch, μειώνεται και ο χρόνος. Παρατηρείται και αύξηση του bitrate όσο μειώνονται τα σημεία.





Σχήμα 6.7: Αποτελέσματα για το 117_11_4_14_33_46.yuv

ΚΕΦΑΛΑΙΟ 7

Επίλογος

7.1 Συμπεράσματα

Η έλευση των κοινωνικών μέσων δικτύωσης και των έξυπνων συσκευών καθιστούν τη χρήση βίντεο ολοένα και πιο σημαντική. Μέσω του συνεχούς αυξανόμενου όγκου δεδομένων καθώς και της αναζήτησης όλο και υψηλότερης ανάλυσης ανακύπτει η ανάγκη για όσο πιο αποδική συμπίεση όπως επίσης και μείωσης του χρόνου.

Με την υπεροχή του HEVC έναντι των άλλων στανταρ δε θα ήταν περίεργο να λάβει στα επόμενα χρόνια μεγαλύτερο μέρος της αγοράς. Μέσω της παραλληλοποίησης με χρήση wavefront και την εξαιρετική μείωσης του χρόνου που τη διέπει η κωδικοποίηση γίνεται πρακτική δυνατότητα σε όλο και περισσότερους υπολογιστές οικιακής χρήσης. Είναι εμφανής η αποτελεσματικότητα με μείωσεις του χρόνου σχεδόν στο μισό για δύο πυρήνες, σχεδόν στο ένα τρίτον για τρεις πυρήνες κοκ.

Επιπρόσθετα, η ραγδαία εξέλιξη των έξυπνων συσκευών έχει ως αποτέλεσμα και την ανάπτυξη όλο και πιο ακριβέστερων αισθητήρων. Όχι μόνο η παρουσία του γυροσκοπίου αποτέλει κάτι το δεδομένο -κάτι που πριν λίγα χρόνια αποτελούσε προνόμιο μόνο των πιο πολυτελών συσκευών στην αγορά- αλλά και η ακρίβεια όλων των οργάνων αυτών μπορούν να καταστήσουν μια συσκευή εμπορικά επιτυχή ή ανεπιτυχή.

Με την εφαρμογή Android που αναπτύχθηκε δίνεται η δυνατότητα στο χρήστη να τραβήξει ένα βίντεο στις μέγιστες αναλύσεις τις οποίες υποστηρίζει η συσκευή του. Παράλληλα, η βελτίωση της κάμερας σε κάθε συσκευή και η δυνατότητα που παρέχουν τα λογισμικά τους για ανάπτυξη εφαρμογών υψηλού επιπέδου αναδεικνύουν την επιτακτικότητα όχι μόνο της χρησιμότητας αλλά και της ανάγκης δεδομένων για την κίνηση της συκευής. Ο περιορισμός των σημείων αναζήτησης της tzSearch κάνουν τη διαδικασία του inter-coding -η οποία λαμβάνει και το μεγαλύτερο χρόνο- αρκετά πιο σύντομη. Όπως δείχθηκε πειραματικά, η ελάχιστη αύξηση του bitrate σε συνδιασμό με το κέρδος στο χρόνο κωδικοποίησης δείχνουν τη χρησιμότητα στη χρήση δεδομένων κατά την κίνηση.

Ο συνδυασμός δε ενός παράλληλου συστήματος μαζί με τη χρήση metadata για την κίνηση της συσκευής κατά τη tzSearch μας δίνουν ικανοποιητικά αποτελέσματα τόσο από τη μεριά του bitrate όσο και από τη μεριά της χρονικής πολυπλοκότητας. Παρατηούμε μιρκή αύξηση του bitrate και περαιτέρω μείωση του χρόνου διεκπεραίωσης όσο μειώνεται ο αριθμός των σημείων αναζήτησης της tzSearch,

πράγμα αναμενόμενο. Θα πρέπει να σημειωθεί ότι ο κωδικοποιητής HM συνεχώς εξελίσσεται και η βελτίωση του θα έπιφέρει καλύτερα χρονικά αποτελέσματα καθώς και ποιότητα και πιστότητα βίντεο.

7.2 Μελλοντική εργασία

Όπως ισχύει για όλες τις εφαρμογές πάντα υπάρχει η δυνατότητα για περαιτέρω εξέλιξη. Παρότι ασχοληθήκαμε αποκλειστικά με το HEVC και τον κωδικοποιητή HM η διαδικασία αυτή θα μπορούσε να πραγματοποιηθεί για άλλους κωδικοποιητές όπως επίσης και για άλλλα στανταρ. Η ανάπτυξη καλύτερων μεθόδων πρόβλψης του σημείου κατά τη διαδικασία mapping καθώς και άλλων μεθόδων τροποποίησης της tzSearch είναι ένας άλλος τρόπος συνέχισης της έρευνας. Η μεταφορά των παράλληλων διαδικασιών αντί του CPU σε GPU, θα μπορούσε να επιφέρει μεγαλύτερη αξιοποίηση της παραλλληλίας και συνεπώς καλύτερα αποτελέσματα όσον αφορά τη χρονική πολυπλοκότητα.

Βιβλιογραφία

1. *Cisco Visual Networking Index: Forecast and Methodology, 2016–2021*
2. *A Brief History of Video Coding by Marco Jacobs and Jonah Probell ARC International*
3. *Av Overview of H.26x Series and its Applications by S.Vetrivel, K. Suba and Dr G.Athisha,* International Journal of Engineering Science and Technology *Vol 2*
4. *Overview of the H.264/AVC Video Coding Standard Thomas Wiegand, Gary J. Sullivan, Gisle Bjøntegaard, and Ajay Luthra,* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 13, NO. 7 2003
5. *Test Video Quality 720P 1080P 1440P 2160P 4320P Max Bitrate Which Compresses Youtube* https://www.tutorialguidacomefare.com/test-video-quality-720p-1080p-1440p-2160p-max-bitrate-which-compresses-youtube/
6. *Performance Comparison of AV1, JEM, VP9, and HEVC Encoders by Dan Grois, Tung Nguyen, and Detlev Marpe, (Conference Presentation). 120. 10.1117/12.2283428, (2017)*
7. Performance Comparison of H.265/MPEG-HEVC, VP9, and H.264/MPEG-AVC Encoders by Dan Grois and Detlev Marpe, 2013 Picture Coding Symposium (PCS), 394-397. 10.1109/PCS.2013.6737766, (2013)
8. *Comparison Of High Efficiency Video Coding (HEVC) Performance With H.264 Advanced Video Coding (AVC) by A. A. Basri and N. Zainal, Journal of Engineering Science and Technology Special Issue on 4th International Technical Conference 2014, 10. 102-111,(2015)*
9. *HEVC Advance Patent Pool Creates Confusion* http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/HEVC-Advance-Patent-Pool-Creates-Confusion-Lacks-Transparency-105235.aspx
10. *AV1 Announcement* https://blogs.cisco.com/collaboration/av1-video-codec av1
11. *HM encoder Official Site* https://hevc.hhi.fraunhofer.de
12. *High Efficiency Video Coding(HEVC), by Vivienne Sze, Madhukar Budagavi, Gary J. Sullivan, published by Springer, published by Springer*
13. *Combining tile parallelism with slice partitioning in video coding by Maria Koziri, Panos Papadopoulos and Thanasis Loukopoulos, 23. 10.1117/12.2320937,(2018)*
14. *Overview of the High Efficiency Video Coding (HEVC) Standard by Gary J. Sullivan, Jens-Rainer Ohm, Woo-Jin Han and Thomas Wiegand,* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 22

