Πολυτεχνείο Κρήτης
Τμήμα Ηλεκτρονικών Μηχανικών & Μηχανικών Υπολογιστών
Διπλωματική Εργασία

Αναγνώριση κινούμενων αντικειμένων σε βίντεο και υλοποίηση σε (Digital Signal Processor) εξειδικευμένο επεξεργαστή.

Εκπονήθηκε από:
Γεωργάκης Μιχάλης

Τριμελής Επιτροπή

Καθηγητής Μιχάλης Ζερβάκης (επιβλέπων)

Καθηγητής Κωνσταντίνος Καλαϊτζάκης

Αναπληρωτής Καθηγητής Ευριπίδης Πετράκης

Σεπτέμβριος 2009

Χανιά

Αφιερωμένη στον καθηγητή, δάσκαλο και φίλο Μιχάλη Ζερβάκη για την υπομονή και την βοήθεια του.

Περίληψη

Το πρόβλημα της ανίχνευσης αντικειμένων μέσα σε βίντεο συνίσταται στον εντοπισμό ανθρώπων, αυτοκινήτων η άλλων περιοχών ενδιαφέροντος μέσα σε αυτό. Η λύση που προτείνεται είναι η χρήση δύο διαφορετικών μεθόδων που λειτουργούν παράλληλα προκειμένου να επιτευχθεί ο βέλτιστος προσδιορισμός αυτών. Οι δύο αυτές μέθοδοι είναι η αφαίρεση του τρέχοντος frame από ένα κενό frame φόντου (Background Subtraction) και ο εντοπισμός ακμών μέσα στο τρέχον frame (Edge Detection). Οι δύο αυτές μέθοδοι αν και συνιστούν μία πολύ ισχυρή μέθοδο εντοπισμού έχουν και οι δύο κάποια τρωτά σημεία. Η αφαίρεση του τρέχοντος frame από ένα κενό frame φόντου βασίζει την λειτουργία της σε μία παράμετρο κατωφλίου η οποία αν οριστεί λάθος δημιουργείται έντονο πρόβλημα ενώ η ανίχνευση ακμών σε πραγματικό χρόνο για κάθε frame είναι μία ιδιαίτερα χρονοβόρα διαδικασία. Η εργασία μας βασίστηκε στην εξομάλυνση των δύο αυτών προβλημάτων. Υλοποιήθηκε ένας σύνθετος αλγόριθμος επαναπροσδιορισμού της τιμής για την παράμετρο του κατωφλίου ώστε να μεγιστοποιείται η απόδοση της μεθόδου αφαίρεσης του τρέχοντος frame από το δεδομένο frame αναφοράς ενώ προτείνεται μία εναλλακτική συνάρτηση για την ανίχνευση ακμών η οποία απαιτεί τον μισό χρόνο, χωρίς να υπολείπεται σε απόδοση. Παράλληλα προστέθηκε ένας αλγόριθμος παρακολούθησης των αντικειμένων ενδιαφέροντος ο οποίος παρέχει στον χρήστη πληροφορίες για το αντικείμενο παρακολούθησης που αφορούν το μέγεθος, την θέση του και την ταχύτητα του ενώ γίνεται και πρόβλεψη της μελλοντικής του θέσης με βάση την προηγούμενη.

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή

1.1 Εισαγωγή……Σελ.6
1.2 Εφαρμογές των συστημάτων αναγνώρισης……Σελ.7
1.3 Η δική μας υλοποίηση και τα προβλήματα της……..……Σελ.8
1.4 Οι λύσεις που προτείνονται……Σελ.9
1.5 Τι περιέχεται στα κεφάλαια……Σελ.9

Κεφάλαιο 2. Το σύστημα

2.1 Γενικά για το Σύστημα……...……..……Σελ.10
2.2 Προδιαγραφές του συστήματος……Σελ.10
2.3 Είσοδοι συστήματος……..……Σελ.13
2.4 Υλοποίηση του συστήματος……..……Σελ.14
2.5 Έξοδοι συστήματος……...……Σελ.15
2.6 Matrox Imaging Library……...……Σελ.17

Κεφάλαιο 3. Αναγνώριση αντικειμένων σε εικόνα (frame)

3.1 Μέθοδοι εντοπισμού των αντικειμένων……...……Σελ.19
3.2 Αφαίρεση τρέχοντος frame από δεδομένο Background frame (Frame subtraction)……...Σελ.19
3.3 Γενικά για το κατώφλι……..……..……Σελ.19
3.4 Προδιαγραφές μηχανισμού υπολογισμού βέλτιστου κατωφλίου…..……Σελ.20
3.5 Μηχανισμός γραμμικής εκτίμησης βέλτιστου κατωφλίου……...……Σελ.22
3.6 Μηχανισμός γεωμετρικής εκτίμησης βέλτιστου κατωφλίου ……...Σελ.24
3.7 Ανίχνευση ακμών μέσα στο frame (Edge Detection)…….. Σελ.25
3.8 Βελτιστοποιημένη ανίχνευση ακμών……Σελ.27

Κεφάλαιο 4. Blob Tagging, Tracking, Predicting Algorithm

4.1 Γενικά για τον αλγόριθμο ……...……Σελ.29
4.2 Είσοδοι αλγορίθμου…….....…Σελ.30
4.3 Εκκίνηση του αλγορίθμου……Σελ.30
4.4 Λειτουργία αλγορίθμου πρόβλεψης…...…….……Σελ.31
4.5 Έλεγχος επιτυχίας πρόβλεψης κατεύθυνσης (Vector Error) ……...…Σελ.32
4.6 Έλεγχος απόκλισης προβλεπόμενης απόστασης (Pixel Error)……..…Σελ.33

Κεφάλαιο 5. Εφαρμογή – Αποτελέσματα

5.1 Madrid airport video ……Σελ.35
5.2 Traffic camera video ……Σελ.37
5.3 OPTAG video ……..……Σελ.38
5.4 Αποτελέσματα ……..…Σελ.39
5.5 Συμπεράσματα……...…Σελ.39
5.6 Μετρήσεις χρόνων……Σελ.41
5.7 Συνολική βελτίωση με τη χρήση του βελτιστοποιημένου Edge Detection…...…Σελ.41

Κεφάλαιο 6. Βελτιώσεις – Μελλοντικές επεκτάσεις…….Σελ.43

Περιεχόμενα Πινάκων - Εικόνων

Εικόνα 2.1.1 Αρχικό Σύστημα……..……Σελ.10

Εικόνα 2.2.1 Block Diagram Βελτιστοποιήσεων…….……Σελ.11

Εικόνα 2.5.1 Έξοδος στο αρχείο καταγραφής συμβάντων……...……Σελ.15

Εικόνα 2.5.2 Έξοδοι στην οθόνη του συστήματος - κονσόλα……..……Σελ.16

Εικόνα 2.5.3 Έξοδοι στην οθόνη του συστήματος - βίντεο……..……Σελ.17

Εικόνα 3.4.1 Block Diagram Αλγορίθμου εύρεσης βέλτιστου κατωφλίου……..……Σελ.21

Εικόνα 3.5.1 Block Diagram FSM Αλγορίθμου εύρεσης βέλτιστου κατωφλίου…….……Σελ.22

Εικόνα 3.7.1 Block Diagram Αρχικού Edge Detection……...……Σελ.26

Εικόνα 3.8.1 Block Diagram Βελτιστοποιημένου Edge Detection……..……Σελ.27

Εικόνα 4.1.1 Block Diagram Αλγορίθμου Παρακολούθησης……..……Σελ.29

Εικόνα 5.1.1 Βίντεο Αεροδρομίου Μαδρίτης - Αρχικά……...……Σελ.35

Εικόνα 5.1.2 Βίντεο Αεροδρομίου Μαδρίτης - Τελικά.……Σελ.36

Εικόνα 5.1.3 Βίντεο Αεροδρομίου Μαδρίτης – Έξοδος στην κονσόλα……...……Σελ.36

Εικόνα 5.2.1 Βίντεο κυκλοφορίας – Χωρίς Edge Detection……Σελ.37

Εικόνα 5.2.2 Βίντεο κυκλοφορίας – Με Edge Detection…….…Σελ.37

Εικόνα 5.3.1 Στιγμιότυπο από OPTAG βίντεο…….Σελ.38

Πίνακας 5.4.1 Μετρήσεις OPTAG βίντεο χωρίς τη χρήση του αλγορίθμου εύρεσης βέλτιστου κατωφλίου…….Σελ.39

Πίνακας 5.4.2 Μετρήσεις OPTAG βίντεο με τη χρήση του αλγορίθμου εύρεσης βέλτιστου κατωφλίου…….Σελ.39

Πίνακας 5.6.1 Μετρήσεις χρόνων προσθηκών για OPTAG βίντεο……..Σελ.41

Πίνακας 5.7.1 Βελτίωση απόδοσης με τη χρήση του βελτιστοποιημένου Edge Detection……..Σελ.41

Πίνακας 5.7.2 Απόδοση εντοπισμού με τη χρήση του βελτιστοποιημένου Edge Detection……...Σελ.41

Βιβλιογραφία…….……Σελ.44

Παράρτημα: Sobel filter – Prewitt filter theory……Σελ.45

Κεφάλαιο 1. Εισαγωγή

1.1 Εισαγωγή

Η διπλωματική εργασία αυτή βασίζεται στο project OPTAG (Improving airport Efficiency, Security and Passenger Flow by Enhanced Passenger Monitoring) το οποίο διεκπεραιώθηκε από το εργαστήριο ψηφιακής επεξεργασίας σήματος και εικόνας του Τμήματος Ηλεκτρονικών μηχανικών και Μηχανικών υπολογιστών του Πολυτεχνείου Κρήτης. Αποτελεί συνέχεια αυτού και στοχεύει στην βελτίωση της απόδοσης του συστήματος όπως αυτό σχεδιάστηκε καθώς και στην προσθήκη περαιτέρω δυνατοτήτων σε αυτό.

Ο στόχος του προγράμματος με το ακρωνύμιο OPTAG είναι η υλοποίηση ενός συστήματος επιτήρησης χρησιμοποιώντας την πανοραμική εικόνα που λαμβάνεται από διάφορες κάμερες. Σε ένα αερολιμένα το σύστημα αυτό δεν πρόκειται μόνο να εξοικονομήσει χρόνο και να συμβάλλει στην ευκολία των επιβατών, αλλά και να περιορίσει το κόστος των συστημάτων ασφαλείας. Ένα αυτόνομο σύστημα θα είναι σε θέση να προσδιορίσει ανωμαλίες ή απρόβλεπτες καταστάσεις και μπορεί να βοηθήσει ένα ανθρώπινο χειριστή, ακόμη και αν δεν μπορεί να αντικαταστήσει την παρουσία του.

Ένας από τους στόχους του συστήματος είναι να εντοπίσει τα αντικείμενα που μπορεί να είναι ένας άνθρωπος ή κάποια αποσκευή. Κινούμενα αντικείμενα και αντικείμενα που δεν ανήκουν σε ένα προκαθορισμένο πλαίσιο δημιουργούν αλλαγές στην ένταση της εικόνας τις οποίες το σύστημα εντοπίζει και αξιολογεί.

Οι μέθοδοι που χρησιμοποιούνται για το σκοπό αυτό είναι αρκετά αποτελεσματικές, υπό ορισμένες προϋποθέσεις, αλλά οι περισσότεροι από αυτές δεν έχουν σχεδιαστεί για εφαρμογή πραγματικού χρόνου καθώς είναι ακατάλληλες αφού η υπολογιστική πολυπλοκότητα είναι αρκετά υψηλή. Στο OPTAG η εφαρμογή σε πραγματικό χρόνο με αξιόλογα αποτελέσματα είναι υψηλής προτεραιότητας. Αυτό σημαίνει ότι μια ισχυρή μέθοδος για την ανίχνευση αντικειμένων μέσω των συστημάτων επιτήρησης είναι απαραίτητη.

