ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

ΑΛΓΟΡΙΘΜΟΙ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ VIDEO

Περιβολάρης Σταύρος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ
Υπεύθυνος
Δελήμπασης Κωνσταντίνος
Επίκουρος Καθηγητής

Λαμία, Μάρτιος έτος 2015

ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ
ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΜΕ ΕΦΑΡΜΟΓΕΣ ΣΤΗ ΒΙΟΙΑΤΡΙΚΗ

ΑΛΓΟΡΙΘΜΟΙ ΤΜΗΜΑΤΟΠΟΙΗΣΗΣ VIDEO

Περιβολάρης Σταύρος

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Επιβλέπων
Δελήμπασης Κωνσταντίνος
Επίκουρος Καθηγητής

Λαμία, Μάρτιος έτος 2015

Αλγόριθμοι Τμηματοποίησης Video

Περιβολάρης Σταύρος

Τριμελής Επιτροπή:

Δελήμπασης Κωνσταντίνος, Επίκουρος Καθηγητής

Λουκόπουλος Αθανάσιος, Λέκτωρ

Πλαγιανάκος Βασίλης, Αναπληρωτής Καθηγητής

Περιεχόμενα

Περίληψη

Στην παρούσα εργασία έγινε έρευνα σχετικά με την παραλληλοποίηση αλγορίθμου τμηματοποίησης σε video, με χρήση του μοντέλου server/client. Σκοπός ήταν η υλοποίησή του με τέτοιον τρόπο ώστε να επιτυγχάνεται η καλύτερη δυνατή ταχύτητα εκτέλεσής του. Δηλαδή, περισσότερα frames/second σε σχέση με αυτά της σειριακής εκτέλεσης. Χρησιμοποιήθηκαν, δοκιμαστικά, διάφορες αρχιτεκτονικές και πρωτόκολλα επικοινωνίας και έγινε η μετατροπή του κώδικα από περιβάλλον Matlab σε Java και συγκεκριμένα, Processing. Η Java παρείχε την απαιτούμενη ευελιξία όσον αφορά τον χειρισμό των δεδομένων και της επικοινωνίας μεταξύ ετερογενών υπολογιστών. Ειδικότερα, δύο αλγόριθμοι ήταν αυτοί που μετατράπηκαν και επιλέχθηκε αυτός με τις καλύτερες προϋποθέσεις και βελτιώσεις. Φυσικά, έγιναν κάποιες τροποποιήσεις για να ταιριάξουν στην περίπτωση που μελετήθηκε καθώς και κάποιες ακόμα για την βελτιστοποίηση, όσο αυτή ήταν εφικτή, του χρόνου εκτέλεσής του.

Το αρχείο video στο οποίο έγιναν οι δοκιμές προήλθε από την camera που είναι εγκατεστημένη στο πανεπιστήμιο. Μετά την σωστή τμηματοποίηση του video έγιναν και δοκιμές με camera, η οποία ήταν συνδεδεμένη με τον client υπολογιστή. Το Processing ήταν αυτό που παρείχε την δυνατότητα επιλογής και χρήσης της, αφού έχει τις κατάλληλες συναρτήσεις. Ωστόσο, αυτή η χρήση, επειδή το Processing έχει ειδικές βιβλιοθήκες, περιορίζει την υλοποίηση σε περιβάλλοντα που χρησιμοποιούν μόνο το συγκεκριμένο.

Στόχος όλων αυτών είναι η αναγνώριση κίνησης στον χώρο για την παρακολούθηση και τον έλεγχο στην υγεία των ηλικιωμένων και των χρόνια ασθενών. Η τμηματοποίηση είναι αυτή η διαδικασία που εστιάζει στην κίνηση και αποκρύπτει τις υπόλοιπες πληροφορίες, δηλαδή το φόντο (υπόβαθρο) που περιέχει τα στατικά αντικείμενα, για παράδειγμα ένα δωμάτιο. Κρίνεται, λοιπόν, απαραίτητη η καλύτερη δυνατή ταχύτητα ανταλλαγής και επεξεργασίας των δεδομένων για την πρόληψη ατυχών καταστάσεων ή την γρήγορη ανταπόκριση όπου υπάρχει πρόβλημα.

Κεφάλαιο 1 Εισαγωγή

Σε πρώτο στάδιο έγινε χρήση της αρχιτεκτονικής των κατανεμημένων συστημάτων. Παρακάτω, θα παρουσιαστεί μια γενική περιγραφή τους και πώς αυτά λειτουργούν για έναν κοινό σκοπό, που είναι εν ολίγοις η επιτάχυνση της εκτέλεσης των υπολογισμών με χρήση παραπάνω του ενός υπολογιστών. Έπειτα, χρειάζεται να αναφερθεί και ο τρόπος λειτουργίας των sockets που είναι αυτά που επιτρέπουν στους υπολογιστές να επικοινωνούν μεταξύ τους. Σε επόμενο στάδιο θα γίνει λόγος σχετικά με τα χαρακτηριστικά που διέπουν την εικόνα, την αναγκαιότητα και τον τρόπο της διάσπασής της και τέλος τον αλγόριθμο τμηματοποίησης που χρησιμοποιήθηκε.

Συνοψίζοντας, ο παραλληλισμός είναι ένας τρόπος κατά τον οποίο οι υπολογισμοί γίνονται ταυτόχρονα. Ουσιαστικά, μεγάλα προβλήματα διασπώνται σε μικρότερα, τα οποία είναι γρηγορότερα και ευκολότερα στην εκτέλεση, με χρήση των πολύ-επεξεργαστικών συστημάτων. Τελικά, τα αποτελέσματα αυτά συγκεντρώνονται για να προκύψει το συνολικό που θα πρέπει να είναι αυτό που θα προέκυπτε αν οι υπολογισμοί γίνονταν σειριακά. Οι δυσκολίες που έγκεινται σε αυτή την μέθοδο είναι σχετικές με την επικοινωνία, τον συγχρονισμό καθώς και τη διαχείριση των πόρων που θεωρούνται κατάλληλοι για χρήση. Για το τελευταίο, μάλιστα, έχουν προταθεί κάποιες μαθηματικές λύσεις, ώστε να γίνεται, θεωρητικά, ο υπολογισμός των πόρων που χρειάζονται αλλά και η επιτάχυνση που επιτυγχάνεται.

1. ΚΑΤΑΝΕΜΗΜΕΝΑ ΣΥΣΤΗΜΑΤΑ

Μια μορφή παράλληλης επεξεργασίας, η οποία ήταν και η κατάλληλη για τον σκοπό της εργασίας, είναι αυτή των κατανεμημένων συστημάτων. Χωρίς να ξεφεύγει από τα πρότυπα του παράλληλου προγραμματισμού, σ’ αυτήν χρησιμοποιούνται επιπλέον υπολογιστές για την επίτευξη της γρηγορότερης εκτέλεσης των αλγορίθμων, μέσω των αρχιτεκτονικών του δικτύου. Υπάρχει μια μορφή επικοινωνίας μεταξύ τους, κλειδί για τον αντικειμενοστραφή προγραμματισμό, γνωστή και ως «ανταλλαγή μηνυμάτων». Σαν «μήνυμα» θεωρείται το «αντικείμενο», που μπορεί να ‘ναι μια μεταβλητή, μια δομή δεδομένων ή και μια συνάρτηση. Το «μήνυμα» - αντικείμενο» αυτό αποστέλλεται σε άλλον υπολογιστή – σύστημα, στο οποίο εκτελείται μια διεργασία, και ταυτόχρονα αποτελεί σήμα για τη διεργασία να ξεκινήσει την επεξεργασία πάνω στο «μήνυμα». Αφού όμως χρησιμοποιήθηκε μέσω Processing η γλώσσα προγραμματισμού Java, σαν «μήνυμα» ακόμα

συγκαταλέγονται και οι συνδυασμοί δομών δεδομένων, συναρτήσεων και μεταβλητών, μιας και η ύπαρξη κλάσεων με όλες τις ιδιότητές τους είναι αναπόφευκτη.

Τα κατανεμημένα συστήματα δεν μπορούν να θεωρηθούν ότι ανήκουν στην ιδέα του master – slave. Στην ουσία, είναι ξέχωρα, αυτόνομα συστήματα, δηλαδή ένα σύνολο hardware και software που βρίσκονται σε διαφορετικούς υπολογιστές, τα οποία λειτουργούν «μαζί» για την επίτευξη ενός κοινού στόχου. Σαν ένα συγκεκριμένο πρόγραμμα να εκτελείται σε όλα τα μηχανήματα και οι διαφορετικές ενότητές του να εκτελούνται με την σειρά τους σε άλλα μηχανήματα. Τέλος, τα κατανεμημένα συστήματα χρησιμοποιούν, το καθένα, την δική του μνήμη κι έτσι δεν υπάρχει χάσιμο χρόνου σε προσπελάσεις και διευθυνσιοδοτήσεις σε κοινή μνήμη.

2. SOCKETS – ΠΡΩΤΟΚΟΛΛΟ ΕΠΙΚΟΙΝΩΝΙΑΣ

Σαν sockets ορίζονται τα άκρα από έναν δίαυλο επικοινωνίας διπλής κατεύθυνσης μεταξύ δύο προγραμμάτων που εκτελούνται στο δίκτυο. Οι αναγκαίες βιβλιοθήκες για την χρήση και τον χειρισμό τους παρέχονται με την σειρά τους από το λειτουργικό σύστημα.

Αποτελούνται από δύο μέρη τα οποία τα περιγράφουν, την διεύθυνση IP του υπολογιστή και τον αριθμό του port του άκρου που ανήκει. Το λειτουργικό σύστημα προωθεί τα δεδομένα των εισερχόμενων IP πακέτων στην εκάστοτε εφαρμογή, αφαιρώντας τα headers επικοινωνίας και ασφάλειας, που παρέχει το πρωτόκολλο επικοινωνίας, για την αναγνώριση του προορισμού τους, διαθέτοντάς τα προς επεξεργασία.

Επειδή χρησιμοποιούνται στο μοντέλο server / client, όταν ο server δεχτεί την σύνδεση ενός client δημιουργείται μια διεργασία – νήμα για την εξυπηρέτησή του. Ο αριθμός των διεργασιών αυτών περιορίζεται από τους επεξεργαστές που διαθέτει ο υπολογιστής. Έτσι, με το άνοιγμα τόσων socket όσοι είναι οι επεξεργαστές μπορούν να εξυπηρετούνται ταυτόχρονα τόσοι client. Ένα ακόμα χαρακτηριστικό των socket είναι το πρωτόκολλο επικοινωνίας για τα προγράμματα που εκτελούνται στο δίκτυο. Το πρωτόκολλο που χρησιμοποιήθηκε εδώ είναι το UDP (User Datagram Protocol). Σαν client είναι ο υπολογιστής που αποστέλλει το αρχείο video και προβάλλει το επεξεργασμένο και σαν server αυτός που δέχεται το αρχείο για να το επεξεργαστεί και να το επιστρέψει.

Το UDP είναι ένα από τα κύρια πρωτόκολλα επικοινωνίας υπολογιστών σε δίκτυο. Χρησιμοποιεί την επικοινωνία χωρίς σύνδεση, πράγμα που σημαίνει πως η

μετάδοση των πακέτων – δεδομένων γίνεται μεμονωμένα, αφού στο καθένα παρέχονται οι ακριβείς πληροφορίες για τον προορισμό του μέσα στο δίκτυο και δεν ορίζονται μια φορά για όλα. Τα πακέτα αυτά καλούνται Datagrams. Επιπλέον, αυτά στέλνονται χωρίς να εξασφαλίζεται πως ο παραλήπτης είναι έτοιμος να τα δεχτεί. Η επιλογή αυτού του πρωτοκόλλου έγινε διότι ο φόρτος επικοινωνίας είναι μικρός και επειδή επιτρέπει το λεγόμενο broadcast. Το τελευταίο ορίζεται ως η δυνατότητα μετάδοσης όλων των δεδομένων προς αποστολή σε μια συγκεκριμένη χρονική στιγμή σε όλους τους διαθέσιμους παραλήπτες., Ο περιορισμένος φόρτος επικοινωνίας, οφείλεται στο ότι δεν υπάρχουν μηχανισμοί που να αποτρέπουν την απώλεια δεδομένων, το αν παραλήφθηκε όντως το μήνυμα στον server ή αν έγινε με λάθος σειρά η αποστολή του πακέτου. Το λεγόμενο, δηλαδή, handshaking που αποτελεί τους κανόνες με τους οποίους θα γίνει η επικοινωνία των υπολογιστών απουσιάζει στο πρωτόκολλο UDP. Παρ' όλα αυτά όμως παρέχει μηχανισμούς για την ακεραιότητα των δεδομένων όπως και για τον εντοπισμό λάθους σε αυτά.

Το header του UDP αποτελείται από τέσσερα πεδία, που το καθένα είναι μεγέθους δύο bytes. Συγκεκριμένα, το πρώτο είναι ο αριθμός του port του αποστολέα, όταν αυτό έχει νόημα ή χρειάζεται απάντηση από τον παραλήπτη. Διαφορετικά η τιμή του είναι μηδέν. Δεύτερο είναι, αριθμός port του προορισμού. Ακολουθεί το μήκος, σε bytes, του header και των δεδομένων. Το μέγιστο μέγεθος είναι το 65,535 και ειδικότερα τα οχτώ απ' αυτά είναι για το header και τα υπόλοιπα για τα δεδομένα. Έπειτα ο υπόλοιπος χώρος που απομένει είναι για την συνάρτηση που παρέχει την ακεραιότητα των δεδομένων, που αναφέρθηκε προηγουμένως.

3. ΨΗΦΙΑΚΗ ΕΙΚΟΝΑ

Η ψηφιακή εικόνα στους υπολογιστές αναπαρίσταται με την βοήθεια των pixels (εικονοστοιχεία). Κάθε pixel αποτελεί κι ένα σημείο της εικόνας και το πλήθος τους είναι τόσο, όσο και η ανάλυση της εικόνας (γινόμενο γραμμών επί στήλες). Ακόμη, τα χρώματα που μπορούν να αναπαρασταθούν από κάθε pixel εξαρτώνται από τον αριθμό των bits ανά pixel (bpp = bits per pixel). Έτσι, για παράδειγμα, μια δυαδική εικόνα σημαίνει πως χρειάζεται δύο χρώματα για την απεικόνισή της οπότε και 1 bit per pixel (bpp) αφού $2^1 = 2$ χρώματα. Επομένως, για την δυαδική εικόνα έχουμε τις τιμές μηδέν και ένα, όπου, αντίστοιχα, το μεν αντιπροσωπεύει το μαύρο και το δε το λευκό. Τυπική τιμή του πλήθους των bpp είναι το 8, για το οποίο η μέγιστη μη προσημασμένη τιμή ενός pixel είναι η 255 ($2^8 = 256$) που χρησιμοποιείται για το λευκό και η ελάχιστη το 0 που είναι για το μαύρο.