15. *H.264 and MPEG-4 Video Compression, by Iain E. G. Richardson, published by Wiley*
16. *Android Documentation https://developer.android.com/docs/*
17. *MACHINE VISION, by Ramesh Jain, Rangachar Kasturi, Brian G. Schunck, Published by McGraw-Hill*
18. *HM Software Documentation https://hevc.hhi.fraunhofer.de/HM-doc/*

Παράρτημα

Π1. ΜέθοδοςRect

```
__inline VoidTEncSearch::xTZ8PointDiamondSearch(const TComPattern* const
pcPatternKey,
   Int TZSearchStruct& rcStruct,
   const TComMv* const pcMvSrchRngLT,
   const TComMv* const pcMvSrchRngRB,
   const Int iStartX,
   const Int iStartY,
   const Inti Dist,
   const Bool bCheckCornersAtDist1,
   int vx, int vy)
{
   Const Int   iSrchRngHorLeft = pcMvSrchRngLT->getHor();
   Const Int   iSrchRngHorRight = pcMvSrchRngRB->getHor();
   const Int   iSrchRngVerTop = pcMvSrchRngLT->getVer();
   const Int   iSrchRngVerBottom = pcMvSrchRngRB->getVer();

   // 8 point search,                  //   1 2 3
   // search around the start point    //   4 0 5
   // with the required  distance      //   6 7 8
   assert(iDist != 0);
   const Int iTop = iStartY - iDist;
   const Int iBottom = iStartY + iDist;
   const Int iLeft = iStartX - iDist;
   const Int iRight = iStartX + iDist;
   rcStruct.uiBestRound += 1;

   if (iDist == 1)
   {

      if (iTop >= iSrchRngVerTop && iTop - 5<vy) // check top
      {
         if (bCheckCornersAtDist1)
         {
            if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check top-left
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iTop, 1, iDist);
            }
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
            if (iRight <= iSrchRngHorRight) // check middle right
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iRight, iTop, 3, iDist);
            }
```

```
        }
        else
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
        }
      }
      if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check middle left
      {
        xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
      }
      if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check middle right
      {
        xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
      }
      if (iBottom <= iSrchRngVerBottom && iBottom + 5>vy) // check bottom
      {
        if (bCheckCornersAtDist1)
        {
          if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check top-left
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iBottom, 6, iDist);
          }
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7, iDist);
          if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check middle
right
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iBottom, 8, iDist);
          }
        }
        else
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7, iDist);
        }
      }
    }
    else
    {
      if (iDist<= 8)
      {
        const Int iTop_2 = iStartY - (iDist>> 1);
        const Int iBottom_2 = iStartY + (iDist>> 1);
        const Int iLeft_2 = iStartX - (iDist>> 1);
        const Int iRight_2 = iStartX + (iDist>> 1);

        if (iTop >= iSrchRngVerTop && iLeft >= iSrchRngHorLeft &&
          iRight <= iSrchRngHorRight && iBottom <= iSrchRngVerBottom) // check
border
        {
          if (iTop - 5 <vy){ xTZSearchHelp(pcPatternKey, rcStruct, iStartX,
iTop, 2, iDist); }
          if (iLeft_2 - 5 <vx){
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iTop_2, 1, iDist>>
1);
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iBottom_2, 6,
iDist>> 1);
          }
          if (iRight_2 + 5 >vx){
            xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iTop_2, 3,
iDist>> 1);
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
```

```
            xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iBottom_2, 8,
iDist>> 1);
          }
          if (iBottom + 5 >vy){ xTZSearchHelp(pcPatternKey, rcStruct, iStartX,
iBottom, 7, iDist); }
        }
        else// check border
        {
          if (iTop >= iSrchRngVerTop && iTop - 5<vy) // check top
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
          }
          if (iTop_2 >= iSrchRngVerTop && iTop_2 - 5<vy) // check half top
          {
            if (iLeft_2 >= iSrchRngHorLeft && iLeft_2 - 5 <vx) // check half
left
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iTop_2, 1,
(iDist>> 1));
            }
            if (iRight_2 <= iSrchRngHorRight && iRight_2 + 5>vx) // check
half right
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iTop_2, 3,
(iDist>> 1));
            }
          } // check half top
          if (iLeft >= iSrchRngHorLeft  && iLeft - 5 <vx) // check left
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
          }
          if (iRight <= iSrchRngHorRight  && iRight + 5 >vx) // check right
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
          }
          if (iBottom_2 <= iSrchRngVerBottom && iBottom_2 + 5>vy) // check
half bottom
          {
            if (iLeft_2 >= iSrchRngHorLeft && iLeft_2 - 5 <vx) // check half
left
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iBottom_2, 6,
(iDist>> 1));
            }
            if (iRight_2 <= iSrchRngHorRight && iRight_2 - 5 <vx) // check
half right
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iBottom_2, 8,
(iDist>> 1));
            }
          } // check half bottom
          if (iBottom <= iSrchRngVerBottom && iBottom - 5<vy) // check bottom
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7,
iDist);
          }
        } // check border
      }
      else// iDist > 8
      {
        if (iTop >= iSrchRngVerTop && iLeft >= iSrchRngHorLeft &&
```

```
            iRight <= iSrchRngHorRight && iBottom <= iSrchRngVerBottom) // check
border
        {
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 0, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 0, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 0, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 0, iDist);
            for (Int index = 1; index < 4; index++)
            {
               const Int iPosYT = iTop + ((iDist>> 2) * index);
               const Int iPosYB = iBottom - ((iDist>> 2) * index);
               const Int iPosXL = iStartX - ((iDist>> 2) * index);
               const Int iPosXR = iStartX + ((iDist>> 2) * index);
               if (iPosXL - 5 <vx){
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYT, 0,
iDist);
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYB, 0,
iDist);
               }
               if (iPosXR + 5 >vx){
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYT, 0,
iDist);
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYB, 0,
iDist);
               }
            }
         }
         else// check border
         {
            if (iTop >= iSrchRngVerTop && iTop - 5<vy) // check top
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 0, iDist);
            }
            if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check left
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 0, iDist);
            }
            if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check right
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 0, iDist);
            }
            if (iBottom <= iSrchRngVerBottom && iBottom + 5>vy) // check bottom
            {
               xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 0,
iDist);
            }
            for (Int index = 1; index < 4; index++)
            {
               const Int iPosYT = iTop + ((iDist>> 2) * index);
               const Int iPosYB = iBottom - ((iDist>> 2) * index);
               const Int iPosXL = iStartX - ((iDist>> 2) * index);
               const Int iPosXR = iStartX + ((iDist>> 2) * index);

               if (iPosYT >= iSrchRngVerTop && iPosYT - 5<vy) // check top
               {
                  if (iPosXL >= iSrchRngHorLeft && iPosXL - 5<vx) // check left
                  {
                     xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYT, 0,
iDist);
                  }
                  if (iPosXR <= iSrchRngHorRight && iPosXR + 5>vx) // check right
                  {
```

```
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYT, 0,
iDist);
               }
            } // check top
            if (iPosYB <= iSrchRngVerBottom && iPosYB + 5>vy) // check bottom
            {
               if (iPosXL >= iSrchRngHorLeft && iPosXL - 5<vx) // check left
               {
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYB, 0,
iDist);
               }
               if (iPosXR <= iSrchRngHorRight && iPosXR + 5>vx) // check right
               {
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYB, 0,
iDist);
               }
            } // check bottom
         } // for ...
      } // check border
   } // iDist <= 8
} // iDist == 1
}
```