Το πρώτο πρόβλημα που πρέπει να αντιμετωπιστεί είναι η κίνηση των αντικειμένων στο εσωτερικό της σκηνής. Η προσέγγιση που χρησιμοποιείται, αναπτύσσει δύο συστήματα ανίχνευσης, το πρώτο βασίζεται στην διαφορά της έντασης της τρέχουσας εικόνας της ακολουθίας σε σχέση με το μία εικόνα αναφοράς και το άλλο με βάση το χρώμα στα άκρα της τρέχουσας εικόνας. Οι περιοχές που εντοπίζονται μέσω μίας πράξης AND αποτελούν τις περιοχές ενδιαφέροντος. Η συγχώνευση των δύο μεθόδων παρέχει ένα ισχυρό αλγόριθμο για την ανίχνευση αντικειμένων που παρουσιάζουν ενδιαφέρον. Με την πάροδο του χρόνου υπάρχουν αλλαγές φωτισμού στις εικόνες. Συνεπώς η ενημέρωση της εικόνας αναφοράς, είναι αναγκαία και εκτελείται παράλληλα, προκειμένου να καταστεί το σύστημα προσαρμόσιμο.

Ωστόσο οι δύο μέθοδοι , παρόλο που προσφέρουν ένα ισχυρό σύστημα αναγνώρισης έχουν και σημεία τα οποία επιδέχονται βελτίωσης. Μέσω αυτής της διπλωματικής εργασίας αυτός ακριβώς ήταν και ο σκοπός μας.

1.2 Εφαρμογές των συστημάτων αναγνώρισης

Μέχρι σήμερα έχει γίνει σε παγκόσμιο επίπεδο αρκετή έρευνα πάνω στην ανίχνευση αντικειμένων μέσα σε βίντεο. Η ίδια η Google μέσω των υπηρεσιών της επιχειρεί να ταξινομήσει τα βίντεο με βάση το περιεχόμενο τους. Παράλληλα στον τομέα της ασφάλειας μεγάλη προσπάθεια έχει γίνει προκειμένου η παρακολούθηση μέσω καμερών να καταστεί αποτελεσματικότερη.

Η δική μας εφαρμογή προορίζεται να συστήσει, αν όχι εξολοκλήρου από μόνη της, ένα σύστημα ασφαλείας με δυνατότητες αναγνώρισης αντικειμένων, μελέτης αυτών και εξαγωγής συμπερασμάτων από την δραστηριότητα μέσα στο βίντεο. Ιδανικά το σύστημα μας θα μπορούσε να λειτουργεί απολύτως αυτόνομα, ωστόσο η ανθρώπινη παρουσία χρειάζεται. Το σημείο που αξίζει να αναφερθεί είναι η απόλυτη προσαρμοστικότητα του συστήματος. Εφόσον αρχικοποιηθεί την πρώτη φορά λειτουργίας του μπορεί να λειτουργήσει απολύτως αυτόνομα χωρίς να χρειάζεται περαιτέρω εποπτεία των παραμέτρων του. Ο ανθρώπινος παρατηρητής λοιπόν δεν χρειάζεται να κάνει τίποτε άλλο από το να παρακολουθεί τις εξόδους αυτού και να επεμβαίνει όταν το σύστημα εμφανίζει κάποιο σήμα λάθους ή τον ενημερώνει για κάποιο συμβάν μέσα στο βίντεο.

Το σύστημα αν και σχεδιάστηκε για αεροδρόμια (με χρήση πανοραμικών καμερών και RF-tagging) μπορεί κάλλιστα να χρησιμοποιηθεί και αυτόνομα σαν ένα πλήρες σύστημα ασφαλείας το οποίο θα επεξεργάζεται σε πραγματικό χρόνο την πληροφορία που λαμβάνεται από τις κάμερες θα την αξιολογεί και θα ενημερώνει τον χειριστή για τα αποτελέσματα. Η λειτουργικότητα του είναι αρκετά ευρεία, γεγονός που μας επέτρεψε να το δοκιμάσουμε τόσο σε πραγματικά δεδομένα αεροδρομίου όσο και σε βίντεο από κάμερες ελέγχου κυκλοφορίας (traffic control cameras).

Υπάρχουν αρκετά εφαρμογές πάνω στο ίδιο αντικείμενο όπως:

- “Active People Tracking in Camera Images (Visual Surveillance)” του Nils T Siebel από το University of Kiel στην Γερμανία.
- “Active Surveillance Using Dynamic Background Subtraction” του Kenneth M. Dawson-howe από το Pennsylvania University στις Ηνωμένες Πολιτείες.
- Αυτή των Arun Hampapur, Jonathan H. Connell, Andrew W. Senior, Chiao-Fe Shu και Ying-Li Tian “System and method for managing moving surveillance cameras”.

Το κοινό χαρακτηριστικό όλων αυτών είναι η ιδιαίτερη βαρύτητα που δίνουν πάνω στον εντοπισμό αντικειμένων μέσω αφαίρεσης του τρέχοντος frame από ένα δεδομένο frame αναφοράς (Background Subtraction). Στην δική μας υλοποίηση το Background Subtraction αποτελεί ένα μόνο κομμάτι του συνολικού αλγορίθμου χωρίς να βασιζόμαστε στην πληροφορία που εξάγουμε αποκλειστικά μέσω αυτής της μεθόδου.

Παράλληλα υπάρχει πληθώρα εφαρμογών τόσο στον τομέα της παρακολούθησης (surveillance) όσο και στον τομέα της βιοιατρικής οι οποίες χρησιμοποιούν την ανίχνευση ακμών (Edge Detection) στην υλοποίηση των συστημάτων τους.

- Αυτόματη ανίχνευση των περιγραμμάτων των αγγειακών τοιχωμάτων σε εικόνες ενδοστεφανιαίου υπερηχογραφήματος, Καρπετάς Γεώργιος, Τμήμα Ιατρικής, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης.

- Ολοκληρωμένο σύστημα οδομετρίας για κινούμενα ρομπότ με χρήση μετρήσεων από πολλαπλούς αισθητήρες, Κελασίδη Ελένη, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών.
- Μελέτη και ανάπτυξη αλγορίθμων για την ανάλυση βρογχοσκοπικών εικόνων σε πραγματικό χρόνο, Καραργύρης Αλέξανδρος, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο.
- Ανίχνευση Κειμένου σε Βίντεο, Παπαβασιλείου Βασίλης Σταφυλάκης Θέμος Κατσούρος Βασίλης Καραγιάννης Γεώργιος, Ινστιτούτο Επεξεργασίας του Λόγου/Ε.Κ «Αθηνά».

Ακόμα ένα στοιχείο που αξίζει να αναφερθεί είναι η επεκτασιμότητα του συστήματος τόσο με την χρήση πολλαπλών καμερών που μπορούν να ενεργοποιηθούν η όχι κατά τη βούληση του χρήστη καθώς και μέσω της χρήσης πανοραμικής εικόνας. Η πανοραμική εικόνα μπορεί να προσφέρει ακόμα μεγαλύτερη ευελιξία καθώς το σύστημα μας είναι σχεδιασμένο για να λειτουργεί (κατά προτίμηση) με σταθερή εικόνα από μία η πολλαπλές κάμερες (μέσω πανοραμικού βίντεο).

Η καινοτομία στην δική μας υλοποίηση έγκειται στην χρήση και των δύο παραπάνω μεθόδων προκειμένου να υλοποιηθεί ένα ισχυρό σύστημα το οποίο να μπορεί να αναγνωρίσει και να παρακολουθήσει κινούμενα αντικείμενα μέσα σε βίντεο. Ακόμα ήταν απαραίτητο το σύστημα μας να διαφοροποιηθεί από την συνήθη κατάσταση η οποία είναι η θεώρηση μίας από τις παραπάνω μεθόδους κατάλληλη για κάθε μία εφαρμογή, προτείνοντας μία συνδυαστική προσέγγιση των δύο μεθόδων (ανίχνευσης ακμών και αφαίρεσης frame) προκειμένου να πετύχει την βέλτιστη αναγνώριση των αντικειμένων.

Επιπλέον μέσω των αλγορίθμων και των μηχανισμών που αναπτύχθηκαν προέκυψε ένα εντελώς νέο σύστημα το οποίο είναι εξαιρετικά προσαρμόσιμο όσον αφορά την είσοδο του (το βίντεο δηλαδή), είναι βελτιστοποιημένο επεξεργαστικά απαιτώντας πολύ λιγότερους πόρους (στοιχείο που μας προσφέρει ευελιξία ώστε στο μέλλον να μπορούμε να μιλάμε για παρακολούθηση πολλαπλών βίντεο υψηλής ανάλυσης δεδομένου του πλεονάσματος της ισχύος που έχουμε) ενώ παρέχει στον χρήστη πολλαπλές πληροφορίες τόσο όσον αφορά την λειτουργία του σε πραγματικό χρόνο όσο και την κατάσταση του συστήματος, μέσω αρχείο καταγραφής, εμφάνισης πληροφοριών σε πραγματικό χρόνο και αντίστοιχα έξοδο στην κονσόλα του λειτουργικού συστήματος.

1.3 Η δική μας υλοποίηση και τα προβλήματα της

Οι δύο μέθοδοι που χρησιμοποιούνται όπως αναφέρθηκε είναι ή χρήση αφαίρεσης ενός του κάθε frame από ένα δεδομένο, “κενό” frame αναφοράς (Frame Subtraction) και παράλληλα η ανίχνευση ακμών μέσα στην εικόνα μας (Edge Detection) προκειμένου να εντοπίσουμε τις ζητούμενες περιοχές ενδιαφέροντος. Οι δύο αυτές μέθοδοι σε συνδυασμό με την συνάρτηση BackgroundUpdate, η οποία ανανεώνει το δεδομένο frame αναφοράς που χρησιμοποιείται κατά την αφαίρεση, συνιστούν ένα σύστημα εντοπισμού το οποίο λειτουργεί ικανοποιητικά. Επιθυμώντας να βελτιώσουμε την απόδοση του συστήματος παρατηρήθηκαν τα παρακάτω:

- Τα σύστημα βασίζει πολύ την καλή λειτουργία του σε παραμέτρους που εισάγονται από τον χρήστη κατά την ενεργοποίηση του, οι οποίες αν είναι λανθασμένες αυτό υπολειτουργεί.

- Το σύστημα είναι απαραίτητο να είναι όσο πιο γρήγορο γίνεται. Όσο πιο υψηλό framerate έχουν οι συνεχώς αναβαθμιζόμενες κάμερες τόσο και αυτό οφείλει να συμβαδίζει.
- Από το σύστημα απουσιάζει ένας αλγόριθμος που να εξάγει πληροφορίες από το βίντεο εάν αυτό είναι εφικτό.
- Το tracking των εντοπιζόμενων αντικειμένων οφείλει να προσφέρει στον χρήστη αρκετές πληροφορίες για το αντικείμενο όπως, την θέση του, το μέγεθος του, την πορεία του και την μελλοντική πορεία του.

1.4 Οι λύσεις που προτείνονται

- Αυτοματοποίηση των παραμέτρων του συστήματος, τόσο κατά την εκκίνηση όσο και κατά την λειτουργία του σε πραγματικό χρόνο.
- Εύρεση βέλτιστων παραμέτρων λειτουργίας επίσης σε πραγματικό χρόνο.
- Ειδοποίηση όταν οι παράμετροι του συστήματος αλλάζουν δραστικά, γεγονός που ενδεχομένως υποδηλώνει κάποιο συμβάν μέσα στο βίντεο (Event triggered mechanism).
- Βελτίωση του χρόνου που απαιτείται υπολογιστικά για την επεξεργασία του βίντεο.
- Tracking των αντικειμένων μέσα σε αυτό.

Η καινοτομία αυτών:

- Πλήρως αυτόματη αξιολόγηση και παραμετροποίηση του συστήματος σε πραγματικό χρόνο.
- Μείωση του χρόνου υπολογιστικής επεξεργασίας κατά 40%.
- Tagging, Tracking και Predicting των περιοχών ενδιαφέροντος.