Για την αναπαράσταση των χρωμάτων , χρησιμοποιούνται ακόμα και διάφορα χρωματικά μοντέλα. Το απλούστερο μοντέλο – RGB- χρησιμοποιεί N bits για την αναπαράσταση των τιμών κάθε ενός από τα τρία βασικά χρώματα συνήθως κόκκινο (R), στο πράσινο (G) και στο μπλε(B), . Οι συνδυασμοί RGB των pixel σε κάθε θέση του πίνακα δίνουν την πληθώρα των χρωμάτων που αποτυπώνονται στην εικόνα που βλέπουμε. Αν και τελευταία τα χρώματα μπορούν να απεικονιστούν και με παραπάνω από οχτώ bpp. Τα παραπάνω σημαίνουν πως η ανάλυση της εικόνας αποτυπώνεται στον υπολογιστή μέσω ενός δισδιάστατου πίνακα, ενώ αν υπάρχουν και χρωματικά μοντέλα ο πίνακας γίνεται τρισδιάστατος (3 προαναφερθέντες δισδιάστατοι πίνακες). Σε κάθε θέση δηλαδή έχουμε μια τιμή για το κόκκινο, μια για το πράσινο και μια για το μπλε. Σαν αρχικό σημείο της εικόνας λαμβάνεται η θέση (0,0) του πίνακα που είναι και το πάνω αριστερό μέρος της εικόνας.

4. ΠΑΡΑΛΛΗΛΟΠΟΙΗΣΗ ΕΠΕΞΕΡΓΑΣΙΑΣ ΕΙΚΟΝΑΣ

1. ΠΕΡΙΓΡΑΦΗ ΕΝΝΟΙΑΣ

Η παραλληλοποίηση της επεξεργασίας εικόνας είναι μια έννοια που περιλαμβάνει την διάσπασή της αρχικής σε υπό – εικόνες και την επιθυμητή επεξεργασία που εφαρμόζεται στα κομμάτια αυτά. Η διάσπαση γίνεται εφικτή χωρίζοντας την εικόνα είτε διαιρώντας το πλάτος, είτε το ύψος της, είτε και τα δύο. Συνηθίζεται, ο διαχωρισμός αυτός να γίνεται μέσω των δυνάμεων του δύο αφού και οι περισσότερες αναλύσεις διαιρούνται τέλεια με τον αριθμό αυτό.

Σε υπολογιστικό επίπεδο μπορούμε να πούμε πως κάθε επεξεργαστής αναλαμβάνει μια από αυτές τις υπό – εικόνες. Οπότε, έχουμε τα στάδια της διαμοίρασης και της συλλογής. Το κύριο πρόγραμμα τις διανέμει στον κάθε επεξεργαστή, που έχει αναλάβει να πραγματοποιήσει μια συγκεκριμένη διεργασία, και στην συνέχεια, όταν αυτή τελειώνει, επιστρέφει τα αποτελέσματα πίσω. Προφανώς, όταν χρησιμοποιείται ένας μόνο υπολογιστής υπάρχει περιορισμένος αριθμός επεξεργαστών και συνεπώς αριθμός αλγορίθμων που μπορούν να εφαρμοστούν στα αρχικά δεδομένα.

2. ΣΧΕΤΙΚΕΣ ΑΝΑΦΟΡΕΣ

Κάποιες σχετικές εφαρμογές που χρησιμοποιούν αυτόν τον τρόπο για την επιτάχυνση των διαδικασιών, σε επίπεδο εικόνας παρουσιάζονται παρακάτω:

Pipeline:

Σκοπός αυτής της έρευνας [7] είναι η επιτάχυνση της διαδικασίας ανίχνευσης ακμών και του median φίλτρου. Η τεχνική που χρησιμοποιείται, διασπά την εικόνα σε δύο κομμάτια και εφαρμόζει τις δυο παραπάνω διαδικασίες ταυτόχρονα.

Η μάσκα φίλτρου που χρησιμοποιείται είναι μεγέθους 3x3. Για τον λόγο αυτό, αντί για μέση τιμή από 9 pixel γύρω από το κεντρικό, η μέση τιμή εξάγεται από τα τρία pixel κάθε γραμμής κι έπειτα η τελική προκύπτει από αυτές τις τρεις. Στην άλλη περίπτωση, για την ανίχνευση ακμών, χρησιμοποιούνται οχτώ 3x3 πίνακες συνέλιξης και για το κεντρικό pixel προκύπτει η μέγιστη από τις οχτώ που το περιβάλλουν. Ωστόσο, παρουσιάζονται δύο στοιχεία, Gx και Gy, για την κατακόρυφη και την οριζόντια ακμή αντίστοιχα. Το απόλυτο άθροισμα αυτών αναδεικνύει την ένταση της κλίσης. Αυτοί οι πίνακες, λοιπόν, μπορούν να εφαρμοστούν χωριστά στην εικόνα για παραγωγή ξέχωρων μετρήσεων για τις κλίσεις σε κάθε προσανατολισμό (Gx, Gy). Συνδυαστικά, βρίσκεται το απόλυτο μέγεθος και η κατεύθυνση της κλίσης σε κάθε σημείο.

Επομένως στο κάθε κομμάτι της εικόνας που διασπάστηκε εφαρμόζονται ταυτόχρονα οι δύο αυτοί αλγόριθμοι, με τις τροποποιήσεις τους. Ο συνδυασμός των αποτελεσμάτων τους παράγει την τελική πληροφορία που θέλουμε να εξάγουμε για την εικόνα.

Δίκτυο:

Η παρούσα έρευνα [8] χρησιμοποιεί τους υπολογισμούς μιας περιοχής γειτόνων για την εξαγωγή συμπερασμάτων και συγκεκριμένα την συνέλιξη. Το αποτέλεσμα που προκύπτει από τις πράξεις αυτές είναι μια συνάρτηση του τρέχοντος pixel, καθώς και των γειτονικών, από 2x2 μέχρι 21x21 pixels. Η διαδικασία υπολογισμού μπορεί να γίνει σε διαφορετικά σημεία της εικόνας, ταυτόχρονα. Για να υπολογιστεί, όμως, η συνέλιξη στις υπό – εικόνες χρειάζονται τιμές από τα όρια τους και αυτές βρίσκονται μοιρασμένες στους διαθέσιμους υπολογιστές. Έτσι, προκύπτει παραπάνω φόρτος αφού χρειάζεται η επικοινωνία μεταξύ των υπολογιστών για ανταλλαγή τιμών pixel, που εξαρτάται από τα μεγέθη των ορίων των υπό – εικόνων καθώς και του πίνακα συνέλιξης. Η τεχνική που επιλέχθηκε για το

πρόβλημα αυτό ονομάζεται overlap mapping. Υποδηλώνει πως σε κάθε υπολογιστή θα παρέχεται η υπό – εικόνα μαζί με τα αναγκαία οριακά pixel από τις άλλες, για να επιτευχθεί ο υπολογισμός. Η εικόνα διαμοιράζεται κατάλληλα με χρήση μιας ευριστικής μεθόδου σε σχέση με τον αριθμό των διαθέσιμων υπολογιστών. Συνοπτικά, προκύπτουν ακαθόριστου σχήματος τομείς της εικόνας που έχουν, όμως, τον ίδιο αριθμό pixel.

Το μοντέλο παραλληλοποίησης που χρησιμοποιείται είναι αυτό της χρήσης υπολογιστών ενός δικτύου και δίνει, τελικά, την δυνατότητα του υπολογισμού του χρόνου των ακόλουθων φάσεων:

Ta = χρόνος που χρειάζεται για το σπάσιμο της εικόνας με την ευριστική μέθοδο στον αποστολέα

Tb = χρόνος επικοινωνίας για αποστολή όλων των υπό – εικόνων και του πίνακα συνέλιξης στους διαθέσιμους υπολογιστές.

Tc = μέσος χρόνος υπολογισμού της συνέλιξης σε έναν υπολογιστή.

Td = χρόνος συλλογής αποτελεσμάτων από έναν υπολογιστή.

Έτσι, αυτές οι τέσσερις φάσεις δείχνουν με σχετική ακρίβεια την λειτουργικότητα και την επιτάχυνση όσο αλλάζει ο αριθμός των υπολογιστών

Ο χρόνος εκτέλεσης του παράλληλου προγράμματος q είναι το άθροισμα των τεσσάρων παραπάνω. Συμπεραίνουμε πως όσο ο αριθμός των διαθέσιμων υπολογιστών αυξάνεται, το Tc και το Td μπορεί να ελαττώνονται και το Tb θα αυξάνεται. Σε περίπτωση που ο χρόνος υπολογισμού είναι αρκετά μικρός, μερικοί κόμβοι (υπολογιστές του δικτύου) θα στείλουν τα αποτελέσματα τους, προτού γίνει η διανομή των υπό – εικόνων παντού. Διαμορφώνεται, έτσι, ο χρόνος ως T = Ta + Tb + nxTd, με n ο αριθμός των διαθέσιμων υπολογιστών στο δίκτυο.

Η έρευνα αυτή παρουσιάζει έναν αποτελεσματικό τρόπο για την διανομή των υπό – εικόνων για την αποφυγή περιττού φόρτου επικοινωνίας. Συμπεριλαμβάνεται, επίσης και ένας ακριβής αριθμός υπολογιστών προς χρήση για το καλύτερο δυνατό αποτέλεσμα στην επιτάχυνση. Ωστόσο, υπάρχουν και παράγοντες που επηρεάζουν το διαθέσιμο, λόγου χάρη,

bandwidth του δικτύου, σχετικό με την γρήγορη διανομή των εικόνων στους υπολογιστές, που δύσκολα μπορεί να υπολογίσει κανείς.

Πολύ – επεξεργαστικοί Υπολογιστές:

Η κύρια ιδέα που παρουσιάζεται και στην έρευνα αυτή [9] για την επεξεργασία της εικόνας είναι η διάσπαση του προβλήματος σε μικρότερα και η επίλυση αυτών ταυτόχρονα, ώστε ο συνολικός χρόνος να μπορεί να διανέμεται βάσει των εργασιών. Η παράλληλη επεξεργασία εικόνας δεν μπορεί να εφαρμοστεί σε όλα τα σχετικά προβλήματα, γι' αυτό κι εδώ εισάγονται χαρακτηριστικά που απαιτούνται για μια σωστή και αποτελεσματική εκτέλεση. Τα χαρακτηριστικά αυτά είναι:

Διακριτότητα – Ο αριθμός των βασικών μονάδων του προβλήματος και ο διαχωρισμός σαν: *coarse-grained*, όπου έχουμε λίγες λειτουργίες με βαρείς υπολογισμούς και *fine-grain*, όπου έχουμε μεγάλο αριθμό μικρότερων υπό - μονάδων και λιγότερο βαρείς υπολογισμούς.

Είδος παράλληλης επεξεργασίας – *Explicit*: ο αλγόριθμος περιέχει οδηγίες που συγκεκριμενοποιούν ποιες διεργασίες θα φτιαχτούν και θα εκτελεστούν παράλληλα και *Implicit*: ο compiler αναλαμβάνει το έργο της εισαγωγής των απαραίτητων οδηγιών για την εκτέλεση του προγράμματος στον υπολογιστή.

Συγχρονισμός – Αποκλείει την επικάλυψη δύο ή περισσότερων διεργασιών μεταξύ τους.

Καθυστέρηση – Ο χρόνος μεταβίβασης της πληροφορίας από τον αποστολέα στον δέκτη.

Κλιμάκωση – Η ικανότητα του αλγορίθμου να διατηρεί την επίδοσή του, καθώς αυξάνεται ο αριθμός των επεξεργαστών και το μέγεθος του προβλήματος στην ίδια αναλογία.

Το κύριο σημείο στους αλγόριθμους που χρησιμοποιήθηκαν είναι ο καθορισμός των κομματιών που χρειάζεται η αρχική εικόνα για την διάσπαση. Συσχετίζεται, δηλαδή, με τον αριθμό των thread. Με ένα thread ο υπολογισμός είναι διαδοχικός ενώ με παραπάνω η εικόνα χωρίζεται σε

διακριτές περιοχές. Η περιοχή που ανατίθεται σε κάθε thread το καθιστά υπεύθυνο για την διεξαγωγή των υπολογισμών, διατηρώντας ταυτόχρονα και τον συγχρονισμό μεταξύ των επεξεργαστών για αποφυγή deadlock. Ενδεικτικά, δύο από τους τέσσερις αλγορίθμους που εκτελέστηκαν παράλληλα είναι η εξίσωση ιστογράμματος και complex noise reduction χρησιμοποιώντας παράλληλους υπολογισμούς.

Για τον πρώτο, τα βήματα που ακολουθήθηκαν είναι τα εξής:

- Φόρτωση εικόνας στον client επεξεργαστή.
- Δημιουργία αντιγράφων της εικόνας στον client επεξεργαστή.
- Σύμφωνα με τις απαιτήσεις, απασχόληση σχετικού αριθμού επεξεργαστών.
- Αποστολή αντιγράφου σε διαφορετικούς επεξεργαστές.
- Κάθε επεξεργαστής αποτελείται από ξεχωριστά φίλτρα όπως median filter, Wiener filter, order statistical filter, min-max filter κτλ
- Συνέλιξη για εύρεση PSNR (Peak signal-to-noise ratio) και MSE (mean statistical error) της εικόνας σε κάθε επεξεργαστή.
- Αποστολή των τιμών PSNR, MSE στον client επεξεργαστή
- Σύγκριση των αποτελεσμάτων από όλους τους επεξεργαστή στον client επεξεργαστή.
- Προβολή της εικόνας με το μεγαλύτερο PSNR και χαμηλότερο MSE
- Αποδέσμευση των επεξεργαστών.

Για τον δεύτερο:

- Διαχωρισμός εικόνας είτε οριζόντια, είτε κάθετα και απασχόληση του κατάλληλου αριθμού επεξεργαστών.
- Δημιουργία του αθροιστικού ιστογράμματος (cumulative histogram) σε κάθε τμήμα της εικόνας, του κάθε επεξεργαστή.
- Κανονικοποίηση της τιμής που προκύπτει διαιρώντας με τον συνολικό αριθμό των pixels.

- Πολλαπλασιασμός των τιμών αυτών με την μέγιστη απόχρωση του γκρι και στρογγυλοποίηση αυτού που προκύπτει.
- Ένα προς ένα συσχέτιση της αρχικής τιμής με την τιμή του βήματος 3.
- Αποστολή του εξισωμένου ιστογράμματος στον client επεξεργαστή.

Διαφαίνεται, λοιπόν, στην έρευνα αυτή ότι όσον αφορά ακολουθιακούς υπολογισμούς ο διαχωρισμός και η εκτέλεση τους είναι εύκολη και έχει τα επιθυμητά αποτελέσματα. Ωστόσο, χρησιμοποιείται η αρχιτεκτονική ενός μόνο υπολογιστή.