Π2 ΜέθοδοςAngle

```
_inline Void TEncSearch::xTZ8PointDiamondSearch(const TComPattern* const
pcPatternKey,
   Int TZSearchStruct& rcStruct,
   const TComMv* const pcMvSrchRngLT,
   const TComMv* const pcMvSrchRngRB,
   const Int iStartX,
   const Int iStartY,
   const Int iDist,
   const Bool bCheckCornersAtDist1,
   int vx)
{
   const Int   iSrchRngHorLeft = pcMvSrchRngLT->getHor();
   const Int   iSrchRngHorRight = pcMvSrchRngRB->getHor();
   const Int   iSrchRngVerTop = pcMvSrchRngLT->getVer();
   const Int   iSrchRngVerBottom = pcMvSrchRngRB->getVer();

   // 8 point search,                   //   1 2 3
   // search around the start point     //   4 0 5
   // with the required  distance       //   6 7 8
   assert(iDist != 0);
   const Int iTop = iStartY - iDist;
   const Int iBottom = iStartY + iDist;
   const Int iLeft = iStartX - iDist;
   const Int iRight = iStartX + iDist;
   rcStruct.uiBestRound += 1;

   if (iDist == 1)
   {

      if (iTop >= iSrchRngVerTop) // check top
      {
         if (bCheckCornersAtDist1)
```

```
        {
          if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check top-left
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iTop, 1, iDist);
          }
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
          if (iRight <= iSrchRngHorRight) // check middle right
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iTop, 3, iDist);
          }
        }
        else
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
        }
      }
      if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check middle left
      {
        xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
      }
      if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check middle right
      {
        xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
      }
      if (iBottom <= iSrchRngVerBottom) // check bottom
      {
        if (bCheckCornersAtDist1)
        {
          if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check top-left
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iBottom, 6, iDist);
          }
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7, iDist);
          if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check middle
right
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iBottom, 8, iDist);
          }
        }
        else
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7, iDist);
        }
      }
    }
    else
    {
      if (iDist <= 8)
      {
        const Int iTop_2 = iStartY - (iDist >> 1);
        const Int iBottom_2 = iStartY + (iDist >> 1);
        const Int iLeft_2 = iStartX - (iDist >> 1);
        const Int iRight_2 = iStartX + (iDist >> 1);

        if (iTop >= iSrchRngVerTop && iLeft >= iSrchRngHorLeft &&
          iRight <= iSrchRngHorRight && iBottom <= iSrchRngVerBottom) // check
border
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
          if (iLeft_2 - 5 <vx) {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iTop_2, 1, iDist
>> 1);
```

```
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iBottom_2, 6,
iDist >> 1);
          }
          if (iRight_2 + 5 >vx) {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iTop_2, 3, iDist
>> 1);
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iBottom_2, 8,
iDist >> 1);
          }
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7, iDist);
        }
        else// check border
        {
          if (iTop >= iSrchRngVerTop) // check top
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 2, iDist);
          }
          if (iTop_2 >= iSrchRngVerTop) // check half top
          {
            if (iLeft_2 >= iSrchRngHorLeft && iLeft_2 - 5 <vx) // check half
left
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iTop_2, 1,
(iDist >> 1));
            }
            if (iRight_2 <= iSrchRngHorRight && iRight_2 + 5>vx) // check
half right
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iTop_2, 3,
(iDist >> 1));
            }
          } // check half top
          if (iLeft >= iSrchRngHorLeft  && iLeft - 5 <vx) // check left
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 4, iDist);
          }
          if (iRight <= iSrchRngHorRight  && iRight + 5 >vx) // check right
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 5, iDist);
          }
          if (iBottom_2 <= iSrchRngVerBottom) // check half bottom
          {
            if (iLeft_2 >= iSrchRngHorLeft && iLeft_2 - 5 <vx) // check half
left
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iLeft_2, iBottom_2, 6,
(iDist >> 1));
            }
            if (iRight_2 <= iSrchRngHorRight && iRight_2 - 5 <vx) // check
half right
            {
              xTZSearchHelp(pcPatternKey, rcStruct, iRight_2, iBottom_2, 8,
(iDist >> 1));
            }
          } // check half bottom
          if (iBottom <= iSrchRngVerBottom) // check bottom
          {
            xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 7,
iDist);
          }
```

```
      } // check border
    }
    else// iDist > 8
    {
      if (iTop >= iSrchRngVerTop && iLeft >= iSrchRngHorLeft &&
        iRight <= iSrchRngHorRight && iBottom <= iSrchRngVerBottom) // check
border
      {
        xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 0, iDist);
        xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 0, iDist);
        xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 0, iDist);
        xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 0, iDist);
        for (Int index = 1; index < 4; index++)
        {
          const Int iPosYT = iTop + ((iDist >> 2) * index);
          const Int iPosYB = iBottom - ((iDist >> 2) * index);
          const Int iPosXL = iStartX - ((iDist >> 2) * index);
          const Int iPosXR = iStartX + ((iDist >> 2) * index);
          if (iPosXL - 5 <vx) {
            xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYT, 0,
iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYB, 0,
iDist);
          }
          if (iPosXR + 5 >vx) {
            xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYT, 0,
iDist);
            xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYB, 0,
iDist);
          }
        }
      }
      else// check border
      {
        if (iTop >= iSrchRngVerTop) // check top
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iTop, 0, iDist);
        }
        if (iLeft >= iSrchRngHorLeft && iLeft - 5<vx) // check left
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iLeft, iStartY, 0, iDist);
        }
        if (iRight <= iSrchRngHorRight && iRight + 5>vx) // check right
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iRight, iStartY, 0, iDist);
        }
        if (iBottom <= iSrchRngVerBottom) // check bottom
        {
          xTZSearchHelp(pcPatternKey, rcStruct, iStartX, iBottom, 0,
iDist);
        }
        for (Int index = 1; index < 4; index++)
        {
          const Int iPosYT = iTop + ((iDist >> 2) * index);
          const Int iPosYB = iBottom - ((iDist >> 2) * index);
          const Int iPosXL = iStartX - ((iDist >> 2) * index);
          const Int iPosXR = iStartX + ((iDist >> 2) * index);

          if (iPosYT >= iSrchRngVerTop) // check top
          {
            if (iPosXL >= iSrchRngHorLeft && iPosXL - 5<vx) // check left
            {
```

```
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYT, 0,
iDist);
               }
               if (iPosXR <= iSrchRngHorRight && iPosXR + 5>vx) // check right
               {
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYT, 0,
iDist);
               }
            } // check top
            if (iPosYB <= iSrchRngVerBottom) // check bottom
            {
               if (iPosXL >= iSrchRngHorLeft && iPosXL - 5<vx) // check left
               {
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXL, iPosYB, 0,
iDist);
               }
               if (iPosXR <= iSrchRngHorRight && iPosXR + 5>vx) // check right
               {
                  xTZSearchHelp(pcPatternKey, rcStruct, iPosXR, iPosYB, 0,
iDist);
               }
            } // check bottom
         } // for ...
      } // check border
   } // iDist <= 8
} // iDist == 1
}
```