1.5 Τι περιέχεται στα κεφάλαια

- Στο κεφάλαιο 2 γίνεται μία λεπτομερής περιγραφή του συστήματος και των μερών που το αποτελούν. Περιγράφεται ο τρόπος αναγνώρισης και ταξινόμησης των περιοχών ενδιαφέροντος μέσα στο βίντεο, οι προδιαγραφές του συστήματος, οι περιπτώσεις που δεν εντοπίζονται περιοχές ενδιαφέροντος, καθώς και οι είσοδοι και οι έξοδοι αυτού. Ακόμα γίνεται αναφορά στο περιβάλλον εργασίας και στα εργαλεία που χρησιμοποιήθηκαν.
- Στο κεφάλαιο 3 αναλύονται οι μέθοδοι που χρησιμοποιεί το σύστημα για να αναγνωρίσει τις περιοχές ενδιαφέροντος καθώς και τα προβλήματα που αυτές αντιμετωπίζουν καθώς και οι λύσεις τους.
- Στο κεφάλαιο 4 γίνεται αναφορά στον αλγόριθμο πρόβλεψης της κίνησης των αντικειμένων.
- Στο κεφάλαιο 5 γίνεται μελέτη και αξιολόγηση της λειτουργίας του συστήματος με πραγματικά δεδομένα από πολλαπλά βίντεο ενώ γίνεται και μία σύνοψη των αποτελεσμάτων και της απόδοσης των βελτιώσεων που έγιναν στο αρχικό σύστημα με βάση τα βίντεο που μελετήθηκαν.
- Στο κεφάλαιο 6 περιγράφονται οι ενδεχόμενες μελλοντικές επεκτάσεις του συστήματος καθώς και γενικές παρατηρήσεις για αυτό.

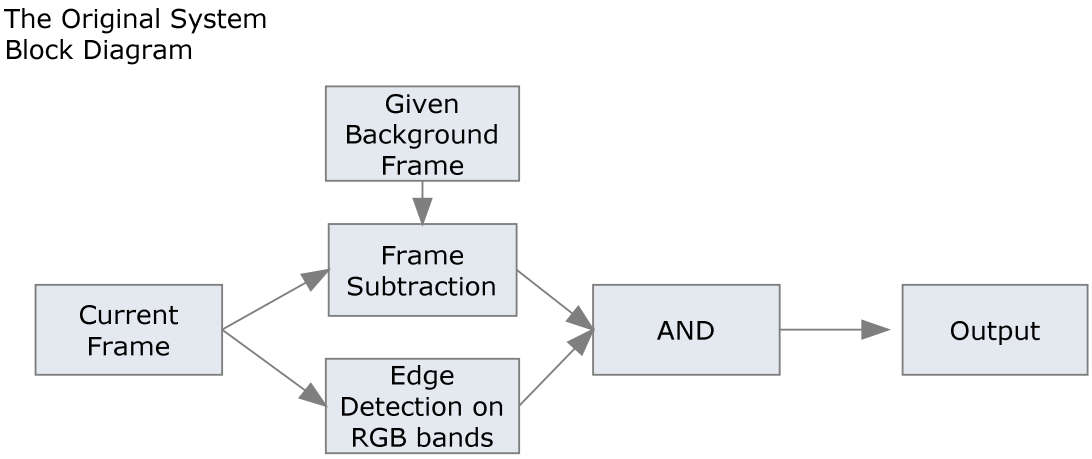
Κεφάλαιο 2. Το σύστημα

2.1 Γενικά για το σύστημα

Το ήδη υπάρχων σύστημα δεχόταν σαν είσοδο μία δεδομένη εικόνα φόντου (background frame) και παράλληλα με ένα επαναληπτικό βρόγχο προχωρούσε στις εξής δύο ενέργειες:

- Αφαιρούσε κάθε επόμενο frame από το δοσμένο frame φόντου εντοπίζοντας έτσι τις διαφορές pixel προς pixel στην ένταση της εικόνας με αποτέλεσμα να εντοπίζει που έχει υπάρξει αλλαγή στο περιεχόμενο της εικόνας.
- Σε κάθε ένα από τα επόμενα frame προχωρούσε σε εύρεση των ακμών μέσα σε αυτό.
- Με μία AND πράξη εντόπιζε τα σημεία ενδιαφέροντος και τα παρουσίαζε στην οθόνη μέσω ενός περιγράμματος.

Το block diagram του αρχικού συστήματος είναι το εξής:



Εικόνα 2.1.1 Αρχικό Σύστημα

2.2 Προδιαγραφές του συστήματος

Ο κώδικας πάνω στον οποίο βασίστηκε η ανάπτυξη εντόπιζε τα blob μέσα σε κάποιο βίντεο. Ο σκοπός της ανάπτυξης του εν λόγω κώδικα ήταν, σε συνδυασμό με διάφορες άλλες μεθόδους, να αποτελέσει ένα νέο, καινοτόμο σύστημα ασφαλείας αεροδρομίων. Σαν είσοδο έχουμε βίντεο από κάμερες ασφαλείας του αεροδρομίου. Τα χαρακτηριστικά αυτών είναι:

- Είναι ακίνητες σε ένα σημείο ή κινούνται αποκλειστικά στους 2 άξονες όντας τοποθετημένες.
- Είναι τοποθετημένες σε ένα σχετικά ψηλό σημείο ώστε η θέση τους να τους να τους επιτρέπει να καλύπτουν οπτικά ένα ευρύ οπτικό φάσμα.

Τα blob τα οποία πρωταρχικά μας ενδιέφεραν λοιπόν ήταν ο εντοπισμός των διαφόρων επιβατών και του προσωπικού του αεροδρομίου καθώς και τυχόν αποσκευές οι οποίες είχαν αφεθεί χωρίς επιτήρηση.

Η επέκταση του κώδικα ήταν προς τις εξής κατευθύνσεις:

- Ορισμός ταυτότητας για κάθε blob μέσα στο frame.

- Αναγνώριση του κάθε blob με βάση το μέγεθος του.
- Ανάλυση του κάθε blob με βάση την συμπεριφορά του μέσα στο βίντεο συνολικά.
- Εφόσον το blob κινείται πρόβλεψη της κίνησης του, τόσο όσον αφορά την κατεύθυνση όσο και την απόσταση.
- Μηχανισμός αυτοδιόρθωσης των παραμέτρων του συστήματος.
- Μηχανισμός εύρεσης βέλτιστων παραμέτρων για το σύστημα σε πραγματικό χρόνο.
- Μεγαλύτερη προσαρμοστικότητα του κώδικα σε εξωτερικές αλλαγές, όπως αλλαγή της φωτεινότητας.
- Όσο το δυνατόν οι όποιες αλλαγές να μην επιβαρύνουν ιδιαίτερα επιπλέον τον ήδη υπάρχον κώδικα, τόσο χρονικά όσο και από άποψη επεξεργαστικής ισχύος.

Προς αυτές τις κατευθύνσεις υλοποιήθηκαν δύο ανεξάρτητοι αλγόριθμοι:

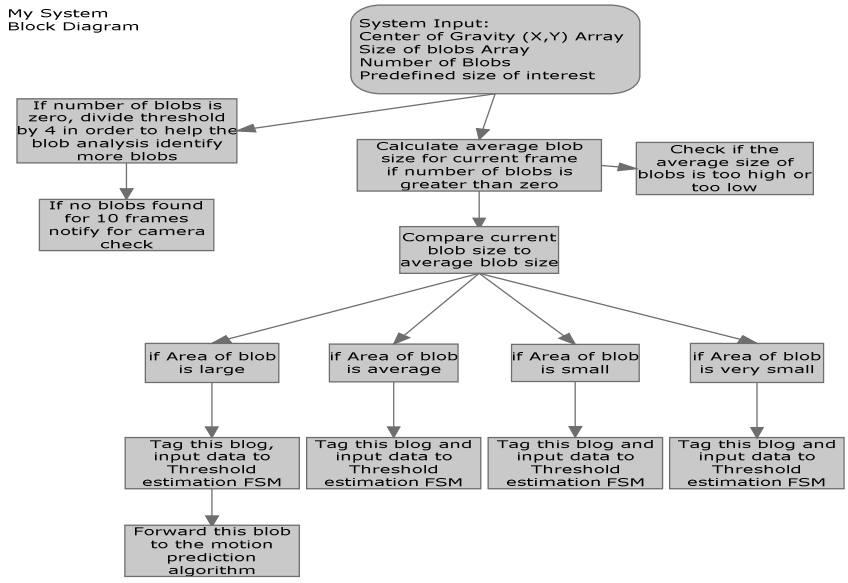
- Ένας αλγόριθμος πρόβλεψης της κίνησης.
- Ένας αλγόριθμος αξιολόγησης της τιμής κατωφλίου του συστήματος και σε περίπτωση που αυτή δεν είναι βέλτιστη, εύρεση αυτής.

Σαν δεδομένο είχαμε πάντα πως οι προσθήκες όφειλαν να συνάδουν με τις προδιαγραφές του αρχικού project δηλαδή:

- Χρήση του συστήματος σε πραγματικό χρόνο.
- Ταχύτητα στις λειτουργίες του ώστε ανά πάσα στιγμή είτε να μην χάνεται καθόλου πληροφορία είτε να χάνεται η λιγότερη δυνατή.
- Δυνατότητα διαχείρισης μέχρι και εκατό blob ανά frame.
- Μη ανάγκη επιτήρησης και επαναπροσδιορισμού των παραμέτρων του συστήματος.

Παράλληλα εντοπίστηκε πως το κυριότερο κόστος επεξεργαστικά-υπολογιστικά το είχε η εύρεση ακμών (Edge Detection) που κάνει το σύστημα. Για αυτό το λόγο κρίθηκε απαραίτητο να υλοποιηθεί ένας εναλλακτικός τρόπος να γίνεται αυτό που να απαιτεί λιγότερο χρόνο.

Με βάση τα παραπάνω το block diagram του συστήματος που υλοποιήθηκε είναι το εξής:



Εικόνα 2.2.1 Block Diagram Βελτιστοποιήσεων

Το σύστημα μας βασίζεται στην ύπαρξη blob για την λειτουργία του. Το να μην βρεθούν καθόλου blob σημαίνει ένα από τα παρακάτω:

- Μη ύπαρξη κίνησης μέσα στο frame.
- Ορισμός εσφαλμένου κατωφλίου (threshold).
- Βλάβη της κάμερας.

Το σύστημα μας είναι σχεδιασμένο ώστε από το κάθε frame να εξάγεται το μέγιστο ποσό πληροφόρησης. Η μη ύπαρξη πληροφορίας για επεξεργασία το ωθεί σε επαναπροσδιορισμό του κατωφλίου και συγκεκριμένα σε μείωση αυτού κατά 75% ώστε να βοηθηθεί το σύστημα στην εύρεση blob. Εφόσον η κάμερα λειτουργεί κανονικά το σύστημα θα εντοπίσει blob και κατόπιν το κατώφλι μας θα επαναπροσδιοριστεί στην βέλτιστη τιμή του. Παρά την μείωση κατά 75% του κατωφλίου, αν για περισσότερα από δέκα frame δεν εντοπίζονται blob τότε το σύστημα διακόπτει την λειτουργία του και εμφανίζει μήνυμα λάθους το οποίο προτρέπει τον χειριστή να ελέγξει την κάμερα για το αν λειτουργεί κανονικά. Ο κώδικας σε C είναι ο εξής:

```
if (TotalBlobs == 0)
        noblob++;

        if (noblob>9)
                printf("Please check the camera. No blobs found for %d frames.\n",noblob);
                noblob=0;

        THRESHOLD=THRESHOLD/4;
        printf("New THRESHOLD = %d(/4).\n",THRESHOLD);
```

Ένα από τα κύρια χαρακτηριστικά του συστήματος μας είναι η ταυτοποίηση των blob με βάση το μέγεθος τους. Προκειμένου να επιτευχθεί αυτό χρειαζόμασταν ένα αντικειμενικό, μη δεσμευτικό και ευέλικτο μέγεθος με το οποίο θα συγκρίναμε τα blob προκειμένου να τα αναγνωρίσουμε. Το μέγεθος που χρησιμοποιήσαμε είναι ο μέσος όρος των επιφανειών των blob σε κάθε frame. Αυτό επιλέχθηκε διότι:

- Δεν χρειάζεται να εισάγουμε διαφορετικά μεγέθη σύγκρισης για κάθε εγκατάσταση της εφαρμογής μας (πχ ασφάλεια αεροδρομίων, κάμερα ελέγχου κυκλοφορίας κτλ).
- Δεν επηρεάζεται η αναγνώριση των blob και η ταυτοποίηση τους από την απόσταση της κάμερας από αυτά. Όσο πιο μακριά η κοντά τοποθετούμε την κάμερα τόσο μικρότερα ή μεγαλύτερα αναλογικά γίνονται τα blob.
- Μας παρέχει ένα ισχυρό μέσο εκτίμησης της πληροφορίας μέσα στο frame. Ένας μικρός μέσος όρος σημαίνει πως δεν έχουμε blob ικανού μεγέθους μέσα στο frame μας ενώ ένας μεγάλος σημαίνει πως υπάρχει περίσσεια πληροφορίας μέσα σε αυτό. Συνήθως η τιμή του μέσου όρου, σε κανονικές συνθήκες, είναι κοντά στην τιμή ενδιαφέροντος που έχουμε σαν είσοδο. Αυτό σημαίνει πως υπάρχουν τόσο μεγάλα blob τα οποία και προωθούμε στον αλγόριθμο πρόβλεψης κίνησης αλλά και μικρά τα οποία οφείλονται σε σκιές ή στιγμιαίες αλλαγές της φωτεινότητας.
- Σε συνδυασμό με την ελάχιστη επιφάνεια ενδιαφέροντος που έχουμε σαν είσοδο, μας δίνεται η δυνατότητα να εντοπίσουμε και να εκτιμήσουμε τα blob ενδιαφέροντος πολύ ευκολότερα.
- Ο αλγόριθμος εύρεσης βέλτιστου κατωφλίου χρησιμοποιεί την ποσότητα αυτή για να εκτιμήσει την ροή της πληροφορίας μας και κατόπιν να επαναπροσδιορίσει το κατώφλι εκ νέου εάν αυτό χρειάζεται.