5. ΤΜΗΜΑΤΟΠΟΙΗΣΗ ΕΙΚΟΝΑΣ

1. ΠΕΡΙΓΡΑΦΗ ΕΝΝΟΙΑΣ

Η τμηματοποίηση των ψηφιακών εικόνων είναι η διαδικασία που διαχωρίζει την εικόνα σε ομάδες pixel με τιμές που οριοθετούν την περιοχή που καταλαμβάνει ένα ή περισσότερα αντικείμενα ενδιαφέροντος., όπως ανατομικά όργανα του σώματος σε ιατρικές εικόνες. Τα pixel που ανήκουν στο ίδιο αντικείμενο έχουν κάποια κοινά ή κοντινά σε τιμή χαρακτηριστικά όπως το χρώμα, η φωτεινότητά τους και η υφή στην περιοχή.

Αποτέλεσμα αυτής της διαδικασίας, είναι η απομόνωση του αντικειμένου, μέσω της εξαγωγής μιας νέας εικόνας με διαφορετικό χρωματικό μοντέλο ή το βάθος χρώματος (bpp), όπως για παράδειγμα η δημιουργία μιας δυαδικής εικόνας από μια έγχρωμη. Κατ' επέκταση όταν έχουμε μια αλληλουχία εικόνων (video) μπορούμε να μελετήσουμε την κίνηση ενός αντικειμένου στον χώρο με χρήση στατιστικών μεθόδων ή αλγορίθμων επεξεργασίας εικόνων.

2. ΣΧΕΤΙΚΕΣ ΑΝΑΦΟΡΕΣ

Δεν θα μπορούσε να παραληφθεί και η διατύπωση μερικών αλγορίθμων σχετικά με την τμηματοποίηση. Από τους παρακάτω [3], [20], επιλέχθηκε ο τελευταίος για υλοποίηση στην παρούσα πτυχιακή εργασία. Ο σκοπός, παρ' όλα αυτά, ήταν η χρήση της τεχνικής αυτής για αναγνώριση κίνησης σε video.

Η πρώτη προσέγγιση έχει να κάνει με μια κατηγορία αυτών των αλγορίθμων που είναι η αφαίρεση του φόντου. Συγκεκριμένα, το αποτέλεσμα προκύπτει συγκρίνοντας το τρέχον frame με ένα frame αναφοράς που καλείται background image. Προκύπτει λογικά πως το τελευταίο δεν θα πρέπει να περιέχει κίνηση και συνεχώς να ενημερώνεται. Οι κυριότεροι αλγόριθμοι μοντελοποίησης υποβάθρου είναι οι: running Gaussian average, kernel density estimation, sequential kernel density approximation, co – occurrence of image backgrounds, mixture of Gaussians και eigenbackgrounds.

Ξεκινώντας, η τεχνική του running *Gaussian average* έχει σαν θέμα την ανεξάρτητη μοντελοποίηση του φόντου σε κάθε περιοχή pixel με χρήση της Gaussian συνάρτησης πυκνότητας πιθανότητας (pdf) στις τελευταίες n τιμές των pixel των τελευταίων frames. Για την προσαρμογή στον χρόνο των frames υπολογίζεται ο λεγόμενος running average, που λαμβάνει υπ' όψιν την τιμή του τρέχοντος frame και τον προηγούμενο μέσο όρο και μια βαρύτητα που υπολογίζεται εμπειρικά. Η τελευταία προσαρμόζεται στον μέσο όρο και στην τιμή. Τέλος, κάποιες επεκτάσεις αυτής της τεχνικής σχετίζονται και με άλλους χρωματικούς χώρους, όπως επίσης και με την πραγματικού χρόνου επεξεργασία, μειώνοντας τον υπολογιστικό φόρτο με την μέση τιμή και την τυπική απόκλιση να προκύπτουν πιο αραιά συγκριτικά με το frame rate.

Έπειτα, η επόμενη τεχνική χειρίζεται την προσέγγιση της συνάρτησης πυκνότητας πιθανότητας (pdf) του φόντου που προκύπτει κάνοντας χρήση του ιστογράμματος των πιο πρόσφατων τιμών, που έχουν κατηγοριοποιηθεί σαν τιμές του φόντου. Ο αριθμός των δειγμάτων, όμως, είναι περιορισμένος. Αυτό, πρακτικά, σημαίνει ότι η προσέγγιση μπορεί να παρέχει ανακριβή μοντελοποίηση της πραγματικής, άγνωστης pdf με τις τελευταίες τιμές της πραγματικής, συχνά να λείπουν και το ιστόγραμμα, επομένως, να μην ανταποκρίνεται κατάλληλα. Εδώ, χρησιμοποιώντας ένα μη – παραμετρικό μοντέλο που βασίζεται στο *kernel density estimation* (KDE) στον buffer των τελευταίων n τιμών, μπορεί να μοντελοποιηθεί η διανομή των τιμών φόντου. Το κέρδος σ' αυτό είναι μια συνεχής και με εξομάλυνση εκδοχή των ιστογραμμάτων που ακολουθούν. Ένα άθροισμα των Gaussian kernels παρέχει το pdf του φόντου, κεντραρισμένο στις n πιο πρόσφατες τιμές του. Η ενημέρωση του μοντέλου, τέλος, γίνεται αντικαθιστώντας απλώς τις τιμές φόντου του buffer με σειρά fifo(first in – first out).

Συνεχίζοντας με τους αλγορίθμους τμηματοποίησης, γίνεται αναφορά για την λεγόμενη *sequential kernel density approximation*. Υλοποιήσεις του mean – shift vector χρησιμοποιούνται αρκετά για προβλήματα σχετικά με την τμηματοποίηση εικόνας και την παρακολούθηση. Είναι μια αποτελεσματική τεχνική υπολογισμού της κλίσης ανάβασης που είναι ικανή να εντοπίζει την πραγματική pdf κατ' ευθείαν από το δείγμα δεδομένων. Ωστόσο, το υπολογιστικό κόστος είναι υψηλό, αφού αποτελεί μια επαναληπτική διαδικασία και χρειάζεται την μελέτη σύγκλισης σε ένα χώρο δεδομένων. Οπότε, δεν μπορεί να εφαρμοστεί στην pdf φόντου σε επίπεδο pixel.

Το mean – shift vector, στην συγκεκριμένη περίπτωση, χρησιμοποιείται μόνο για αρχικοποίηση του μοντέλου και ειδικότερα για εντοπισμό του pdf φόντου από το αρχικό δείγμα δεδομένων. Σ' αυτό το βήμα, το αρχικό set των Gaussian modes του pdf φόντου εντοπίζεται από ένα αρχικό set δειγμάτων. Σε πραγματικό χρόνο, τώρα, η ενημέρωση γίνεται παρέχοντας απλές ευριστικές μεθόδους αντιμετωπίζοντας έτσι την προσαρμογή, τη δημιουργία και την συγχώνευση αυτών των modes.

Σε επόμενο στάδιο έχουμε την χωρική αξιοποίηση του *co – occurrence of image variations* αλγόριθμο. Η ιδέα, εδώ, είναι πως τα γειτονικά block των pixel που ανήκουν στο φόντο θα πρέπει να επιδέχονται παρόμοιες μεταβολές με τον χρόνο. Αν κι αυτή η υπόθεση είναι αληθής για συγκεκριμένες περιπτώσεις, εν μέρει δεν ισχύει για τα block που περιέχουν τα όρια των διακριτών αντικειμένων του φόντου. Τα βήματα του αλγορίθμου αυτού συνοψίζονται στα εξής: όπως αναφέρθηκε, χρησιμοποιούνται N x N περιοχές pixel με ένα trade – off μεταξύ της ανάλυσης και της σταθερότητας, ταχύτητας. Έπειτα, για κάθε block ένας συγκεκριμένος αριθμός χρονικών δειγμάτων αποκτάται, υπολογίζοντας πρώτα τον μέσο όρο στο χρόνο και εν συνεχεία τις διαφορές μεταξύ αυτού και των δειγμάτων, που αποτελούν τα λεγόμενα image variations. Ο πίνακας συνδιασποράς υπολογίζεται σχετικά με τον μέσο όρο και ένας μετασχηματισμός ιδιοδιανύσματος εφαρμόζεται, ώστε να ελαττωθούν οι διαστάσεις των image variations. Ύστερα εφαρμόζονται κάποιες τεχνικές για την κατηγοριοποίηση του τρέχοντος block για το αν αυτό ανήκει ή όχι στο φόντο, με εφαρμογή της αθροιστικής πιθανότητας για οχτώ γειτονικά block.

Η αναφορά που θα υπάρξει στην συνέχεια είναι για τον αλγόριθμο του mixture of Gaussians. Συγκεκριμένα, εξετάστηκε η περίπτωση κατά την οποία ένα μοντέλο φόντου με πολλές τιμές θα μπορεί να αντιμετωπιστεί με πολλαπλά αντικείμενα. Η πιθανότητα σ' αυτή την περίπτωση της παρακολούθησης μιας συγκεκριμένης τιμής pixel στον χρόνο θα μπορεί να υπολογιστεί μέσω του mixture of Gaussians. Καθεμιά Gaussian κατανομή θα περιγράφει αποκλειστικά και μόνο είτε τα αντικείμενα που προβάλλονται στο προσκήνιο, είτε στο φόντο. Για να μπορέσει όμως το mixture να γίνει μέρος του φόντου θα πρέπει να υπάρξει κάποιο κριτήριο ώστε να διαχωρίζονται αυτές οι κατανομές. Υπάρχει μια εκτίμηση για την καθεμιά, βάσει του πλάτους κορυφής και της τυπικής απόκλισης που την συγκαταλέγει στις δυο κατηγορίες. Η υπόθεση είναι πως η μεγαλύτερη και πιο συμπαγής ανήκει κατά κύριο λόγο στο φόντο.

Το πρόβλημα που πρέπει να επιλυθεί σχετίζεται με το χρονικό πλαίσιο που «κινούνται» τα frames. Ταυτόχρονα, λοιπόν, θα πρέπει η κάθε τιμή του pixel που παρακολουθείται να εκχωρείται στην κατανομή που ταιριάζει και να υπολογίζονται οι παράμετροι του προσφάτως αλλαγμένου φόντου.

Προτελευταίος είναι αυτός των *Eigenbackgrounds*. Η προσέγγιση που παρουσιάζεται σ' αυτόν βασίζεται σε ανάλυση ιδιοτιμών σε ολόκληρη όμως την εικόνα. Κάνοντας χρήση ενός τόσο μεγάλου χώρου αποφεύγονται ανεπιθύμητα αποτελέσματα κατά τον διαχωρισμό των block.

Η πρώτη φάση ορίζει την απόκτηση ενός αριθμού frame και τον υπολογισμό του μέσου frame. Σ' αυτό, τώρα, γίνεται υπολογισμός του πίνακα συνδιασποράς και τα καλύτερα ιδιοδιανύσματα αποθηκεύονται σε έναν πίνακα. Η δεύτερη φάση, που αποτελεί την κατηγοριοποίηση, προβάλλει κατάλληλα κάθε καινούριο frame σε αντίστοιχο ιδιοχώρο. Το αποτέλεσμα αυτό χρησιμοποιείται για να επιστραφεί στον χώρο με τις υπόλοιπες, μέσω του αθροίσματος του μέσου frame που υπολογίστηκε στην πρώτη φάση και του γινομένου του προηγούμενου ιδιοχώρου με τον πίνακα ιδιοδιανυσμάτων. Τελικά, τα σημεία που ανήκουν στο προσκήνιο εντοπίζονται στις περιοχές όπου η απόλυτη τιμή της διαφοράς του τρέχοντος και καινούριου frame με την προβολή του στον χώρο των υπόλοιπων είναι μεγαλύτερη από μια τιμή κατωφλίου T.

Ο τελευταίος αλγόριθμος [20] ο οποίος και θα αναλυθεί περισσότερο, είναι αυτός που επιλέχθηκε για να υλοποιηθεί. Αποτελεί την δεύτερη και τελευταία προσέγγιση η οποία λύνει το πρόβλημα της ενημέρωσης του φόντου όταν σ' αυτό υπάρχει μια ξαφνική αλλαγή στον φωτισμό, πράγμα που δυσκολεύει τον εντοπισμό της κίνησης. Εδώ, λοιπόν, παρουσιάζεται μια μέθοδος εκτίμησης φωτισμού για τον καθορισμό δύο μορφών φόντου. Το ένα θα 'ναι το φωτεινό frame και το άλλο το σκοτεινό. Βασιζόμενο σ' αυτά, η δυαδική μάσκα (εικόνα) για το κινητό αντικείμενο προκύπτει μέσω μιας συνάρτησης κατωφλίου. Έχουν προταθεί αρκετές μέθοδοι για το πρόβλημα του φωτισμού, ωστόσο, αυτή που θεωρήθηκε κατάλληλη ήταν η background subtraction, διότι είναι υπολογιστικά χαμηλή και ακριβής στην αναγνώριση κίνησης.

Ο τρόπος που προτίθεται είναι της δημιουργίας τριών μερών με το πρώτο να περιέχει το φόντο, το δεύτερο την εκτίμηση του φωτισμού στο τρέχον frame και το τρίτο την αναγνώριση του αντικειμένου. Ξεκινώντας με την περιγραφή τους, εν αντιστοιχία, παρέχεται μια ολοκληρωμένη εικόνα της διαδικασίας καθώς και η αναγκαιότητα του διαχωρισμού.

Έχουμε, λοιπόν, αρχικά την μέση χρονική τιμή η οποία δεν μπορεί να προσαρμοστεί στον φωτισμό του επερχόμενου frame. Γι' αυτό τον λόγο χρησιμοποιείται ο υπολογισμός της μέσω του running average που περιλαμβάνει το τρέχον φόντο, το προηγούμενο και το τρέχον frame. Στα δύο τελευταία προστίθεται κι ένας παράγοντας που αντιπροσωπεύει μια προσαρμοστική παράμετρο.

Έπειτα, με χρήση της θεωρίας της εντροπίας , ορίζουμε σαν σκοτεινό frame αυτό που η εν λόγω τιμή του είναι χαμηλή και αντίθετα. Αυτό το χαρακτηριστικό είναι απαραίτητο για τον εντοπισμό στην αλλαγή του φωτισμού. Υπολογίζεται η pdf του frame, με μια τροποποίηση λαμβάνοντας υπ' όψιν και την ανάλυση του, και η εντροπία προκύπτει έχοντας σαν όριο την μέγιστη και ελάχιστη τιμή φωτισμού στο frame. Συγκρίνοντας, τελικά, την διαφορά της εντροπίας του τρέχοντος με του προηγουμένου με μια τιμή κατωφλίου, γίνεται αντιληπτό αν υπήρξε αλλαγή στον φωτισμό. Έτσι, αν υπήρξε, βάσει της μέσης τιμής του φωτισμού των δύο μορφών αυτές αντικαθίστανται κατάλληλα και χρησιμοποιούνται σαν αναφορά για την αλλαγή του φωτισμού στο τρέχον φόντο.