Π4. Μέθοδος 5Point

```
for (int i = 0; i < 5; i++){
   if (pcCU->getSlice()->board[i][0] <= iSrchRngHorRight &&pcCU->getSlice()-
>board[i][0] >= iSrchRngHorLeft &&pcCU->getSlice()->board[i][1] <=
iSrchRngVerTop &&pcCU->getSlice()->board[i][1] >= iSrchRngVerBottom){

         xTZSearchHelp(pcPatternKey, cStruct, pcCU->getSlice()->board[i][0],
      pcCU->getSlice()->board[i][1], 0, iRaster);
      }
   }
```

Π5 Μέθοδοςfullrect

```
Void TEncSearch::xPatternSearch(const TComPattern* const pcPatternKey,
   Const Pel*              piRefY,
   Const Int iRefStride,
   Const TComMv* const pcMvSrchRngLT,
   Const TComMv* const pcMvSrchRngRB,
   TComMv& rcMv,
   Distortion& ruiSAD,TComDataCU* pcCU)
{
   Int    iSrchRngHorLeft = pcMvSrchRngLT->getHor();
   Int    iSrchRngHorRight = pcMvSrchRngRB->getHor();
   Int    iSrchRngVerTop = pcMvSrchRngLT->getVer();
   Int    iSrchRngVerBottom = pcMvSrchRngRB->getVer();

   Distortion  uiSad;
   Distortion  uiSadBest = std::numeric_limits<Distortion>::max();
   Int         iBestX = 0;
   Int         iBestY = 0;

   //-- jclee for using the SAD function pointer
   m_pcRdCost->setDistParam(pcPatternKey, piRefY, iRefStride, m_cDistParam);

   // fast encoder decision: use subsampled SAD for integer ME
   if (m_pcEncCfg->getFastInterSearchMode() == FASTINTERSEARCH_MODE1 ||
m_pcEncCfg->getFastInterSearchMode() == FASTINTERSEARCH_MODE3)
   {
      if (m_cDistParam.iRows > 8)
      {
         m_cDistParam.iSubShift = 1;
      }
   }
   int xm = pcCU->getSlice()->board[0][0];
   int ym = pcCU->getSlice()->board[0][1];
   piRefY += (iSrchRngVerTop * iRefStride);

   for (Int i = 0; i < 5; i++){

      Int x = xm;
      Int y = ym;
      if (x>iSrchRngHorLeft && x<iSrchRngHorRight && y>iSrchRngVerTop && y <
iSrchRngVerBottom){
         m_cDistParam.pCur = piRefY + y*iRefStride + x;
         setDistParamComp(COMPONENT_Y);

         m_cDistParam.bitDepth = pcPatternKey->getBitDepthY();
         uiSad = m_cDistParam.DistFunc(&m_cDistParam);

         // motion cost
         uiSad += m_pcRdCost->getCostOfVectorWithPredictor(x, y);

         if (uiSad < uiSadBest)
         {
            uiSadBest = uiSad;
            iBestX = x;
```

```
            iBestY = y;
            m_cDistParam.m_maximumDistortionForEarlyExit = uiSad;
         }


      }
   }
   rcMv.set(iBestX, iBestY);

   ruiSAD = uiSadBest - m_pcRdCost->getCostOfVectorWithPredictor(iBestX,
iBestY);
   return;
}
```

Π6 Μέθοδοςfull5

```
VoidTEncSearch::xPatternSearch(const TComPattern* const  pcPatternKey,
   Const Pel*               piRefY,
   Const Int iRefStride,
   const TComMv* const pcMvSrchRngLT,
   const TComMv* const pcMvSrchRngRB,
   TComMv& rcMv,
   Distortion& ruiSAD, TComDataCU* pcCU)
{
   Int    iSrchRngHorLeft = pcMvSrchRngLT->getHor();
   Int    iSrchRngHorRight = pcMvSrchRngRB->getHor();
   Int    iSrchRngVerTop = pcMvSrchRngLT->getVer();
   Int    iSrchRngVerBottom = pcMvSrchRngRB->getVer();

   Distortion  uiSad;
   Distortion  uiSadBest = std::numeric_limits<Distortion>::max();
   Int         iBestX = 0;
   Int         iBestY = 0;

   //-- jclee for using the SAD function pointer
   m_pcRdCost->setDistParam(pcPatternKey, piRefY, iRefStride, m_cDistParam);

   // fast encoder decision: use subsampled SAD for integer ME
   if (m_pcEncCfg->getFastInterSearchMode() == FASTINTERSEARCH_MODE1 ||
m_pcEncCfg->getFastInterSearchMode() == FASTINTERSEARCH_MODE3)
   {
      if (m_cDistParam.iRows > 8)
      {
         m_cDistParam.iSubShift = 1;
      }
   }
   int xm = pcCU->getSlice()->board[0][0];
   int ym = pcCU->getSlice()->board[0][1];
   piRefY += (iSrchRngVerTop * iRefStride);

   for (Int y = iSrchRngVerTop; y <= iSrchRngVerBottom; y++)
   {
      for (Int x = iSrchRngHorLeft; x <= iSrchRngHorRight; x++)
      {
         if (x - 5 < xm && x + 5 > xm && y - 5 < ym && y + 5 > ym){
            //  find min. distortion position
            m_cDistParam.pCur = piRefY + x;

            setDistParamComp(COMPONENT_Y);

            m_cDistParam.bitDepth = pcPatternKey->getBitDepthY();
```

```
            uiSad = m_cDistParam.DistFunc(&m_cDistParam);

            // motion cost
            uiSad += m_pcRdCost->getCostOfVectorWithPredictor(x, y);

            if (uiSad < uiSadBest)
            {
               uiSadBest = uiSad;
               iBestX = x;
               iBestY = y;
               m_cDistParam.m_maximumDistortionForEarlyExit = uiSad;
            }
         }
      }
      piRefY += iRefStride;
   }

   rcMv.set(iBestX, iBestY);

   ruiSAD = uiSadBest - m_pcRdCost->getCostOfVectorWithPredictor(iBestX,
iBestY);
   return;
}
```