Ο τρόπος υπολογισμού του μέσου όρου είναι απλός: Αθροίζουμε τις επιφάνειες όλων των blob που έχουν βρεθεί και κατόπιν τις διαιρούμε με τον αριθμό τους. Δηλαδή:

```
size=0;

for(ij=0;ij<TotalBlobs;ij++)
size=size+Area[ij];
if(TotalBlobs>0)
size=size/TotalBlobs;
```

Όπου TotalBlobs ο αριθμός των blob που έχουν βρεθεί, Area[ij] η επιφάνεια του κάθε blob και size ο μέσος όρος αυτών. Το size επαναϋπολογίζεται σε κάθε νέο frame καθώς η πληροφορία που αυτό περιέχει είναι δυναμικό μέγεθος που διαρκώς μεταβάλλεται.

2.3 Είσοδοι συστήματος

Το σύστημα μας χρησιμοποίει σαν εισόδους δεδομένα από τον ήδη υπάρχον κώδικα καθώς και μια νέα είσοδο. Συγκεκριμένα:

- Τον αριθμό των blob που έχει βρει ο ήδη υπάρχων κώδικας.
- Έναν πίνακα με τα μεγέθη των blob που έχουν βρεθεί.
- Έναν πίνακα με τις θέσεις στον X άξονα των κέντρων βαρύτητας των blob που έχουν βρεθεί.
- Έναν πίνακα με τις θέσεις στον Y άξονα των κέντρων βαρύτητας των blob που έχουν βρεθεί.

Επιπλέον το σύστημα μας σαν μέσο ελέγχου χρησιμοποίει σαν είσοδο και ένα δεδομένο μέγεθος ενδιαφέροντος (size of interest). Αυτό το δεδομένο είναι ανεξάρτητο για κάθε υλοποίηση του συστήματος, πρέπει να οριστεί μία φορά, και έχει τις εξής δύο χρήσεις:

- Αποτελεί το ελάχιστο μέγεθος που πρέπει να έχει ένα blob προκειμένου να προχωρήσουμε σε πρόβλεψη της κίνησης αυτού.
- Ανάλογα με την κάθε υλοποίηση και εγκατάσταση ο αλγόριθμος εύρεσης του βέλτιστου κατωφλίου το χρησιμοποιεί προκειμένου να το επαναϋπολογίσει εάν αυτό κριθεί απαραίτητο.

Η εύρεση της τιμής αυτής της παραμέτρου είναι κατά βάση εμπειρική καθώς αυτή ποικίλει και βασίζεται κυρίως στην απόσταση της κάμερας από τα αντικείμενα ενδιαφέροντος ανάλογα με την εφαρμογή, την υλοποίηση και το επιθυμητό αποτέλεσμα. Όσο πιο μικρή επιφάνεια τα αντικείμενα ενδιαφέροντος καταλαμβάνουν μέσα στο frame τόσο πιο μικρή οφείλει να είναι και η τιμή της. Η τιμή αυξάνεται σε κάποια διαφορετική υλοποίηση του συστήματος, όπως κάμερες κυκλοφορίας όπου τα αντικείμενα παρατήρησης είναι τα οχήματα τα οποία ενδεχομένως θα καταλαμβάνουν μεγαλύτερη επιφάνεια μέσα στο frame. Η παράμετρος αυτή του συστήματος δίνεται μαζί με το αρχείο παραμέτρων (configuration file) στο σύστημα κάθε φορά που αυτό ξεκινά την λειτουργία του και δεν χρειάζεται να τροποποιηθεί εφόσον δεν αλλάξει το αντικείμενο παρατήρησης ή η απόσταση της κάμερας από αυτό.

Ακόμα το σύστημα μας χρησιμοποιεί πλήθος μεταβλητών που παίζουν τον ρόλο των παραμέτρων για αυτό. Οι μεταβλητές που χρησιμοποιήσαμε ήταν τύπων integer, double, long και char. Όλες αρχικοποιούνται στον αριθμό μηδέν, ενώ στην περίπτωση που

χρησιμοποιούμε πίνακα μεταβλητών ένα loop αρχικοποιεί κάθε μία θέση αυτού και πάλι στο μηδέν.

Οφείλει να αναφερθεί πως τα βίντεο τα οποία μελετούμε είναι καλής ποιότητας και με σαφή πληροφορία που επιθυμούμε να εξάγουμε. Η matrox imaging library παρουσιάζει συνοχή στην ονοματολογία των blob. Αυτό σημαίνει πως ένα blob που παρουσιάζεται σε μια συνέχεια από frame έχει διαρκώς τον ίδιο αριθμό (0,1,2 κτλ). Η MIL χρησιμοποιεί ένα σύστημα ονοματολογίας κάνοντας χρήση πινάκων όπου κάθε θέση του κάθε πίνακα περιέχει τα δεδομένα που αφορούν το ίδιο blob κάθε φορά. Συνεπώς η θέση 0 του πίνακα επιφανειών περιέχει το μέγεθος του blob 0, ενώ η αντίστοιχη θέση στους πίνακες συντεταγμένων περιέχει τις συντεταγμένες του blob 0.

2.4 Υλοποίηση του συστήματος

Το σύστημα μας υλοποιείται με ένα loop το οποίο εκτελείται τόσες φορές όσα και τα blob που έχουν βρεθεί. Μέσα στο loop λειτουργούν τρία διαφορετικά υποσυστήματα:

- Ο αλγόριθμος πρόβλεψης κίνησης blob.
- Ο μηχανισμός αξιολόγησης και εύρεσης βέλτιστου κατωφλίου συστήματος.
- Η αναγνώριση και κατηγοριοποίηση κάθε blob με βάση το μέγεθος του.

Σε κάθε επανάληψη ελέγχεται με μία συνθήκη ελέγχου κάθε ένα από τα blob που έχουν βρεθεί μέσα στο τρέχον frame. Αυτό που ελέγχεται συγκεκριμένα είναι η επιφάνεια του κάθε ενός σε σχέση με το μέσο όρο όλων των επιφανειών που υπολογίστηκε προηγουμένως και την επιφάνεια ενδιαφέροντος που έχουμε σαν είσοδο την οποία ονομάζουμε MY_SIZE. Τα blob αξιολογούνται ως εξής:

- Blob μεγαλύτερο από το μέσο όρο επιφανειών και μεγαλύτερο από την δοσμένη επιφάνεια ενδιαφέροντος. Δηλαδή: Area[ij]>MY_SIZE && Area[ij]>size. Σε αυτήν την περίπτωση θεωρούμε πως το blob αποτελεί blob ενδιαφέροντος και το προωθούμε στον αλγόριθμο πρόβλεψης κίνησης.
- Blob μεγαλύτερο από το μέσο όρο επιφανειών και μικρότερο από την δοσμένη επιφάνεια ενδιαφέροντος. Δηλαδή: Area[ij]>size && Area[ij]<MY_SIZE. Σε αυτήν την περίπτωση θεωρούμε πως το blob αποτελεί ένα ενδεχόμενο blob ενδιαφέροντος. Δεν το προωθούμε στον αλγόριθμο πρόβλεψης ενδιαφέροντος αλλά ελέγχουμε αν κινείται. Αναμένουμε να δούμε αν αποτελεί μέρος από ένα μεγαλύτερο blob το οποίο κινείται και τώρα πρωτοεμφανίζεται στο frame.
- Blob μικρότερο από το μέσο όρο επιφανειών και μεγαλύτερο από το 25% αυτού. Δηλαδή: Area[ij]<=size && Area[ij]>=size/2. Σε αυτήν την περίπτωση το blob αυτό για το σύστημα μας δεν παρουσιάζει ενδιαφέρον. Και πάλι ελέγχουμε αν κινείται ωστόσο. Αναμένουμε να δούμε την εξέλιξη του, δηλαδή αν στα επόμενα frame το blob αυτό θα μεγαλώσει η εάν θα μικρύνει.
- Blob μικρότερο από το 25% του μέσου όρου των επιφανειών. Δηλαδή: Area[ij]<size/2 Αυτά τα blob παρουσιάζουν το μικρότερο ενδιαφέρον. Κατά κύριο λόγο τα θεωρούμε θόρυβο καθώς δεν έχουν συνέχεια και οφείλονται συνήθως σε σκιές η σε στιγμιαίες αλλαγές του φωτισμού που παρουσιάζονται μέσα στο frame μας.

Παράλληλα με την ταυτοποίηση και κατάταξη όλων των blob του frame σε κάποια από τις παραπάνω κατηγορίες το σύστημα προχωρά και στα εξής:

- Τυπώνει στην οθόνη του συστήματος για κάθε ένα blob την επιφάνεια του και τη θέση του στον X και στον Y άξονα και ανάλογα με το μέγεθος του και το αν κινείται ή όχι το χαρακτηρίζει ως:

    - Large Blob (predicted)
    - Above average size Blob
    - Average size Blob
    - Particle
    - Moving
    - Non-moving

- Κάθε blob το οποίο εξετάζεται ορίζει και μία μοναδική κατάσταση στο σύστημα μας. Οι καταστάσεις αυτές αποτελούν είσοδο στον αλγόριθμο υπολογισμού βέλτιστου κατωφλίου και συγκεκριμένα στην μηχανή πεπερασμένων καταστάσεων (Finite State Machine – FSM) αυτού.

2.5 Έξοδοι συστήματος

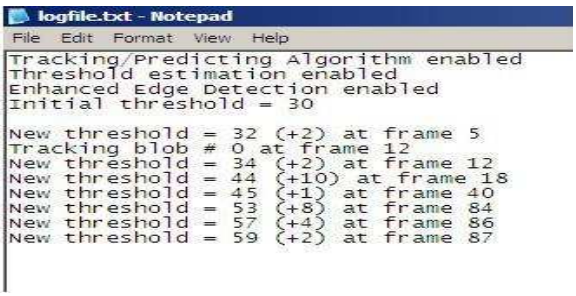
Προκειμένου να είναι λειτουργικό το σύστημα που υλοποιήθηκε, ιδιαίτερη προσοχή δόθηκε στις εξόδους αυτού προς τον χρήστη. Οι έξοδοι είναι τεσσάρων μορφών:

- Στην κονσόλα του συστήματος (DOS prompt).
- Στην οθόνη που προβάλει το βίντεο.
- Σε ένα αρχείο καταγραφής συμβάντων (logfile).
- Με ηχητικές ειδοποιήσεις όταν εντοπίζονται γεγονότα που χρήζουν το ενδιαφέρον του χειριστή μέσα στο βίντεο.

Το αρχείο καταγραφής συμβάντων περιέχει στοιχεία όπως:

- Ποια υποσυστήματα είναι ενεργά και ποια όχι.
- Την τιμή αρχικού κατωφλίου.
- Τις εναλλαγές στην τιμή του κατωφλίου και σε ποιο frame αυτές συμβαίνουν.
- Τα blob τα οποία προωθούνται στον αλγόριθμο εντοπισμού και παρακολούθησης αντικειμένων.