Το τρίτο μέρος που αφορά την αναγνώριση της κίνησης του αντικειμένου είναι απόρροια του παραπάνω υπολογισμού. Η απόλυτη διαφορά μεταξύ του εισερχόμενου frame και του φόντου που άλλαξε δίνει τις τιμές στην δυαδική μάσκα της κίνησης. Αν η τιμή στη τρέχουσα θέση του pixel είναι μεγαλύτερη από ένα κατώφλι σε αντίστοιχο πίνακα, τότε στον δυαδικό πίνακα αποθηκεύεται η τιμή ένα, πράγμα που σημαίνει πως αυτή σηματοδοτεί την κίνηση, κι αντίθετα. Το κατώφλι στην συνέχεια αλλάζει είτε με αύξηση κατά ένα αν η απόλυτη διαφορά είναι μεγαλύτερη από την τιμή κατωφλίου και με μείωση όταν γίνεται το άλλο.

Ο αλγόριθμος αυτός δεν τηρήθηκε κατά γράμμα αφού υπήρξαν μικροαλλαγές στην τιμή κατωφλίου και στη διάσταση των πινάκων που συγκρίνονται, για λόγους βελτιστοποίησης. Αποτέλεσμα αυτού είναι η αποτύπωση με λευκό και μαύρο της κίνησης και του φόντου, αντίστοιχα, αφού προβάλλεται στον client η δυαδική μάσκα.

Κεφάλαιο 2 Αρχιτεκτονική Εφαρμογής

1. ΕΠΙΣΚΟΠΗΣΗ ΕΦΑΡΜΟΓΗΣ

Στην διάθεσή μας υπήρχαν ετερογενή υπολογιστικά συστήματα με διαφορετική αρχιτεκτονική το καθένα. Ωστόσο, χρησιμοποιώντας το πολύ δύο εκτελέσεις του ίδιο αλγορίθμου, ανάλογα τις απαιτήσεις σε αριθμό υπό – frames, έγινε εκμετάλλευση της πολύ – επεξεργαστικής αρχιτεκτονικής του κάθε υπολογιστή. Το αποτέλεσμα παραμένει ντετερμινιστικό. Χρησιμοποιήθηκε ο παραλληλισμός της εικόνας, αφού κάθε frame χωριζόταν σε υπό – frames, με την ίδια διαδικασία να εφαρμόζεται στα αντίστοιχα δεδομένα. Το τελευταίο, μάλιστα, υποδηλώνει πως υπήρχε και καταμερισμός της εργασίας.

Αξιοποιήθηκαν μέχρι τρεις διαφορετικοί servers (υπολογιστικά συστήματα που εκτελούν την τμηματοποίηση) και ένας client που συνδεόταν με WiFi με τους υπόλοιπους, που ήταν και ο αποστολέας του video. Η επικοινωνία έγινε μέσω τοπικού δικτύου και χρησιμοποιήθηκαν ένα, δύο, τέσσερα και έξι sockets. Λόγω του περιορισμένου αριθμού διαθέσιμων υπολογιστών, δεν γινόταν να επεκταθούμε στα οχτώ. Το *Processing* χρησιμοποιήθηκε αντί του *OpenCV*, γιατί το τελευταίο δεν διάβαζε τα video από το path σε λειτουργικό Windows.

Οι αλλαγές στις διαστάσεις για την επεξεργασία του/των frame/υπό – frames, γίνονταν χειροκίνητα για κάθε περίπτωση, σύμφωνα με τον αριθμό των servers. Αν το frame χωριζόταν στα δύο τότε το «σπάσιμο» γινόταν χωρίζοντας το ύψος ή το πλάτος και στέλνοντας τα υπό frames στον/στους server. Δεν χρησιμοποιήθηκαν local ports, ακόμα και στην περίπτωση που είχαμε ολόκληρο το frame προς επεξεργασία. Αντίστοιχα, με τέσσερις και έξι servers είχαμε χωρισμό του ύψους και του πλάτους κατά δύο και κατά τρία του ύψους και δύο του πλάτους. Οι διαστάσεις βοηθούσαν να γίνει η τελευταία διαίρεση, αλλά προτιμότερο είναι να γίνεται σύμφωνα με τις δυνάμεις του δύο.

2. ΠΕΡΙΓΡΑΦΗ ΥΛΟΠΟΙΗΣΗΣ

Για την υλοποίησης της εργασίας έγινε χρήση του προγραμματιστικού περιβάλλοντος *Processing*, που αποτελεί open source εργαλείο σε γλώσσα *Java* για την επεξεργασία video, εικόνων και γραφικών. Το πρωτόκολλο επικοινωνίας που επιλέχθηκε για την μεταφορά δεδομένων μέσω του δικτύου ήταν το *UDP*. Φάνηκε ιδανικό διότι απουσιάζουν από αυτό έλεγχοι που αφορούν την απώλεια πακέτων (handshaking), μειώνοντας έτσι τον φόρτο επικοινωνίας. Αυτό χρησιμεύει διότι σε

video streaming υπάρχει μια σχετική ανοχή στις απώλειες και σκοπός ήταν η επεξεργασία να γίνει όσο το δυνατόν πιο κοντά σε πραγματικό χρόνο. Στην περίπτωση του video streaming, το οποίο είναι μια χρονικά ευαίσθητη εργασία είναι προτιμότερο να χαθούν πακέτα δεδομένων, παρά να αναλώνεται χρόνος αναμονής για την λήψη αυτών.

Το *Processing*, ξεκινώντας, έχει δύο κύριες συναρτήσεις, την `setup` και την `draw`. Η `setup`, όντας συνάρτηση αρχικοποίησης, καλείται μία φορά στην αρχή, ενώ η `draw` για κάθε frame που είναι να προβληθεί. Το τμηματοποιημένο frame (η δυαδική μάσκα στην προκειμένη περίπτωση), με την σειρά του, είναι ειδικού τύπου που παρέχεται από το *Processing*, ονόματι `PImage`, και διαθέτει κάποιες διευκολύνσεις στον χειρισμό των pixel. Αυτά αποθηκεύονται σαν διάνυσμα στον πίνακα `pixels`.

Στην μεριά του *client*, στο κομμάτι που διαβάζουμε το video και το στέλνουμε για επεξεργασία, στην αντίστοιχη `setup` δημιουργήθηκε και το `Datagram Socket` στο οποίο θα λαμβάνουμε πίσω τη δυαδική μάσκα και στην `draw` θα αποτυπώνεται το κάθε επεξεργασμένο frame. Για το διάβασμα του video, την εκκίνησή του, την μη επανάληψή του και την σίγαση του ήχου, χρησιμοποιήθηκαν εντολές από το *Processing*. Μια επιπλέον συνάρτηση, η **movieEvent**(*Processing)*, καλείται κάθε φορά που διαβάζεται ένα frame και είναι αυτή που το στέλνει για επεξεργασία. Η αποστολή έγινε μέσω μιας βοηθητικής συνάρτησης, η οποία είναι υπεύθυνη και για την διάσπαση του frame στον κατάλληλο αριθμό υπό – frames.

Σ' αυτήν δημιουργούμε μια `BufferedImage` τύπου εικόνα, που αποθηκεύει το frame. Η διαδικασία αυτή γίνεται για να μπορέσει να πραγματοποιηθεί η αποστολή μέσω του πρωτοκόλλου UDP, που επιλέξαμε. Επειδή, όμως, χρειάζεται να αποστέλλεται για την επικοινωνία πίνακας byte, δημιουργούμε ένα `BufferedOutputStream` στο οποίο θα βρίσκεται ένα `ByteArrayOutputStream`, για την αναγκαία μετατροπή. Επιπλέον, γίνεται η μετατροπή σε **.jpg** της `BufferedImage` εικόνας και τοποθετείται στο `ByteArrayOutputStream`, κάνοντας και την απαραίτητη συμπίεση, αφού το μέγεθος που μπορεί να σταλεί και να ληφθεί μέσω του πρωτοκόλλου επικοινωνίας είναι περιορισμένο. Τέλος, στέλνουμε τα δεδομένα .jpg μέσω του `Datagram Socket` στον/στους server/servers, τα οποία βρίσκονται στον πίνακα byte και αποτελούν το πακέτο. Όλα αυτά είναι υλοποιημένα στην γλώσσα *Java*, με μόνη διαφορά την φόρτωση των pixel σε `BufferedImage` και σε `PImage` μέσω των συναρτήσεων που υπάρχουν στο *Processing*.

Στην μεριά της επεξεργασίας, δηλαδή στον *server*, ορίζουμε αρχικά το μέγεθος του buffer με τον πίνακα byte που θα ληφθεί με μέγιστο το 65536, όπως και στον *client* για την λήψη του τμηματοποιημένου frame. Υπάρχει κι εδώ η συνάρτηση `setup` όπου με τον ίδιο τρόπο δημιουργούμε το `Datagram Socket`, που θα λαμβάνει τα frame προς επεξεργασία κι έπειτα, στην συνάρτηση `draw`, γίνεται η τμηματοποίηση, διότι αυτή διαχειρίζεται κάθε frame/second. Καθώς η επεξεργασία τελειώνει, στο κομμάτι αυτό περιέχεται και η αποστολή πίσω στον *client*.

Αρχικά, μέσω μιας συνάρτησης γίνεται έλεγχος αν υπάρχει frame για να διαβαστεί από το socket. Εκεί, δημιουργούμε ένα `Datagram Packet` που θα λαμβάνει τα δεδομένα από το `Datagram Socket`. Το μέγεθός του θα πρέπει να 'ναι μικρότερο ή ίσο με αυτό του `buffer`. Σε έναν πίνακα byte μεταφέρονται τα δεδομένα που ελήφθησαν στο socket για να μπορέσει αργότερα να μετατραπεί σε εικόνα `PImage`. Για τον λόγο αυτό δημιουργείται ένα `ByteArrayInputStream` και φτιάχνοντας μια `BufferedImage` εικόνα τα διαβάζουμε από το πρώτο, τα αποθηκεύουμε σ' αυτήν και αργότερα μέσω της `getRGB` γίνεται η τοποθέτηση των pixel στην `PImage` που μπορεί να επεξεργαστεί από το *Processing*. Η τελευταία θα χρησιμοποιηθεί στην `draw` για την τμηματοποίηση.

Για το segmentation, χρησιμοποιήθηκαν κυρίως εντολές από το *Processing*. Κάθε φορά που καλείται η `draw`, δηλαδή έχουμε frame για προβολή (συνεπώς επεξεργασία), υπάρχει μια μεταβλητή **frameCount** που αυξάνεται. Στην πρώτη φορά που έχουμε κλήση, μετατρέπουμε το πρώτο frame σε grayscale ώστε να αποθηκευθεί σαν background για την τμηματοποίηση, Κάνοντας χρήση τριών πινάκων, δηλαδή το τρέχον, το φωτεινό και το σκοτεινό φόντο εφαρμόζουμε το πρώτο μέρος του αλγορίθμου τμηματοποίησης.

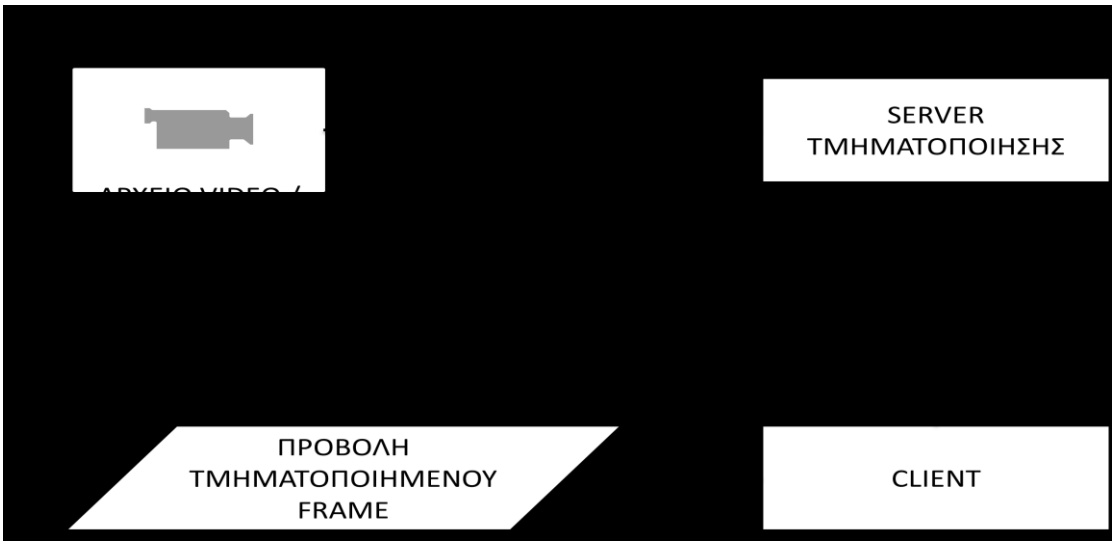
Όταν λάβουμε το δεύτερο frame και μετά, δημιουργούμε δυο πίνακες που θα περιλαμβάνουν την grayscale μορφή του και την διαφορά σε σχέση με το προηγούμενο, αντιστοίχως. Εδώ έγινε η διαδικασία που περιγράφηκε στο τρίτο μέρος του αλγορίθμου που υλοποιήθηκε. Η δυαδική μάσκα αποστέλλεται στην συνέχεια πίσω στον client για την προβολή κι αυτό γίνεται σαν μια συνεχόμενη ροή μέσω των socket.

Η παραπάνω περιγραφή αφορούσε την περίπτωση που μεταφέραμε και λαμβάναμε ολόκληρο το frame. Ωστόσο, όπως αναφέρθηκε, έγιναν δοκιμές διαχωρίζοντάς το σε υπό – frames, που αποθηκεύτηκαν σε πίνακα. Η αποστολή και η λήψη έγινε με τον ίδιο τρόπο, απλώς χρειάστηκε χειροκίνητα στον/στους server να ορίσουμε τα μεγέθη που πρόκειται να ληφθούν και να αποσταλούν. Το κάθε

τμηματοποιημένο υπό – frame αποθηκεύθηκε στην αντίστοιχη θέση του ολικού που πρέπει να συμπληρωθεί με όλα τα αποτελέσματα. Όπως για παράδειγμα, το πάνω μισό που στάλθηκε στον *server*, έγινε το πάνω μισό του frame στον *client* που περιμένει τα αποτελέσματα για προβολή. Αυτή η διαδικασία επαναλήφθηκε για όσα κομμάτια δημιουργήθηκαν. Παρομοίως για την περίπτωση με την camera.

Χρησιμοποιήθηκαν μονοδιάστατοι πίνακες για ευκολία και για ταχύτητα, αφού και ο πίνακας με τα pixel των `PImage` είναι διάνυσμα. Κάθε φορά που είναι να διαμορφωθεί η `PImage` πρέπει πρώτα να φορτώνονται τα pixel της μέσω συνάρτησης του *Processing*, να γίνεται η επεξεργασία και ύστερα τα pixel να ενημερωθούν για την αλλαγή.