П7 encoder_lowdelay_main.cfg

```
#======== File I/O
=====================
                        BitstreamFile                 : str.bin
                        ReconFile                     : rec.yuv
                        #======== Unit definition ================
                        MaxCUWidth                    : 64          # Maximum
                        coding unit width in pixel
                        MaxCUHeight                   : 64          # Maximum
                        coding unit height in pixel
                        MaxPartitionDepth             : 4           # Maximum
                        coding unit depth
                        QuadtreeTULog2MaxSize         : 5           # Log2 of
                        maximum transform size for
                                                                    # quadtree-
                        based TU coding (2...6)
                        QuadtreeTULog2MinSize         : 2           # Log2 of
                        minimum transform size for
                                                                    # quadtree-
                        based TU coding (2...6)
                        QuadtreeTUMaxDepthInter       : 3
                        QuadtreeTUMaxDepthIntra       : 3
                        #======== Coding Structure =============
                        IntraPeriod                   : -1          # Period of I-
                        Frame ( -1 = only first)
                        DecodingRefreshType           : 0           # Random
                        Accesss 0:none, 1:CDR, 2:IDR
                        GOPSize                       : 4           # GOP Size
                        (number of B slice = GOPSize-1)
                        #        Type POC QPoffset QPfactor tcOffsetDiv2
                        betaOffsetDiv2  temporal_id #ref_pics_active #ref_pics
                        reference pictures predict deltaRPS #ref_idcs reference
                        idcs
                        Frame1:  B    1   3        0.4624   0           0
                        0           4               4         -1 -5 -9 -13       0
                        Frame2:  B    2   2        0.4624   0           0
                        0           4               4         -1 -2 -6 -10       1
                        -1       5         1 1 1 0 1
                        Frame3:  B    3   3        0.4624   0           0
                        0           4               4         -1 -3 -7 -11       1
                        -1       5         0 1 1 1 1
                        Frame4:  B    4   1        0.578    0           0
                        0           4               4         -1 -4 -8 -12       1
                        -1       5         0 1 1 1 1
                        ListCombination               : 1           # Use combined
```

```
list for uni-prediction in B-slices
#========== Motion Search =============
FastSearch                    : 1           # 0:Full search
1:TZ search
SearchRange                   : 64          # (0: Search
range is a Full frame)
BipredSearchRange             : 4           # Search range
for bi-prediction refinement
HadamardME                    : 1           # Use of
hadamard measure for fractional ME
FEN                           : 1           # Fast encoder
decision
FDM                           : 1           # Fast Decision
for Merge RD cost
#======== Quantization =============
QP                            : 32          # Quantization
parameter(0-51)
MaxDeltaQP                    : 0           # CU-based
multi-QP optimization
MaxCuDQPDepth                 : 0           # Max depth of
a minimum CuDQP for sub-LCU-level delta QP
DeltaQpRD                     : 0           # Slice-based
multi-QP optimization
RDOQ                          : 1           # RDOQ
RDOQTS                        : 1           # RDOQ for
transform skip
#========== Deblock Filter ============
DeblockingFilterControlPresent: 0           # Dbl control
params present (0=not present, 1=present)
LoopFilterOffsetInPPS         : 0           # Dbl params:
0=varying params in SliceHeader, param = base_param +
GOP_offset_param; 1=constant params in PPS, param =
base_param)
LoopFilterDisable             : 0           # Disable
deblocking filter (0=Filter, 1=No Filter)
LoopFilterBetaOffset_div2     : 0           # base_param: -
13 ~ 13
LoopFilterTcOffset_div2       : 0           # base_param: -
13 ~ 13
#========== Misc. ============
InternalBitDepth              : 8           # codec
operating bit-depth
#========== Coding Tools =================
SAO                           : 1           # Sample
adaptive offset  (0: OFF, 1: ON)
AMP                           : 1           # Asymmetric
```

```
motion partitions (0: OFF, 1: ON)
TransformSkip                 : 1           # Transform
skipping (0: OFF, 1: ON)
TransformSkipFast             : 1           # Fast
Transform skipping (0: OFF, 1: ON)
SAOLcuBoundary                : 0           #
SAOLcuBoundary using non-deblocked pixels (0: OFF, 1: ON)
#============ Slices ================
SliceMode                : 0                # 0: Disable
all slice options.
                                            # 1: Enforce
maximum number of LCU in an slice,
                                            # 2: Enforce
maximum number of bytes in an 'slice'
                                            # 3: Enforce
maximum number of tiles in a slice
SliceArgument            : 1500             # Argument for
'SliceMode'.
                                            # If
SliceMode==1 it represents max. SliceGranularity-sized
blocks per slice.
                                            # If
SliceMode==2 it represents max. bytes per slice.
                                            # If
SliceMode==3 it represents max. tiles per slice.
LFCrossSliceBoundaryFlag : 1               # In-loop
filtering, including ALF and DB, is across or not across
slice boundary.
                                            # 0:not across,
1: across
#============ PCM ================
PCMEnabledFlag                      : 0               # 0:
No PCM mode
PCMLog2MaxSize                      : 5               #
Log2 of maximum PCM block size.
PCMLog2MinSize                      : 3               #
Log2 of minimum PCM block size.
PCMInputBitDepthFlag                : 1               # 0:
PCM bit-depth is internal bit-depth. 1: PCM bit-depth is
input bit-depth.
PCMFilterDisableFlag                : 0               # 0:
Enable loop filtering on I_PCM samples. 1: Disable loop
filtering on I_PCM samples.
#============ Tiles ================
UniformSpacingIdc                   : 0               # 0:
the column boundaries are indicated by ColumnWidth array,
```

```
the row boundaries are indicated by RowHeight array
                                                          # 1:
the column and row boundaries are distributed uniformly
NumTileColumnsMinus1                : 0               #
Number of columns in a picture minus 1
ColumnWidthArray                    : 2 3             #
Array containing ColumnWidth values in units of LCU (from
left to right in picture)
NumTileRowsMinus1                   : 0               #
Number of rows in a picture minus 1
RowHeightArray                      : 2               #
Array containing RowHeight values in units of LCU (from top
to bottom in picture)
LFCrossTileBoundaryFlag           : 1                 #
In-loop filtering is across or not across tile boundary.
                                                          #
0:not across, 1: across
#============ WaveFront ================
WaveFrontSynchro                    : 0               # 0:
No WaveFront synchronisation (WaveFrontSubstreams must be 1
in this case).
                                                          #
>0: WaveFront synchronises with the LCU above and to the
right by this many LCUs.
#=========== Quantization Matrix =================
ScalingList                   : 0                     #
ScalingList 0 : off, 1 : default, 2 : file read
ScalingListFile               : scaling_list.txt      #
Scaling List file name. If file is not exist, use Default
Matrix.
#============ Lossless ================
TransquantBypassEnableFlag: 0  # Value of PPS flag.
CUTransquantBypassFlagValue: 0 # Constant lossless-value
signaling per CU, if TransquantBypassEnableFlag is 1.
#============ Rate Control ======================
RateControl                         : 0               #
Rate control: enable rate control
TargetBitrate                       : 1000000         #
Rate control: target bitrate, in bps
KeepHierarchicalBit                 : 1               #
Rate control: keep hierarchical bit allocation in rate
control algorithm
LCULevelRateControl                 : 1               #
Rate control: 1: LCU level RC; 0: picture level RC
RCLCUSeparateModel                  : 1               #
Rate control: use LCU level separate R-lambda model
```

```
InitialQP                           : 0               #
Rate control: initial QP
RCForceIntraQP                      : 0               #
Rate control: force intra QP to be equal to initial QP
### DO NOT ADD ANYTHING BELOW THIS LINE ###
### DO NOT DELETE THE EMPTY LINE BELOW ###
```