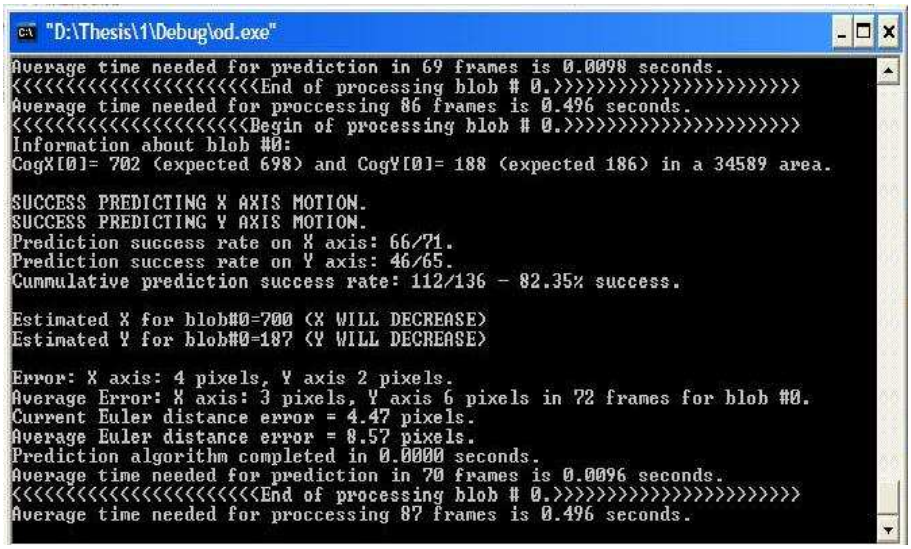
Η εικόνα των περιεχομένων του αρχείου καταγραφής είναι η εξής:



Εικόνα 2.5.1 Έξοδος στο αρχείο καταγραφής συμβάντων

Στην κονσόλα του συστήματος προβάλλονται ποικίλα στατιστικά στοιχεία που αφορούν την λειτουργία των υποσυστημάτων όπως:

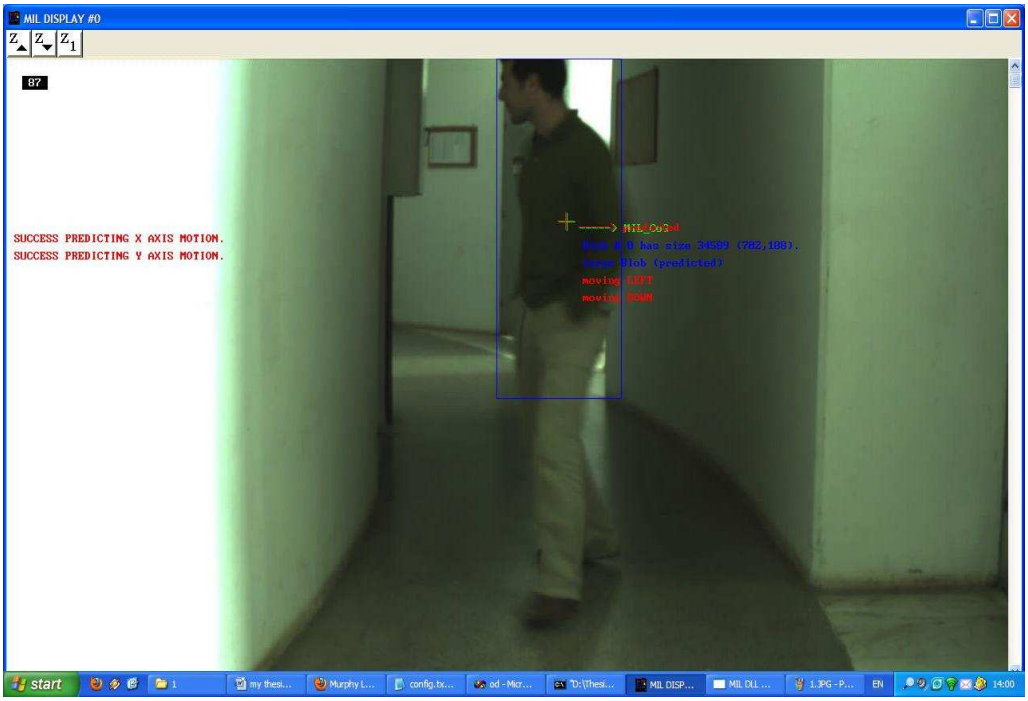
- Τα ποσοστά επιτυχίας των υποσυστημάτων.
- Τους χρόνους που απαιτήθηκαν για επεξεργασία των frame.
- Πληροφορίες για τα blob που έχουν εντοπιστεί.
- Τις εκτιμήσεις του αλγορίθμου πρόβλεψης για την μελλοντική κίνηση του blob.
- Τυχόν νέες τιμές κατωφλίου όπως αυτό έχει υπολογιστεί από τον αντίστοιχο μηχανισμό.



Εικόνα 2.5.2 Έξοδοι στην οθόνη του συστήματος - κονσόλα

Στην οθόνη του συστήματος που προβάλει το βίντεο προβάλλονται οι εξής πληροφορίες:

- Εμφανίζεται περίγραμμα που περικλείει το κάθε blob που έχει εντοπιστεί.
- Δίπλα στο κάθε blob εμφανίζονται πληροφορίες που αφορούν την θέση του, το μέγεθος του και την κίνηση ή μη αυτού.
- Εάν το blob ανήκει σε αυτά που προβλέπεται η κίνηση τους τότε τυπώνεται η πραγματική του θέση, η προβλεπόμενη θέση του, στοιχεία που αφορούν την κατεύθυνση του συγκεκριμένου blob, το αν η πρόβλεψη αυτή ήταν επιτυχημένη ή μη ανά άξονα καθώς και η ταχύτητα του.



Εικόνα 2.5.3 Έξοδοι στην οθόνη του συστήματος - βίντεο

2.6 Matrox Imaging Library

Η matrox imaging library (MIL) είναι ένα ισχυρό μέσο επεξεργασίας εικόνας. Αποτελείται από ένα σύνολο βιβλιοθηκών που προσφέρουν εντολές που αφορούν την επεξεργασία και την εξαγωγή πληροφοριών από εικόνες. Η πλατφόρμα πάνω στην οποία είναι δυνατή η χρήση της MIL είναι η Visual Basic αλλά και η Visual C++. Στην δική μας εφαρμογή χρησιμοποιήσαμε σαν γλώσσα προγραμματισμού την Visual C++ και σαν περιβάλλον εργασίας το Visual Studio 6.

Η MIL προσφέρει πληθώρα εντολών. Οι εντολές της αφορούν την χρήση πολλαπλών οθόνων και καμερών, ως μέσα εισόδου και εξόδου αντίστοιχα, και τον απόλυτο έλεγχο αυτών, τον πάσης φύσεως αριθμητικών και λογικών πράξεων με εικόνες, το φιλτράρισμα και την ομαλοποίηση αυτών, πλήρη ανάλυση δεδομένων εξαγόμενων από αυτές ως blob (binary large object), την αναγνώριση προτύπων, την εύρεση άκμων (Edge Detection) σε εικόνες και την διαμόρφωση μοντέλων OCR μεταξύ άλλων.

Η MIL δεν βασίζεται κατά κανόνα σε κοινές μεταβλητές της ANSI C για να αποθηκεύσει – διαχειριστεί τα δεδομένα. Αντίθετα χρησιμοποίει καταχωρητές (buffer) οι οποίοι και διαμορφώνονται ανάλογα την φύση της πληροφορίας. Οι καταχωρητές λειτουργούν ουσιαστικά σαν πίνακες πολλαπλών διαστάσεων. Για παράδειγμα μια RGB εικόνα απαιτεί την χρήση ενός πίνακα τριών διαστάσεων ενώ μια grayscale μίας διάστασης. Ασφαλώς παράμετροι όπως το βάθος χρώματος η το μέγεθος της εικόνας (άρα και του καταχωρητή) με βάση τα pixel της καθορίζονται κατά την δήλωση του καταχωρητή.

Οι εντολές της MIL είναι βελτιστοποιημένες ώστε να εκτελούνται όσο πιο γρήγορα και πιο βελτιστοποιημένα γίνεται. Η πραγματική απόδοση ωστόσο φαίνεται όταν εκτελούνται πάνω στην αντίστοιχη κάρτα του ψηφιακού επεξεργαστή σήματος (Digital Signal Processor – DSP). Όντας εντολές δομημένες πάνω στην αρχιτεκτονική του DSP η χρήση τους σε

λειτουργία συμβατότητας (μέσω του αντίστοιχου Hasp) δεν φανερώνει τις πραγματικές δυνατότητες της βιβλιοθήκης.

Παρά τις δυνατότητες της βιβλιοθήκης της Matrox ο αλγόριθμος που υλοποιήθηκε βασίζεται σε τυπική C καθώς το πρόβλημα της πρόβλεψης της κίνησης μέσα στο video που κληθήκαμε να επιλύσουμε δεν μπορούσε να επωφεληθεί από τις βελτιστοποιημένες εντολές της MIL. Τα δεδομένα εισόδου του αλγορίθμου ωστόσο προέρχονται από την επεξεργασία των εικόνων με χρήση των εντολών της εν λόγω βιβλιοθήκης και συγκεκριμένα των Blob Analysis εντολών.

Στην υλοποίηση του συστήματος μας γίνεται εκτενής αναφορά στον όρο blob. Blob (binary large object) είναι μια περιοχή ενδιαφέροντος μέσα στο βίντεο. Μπορεί να είναι οποιουδήποτε μεγέθους, να βρίσκεται οπουδήποτε μέσα στο frame και να αποτελεί είτε θόρυβο είτε χρήσιμη πληροφορία. Κάνοντας χρήση των εντολών Blob Analysis της MIL μπορούμε να μελετήσουμε την συμπεριφορά των blob και να εξάγουμε χρήσιμα συμπεράσματα για την δραστηριότητα μέσα στο βίντεο.

Κεφάλαιο 3. Αναγνώριση αντικειμένων σε εικόνα (frame)

3.1 Μέθοδοι εντοπισμού των αντικειμένων

Το σύστημα μας χρησιμοποιεί δύο μεθόδους για να εντοπίσει αντικείμενα μέσα στο frame:

- Frame subtraction [1]
- Edge detection [2], [Παράρτημα Α]

Ο συνδυασμός αυτών των μεθόδων είναι που συνιστά τόσο αποτελεσματικό το σύστημα μας. Οι περιοχές που συναληθεύονται μέσα στο frame με βάση τα αποτελέσματα των παραπάνω είναι οι ζητούμενες περιοχές ενδιαφέροντος. Σκοπός μας ήταν η βελτίωση της απόδοσης των δύο αυτών μεθόδων. Συγκεκριμένα είχαμε σαν σκοπό:

- Την απεξάρτηση της αφαίρεσης των frame που υλοποιεί η πρώτη μέθοδος από την αρχική εισαγωγική παράμετρο κατωφλίου
- Την μείωση του απαιτούμενου χρόνου για την ανίχνευση ακμών.

3.2 Αφαίρεση τρέχοντος frame από δεδομένο Background frame (Frame subtraction)

Η αφαίρεση υλοποιείται με μία αριθμητική πράξη μεταξύ του τρέχοντος frame και ενός κενού frame φόντου (Background frame). Οι περιοχές που εντοπίζονται αποτελούν σημεία στα οποία η διαφορά της έντασης είναι διαφορετική λόγο ύπαρξης κάποιου νέου αντικειμένου μέσα στο frame συγκριτικά με το κενό frame αναφοράς. Ο αριθμός και το μέγεθος των περιοχών αυτών είναι άμεσα εξαρτημένος από το κατώφλι που ορίζεται κατά την εκκίνηση του προγράμματος. Μέσω της συνάρτησης BackgroundUpdate η δεδομένη εικόνα αναφοράς αλλάζει ανά τακτά χρονικά διαστήματα σύμφωνα με την υπόθεση πως όταν ένα pixel μένει ίδιο για περισσότερα από τα μισά frame από την προηγούμενη ανανέωση τότε το τοποθετούμε στην νέα εικόνα φόντου. Ακόμα και έτσι το κατώφλι δεν είναι βέβαιο πως θα είναι το απόλυτα σωστό για την νέα εικόνα φόντου λόγω π.χ. αλλαγής του φωτισμού. Ένας μηχανισμός αξιολόγησης και επανεκτίμησης της τιμής του κατωφλίου λοιπόν ήταν απαραίτητος.