Το Σχήμα 1 παρουσιάζει μια γενική εικόνα της διαδικασίας. Ξεκινώντας έχουμε το, αποθηκευμένο στον client, αρχείο video ή το video της λήψης από camera (πραγματικός χρόνος). Αυτό αποστέλλεται στον server τμηματοποίησης που περιλαμβάνει κι άλλους servers, ανάλογα τα υπό – frames που δημιουργήθηκαν. Γίνεται η επεξεργασία της τμηματοποίησης σ' αυτό που έλαβε ο server και μόλις τελειώσει επιστρέφει στον client για την προβολή. Όσα υπό – frames και να υπάρχουν, ένας client είναι αυτός που τα λαμβάνει.

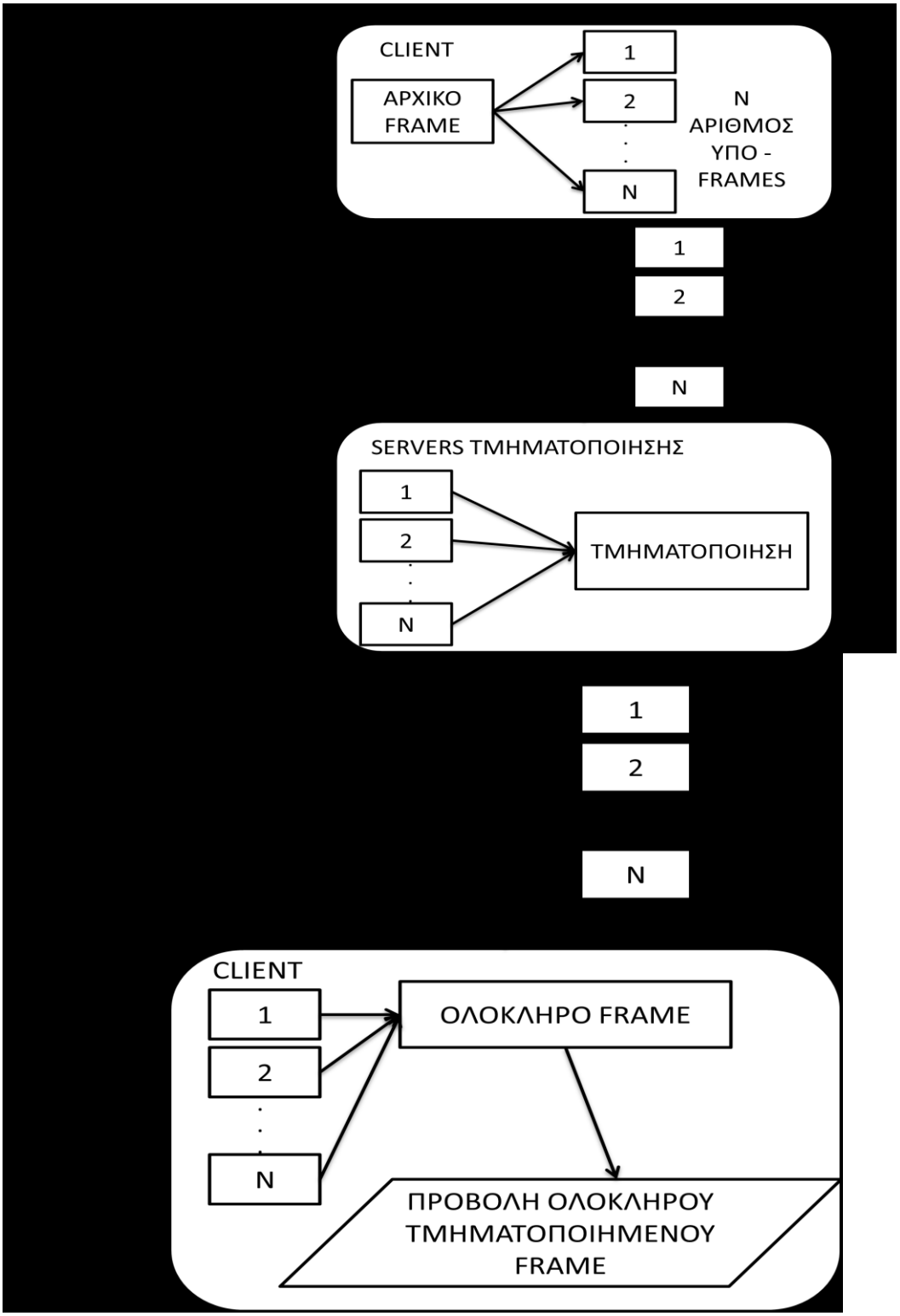


Σχήμα 1 – Τα βήματα της διαδικασίας για κάθε frame με χρήση ενός client και ενός server.

Το Σχήμα 2 δείχνει την περίπτωση στην οποία δημιουργούμε Ν υπό – frames. Ο αριθμός αυτός δηλώνει και τον αριθμό των servers τμηματοποίησης που θα χρειαστεί, ο οποίος είναι φυσικά γνωστός εκ των προτέρων. Έτσι, πραγματοποιείται η διαδικασία της διάσπασης και της διαμοίρασης από τον client. Τα υπό – frames φτάνουν σε ξεχωριστούς servers, που εκτελούν την ίδια διαδικασία.

Όταν τελειώσει η διαδικασία ο client είναι έτοιμος να δεχτεί όλα τα αποτελέσματα γι' αυτό και ανοίγει Ν sockets. Επιλέγοντας ποιο κομμάτι του αρχικού frame στάλθηκε σε ποιον server, η σύνθεση του ολικού είναι ένα γέμισμα μιας νέας εικόνας, ίδιων διαστάσεων με το αρχικό, στις αντίστοιχες θέσεις. Μόλις τελειώσει αυτό έχουμε την προβολή του αποτελέσματος της τμηματοποίησης.

Τα sockets, τα οποία άνοιξαν, διατηρούνται καθ' όλη την διάρκεια που έχουμε κάποιο διαθέσιμο frame, είτε αυτό είναι από την camera, είτε από το αποθηκευμένο video. Επιτυγχάνεται μια συνεχής ροή του video στο μοντέλο client/server που χρησιμοποιείται.

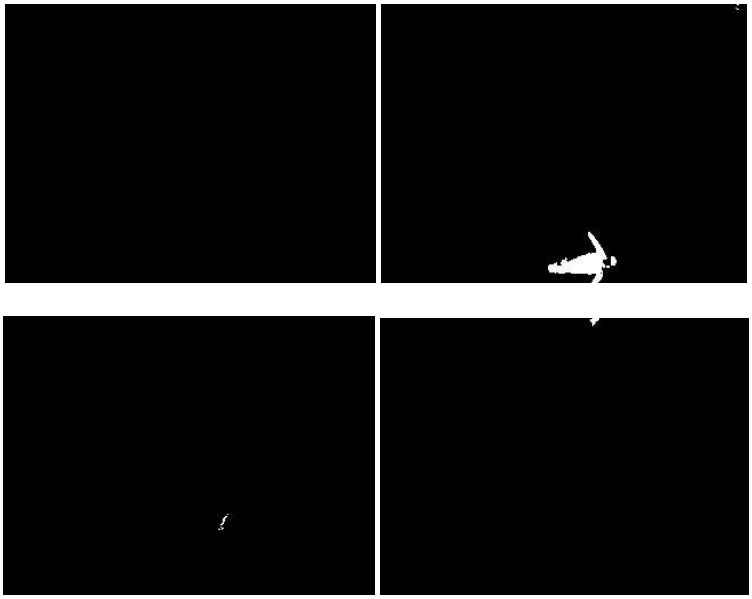


Σχήμα 2 – Τα βήματα της διαδικασίας για κάθε frame με τη χρήση N servers και η επιστροφή πίσω στον client.

Το Σχήμα 3, το Σχήμα 4, το Σχήμα 5 και το Σχήμα 6 παρουσιάζουν την περίπτωση που ο αριθμός N από το Σχήμα 2 είναι τέσσερα και έχουμε video από αρχείο. Ξεκινώντας (Σχήμα 3), έχουμε το αρχικό frame που διαβάζεται. Έπειτα, την διαδικασία της διάσπασης (Σχήμα 4) σε ανάλογο αριθμό υπό – frames. Οι διαστάσεις βοηθούν στο να γίνει διαχωρισμός βάσει ύψους και πλάτους. Αυτά αποστέλλονται σε τέσσερις servers τμηματοποίησης, για να εφαρμοστεί η εν λόγω επεξεργασία (Σχήμα 5). Το πάνω αριστερά, για παράδειγμα πηγαίνει σε συγκεκριμένο server κι από το αποτέλεσμα αυτού, συμπληρώνεται το πάνω αριστερά κομμάτι της εικόνας στον client που φαίνεται στο Σχήμα 6 κ.ο.κ.

Σχήμα 3 – Το frame που διαβάζεται στον client από το αρχείο video.

Σχήμα 4 – Η διάσπαση του frame σε κομμάτια και η λήψη αυτών στους τέσσερις servers τμηματοποίησης.



Σχήμα 5 – Η εφαρμογή του αλγορίθμου τμηματοποίησης, ξεχωριστά, σε κάθενα από τους τέσσερις servers τμηματοποίησης και η παραγωγή των δυαδικών υπό - εικόνων για επιστροφή πίσω στον client.

Σχήμα 6 – Η σύνθεση και προβολή της τελικής δυαδικής εικόνας που απαρτίζεται από τα τέσσερα κομμάτια, αποτελέσματα των servers τμηματοποίησης.

3. ΠΕΡΙΓΡΑΦΗ ΕΦΑΡΜΟΓΗΣ ΜΕ ΧΡΗΣΗ ΨΕΥΔΟΚΩΔΙΚΑ

Client :

Συνάρτηση αρχικοποίησης **setup**

- Δημιουργία client socket
- Ορισμός παραθύρου προβολής
- Διάβασμα αρχείου video / Λήψη από camera

Συνάρτηση **movieEvent**

- Διάβασμα frame ως `PImage`
- Συνάρτηση **broadcast** για αποστολή frame στους servers

Συνάρτηση **broadcast**

- Δημιουργία `BufferedImage` με ανάλυση όση του frame και μεταφορά των pixel από την `PImage` του frame
- Δημιουργία `ByteArrayOutputStream`
- Δημιουργία `BufferedOutputStream` με παράμετρο τη μεταβλητή του `ByteArrayOutputStream` για την απαραίτητη μετατροπή
- Τοποθέτηση της `BufferedImage` στον `BufferedOutputStream` και συμπίεση σε **.jpg**
- Λήψη του πίνακα byte που θα στείλουμε μέσω του UDP
- Δημιουργία `DatagramPacket`
- Αποστολή πακέτου στον server τμηματοποίησης

Συνάρτηση προβολής **draw**

- Λήψη τμηματοποιημένου frame μέσω συνάρτησης **editedImage**
- Προβολή τμηματοποιημένου frame ως `PImage`

Συνάρτηση λήψης **editedImage**

- Δημιουργία `DatagramPacket` βάσει του `inputBuffer`
- Λήψη δεδομένων από το socket και αποθήκευση σε πίνακα byte
- Διάβασμα των δεδομένων μέσω `ByteArrayInputStream`
- Αποσυμπίεση του **.jpg** με δημιουργία νέου `BufferedImage`
- Τοποθέτηση των pixel του `BufferedImage` στην `PImage` για προβολή

Server :

Συνάρτηση αρχικοποίησης **setup**

- Δημιουργία server socket
- Ορισμός παραθύρου για προβολή του frame που λαμβάνεται από τον client

Συνάρτηση προβολής **draw**

- Συνάρτηση λήψης εικόνας **checkForImage**
- Δημιουργία βοηθητικών πινάκων grayscale, για την τμηματοποίηση, από το πρώτο frame
- Από το δεύτερο frame και έπειτα μετατροπή αυτών σε grayscale πίνακες
- Όσο είναι η διάσταση του frame, υπολογισμός διαφοράς στις τιμές των pixel του φόντου του προηγούμενου με το τρέχον και ενημέρωση τιμών
- Υπολογισμός μέσης τιμής στους τρεις πίνακες φόντου και αντικατάσταση τους βάσει συγκρίσεων των τιμών αυτών
- Δημιουργία πίνακα για αποθήκευση τιμών διαφοράς του ενημερωμένου grayscale φόντου με το φόντο του τρέχοντος frame
- Αποστολή του τελευταίου πίνακα που περιέχει το τμηματοποιημένο frame πίσω στον client μέσω συνάρτησης **sendImage** (παρόμοια με συνάρτηση **broadcast**)

Κεφάλαιο 3 Αποτελέσματα

1. ΕΞΟΠΛΙΣΜΟΣ

Οι υπολογιστές server που χρησιμοποιήθηκαν είχαν λειτουργικό Windows 7 32-bit, 2GB RAM, κάρτα γραφικών NVIDIA GeForce 9500 GT και επεξεργαστή Intel® Core™2 Quad Q8200 @ 2,33 GHz με μνήμη cache 3MB. Ο client από την άλλη είχε λειτουργικό Windows 7 64-bit, 4GB RAM, κάρτα γραφικών ATI Mobility Radeon™ HD 4650 και επεξεργαστή Intel® Core™ i5 @ 2.27 GHz με 3MB cache επίσης. Η camera που βρισκόταν στον client ήταν ενσωματωμένη και το μοντέλο της ήταν USB2.0 UVC WebCam. Το δίκτυο WiFi ήταν ταχύτητας 4Mbps.

2. ΠΕΡΙΓΡΑΦΗ ΠΕΙΡΑΜΑΤΩΝ

Χρησιμοποιήθηκαν, σε πρώτη φάση, διαφορετικές αναλύσεις video. Αυτό βοήθησε στο να μελετηθεί η επιρροή της ανάλυσης στον χρόνο εκτέλεσης αλλά και στον χρόνο διαμοίρασης – συλλογής. Ο χρόνος διαμοίρασης είναι αυτός που περιλαμβάνει το διάβασμα του frame, το σπάσιμο σε υπό - frames καθώς και την αποστολή στους servers τμηματοποίησης, μέχρι να σταλεί το πακέτο, ενώ ο χρόνος συλλογής το χρόνο που χρειάζεται για να ληφθούν όλα τα πακέτα των υπό - frames, να συντεθεί η αρχική εικόνα και τέλος να προβληθεί. Το μέγεθος του πακέτου που αποστέλλεται μέσω του UDP είναι περίπου στα 42000 bytes για 640x480 ανάλυση και περιλαμβάνει ολόκληρο το frame. Στους πίνακες που ακολουθούν βρίσκεται η μέση τιμή που υπολογίστηκε για κάθε περίπτωση, πραγματοποιώντας πέντε διαφορετικές εκτελέσεις της διαδικασίας.