Π8 Εγγραφή στοιχείων στο external Storage Directory της Android Συσκευής

```
File f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+ "rawvideo/" +
name+"_"+"a_"+camera.getParameters().getPreviewSize().wid
th+"_"+camera.getParameters().getPreviewSize().height+".t
xt");
f.createNewFile();
FileOutputStream fos = new FileOutputStream(f);
DataOutputStream dos = new DataOutputStream(fos);
dos.writeChars(obj1.toString());
dos.flush();
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"u_"+camera.getParameters().getPreviewSize().wid
th+"_"+camera.getParameters().getPreviewSize().height+".t
xt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars(obj2.toString());
dos.flush();
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"agrav_"+camera.getParameters().getPreviewSize()
.width+"_"+camera.getParameters().getPreviewSize().height
+".txt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars(obj3.toString());
dos.flush();
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"rot_"+camera.getParameters().getPreviewSize().w
idth+"_"+camera.getParameters().getPreviewSize().height+"
.txt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars(obj4.toString());
dos.flush();
```

```
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"zoom_"+camera.getParameters().getPreviewSize().
width+"_"+camera.getParameters().getPreviewSize().height+
".txt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars(obj5.toString());
dos.flush();
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"focal_"+camera.getParameters().getPreviewSize()
.width+"_"+camera.getParameters().getPreviewSize().height
+".txt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars(obj6.toString());
dos.flush();
dos.close();
f=new
File(Environment.getExternalStorageDirectory().getAbsolut
ePath()+File.separator+"rawvideo/" +
name+"_"+"sensosize_"+camera.getParameters().getPreviewSi
ze().width+"_"+camera.getParameters().getPreviewSize().he
ight+".txt");
f.createNewFile();
fos = new FileOutputStream(f);
dos = new DataOutputStream(fos);
dos.writeChars("width: "+width+" height: "+height);
dos.flush();
dos.close();
```