3.3 Γενικά για το κατώφλι

Το κατώφλι (threshold) αποτελεί την κυριότερη παράμετρο του συστήματος μας. Στην πρώτη έκδοση του προγράμματος το κατώφλι είχε μία σταθερή τιμή. Αυτή η τιμή ήταν ξεχωριστή για κάθε υλοποίηση του συστήματος, εξαρτιόταν από τις συγκεκριμένες παραμέτρους αυτού και ο υπολογισμός της απαιτούσε χρόνο καθώς ήταν καθαρά εμπειρικός και βρισκόταν πειραματικά. Άμεση ήταν λοιπόν η ανάγκη εύρεσης ενός αυτόματου μηχανισμού ελέγχου και προσδιορισμού του βέλτιστου κατωφλίου για κάθε υλοποίηση. Το κατώφλι οφείλει να επαναπροσδιορίζεται με κάθε αλλαγή φωτισμού μέσα στο βίντεο ή εν γένει όποτε συμβαίνει κάτι που να προσδίδει θόρυβο κατά την επεξεργασία και ανάλυση των blob μέσα σε αυτό. Ο σκοπός της έρευνας μας ήταν η υλοποίηση αυτού ακριβώς του μηχανισμού εκτίμησης βέλτιστου κατωφλίου σε πραγματικό χρόνο. Μέσω αυτού του μηχανισμού εκτιμάται ο θόρυβος που παρουσιάζεται μέσα στο frame κατά την λειτουργία του συστήματος ενώ παράλληλα είναι δυνατή η ανίχνευση γεγονότων που προκαλούν μεγάλες αλλαγές μέσα στην εικόνα.

Το κατώφλι, δηλαδή η μεταβλητή THRESHOLD του συστήματος, χρησιμοποιείται αποκλειστικά σε μία εντολή αυτού. Παρόλο που χρησιμοποιείται μόνο μία φορά, μια λάθος επιλογή στην τιμή του αρκεί για να καταστήσει το σύστημα ανίκανο να εντοπίσει, να επεξεργαστεί και να προβλέψει την κίνηση blob. Ουσιαστικά λοιπόν η λάθος επιλογή κατωφλίου αρκεί για κάνει την επεξεργασία των blob αδύνατη.

Η εντολή που χρησιμοποιείται η τιμή του THRESHOLD είναι η εξής:

```
MimBinarize(MilRankFiltered, MilBinImage, M_GREATER_OR_EQUAL, THRESHOLD, M_NULL);
```

Σε αυτήν την εντολή όλα τα pixel του buffer εκκίνησης (MilRankFiltered) που έχουν τιμή μεγαλύτερη ή ίση με την τιμή του THRESHOLD περνάνε στον buffer προορισμού (MilBinImage) ώστε να συνεχιστεί η ανάλυση των blob μόνο από pixel που έχουν μία ελάχιστη τιμή, αποκόπτοντας όσο το δυνατόν τον θόρυβο μέσα στο frame. Αξίζει να σημειωθεί πως ο buffer MilRankFiltered περιέχει όλες τις περιοχές που προέκυψαν από αφαίρεση του τρέχοντος frame από το δηλωμένο κενό background frame και αφού η διαφορά αυτή κανονικοποιήθηκε με ένα median φίλτρο. Με άλλα λόγια μία χαμηλή τιμή για το κατώφλι θα αφήσει να περάσει όλος ο θόρυβος μέσα στην επεξεργασία των blob που θα κάνουμε στην συνέχεια προκαλώντας προβλήματα, ενώ μια υψηλή τιμή για αυτό δεν θα επιτρέψει χρήσιμη πληροφορία να περάσει στερώντας από το σύστημα την δυνατότητα να εντοπίσει blob. Η εύρεση της βέλτιστης τιμής για το κατώφλι μπορεί να χρειαστεί πολύ χρόνο ώστε να βρεθεί πειραματικά. Ακόμα και τότε όμως αυτή θα πρέπει να επαναπροσδιοριστεί για κάθε διαφοροποίηση του συστήματος, είτε φυσική (π.χ. φωτισμός), είτε λειτουργική (π.χ. αλλαγή θέσης της κάμερας). Συνεπώς είναι φανερό πως δεν είναι εφικτό να οριστεί μία τιμή για το κατώφλι η οποία να λειτουργεί εξίσου σωστά για κάθε υλοποίηση σε κάθε περιβάλλον και με οποιεσδήποτε συνθήκες. Για αυτό το λόγο είναι απαραίτητη η ύπαρξη ενός μηχανισμού εύρεσης του βέλτιστου κατωφλίου ανά πάσα στιγμή.

3.4 Προδιαγραφές μηχανισμού υπολογισμού βέλτιστου κατωφλίου

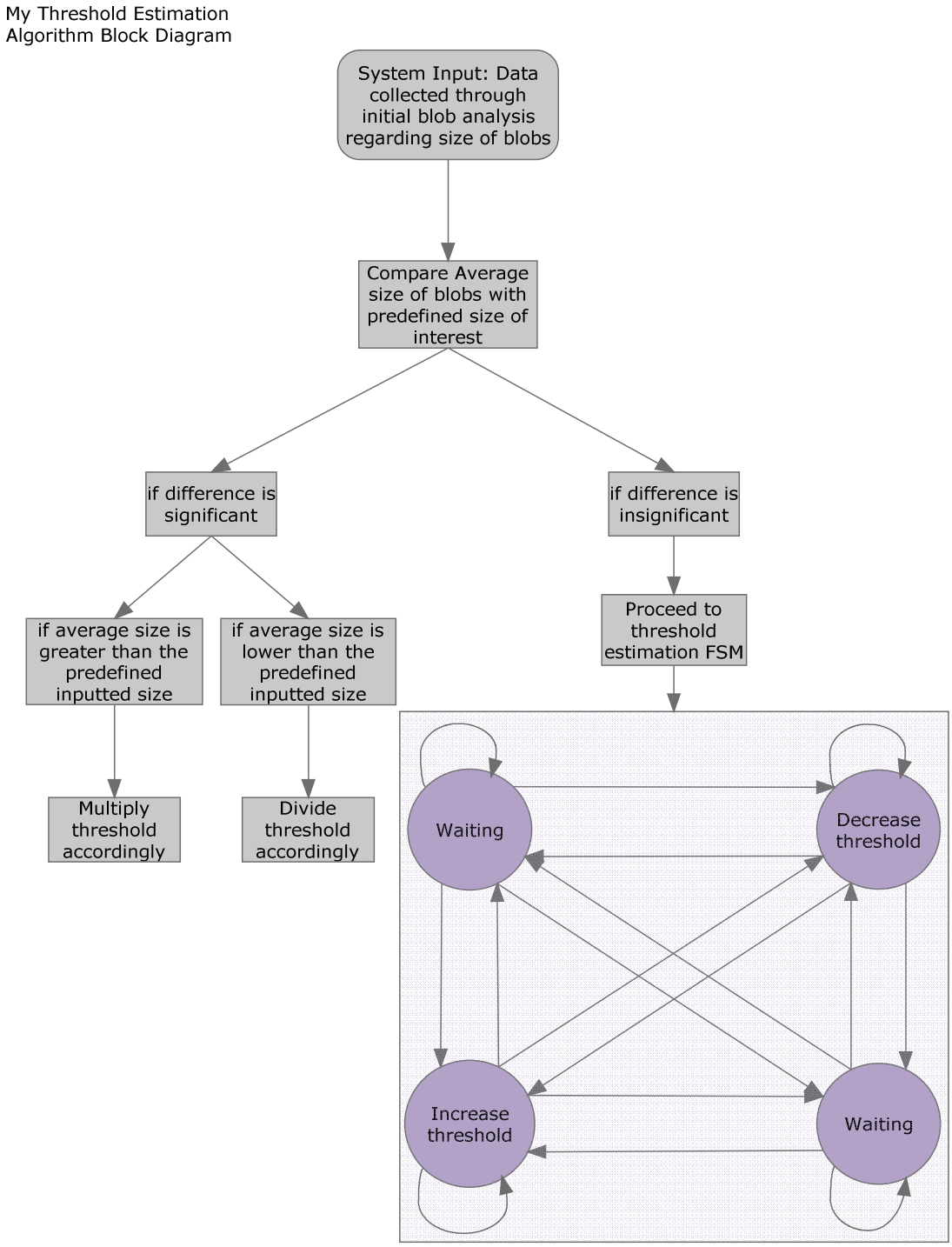
Οι απαιτήσεις που είχαμε από τον μηχανισμό υπολογισμού βέλτιστου κατωφλίου κατά την έρευνα μας ήταν οι εξής:

- Αξιολόγηση κατωφλίου σε πραγματικό χρόνο και σε κάθε frame.
- Γρήγορη σύγκλιση στην βέλτιστη τιμή.
- Όσο δυνατόν απώλεια ελάχιστης πληροφορίας.
- Αυτόματη λειτουργία χωρίς την χρήση καμίας εισόδου - χρήση αποκλειστικά της πληροφορίας μέσα στο βίντεο.
- Περιορισμός και ή δυνατόν απαλοιφή του θορύβου μέσα στο frame.

Για να επιτευχθούν τα παραπάνω υλοποιήθηκε ένας διπλός μηχανισμός ελέγχου και υπολογισμού του κατωφλίου. Το ένα μέρος αυτού λειτουργεί για κάθε blob μέσα σε κάθε frame ενώ το δεύτερο μέρος αξιολογεί την πληροφορία συνολικά μία φορά σε κάθε frame. Ο πρώτος μηχανισμός λειτουργεί γραμμικά πάνω στην τιμή του κατωφλίου, προσθέτοντας ή αφαιρώντας κάποια τιμή από/σε αυτό. Αντίθετα ο δεύτερος μηχανισμός επεμβαίνει γεωμετρικά πάνω στην τιμή του κατωφλίου πολλαπλασιάζοντας ή διαιρώντας την τιμή αυτού. Ο συνδυασμός αυτός επιλέχθηκε ώστε να υπάρχει αφενός η ταχύτητα στην σύγκλιση, και άρα η ελάχιστη απώλεια πληροφορίας, που προσφέρουν οι γεωμετρικές πράξεις, παράλληλα με την βελτιστοποίηση του κατωφλίου κατά μικρότερες τιμές που προσφέρουν οι

προσθαφαιρέσεις. Αποκλειστικά ο ένας ή ο άλλος μηχανισμός δεν θα μπορούσε να επιλεγεί καθώς αν χρησιμοποιούσαμε μόνο πρόσθεση ή αφαίρεση το κατώφλι στην περίπτωση που απείχε πολύ από την βέλτιστη τιμή θα αργούσε πολύ να συγκλίνει σε αυτή, ενώ μόνο με πολλαπλασιασμό ή διαίρεση αυτού θα συγκλίναμε κοντά στην βέλτιστη τιμή του αλλά δεν θα φτάναμε ποτέ ακριβώς σε αυτή.