Ο συνολικός χρόνος ορίζεται ως εξής:

- Χρόνος που χρειάζεται για το άνοιγμα της σύνδεσης (μέχρι αυτή να ομαλοποιηθεί).
- Χρόνος διάσπασης σε υπό – frames, αν αυτά υπάρχουν.
- Χρόνος εκτέλεσης αλγορίθμου στον/στους server/servers τμηματοποίησης.
- Χρόνος συλλογής των αποτελεσμάτων στον client
- Χρόνος σύνθεσης ολόκληρης της εικόνας

Ο client υπολογιστής που έστελνε το video, δεν προέβαλλε την δυαδική μάσκα (τμηματοποιημένο frame) αν δεν είχαν συλλεχθεί όλα τα μέρη που την απαρτίζουν. Ακόμα, το τρέχον frame/υπό – frame δεν επεξεργαζόταν στον server αν δεν είχε τελειώσει η τμηματοποίηση του προηγούμενου. Η κάθε περίπτωση μας δίνει

μια γενική εικόνα για τον ρυθμό των frames/second. Έτσι, έχουμε στον Πίνακα 1 και στον Πίνακα 2:

Πίνακας 1 – Μέση τιμή frames/second στα τετρακόσια frame με διαφορετικό αριθμό υπό – frames για τις τρεις περιπτώσεις.

	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	9,5	16	12,3	9,3
Camera 640x480	9,3	13,7	11,7	8,7
Camera 320x240	22,8	19,6	16,2	11,7

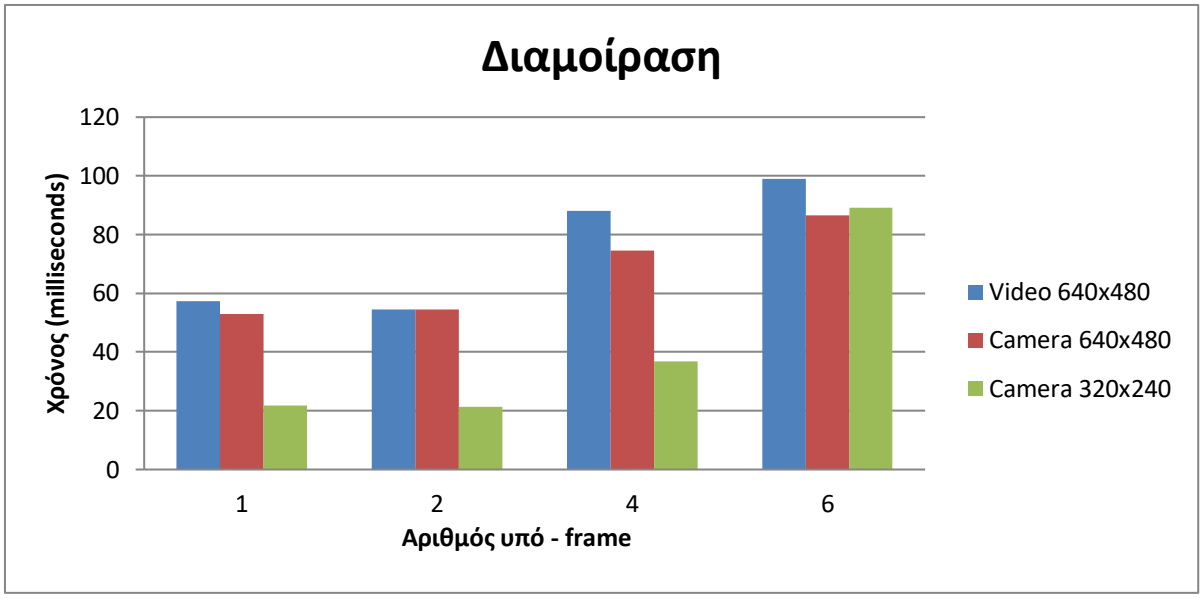
Πίνακας 2 – Μέση τιμή frames/second στα πεντακόσια frame με διαφορετικό αριθμό υπό – frames για τις τρεις περιπτώσεις.

	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	9,5	16,1	12,5	10,6
Camera 640x480	8,6	13,5	11,8	9,5
Camera 320x240	23,5	16,6	16,3	11,9

Ο Πίνακας 3 περιέχει την διαμοίραση του frame ή των υπό – frames στους servers τμηματοποίησης με τον χρόνο όπως ορίστηκε προηγουμένως. Το Σχήμα 7 παρέχει μια αναπαράσταση της εξέλιξης των χρόνων της διαδικασίας της διαμοίρασης, από τον Πίνακα 3, για τις διάφορες περιπτώσεις.

Ο Πίνακας 4 αφορά τον χρόνο για την συλλογή των επιμέρους κομματιών. Εδώ, ο χρόνος άρχιζε αμέσως μετά τον έλεγχο διαθέσιμου κομματιού για συλλογή και τελείωνε όταν όλα τα μέρη είχαν τοποθετηθεί σωστά, λίγο πριν γίνει η προβολή. Με το Σχήμα 8 διαφαίνεται πώς ο αριθμός των υπό – frames επηρεάζει την διαδικασία της συλλογής στον client.

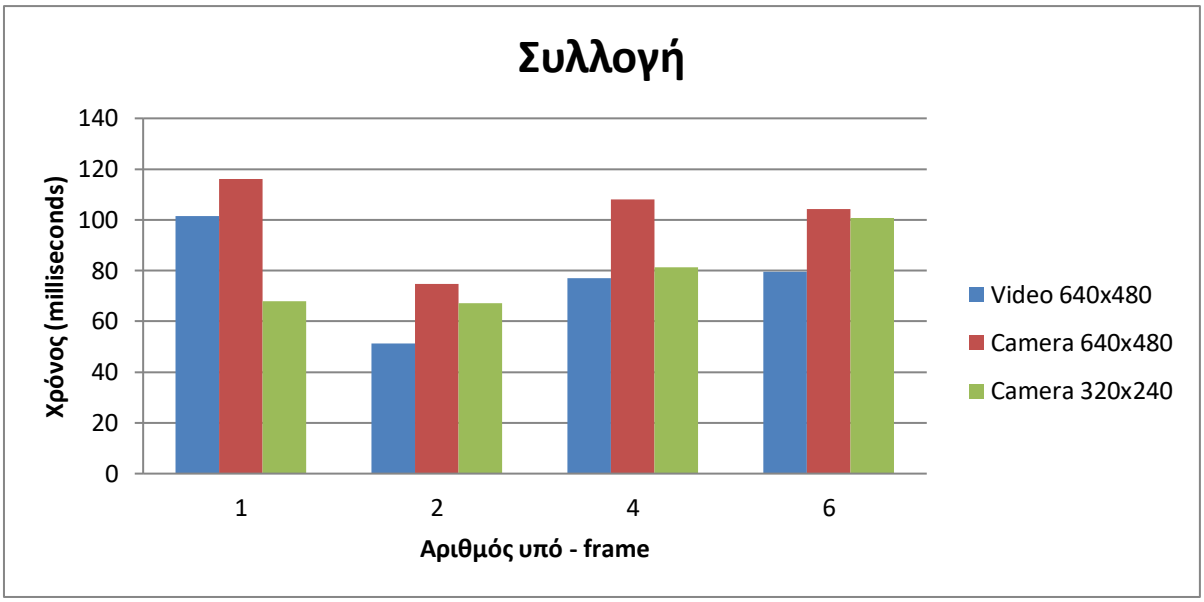
Ο Πίνακας 5 και ο Πίνακας 6 δείχνουν τον χρόνο που χρειάστηκε ολόκληρη η διαδικασία, αφού είχε εξαλειφθεί η αρχική καθυστέρηση από το άνοιγμα των συνδέσεων. Δεν συμπεριλήφθηκαν, δηλαδή, τα αρχικά πενήντα frames. Για τα τετρακόσια frames είχαμε την επεξεργασία από το frame 51 ως το 451 και για τα πεντακόσια από το 452 ως το 952. Το Σχήμα 9 δείχνει σε πρώτη φάση την διάρκεια της διαδικασίας. Στο Σχήμα 10 φαίνεται πως οι αναλογίες με το Σχήμα 9 διατηρούνται κατά κάποιον τρόπο. Σε βάθος χρόνου, όμως, βλέπουμε μια σημαντική αλλαγή στους χρόνους για τις περιπτώσεις των τεσσάρων και των έξι υπό – frames,



Σχήμα 7 – Οι χρόνοι διαμοίρασης (milliseconds) για συγκεκριμένο αριθμό υπό – frames στους servers.

Πίνακας 3 – Μέση τιμή των χρόνων (milliseconds), μαζί με την τυπική απόκλιση, για τη διαμοίραση των υπό – frames στους server τμηματοποίησης με χρήση WiFi.

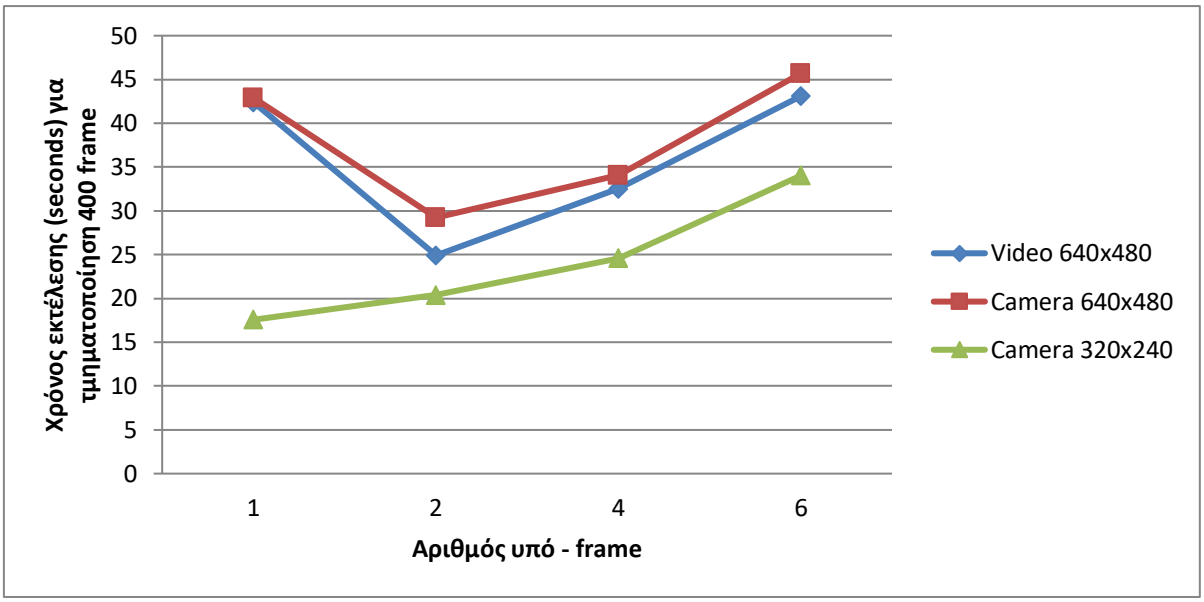
	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	57,23±27,28	54,54±21,74	88,13±21,71	98,90±92,68
Camera 640x480	52,91±32,29	54,40±25,57	74,47±17,11	86,53±34,61
Camera 320x240	22,57±88,13	21,37±35,99	36,78±42,30	89,20±74,48



Σχήμα 8 – Οι χρόνοι συλλογής (milliseconds) πίσω στον client για συγκεκριμένο αριθμό υπό – frames από τους servers.

Πίνακας 4 – Μέση τιμή του χρόνου συλλογής (milliseconds) των υπό – frames στον client μέσω WiFi και σύνθεση της τελικής δυαδικής εικόνας.

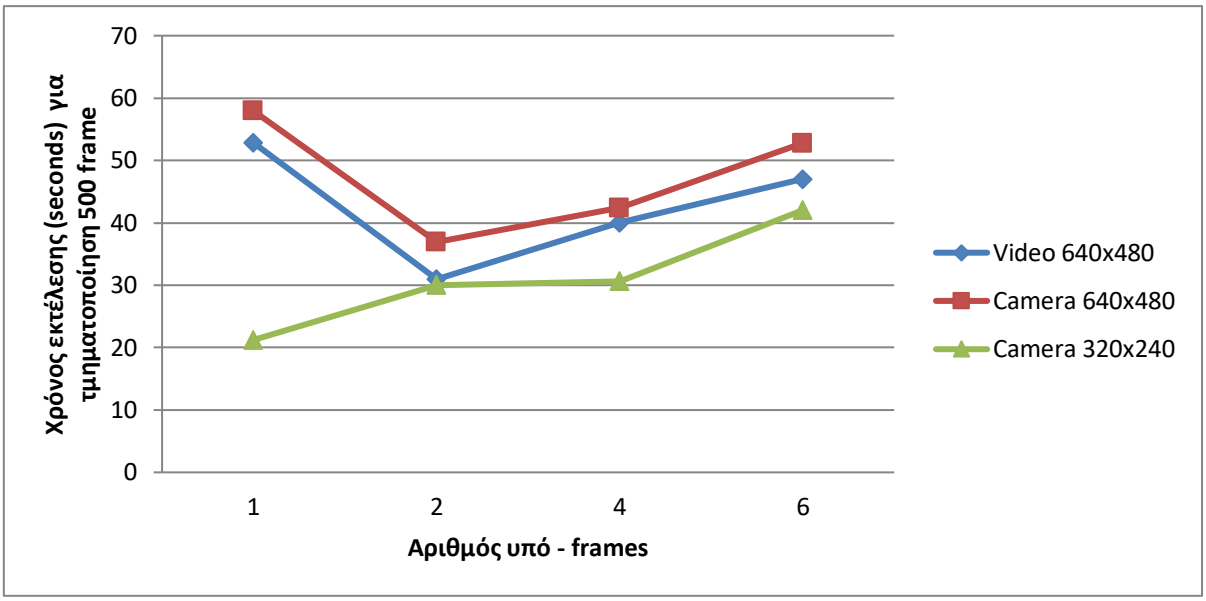
	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	101,40±112,23	51,24±84,21	77,10±122,42	79,55±161,61
Camera 640x480	116,22±296,09	74,80±246,90	107,93±325,60	104,32±297,13
Camera 320x240	67,90±267,70	67,13±233,76	81,35±261,20	100,80±372,27



Σχήμα 9 – Χρόνος εκτέλεσης (seconds) για όλη τη διαδικασία από το frame 51 έως το frame 451.

Πίνακας 5 – Μέση τιμή του χρόνου εκτέλεσης όλης της διαδικασίας για 51 έως 451 frames.

	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	42,385	24,914	32,525	43,095
Camera 640x480	42,916	29,239	34,077	45,692
Camera 320x240	17,567	20,393	24,608	34,024



Σχήμα 10 – Χρόνος εκτέλεσης (seconds) για όλη τη διαδικασία από το frame 452 έως 952.