П9 CompressCU

```
voidTEncSlice::compressCu(UIntctuTsAddr,UIntframeWidthInCtus,
UIntstartCtuTsAddr, UIntboundingCtuTsAddr, TComPic* pcPic,
TComSlice* pcSlice, TEncBinCABAC** pRDSbacCoder,
TComBitCounter* bitcounters,BoolbCompressEntireSlice){
```

```
    constUInt ctuRsAddr = pcPic->getPicSym()-
>getCtuTsToRsAddrMap(ctuTsAddr);

    int row = ctuRsAddr / 20;
    TComDataCU* pCtu = pcPic->getCtu(ctuRsAddr);
    pCtu->initCtu(pcPic, ctuRsAddr);


    constUInt firstCtuRsAddrOfTile = pcPic->getPicSym()-
>getTComTile(pcPic->getPicSym()->getTileIdxMap(ctuRsAddr))-
>getFirstCtuRsAddr();
    constUInt tileXPosInCtus = firstCtuRsAddrOfTile %
frameWidthInCtus;
    constUInt ctuXPosInCtus = ctuRsAddr % frameWidthInCtus;

    if (ctuRsAddr == firstCtuRsAddrOfTile)
    {
            sbacs[row][0][CI_CURR_BEST]-
>resetEntropy(pcSlice);

    }
    else if (ctuXPosInCtus == tileXPosInCtus && m_pcCfg-
>getEntropyCodingSyncEnabledFlag())
    {
      sbacs[row][0][CI_CURR_BEST]->resetEntropy(pcSlice);
            TComDataCU *pCtuUp = pCtu->getCtuAbove();
      if (pCtuUp && ((ctuRsAddr%frameWidthInCtus + 1)
<frameWidthInCtus))
      {
        TComDataCU *pCtuTR = pcPic->getCtu(ctuRsAddr -
frameWidthInCtus + 1);
        if (pCtu->CUIsFromSameSliceAndTile(pCtuTR))
        {
          sbacs[row][0][CI_CURR_BEST]-
>loadContexts(contexts[row]);
        }
      }
    }


    entropyCoders[row]->setEntropyCoder(goOnSbacs[row]);

    entropyCoders[row]->setBitstream(&bitcounters[row]);
    bitcounters[row].resetBits();
    goOnSbacs[row]->load(sbacs[row][0][CI_CURR_BEST
    ((TEncBinCABAC*)goOnSbacs[row]->getEncBinIf())-
>setBinCountingEnableFlag(true);
    Double oldLambda = rdCosts[row]->getLambda();
    if (m_pcCfg->getUseRateCtrl())
```

```
    {
      Int estQP = pcSlice->getSliceQp();
      Double estLambda = -1.0;
      Double bpp = -1.0;

      if ((pcPic->getSlice(0)->getSliceType() == I_SLICE&&
m_pcCfg->getForceIntraQP()) || !m_pcCfg->getLCULevelRC())
      {
        estQP = pcSlice->getSliceQp();
      }
      else
      {

        bpp = rateControls[row]->getRCPic()-
>getLCUTargetBpp(pcSlice->getSliceType());
        if (pcPic->getSlice(0)->getSliceType() == I_SLICE)
        {
                    estLambda = rateControls[row]-
>getRCPic()->getLCUEstLambdaAndQP(bpp, pcSlice->getSliceQp(),
&estQP);
        }
        else
        {
                    estLambda = rateControls[row]-
>getRCPic()->getLCUEstLambda(bpp);
                    estQP = rateControls[row]->getRCPic()-
>getLCUEstQP(estLambda, pcSlice->getSliceQp());
        }

        estQP = Clip3(-pcSlice->getSPS()-
>getQpBDOffset(CHANNEL_TYPE_LUMA), MAX_QP, estQP);
        rdCosts[row]->setLambda(estLambda, pcSlice->getSPS()-
>getBitDepths());
#ifRDOQ_CHROMA_LAMBDA
                  constDouble chromaLambda = estLambda /
rdCosts[row]->getChromaWeight       constDouble
lambdaArray[MAX_NUM_COMPONENT] = { estLambda, chromaLambda,
chromaLambda };
                  trQuants[row]->setLambdas(lambdaArray);
#else
        m_pcTrQuant->setLambda(estLambda);
#endif
      }

      rateControls[row]->setRCQP(estQP);
#ifADAPTIVE_QP_SELECTION
      pCtu->getSlice()->setSliceQpBase(estQP);
#endif
    }
```

```
    cuCoders[row]->compressCtu(pCtu);

    entropyCoders[row]-
>setEntropyCoder(sbacs[row][0][CI_CURR_BEST]);
        entropyCoders[row]->setBitstream(&bitcounters[row]);
    pRDSbacCoder[row]->setBinCountingEnableFlag(true);

    sbacs[row][0][CI_CURR_BEST]->resetBits();
    pRDSbacCoder[row]->setBinsCoded(0);

    m1.lock();
    cuCoders[row]->encodeCtu(pCtu);
    m1.unlock();
    pRDSbacCoder[row]->setBinCountingEnableFlag(false);
    constInt numberOfWrittenBits = entropyCoders[row]-
>getNumberOfWrittenBits();

    constUInt validEndOfSliceCtuTsAddr = ctuTsAddr +
(ctuTsAddr == startCtuTsAddr ? 1 : 0);
        if (pcSlice->getSliceMode() ==
FIXED_NUMBER_OF_BYTES&&pcSlice->getSliceBits() +
numberOfWrittenBits > (pcSlice->getSliceArgument() << 3))
    {
      pcSlice-
>setSliceSegmentCurEndCtuTsAddr(validEndOfSliceCtuTsAddr);
      pcSlice-
>setSliceCurEndCtuTsAddr(validEndOfSliceCtuTsAddr);
      boundingCtuTsAddr = validEndOfSliceCtuTsAddr;
    }
    elseif ((!bCompressEntireSlice) &&pcSlice-
>getSliceSegmentMode() == FIXED_NUMBER_OF_BYTES&&pcSlice-
>getSliceSegmentBits() + numberOfWrittenBits > (pcSlice-
>getSliceSegmentArgument() << 3))
    {
      pcSlice-
>setSliceSegmentCurEndCtuTsAddr(validEndOfSliceCtuTsAddr);
      boundingCtuTsAddr = validEndOfSliceCtuTsAddr;
    }

    if (boundingCtuTsAddr<= ctuTsAddr)
    {
      return;
    }

    pcSlice->setSliceBits((UInt)(pcSlice->getSliceBits() +
numberOfWrittenBits));
```

```
    pcSlice->setSliceSegmentBits(pcSlice-
>getSliceSegmentBits() + numberOfWrittenBits);
    if (ctuXPosInCtus == tileXPosInCtus + 1 && m_pcCfg-
>getEntropyCodingSyncEnabledFlag())
    {

      contexts[row]-
>loadContexts(sbacs[row][0][CI_CURR_BEST]);
    }


    if (m_pcCfg->getUseRateCtrl())
    {
      Int actualQP = g_RCInvalidQPValue;
      Double actualLambda = rdCosts[row]->getLambda    Int
actualBits = pCtu->getTotalBits();
      Int numberOfEffectivePixels = 0;
      for (Int idx = 0; idx <pcPic->getNumPartitionsInCtu();
idx++)
      {
        if (pCtu->getPredictionMode(idx) !=
NUMBER_OF_PREDICTION_MODES&& (!pCtu->isSkipped(idx)))
        {
          numberOfEffectivePixels = numberOfEffectivePixels +
16;
          break;
        }
      }

      if (numberOfEffectivePixels == 0)
      {
        actualQP = g_RCInvalidQPValue;
      }
      else
      {
        actualQP = pCtu->getQP(0);
      }
      rdCosts[row]->setLambda(oldLambda, pcSlice->getSPS()-
>getBitDepths());
      rateControls[row]->getRCPic()-
>updateAfterCTUrateControls[row]->getRCPic()->getLCUCoded(),
actualBits, actualQP, actualLambda,
        pCtu->getSlice()->getSliceType() == I_SLICE ? 0 :
m_pcCfg->getLCULevelRC());
    }
    m2.lock();
    m_uiPicTotalBits += pCtu->getTotalBits();
    m_dPicRdCost += pCtu->getTotalCost();
    m_uiPicDist += pCtu->getTotalDistortion();
```

```
        m2.unlock();
    }
```