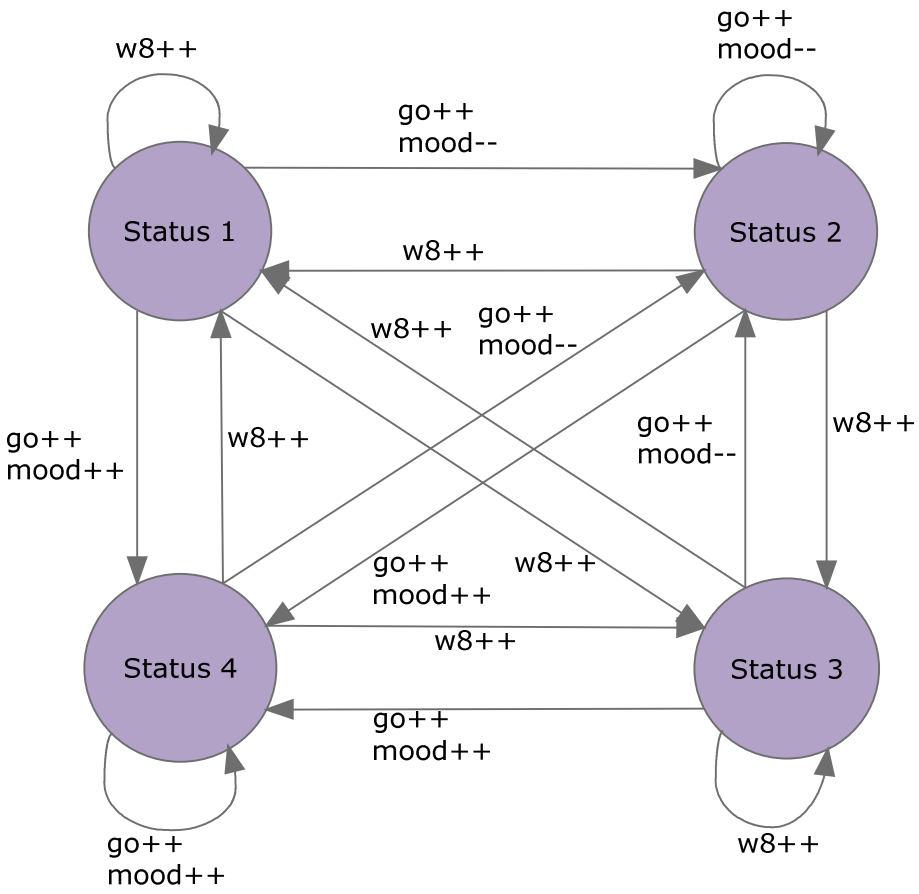
Το block diagram του μηχανισμού εκτίμησης κατωφλίου είναι το εξής:



Εικόνα 3.4.1 Block Diagram Αλγορίθμου εύρεσης βέλτιστου κατωφλίου

3.5 Μηχανισμός γραμμικής εκτίμησης βέλτιστου κατωφλίου

Ο μηχανισμός γραμμικής σύγκλισης κατωφλίου υλοποιείται με μία μηχανή πεπερασμένων καταστάσεων (Finite State Machine – FSM) η οποία λειτουργεί για κάθε blob που έχει βρεθεί μέσα στο frame και για όσα frame χρειαστεί μέχρι να προβεί σε αλλαγή της τιμής του κατωφλίου. Εφόσον η FSM κρίνει πως η αλλαγή της τιμής του κατωφλίου είναι απαραίτητη τότε προχωράει σε αυτήν και επαναρχικοποιούμε τις παραμέτρους της. Το Block Diagram της FSM είναι το εξής:



Εικόνα 3.5.1 Block Diagram FSM Αλγορίθμου εύρεσης βέλτιστου κατωφλίου

Στο σύστημα κατά την ταξινόμηση των blob με βάση το μέγεθος τους (με βάση των μέσο όρο των επιφανειών αυτών) ορίζουμε μία μεταβλητή κατάστασης τύπου integer αυτού με το όνομα status[ij] σε μία ανάλογη τιμή. Συγκεκριμένα:

- Status = 1 εάν το blob το οποίο βρέθηκε είναι μεγάλο και κάνουμε πρόβλεψη αυτού.
- Status = 2 εάν το blob το οποίο βρέθηκε είναι μεγάλο αλλά δεν κάνουμε πρόβλεψη αυτού.
- Status = 3 εάν το blob το οποίο βρέθηκε είναι μικρό αλλά όχι τόσο ώστε να το θεωρήσουμε θόρυβο.
- Status = 4 εάν το blob το οποίο βρέθηκε είναι πολύ μικρό οπότε και το θεωρούμε θόρυβο.

Όπως προαναφέρθηκε η τιμή του κατωφλίου ορίζει την ποσότητα της πληροφορίας που περνάει στο σύστημα. Όσο πιο μεγάλη η τιμή της τόσο λιγότερη πληροφορία περνάει και όσο μικρότερο τόσο περισσότερη. Ιδανικά η τιμή του κατωφλίου θα ήταν τέτοια ώστε θα επέτρεπε να περάσει αποκλειστικά μόνο χρήσιμη πληροφορία και καθόλου θόρυβος. Χρήσιμη πληροφορία θεωρούμε blob τα οποία είτε επιθυμούμε να προβλέψουμε την κίνηση τους είτε υποψήφια blob που αφορούν χρήσιμα αντικείμενα κινούμενα ή μη.

Η υλοποίηση του μηχανισμού αυτού υλοποιείται μέσω της FSM και τριών μεταβλητών με όνομα mood, go και w8 τύπου integer. Η προηγούμενη κατάσταση του συστήματος αποθηκεύεται σε μία μεταβλητή με όνομα oldstatus[ij] επίσης τύπου integer. Η θεωρία πίσω από την υλοποίηση αυτή είναι η εξής:

- Αν έχουμε ένα blob το οποίο προβλέπουμε (δηλαδή status = 1), θεωρούμε πως το κατώφλι μας είναι σε καλό σημείο και περιμένουμε (αυξάνουμε το w8).
- Αν έχουμε ένα blob το οποίο δεν είναι αρκετά μεγάλο για να προβούμε σε πρόβλεψη αυτού (δηλαδή status = 2), θεωρούμε πως το κατώφλι μας είναι μάλλον υψηλό και δεν επιτρέπει σε αυτό το blob να προωθηθεί στον αλγόριθμο πρόβλεψης κίνησης (αυξάνουμε το go και μειώνουμε το mood).
- Αν έχουμε ένα blob το οποίο είναι μικρό αλλά όχι τόσο ώστε να το θεωρήσουμε θόρυβο (δηλαδή status = 3), περιμένουμε να δούμε αν αυτό το blob θα μειωθεί και θα γίνει θόρυβος ή θα αυξηθεί και θα γίνει υποψήφιο blob για πρόβλεψη (αυξάνουμε το w8).
- Αν έχουμε ένα blob το οποίο είναι πολύ μικρό οπότε και το θεωρούμε θόρυβο (δηλαδή status = 4), θεωρούμε πως το κατώφλι μας είναι χαμηλό αφού επιτρέπει την διέλευση θορύβου μέσα στο frame (αυξάνουμε το go και το mood).

Η FSM σε κώδικα C υλοποιείται ως εξής:

```
if (status[ij]==1 && oldstatus[ij]==1)
w8++;
if (status[ij]==1 && oldstatus[ij]==2)
w8++;
if (status[ij]==1 && oldstatus[ij]==3)
w8++;
if (status[ij]==1 && oldstatus[ij]==4)
w8++;
```

```
if (status[ij]==2 && oldstatus[ij]==1)
mood--;
go++;
if (status[ij]==2 && oldstatus[ij]==2)
mood--;
go++;
if (status[ij]==2 && oldstatus[ij]==3)
mood--;
go++;
if (status[ij]==2 && oldstatus[ij]==4)
mood--;
go++;
```

```
if (status[ij]==3 && oldstatus[ij]==1)
w8++;
if (status[ij]==3 && oldstatus[ij]==2)
w8++;
if (status[ij]==3 && oldstatus[ij]==3)
w8++;
if (status[ij]==3 && oldstatus[ij]==4)
w8++;
```

```
if (status[ij]==4 && oldstatus[ij]==1)
mood++;
go++;
if (status[ij]==4 && oldstatus[ij]==2)
mood++;
go++;
if (status[ij]==4 && oldstatus[ij]==3)
mood++;
go++;
if (status[ij]==4 && oldstatus[ij]==4)
mood++;
go++;
```

```
if (mood>10 || mood<-10)
mood=0;
```

```
if (go>w8)
THRESHOLD=THRESHOLD+mood;
if (mood>0)
printf("New THRESHOLD = %d(+%d).\n",THRESHOLD,mood);

if (mood<0)
printf("New THRESHOLD = %d(%d).\n",THRESHOLD,mood);

w8=0;
go=0;
mood=0;
oldstatus[ij]=status[ij];
```

Σε κάθε επανάληψη του loop ελέγχου των blob ελέγχεται η τιμή του mood του go και του w8. Εάν το go είναι μεγαλύτερο από το w8, αυτό σημαίνει πως η ανάγκη για μεταβολή του κατωφλίου είναι άμεση. Για να συμβαίνει αυτό πρέπει είτε να υπάρχει πολύς θόρυβος μέσα στο frame, γεγονός που σημαίνει πως το κατώφλι είναι χαμηλό, είτε να υπάρχουν αρκετά υποψήφια blob για πρόβλεψη τα οποία όμως λόγω υψηλού κατωφλίου περιορίζονται. Στην περίπτωση που το w8 είναι μεγαλύτερο από το go η FSM συνεχίζει να συλλέγει δεδομένα και στα επόμενα frame αναμένοντας το go να γίνει μεγαλύτερο από το w8. Παράλληλα αν το mood είναι πάνω από το 10 τότε το επαναρχικοποιούμε στην τιμή 0 καθώς δεν επιθυμούμε να κάνουμε με αυτόν τον μηχανισμό δραστικές αλλαγές στην τιμή του κατωφλίου. Η μέγιστη μεταβολή λοιπόν που επιφέρει ο μηχανισμός αυτός στην τιμή του κατωφλίου είναι +10 ή -10. Μόλις η τιμή του go ξεπεράσει την τιμή του w8, η ποσότητα mood προστίθεται (η αφαιρείται αν έχει αρνητική τιμή) από την τιμή του κατωφλίου και οι τρεις παράμετροι (mood, go και w8) παίρνουν την τιμή μηδέν οπότε και η FSM επαναρχικοποιείται.

3.6 Μηχανισμός γεωμετρικής εκτίμησης βέλτιστου κατωφλίου

Η τιμή του size (δηλαδή ο μέσος όρος των επιφανειών όλων των blob μέσα στο τρέχον frame) είναι ενδεικτική της πληροφορίας που υπάρχει μέσα στο frame. Με βάση αυτήν γίνεται και η κατάταξη των blob σε περισσότερο ή λιγότερο χρήσιμα. Υπάρχει περίπτωση ωστόσο η τιμή αυτή να είναι είτε ιδιαίτερα υψηλή είτε ιδιαίτερα μικρή. Την κατάταξη αυτή την κάνουμε με βάση την επιφάνεια ενδιαφέροντος που έχουμε ορίσει εμείς στο σύστημα σαν είσοδο (MY_SIZE). Συγκεκριμένα αν η τιμή του size είναι πολύ μικρότερη του MY_SIZE τότε σίγουρα στο frame δεν υπάρχουν blob ικανά για πρόβλεψη γεγονός που μας υποδεικνύει πως το κατώφλι έχει υπερβολικά υψηλή τιμή. Αντίθετα αν η τιμή του size είναι κατά πολύ μεγαλύτερη της τιμής ενδιαφέροντος συμπεραίνουμε πως στο frame έχουν βρεθεί πολύ μεγάλα blob τα οποία δεν είναι δυνατόν να διαχειριστούμε στο μέγεθος που είναι. Και στις δύο αυτές περιπτώσεις η τιμή του κατωφλίου είναι εσφαλμένη γεγονός που μας οδηγεί στο συμπέρασμα πως η ανάγκη επανυπολογισμού αυτού είναι άμεση.

Μέσα από την έρευνα υπολογίστηκε πως η FSM υπολογισμού κατωφλίου μπορεί να διαχειριστεί με επιτυχία ένα σύνολο blob τα οποία θα έχουν ένα εύρος από μικρότερο του MY_SIZE * 10 και μεγαλύτερο από MY_SIZE / 5, όπου ως MY_SIZE ορίζουμε την ελάχιστη επιφάνεια που πρέπει να έχει ένα blob ώστε να προωθηθεί στον αλγόριθμο πρόβλεψης. Στις περισσότερες των περιπτώσεων αυτό είναι ίσο με το μέγεθος των blob που μας ενδιαφέρει να προβλέψουμε την κίνηση τους μέσα στο frame. Η FSM για size μεγαλύτερο από MY_SIZE * 10 και μικρότερο MY_SIZE / 5 χρειάζεται πολλά frame για να συγκλίνει στην ιδανική τιμή. Αυτό σημαίνει πως όση ώρα προσπαθεί να συγκλίνει η πληροφορία που περιέχουν τα frame χάνεται καθώς η τιμή του κατωφλίου καθυστερεί να συγκλίνει στην ιδανική. Ήταν απαραίτητος λοιπόν ένας μηχανισμός υπολογισμού του κατωφλίου στο φάσμα size < MY_SIZE / 5 και size > MY_SIZE * 10.