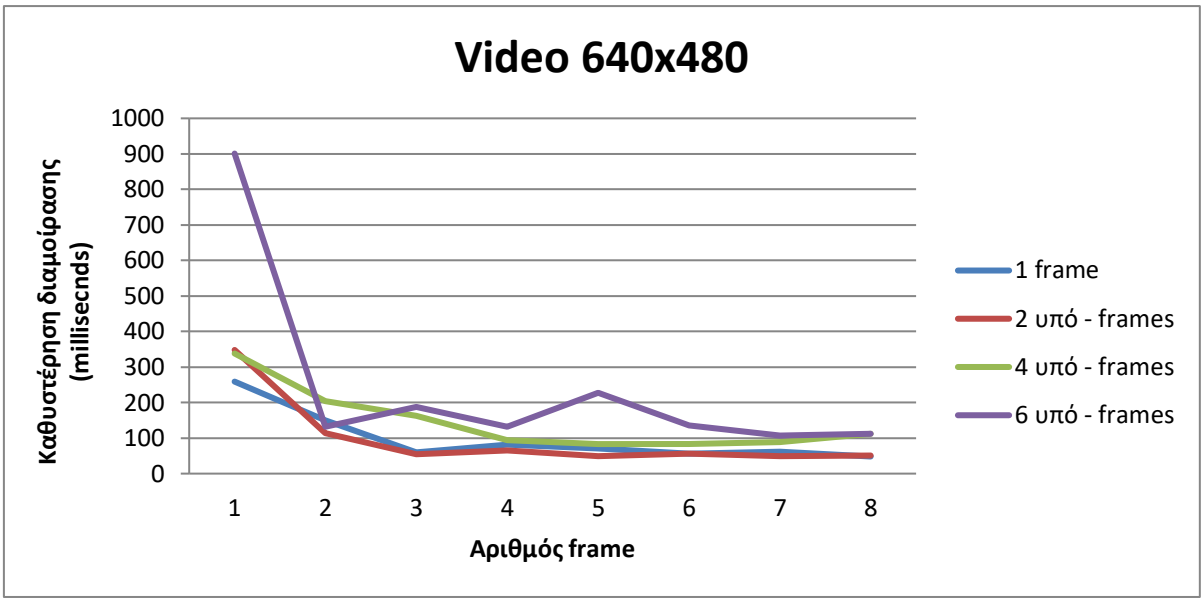
Πίνακας 6 – Μέση τιμή του χρόνου εκτέλεσης όλης της διαδικασίας για τα frame 452 έως 952.

	Πλήθος υπό – frame (servers τμηματοποίησης)			
Ακολουθίες video	**1**	**2**	**4**	**6**
Video 640x480	52,815	30,964	40,048	47,036
Camera 640x480	57,923	36,932	42,4	52,743
Camera 320x240	21,231	30,044	30,65	42,019

Επιπλέον παρατηρήθηκε κάποια καθυστέρηση στη διαμοίραση από τον client και στην συλλογή των υπό – frames από τους server τμηματοποίησης. Στον Πίνακα 7, Πίνακα 8 και Πίνακα 9 έχουμε τον αριθμό των frames που χρειάστηκε να επεξεργαστούν, για την διαμοίραση, για να εξαλειφθεί η καθυστέρηση των αρχικών διαδικασιών στα sockets και αντίστοιχα, για την συλλογή, στον Πίνακα 10, Πίνακα 11, Πίνακα 12. Μπορούμε να πούμε πως αυτές είναι οι διαδικασίες αρχικοποίησης που πραγματοποιεί το socket που αναλαμβάνει μια επεξεργασία σε κάποια δεδομένα, για την εξυπηρέτηση κάποιου αιτήματος. Ύστερα από το πέρας αυτού του αριθμού, οι χρόνοι ομαλοποιούνταν σε ένα συγκεκριμένο εύρος και μετρήθηκαν τα παρακάτω αποτελέσματα.

Ξεκινώντας από το Σχήμα 11 και μέχρι το Σχήμα 16 παρατηρείται μια διαφορά στον μέγιστο χρόνο της διαμοίρασης και σ' αυτόν της συλλογής, με μεγαλύτερο τον τελευταίο. Ο client καλείται να λάβει όλα τα υπό – frames και να τα συνθέσει σε ένα, διαδικασία που διαρκεί περισσότερο από το να τα διαμοιράσει απλώς στους servers τμηματοποίησης. Αυτό γίνεται διότι στους servers δημιουργείται ξεχωριστή διεργασία που δέχεται μόνο ένα υπό – frame, πράγμα που στον client δεν ισχύει, αφού μια διεργασία δέχεται όλα τα υπό – frame. Παρ' όλα αυτά, σε όλες τις περιπτώσεις παρουσιάζεται το ίδιο φαινόμενο, έχοντας διάρκεια οχτώ frames.

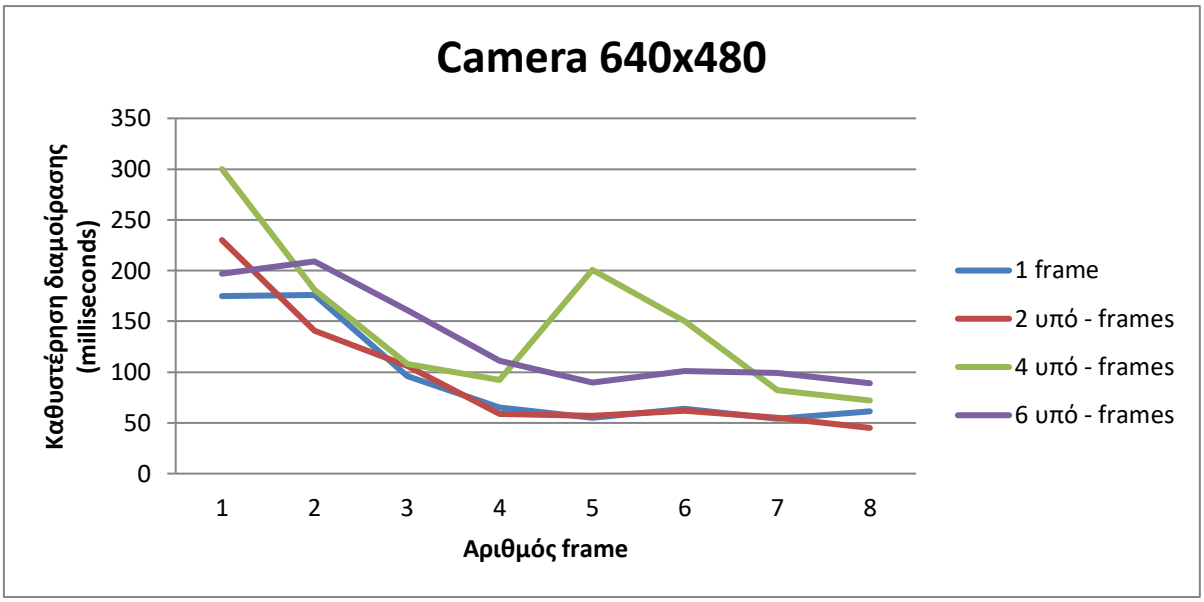
Στο Σχήμα 15 και στο Σχήμα 16 οι χρόνοι ομαλοποιούνται πιο ξεκάθαρα σε σχέση με τα προηγούμενα. Αυτό που έχουμε είναι η επιρροή του δικτύου κατά την εκτέλεση της διαδικασίας. Αυτές οι διαβαθμίσεις του χρόνου, ωστόσο, δεν έπαιξαν ρόλο στον αριθμό των frames που ήταν, ούτως ή άλλως, να επεξεργαστούν για την εξάλειψη της καθυστέρησης και ήταν τοπικές παρεμβάσεις.



Σχήμα 11 - Διάρκεια καθυστέρησης διαμοίρασης όλων των υπό – frames στους servers τμηματοποίησης για το αρχείο video ανάλυσης 640x480.

Πίνακας 7 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) στη διαμοίραση για το αρχείο video ανάλυσης 640x480.

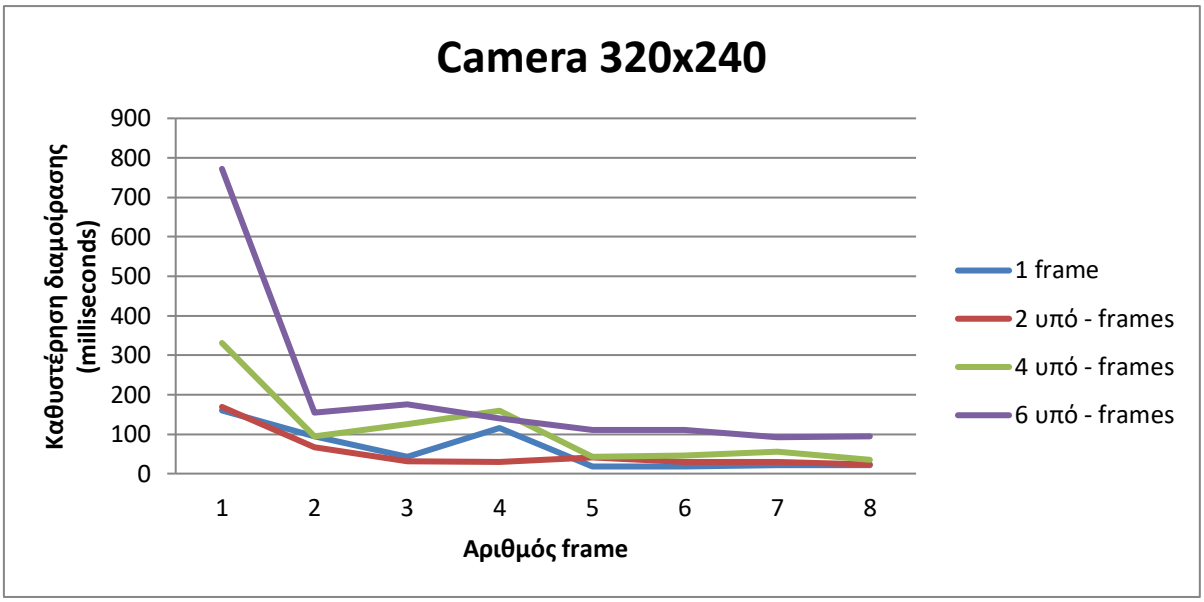
	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	259	150	59	81	71	56	62	48
2	348	113	54	65	49	57	49	50
4	338	204	163	94	83	84	89	112
6	901	131	188	132	227	136	107	112



Σχήμα 12 - Διάρκεια καθυστέρησης διαμοίρασης όλων των υπό – frames στους servers τμηματοποίησης για τη λήψη από camera ανάλυσης 640x480.

Πίνακας 8 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) στη διαμοίραση για την camera 640x480.

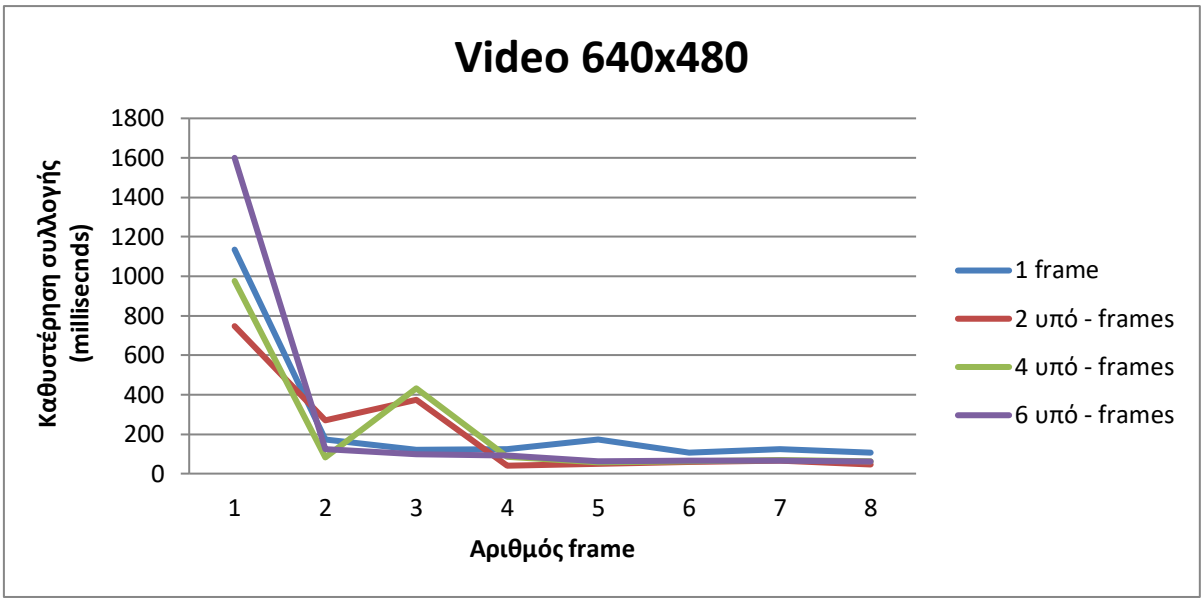
	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	175	176	96	65	55	64	54	61
2	230	141	106	59	57	62	55	45
4	300	181	108	92	201	150	82	72
6	197	209	161	111	90	101	99	89



Σχήμα 13 - Διάρκεια καθυστέρησης διαμοίρασης όλων των υπό – frames στους servers τμηματοποίησης για τη λήψη από camera ανάλυσης 640x480.

Πίνακας 9 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) στη διαμοίραση για την camera 320x240.

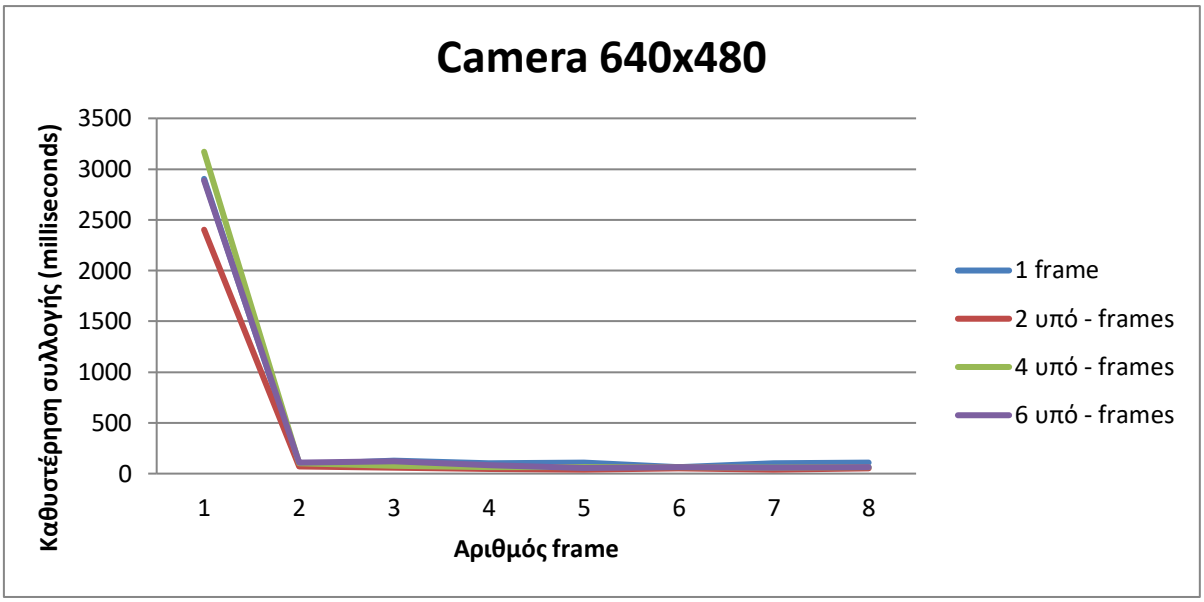
	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	160	94	43	115	18	18	22	22
2	169	66	31	29	41	29	29	23
4	331	95	125	160	43	45	55	35
6	772	155	175	139	111	111	92	95



Σχήμα 14 – Διάρκεια καθυστέρησης συλλογής όλων των υπό – frames και σύνθεση ολόκληρης της εικόνας για το αρχείο video 640x480.