Π10 Init του TEncTop

```
Void TEncTop::init(Bool isFieldCoding)
{
  TComSPS &sps0=*(m_spsMap.allocatePS(0)); // NOTE: implementations that use
more than 1 SPS need to be aware of activation issues.
  TComPPS &pps0=*(m_ppsMap.allocatePS(0));
  // initialize SPS
  xInitSPS(sps0);
  xInitVPS(m_cVPS, sps0);

  if (m_RCCpbSaturationEnabled)
  {
    m_cRateCtrl.initHrdParam(sps0.getVuiParameters()->getHrdParameters(),
m_iFrameRate, m_RCInitialCpbFullness);
  }

  m_cRdCost.setCostMode(m_costMode);

  // initialize PPS
  xInitPPS(pps0, sps0);
  xInitRPS(sps0, isFieldCoding);
  xInitScalingLists(sps0, pps0);

  if (m_wcgChromaQpControl.isEnabled())
  {
    TComPPS &pps1=*(m_ppsMap.allocatePS(1));
    xInitPPS(pps1, sps0);
    xInitScalingLists(sps0, pps1);
  }
  //entropyCoders = (TEncEntropy**)malloc(8 * sizeof(TEncEntropy*));
  //cuCoders = (TEncCu*)malloc(8 * sizeof(TEncCu));
  // initialize processing unit classes
  m_cGOPEncoder.  init( this );
  m_cSliceEncoder.init( this );
  m_cCuEncoder.   init( this );
  m_cCuEncoder.setSliceEncoder(&m_cSliceEncoder);
  for (int i = 0; i < 8; i++){
     cuCoders[i]->init(this);
  }
  // initialize transform & quantization class
  m_pcCavlcCoder = getCavlcCoder();

  m_cTrQuant.init( 1 << m_uiQuadtreeTULog2MaxSize,
                   m_useRDOQ,
                   m_useRDOQTS,
                   m_useSelectiveRDOQ,
                   true
                  ,m_useTransformSkipFast
#if ADAPTIVE_QP_SELECTION
                  ,m_bUseAdaptQpSelect
#endif
                  );
  for (int i = 0; i < 8; i++){
     trQuants[i]->init(1 << m_uiQuadtreeTULog2MaxSize,
        m_useRDOQ,
        m_useRDOQTS,
        m_useSelectiveRDOQ,
        true
        , m_useTransformSkipFast
#if ADAPTIVE_QP_SELECTION
```

```
        , m_bUseAdaptQpSelect
#endif
        );
  }
  // initialize encoder search class

  m_cSearch.init( this, &m_cTrQuant, m_iSearchRange, m_bipredSearchRange,
m_motionEstimationSearchMethod, m_maxCUWidth, m_maxCUHeight, m_maxTotalCUDepth,
&m_cEntropyCoder, &m_cRdCost, getRDSbacCoder(), getRDGoOnSbacCoder() );

  searches = (TEncSearch**)malloc(8 * sizeof(TEncSearch*));
  for (int i = 0; i < 8; i++){
     searches[i] = new TEncSearch;
     rdCosts[i]->setCostMode(m_costMode);
     cuCoders[i]->setEntropy(entropyCoders[i]);
     cuCoders[i]->setSbac(SbacCoders[i]);
     cuCoders[i]->setGoOnSbac(goOnSbacs[i]);
     cuCoders[i]->setRdCost(rdCosts[i]);
     cuCoders[i]->setCABAC(goOnCabacs[i]);
     cuCoders[i]->setRateCrtl(rateControls[i]);
     cuCoders[i]->setTrQuant(trQuants[i]);
     searches[i]->init(this, trQuants[i], m_iSearchRange, m_bipredSearchRange,
m_motionEstimationSearchMethod, m_maxCUWidth, m_maxCUHeight, m_maxTotalCUDepth,
entropyCoders[i], rdCosts[i], SbacCoders[i], goOnSbacs[i]);
       cuCoders[i]->setSearch(searches[i]);
  }
  m_iMaxRefPicNum = 0;
}

Void TEncTop::xInitScalingLists(TComSPS &sps, TComPPS &pps)
{
  // Initialise scaling lists
  // The encoder will only use the SPS scaling lists. The PPS will never be
marked present.
  const Int maxLog2TrDynamicRange[MAX_NUM_CHANNEL_TYPE] =
  {
      sps.getMaxLog2TrDynamicRange(CHANNEL_TYPE_LUMA),
      sps.getMaxLog2TrDynamicRange(CHANNEL_TYPE_CHROMA)
  };
  if(getUseScalingListId() == SCALING_LIST_OFF)
  {
    getTrQuant()->setFlatScalingList(maxLog2TrDynamicRange,
sps.getBitDepths());
    getTrQuant()->setUseScalingList(false);
   for (int i = 0; i < 8; i++){
     trQuants[i]->setFlatScalingList(maxLog2TrDynamicRange,
sps.getBitDepths());
      trQuants[i]->setUseScalingList(false);
   }
    sps.setScalingListPresentFlag(false);
    pps.setScalingListPresentFlag(false);
  }
  else if(getUseScalingListId() == SCALING_LIST_DEFAULT)
  {
    sps.getScalingList().setDefaultScalingList ();
    sps.setScalingListPresentFlag(false);
    pps.setScalingListPresentFlag(false);

    getTrQuant()->setScalingList(&(sps.getScalingList()),
maxLog2TrDynamicRange, sps.getBitDepths());
    getTrQuant()->setUseScalingList(true);
   for (int i = 0; i < 8; i++){
```

```
      trQuants[i]->setScalingList(&(sps.getScalingList()),
maxLog2TrDynamicRange, sps.getBitDepths());
      trQuants[i]->setUseScalingList(true);
   }
  }
  else if(getUseScalingListId() == SCALING_LIST_FILE_READ)
  {
    sps.getScalingList().setDefaultScalingList ();
    if(sps.getScalingList().xParseScalingList(getScalingListFileName()))
    {
      Bool bParsedScalingList=false; // Use of boolean so that assertion
outputs useful string
      assert(bParsedScalingList);
      exit(1);
    }
    sps.getScalingList().checkDcOfMatrix();

sps.setScalingListPresentFlag(sps.getScalingList().checkDefaultScalingList());
    pps.setScalingListPresentFlag(false);
    getTrQuant()->setScalingList(&(sps.getScalingList()),
maxLog2TrDynamicRange, sps.getBitDepths());
    getTrQuant()->setUseScalingList(true);
   for (int i = 0; i < 8; i++){
     trQuants[i]->setScalingList(&(sps.getScalingList()),
maxLog2TrDynamicRange, sps.getBitDepths());
      getTrQuant()->setUseScalingList(true);
   }
  }
  else
  {
    printf("error : ScalingList == %d not supported\n",getUseScalingListId());
    assert(0);
  }

  if (getUseScalingListId() != SCALING_LIST_OFF)
  {
    // Prepare delta's:
    for(UInt sizeId = 0; sizeId < SCALING_LIST_SIZE_NUM; sizeId++)
    {
      const Int predListStep = (sizeId == SCALING_LIST_32x32?
(SCALING_LIST_NUM/NUMBER_OF_PREDICTION_MODES) : 1); // if 32x32, skip over
chroma entries.

      for(UInt listId = 0; listId < SCALING_LIST_NUM; listId+=predListStep)
      {
        sps.getScalingList().checkPredMode( sizeId, listId );
      }
    }
  }
}
```