Πίνακας 10 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) του δικτύου για την συλλογή των τμηματοποιημένων frames του video ανάλυσης 640x480.

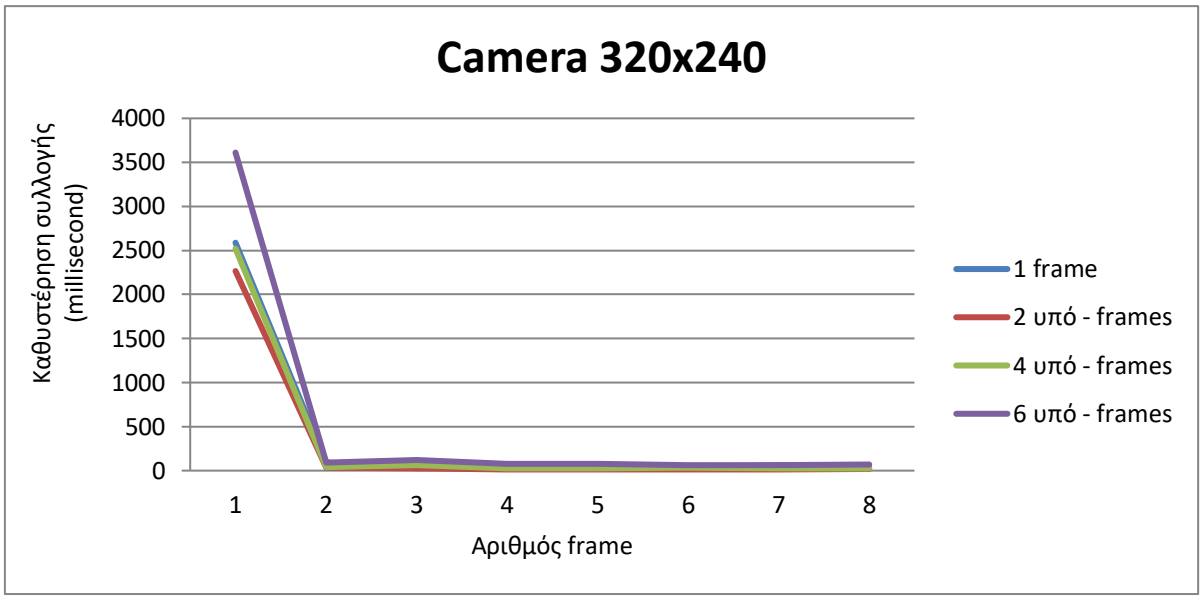
	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	1135	172	119	123	172	106	125	106
2	747	271	373	40	49	58	64	46
4	976	81	431	86	57	63	68	61
6	1600	123	98	91	63	66	66	61



Σχήμα 15 – Διάρκεια καθυστέρησης συλλογής όλων των υπό – frames και σύνθεση ολόκληρης της εικόνας για camera ανάλυσης 640x480.

Πίνακας 11 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) του δικτύου για την συλλογή των τμηματοποιημένων frames στη camera ανάλυσης 640x480.

	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2904	89	129	105	111	63	103	108
2	2403	68	57	45	42	49	39	53
4	3171	99	79	63	65	66	59	62
6	2887	106	120	85	55	62	59	67



Σχήμα 16 – Διάρκεια καθυστέρησης συλλογής όλων των υπό – frames και σύνθεση ολόκληρης της εικόνας για camera ανάλυσης 320x240.

Πίνακας 12 – Μέση τιμή της διάρκειας της καθυστέρησης (milliseconds) του δικτύου για την συλλογή των τμηματοποιημένων frames στη camera ανάλυσης 320x240.

	Αριθμός επεξεργασμένων frame (διάρκεια καθυστέρησης)							
Πλήθος υπό – frame (servers τμηματοποίησης)	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
1	2588	94	28	48	42	30	64	37
2	2267	34	28	13	13	13	13	21
4	2521	41	60	28	29	33	27	25
6	3611	91	122	74	80	66	63	67

Κεφάλαιο 4 Συμπεράσματα

Το πόσο αποδοτικός θα ‘ναι ο παραλληλισμός της εικόνας σε ένα δίκτυο υπολογιστών, καθορίζεται από τον φόρτο επικοινωνίας. Αυτός αποτελείται από την διανομή των υπό – εικόνων, την αποστολή των δεδομένων προς επεξεργασία και την επιστροφή των επεξεργασμένων δεδομένων πίσω στον client, δηλαδή, την συλλογή των αποτελεσμάτων. Εύστοχα, λοιπόν, ο φόρτος αυτός προκύπτει από το πλήθος των υπό – εικόνων και χρειάζεται η καλύτερη διάσπαση σχετικά με τον διαθέσιμο αριθμό υπολογιστών. Αξίζει να προστεθεί πως στην περίπτωση της επεξεργασίας ενός video οι εικόνες δεν είναι ανεξάρτητες μεταξύ τους, πράγμα που δυσχεραίνει κατά κάποιον τρόπο τους υπολογισμούς, αφού υπάρχει συνοχή μεταξύ τους.

Τα αποτελέσματα αυτής της πτυχιακής εργασίας κατέδειξαν την βέλτιστη αρχιτεκτονική για τις διάφορες περιπτώσεις που δοκιμάστηκαν. Υπάρχει, βέβαια, η περίπτωση λόγω του πρωτοκόλλου επικοινωνίας UDP να χάνονται ορισμένα frame κατά την μετάδοση, αλλά η φύση του αλγορίθμου που συνδέει το τρέχον με τα προηγούμενα παρέχει μια ικανοποιητική ανοχή στις απώλειες. Θετικό είναι πως η διαφορά μεταξύ της σειριακής εκτέλεσης και της βέλτιστης, αυτής των δύο υπό - frames, ήταν σχεδόν διπλάσια στο video και σαφώς αρκετή για την camera της 640x480 ανάλυσης. Το αποτέλεσμα από τη camera ανάλυσης 320x240 δείχνει πως αναλώνεται αρκετός χρόνος στην επικοινωνία, αφού όσο αυξάνονται τα υπό – frames η εκτέλεση καθυστερεί. Αυτό μας δείχνει πως στις μικρές αναλύσεις δεν χρειάζεται να διασπάσουμε την εικόνα.

Ωστόσο, ο τρόπος με τον οποίο μετρήθηκαν οι χρόνοι διαμοίρασης και συλλογής εμπεριέχει κάποια χρονική επικάλυψη. Αυτό συμβαίνει διότι η αποστολή και η επεξεργασία του επόμενου frame/υπό – frame δεν γίνεται αν δεν έχει επεξεργαστεί στον/στους server/servers το προηγούμενο frame/υπό – frame και δεν έχει επιστραφεί, με την σειρά του, στον client. Ακόμη, η μετατροπή από `PImage` σε byte array και αντίστροφα εμπεριέχει κι αυτή κάποια χρονική καθυστέρηση. Κάθε υπό – frame για να αποσταλεί στον/στους server/servers αλλά και να σταλεί πίσω στον client υποβάλλεται σ’ αυτή τη διαδικασία.

Συμπεραίνουμε πως η καθυστέρηση που παρουσιάστηκε στους πίνακες 7, 8, 9, 10, 11, 12 οφείλεται κατά κύριο λόγο στο άνοιγμα των συνδέσεων μεταξύ του client και των servers τμηματοποίησης. Επιπλέον, υπάρχει μια κατάσταση για τις αρχικές λειτουργίες που καλείται να εκτελέσει ο/οι server/servers και η οποία αν δεν τελειώσει δεν καταλήγουμε στην ομαλοποίηση των παραπάνω χρόνων. Όσον αφορά τη διαμοίραση των frames, έχουμε μικρότερες καθυστερήσεις αφού τα κομμάτια που στέλνονται είναι μικρά σε μέγεθος. Εδώ συμπεριλαμβάνεται και η μετατροπή των `PImage` σε byte array. Στην συλλογή οι πρώτες τιμές είναι μεγάλες, διότι συλλέγει όλα τα κομμάτια από όλους τους servers, μετατρέποντάς τα και από byte arrays σε `PImage`.

Επιπλέον, παρατηρείται μια διαφορά στον χρόνο εκτέλεσης ολόκληρης της διαδικασίας σε βάθος χρόνου. Πιο συγκεκριμένα, εκεί που έχουμε την μέτρηση για τα τετρακόσια frames παρατηρείται πως η περίπτωση διαχωρισμού σε έξι υπό – frames είναι πιο αργή από το αν είχαμε ολόκληρο το frame, ενώ, στα πεντακόσια όλες οι περιπτώσεις είναι γρηγορότερες. Η αναλογία, παρ' όλα αυτά κατά κάποιον τρόπο διατηρείται.

Ο στόχος, εν ολίγοις, της πτυχιακής πραγματοποιήθηκε, δείχνοντας ταυτόχρονα και την επιρροή του δικτύου στην επικοινωνία. Τα frames/second που επεξεργάζονταν, αποστέλλονταν και λαμβάνονταν έφταναν τον αριθμό των frames που προβάλλονταν στην κανονική αναπαραγωγή του video. Μάλιστα, και τα ελάχιστα frames με θόρυβο δεν κατάφεραν να αλλοιώσουν την εικόνα που ήταν να προβληθεί, με συνέπεια η κίνηση να μπορούσε να αναγνωριστεί.

Βιβλιογραφία

[1] K.G. Margaritis, Parallel Distributed Processing Laboratory (PDP Lab), Department of Applied Informatics, University of Macedonia Thessaloniki, Greece."Parallel Programming & Distributed Software Systems". Internet: http://pdplab.it.uom.gr/project/parallel/index.htm, [Oct. 13, 2014].

[2] "Parallel Computing". Internet: http://en.wikipedia.org/wiki/Parallel_computing#Fine-grained.2C_coarse-grained.2C_and_embarrassing_parallelism, Oct. 4 2004, [Oct. 13, 2014].

[3] Massimo Picardi. "Background subtraction techniques: a review", presented at IEEE International Conference on Systems, Man and Cybernetics, 2004.

[4] "Task Parallelism". Internet: http://en.wikipedia.org/wiki/Task_parallelism, Nov. 6 2013, [Oct. 2, 2014]

[5] "Data Parallelism". Internet: http://en.wikipedia.org/wiki/Data_parallelism, Aug. 19 2013, [Oct. 2, 2014]

[6] Benoit Gennart and Roger D. Hersch. "Computer - Aided Synthesis of Parallel Image Processing Applications", presented at Parallel and Distributed Methods for Image Processing III, SPIE International Symposium on Optical Science, Engineering and Instrumentation, Denver, Colorado, 1999.

[7] Prabhakar Ramachandra Neelawani and Chetan H. "Parallel Architecture for Fast Image Processing" in *Proc. ICCTEM*, 2012, pp. 481-485.

[8] Chi-kin Lee and Mounir Hamdi. "Practical aspects and experiences. Parallel image processing applications on a network of workstations". Hong Kong University of Science and Technology, Kowloon, Hong Kong, 1994.

[9] Sanjay Saxena, Neeraj Sharma and Shiru Sharma. "Image Processing Tasks using Parallel Computing in Multi core Architecture and its applications in Medical Imaging." Internet: http://ijarcce.com/upload/2013/april/53%20-%20sanjay%20saxena%20-%20Image%20Processing%20Tasks%20using.pdf, Apr. 4, 2013 [Oct. 16, 2014].

[10] Seth Copen Goldstein and Nikos Hardavellas. Class Lecture, Topic: "Parallel Architecture Fundamentals", CS 740, School of Computer Science, Carnegie Mellon University, Pittsburgh, Sep. 22, 2003.

[11] Andy Pimentel. "Introduction to Parallel Architecture." PhD Thesis, University of Amsterdam, Netherlands, 1998.

[12] "Multiprocessing". Internet: http://en.wikipedia.org/wiki/Multiprocessing, Oct. 10 2014, [Oct. 18 2014]

[13] Hesham El-Rwini and Mostafa Abd-El-Barr (2005, February). Advanced Computer Architecture and Parallel Processing. (1st edition). [On-line]. Available: http://lyle.smu.edu/~rewini/slides/adv-arch-bk/ [Sep. 25, 2014].

[14] Cristina Soviany, “Embedding Data and Task Parallelism in Image Processing Applicatios.” Thesis, Universitatea Politehnica Bucuresti, Romania, 2003.

[15] “Distributed Computing”. Internet: http://en.wikipedia.org/wiki/Distributed_computing, Oct. 8 2014, [Oct. 18 2014]

[16] S.Schmid and R.Wattenhofer. Web Lecture, Topic: "Principles of Distributed Computing". Swiss Federal Institute of Technology, Zurich, Switzerland.

[17] Muneto Yamamoto and Kunihiko Kaneko. "Parallel Image Database Processing with Map-Reduce and Performance Evaluation in Pseudo Distributed Mode". International Journal of Electronic Commerce Studies, vol.3, No.2, pp. 211-228, 2012.

[18] Barry WIlkinson and Michael Allen. "Parallel Programming: Techniques and Applications using Networked Workstations and Parallel Computers". Internet: http://140.127.182.82/homepage/ccchen/parallel/Slides1.pdf, 1999 [Nov. 1, 2014].

[19] Dennis Lin, Xiaohuang (Victor) Huang, QUang Nguyen, Joshua Blackburn, Christopher Rodrigues, Thomas Huang, Minh N. Do, Sanjay J. Patel and Wen Mei W. Hwu. ”The Parallelization of Video Processing: From Programming models to applications". Internet: http://minhdo.ece.illinois.edu/publications/parallelvideo.pdf, Nov. 2009 [Oct. 11 2014].

[20] Fan-Chieh Cheng, Shih-Chia Huang and Shanq-Jang Ruan. "Illumination-Sensitive Background Modeling Approach for Accurate Moving Object Detection", in *IEEE Transactions on Broadcasting*, vol. 57, No. 4, Dec. 2011.

[21] W. Hu, T. Tan, L. Wang and S. Maybank. “A survey on visual surveillance of object motion and behaviors”, in *IEEE Transactions on Systems, Man and Cybernetics – Part C: Applications and Reviews*, vol. 34, no.3, pp. 334 – 352, Aug. 2004.

[22] Ren Y., Chua C., Ho Y. “Motion detection with non-stationary background”, presented at 11th International Conference on Image Analysis and Processing, Palermo, Italy, 2001.

[23] J. –E. Ha and W.-H. Lee, “Foreground objects detection using multiple difference images”, Opt. Eng., vol. 49, no. 4, p.047201, Apr.2010.

[24] C. Stauffer and W.E.L. Grimson, “Adaptive background mixture models for real-time tracking”, in *Proc. IEEE CVP*, 1999, pp. 246-252.