Ο μηχανισμός αυτός δεν θα μπορούσε να είναι γραμμικός καθώς όπως παρατηρήθηκε θα καθυστερούσε να συγκλίνει στην βέλτιστη τιμή. Και στις δύο περιπτώσεις θα έπρεπε όσο πιο σύντομα να αυξήσουμε το κατώφλι (αν το size είναι πολύ μεγάλο) είτε να το μειώσουμε (αν το size είναι πολύ μικρό). Με βάση αυτό επιλέχθηκε να πολλαπλασιαστεί ή να διαιρεθεί η τιμή του κατωφλίου ώστε να ξεφύγουμε από την εσφαλμένη τιμή αυτού. Οι αριθμοί που επιλέχτηκαν για να διαιρέσουμε ή να πολλαπλασιάσουμε το κατώφλι είναι οι 2,3 και 4. Όσο πιο μεγάλη η διαφορά του size από το MY_SIZE τόσο μεγαλύτερος αριθμός

επιλέγεται. Ο μηχανισμός αυτός λειτουργεί μία φορά σε κάθε frame και εφόσον η τιμή του size δεν επιστρέφει στο φάσμα που μπορεί να διαχειριστεί η FSM (MY_SIZE * 10 > size > MY_SIZE / 5) συνεχίζει να διαιρείται ή να πολλαπλασιάζεται ανάλογα με τις ανάγκες. Παρατηρήθηκε πως απαιτούνται 1 με 2 frame για να συγκλίνει η τιμή του κατωφλίου στο ζητούμενο εύρος οπότε και η FSM αναλαμβάνει να την αυξομειώσει ώστε να λάβει την βέλτιστη τιμή της.

Ο κώδικας C που υλοποιεί το κομμάτι αυτό του μηχανισμού είναι ο εξής:

if (size<MY_SIZE/15) THRESHOLD=THRESHOLD/4;	if (size>30*MY_SIZE) THRESHOLD=THRESHOLD*4;
if (size<MY_SIZE/10 && size>=MY_SIZE/15) THRESHOLD=THRESHOLD/3;	if (size>20*MY_SIZE && size<=30*MY_SIZE) THRESHOLD=THRESHOLD*3;
if (size<MY_SIZE/5 && size>=MY_SIZE/10) THRESHOLD=THRESHOLD/2;	if (size>10*MY_SIZE && size<=20*MY_SIZE) THRESHOLD=THRESHOLD*2;

Επειδή υπήρχε ο κίνδυνος του εγκλωβισμού της τιμής του κατωφλίου έτσι ώστε να διαιρείται και να πολλαπλασιάζεται διαρκώς (να μεταβαίνει από πολύ μικρή σε πολύ μεγάλη τιμή χωρίς να μπορεί να πάρει τον έλεγχο η FSM) επιλέχθηκε η κλιμακωτή διαβάθμιση των πράξεων. Παράλληλα η κλιμάκωση στις πράξεις βοηθάει στην καλύτερη σύγκλιση της τιμής του κατωφλίου καθώς στην πλειοψηφία των περιπτώσεων απαιτείται μόνο ένας πολλαπλασιασμός ή διαίρεση του κατωφλίου, ενώ αν χρειάζονται δύο τότε το κατώφλι συνήθως διαιρείται η πολλαπλασιάζεται με μικρότερο αριθμό την δεύτερη φορά.

3.7 Ανίχνευση ακμών μέσα στο frame (Edge Detection)

Η ανίχνευση ακμών είναι μία διαδικασία που με βάση τις έντονες διαφορές των τιμών έντασης γειτονικών pixel εντοπίζει το περίγραμμα κάποιου αντικειμένου. Αυτή η διαδικασία συμβαίνει για κάθε frame το οποίο έχουμε και σε συνδυασμό με την αφαίρεση των frame από ένα δεδομένο frame αναφοράς συνιστά ένα ισχυρό εργαλείο εντοπισμού αντικειμένων. Η προσέγγιση που χρησιμοποιείται υλοποιείται σε υποδιαίρεση κάθε εικόνας στα τρία βασικά χρώματα κόκκινο, πράσινο και μπλε. Κατόπιν εντοπίζονται όλες οι ακμές μέσα σε κάθε ένα από αυτά μέσω ενός Sobel φίλτρου και τα αποτελέσματα αποθηκεύονται σε έναν buffer.

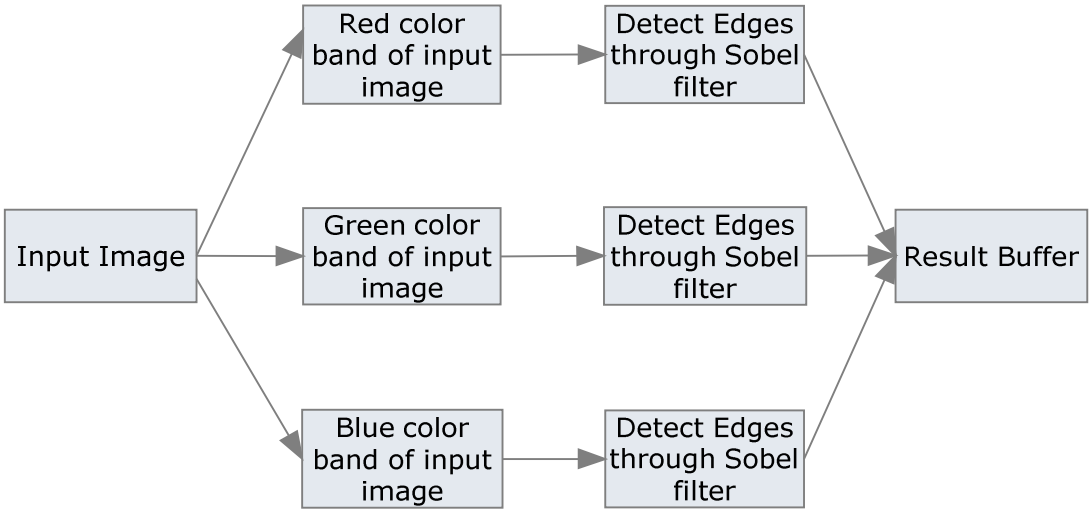
Ο κώδικας που υλοποιεί το Edge detection σε C είναι ο εξής:

```
MIL_ID  RGBEdgeDetection(MIL_ID MilRed, MIL_ID MilGreen, MIL_ID MilBlue, MIL_ID MilSystem)
{
        long    NumEdgeFound,
                TransparentColor;
        MIL_ID  MilEdgeRed,                     /* Edge Image identifier */
                MilEdgeGreen,
                MilEdgeBlue,
                MilEdgeResultRed,       /* Edge Result identifier.*/
                MilEdgeResultGreen,
                MilEdgeResultBlue,
                MilEdge;
        /* Allocate a edge finder context and a result buffer. */
MbufAlloc2d(M_DEFAULT, IMAGE_WIDTH, IMAGE_HEIGHT, 8+M_UNSIGNED, M_IMAGE+M_PROC+M_DISP , &MilEdge);
        MedgeAlloc(MilSystem, M_CONTOUR, M_DEFAULT, &MilEdgeRed);
        MedgeAllocResult(MilSystem, M_DEFAULT, &MilEdgeResultRed);
        MedgeAlloc(MilSystem, M_CONTOUR, M_DEFAULT, &MilEdgeGreen);
        MedgeAllocResult(MilSystem, M_DEFAULT, &MilEdgeResultGreen);
        MedgeAlloc(MilSystem, M_CONTOUR, M_DEFAULT, &MilEdgeBlue);
        MedgeAllocResult(MilSystem, M_DEFAULT, &MilEdgeResultBlue);
        //MdispAlloc(MilSystem, M_DEFAULT , "M_DEFAULT" , M_DEFAULT , &MilDisplay);
```

```
            /* Calculate edges for each color band */
            MedgeCalculate(MilEdgeRed, MilRed, M_NULL, M_NULL, M_NULL, MilEdgeResultRed, M_DEFAULT);
            MedgeCalculate(MilEdgeGreen, MilGreen, M_NULL, M_NULL, M_NULL, MilEdgeResultGreen, M_DEFAULT);
            MedgeCalculate(MilEdgeBlue, MilBlue, M_NULL, M_NULL, M_NULL, MilEdgeResultBlue, M_DEFAULT);
MedgeGetResult(MilEdgeResultRed, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumEdgeFound, M_NULL);/*
Get the number of found edges. */
            //printf("Number of edges (red) : %d\n",NumEdgeFound);
MedgeGetResult(MilEdgeResultGreen, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumEdgeFound, M_NULL);/*
Get the number of found edges. */
            //printf("Number of edges (green) : %d\n",NumEdgeFound);
MedgeGetResult(MilEdgeResultBlue, M_DEFAULT, M_NUMBER_OF_CHAINS+M_TYPE_LONG, &NumEdgeFound, M_NULL);/*
Get the number of found edges. */
            //printf("Number of edges (blue) : %d\n",NumEdgeFound);
            /* write in MilEdge the edge information of all color bands */
            MbufClear(MilEdge, 0); /* black background */
            MgraColor(M_DEFAULT, 255);
            MedgeDraw(M_DEFAULT, MilEdgeResultRed, MilEdge, M_DRAW_EDGE, M_ALL_EDGES , M_DEFAULT);
            MedgeDraw(M_DEFAULT, MilEdgeResultGreen, MilEdge, M_DRAW_EDGE, M_ALL_EDGES , M_DEFAULT);
            MedgeDraw(M_DEFAULT, MilEdgeResultBlue, MilEdge, M_DRAW_EDGE, M_ALL_EDGES , M_DEFAULT);

            MedgeFree(MilEdgeRed);
            MedgeFree(MilEdgeGreen);
            MedgeFree(MilEdgeBlue);
            MedgeFree(MilEdgeResultRed);
            MedgeFree(MilEdgeResultGreen);
            MedgeFree(MilEdgeResultBlue);
            //MdispFree(MilDisplay);
            return MilEdge;
}
```

Το block diagram του Edge detection είναι το εξής:



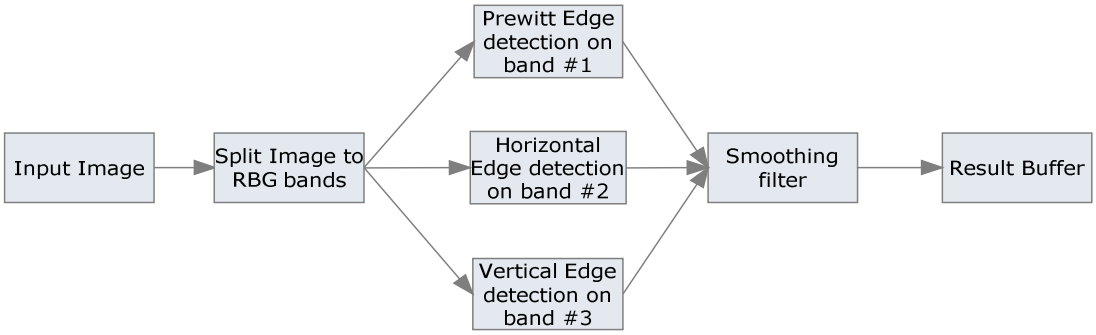
Εικόνα 3.7.1 Block Diagram Αρχικού Edge Detection

Αυτή η διαδικασία, αν και απαραίτητη, χρησιμοποιεί παραπάνω από το μισό επεξεργαστικό χρόνο για το κάθε frame. Συγκεκριμένα για frame μεγέθους 1600x1200 χρειάζεται πάνω από 0,25 δευτερόλεπτα για να ολοκληρωθεί (σε ένα συμβατικό υπολογιστή χωρίς την χρήση του εξειδικευμένου framegrabber). Ο σκοπός λοιπόν ήταν να υλοποιήσουμε την παραπάνω διαδικασία με έναν τρόπο λιγότερο δαπανηρό επεξεργαστικά. Με αυτόν τον τρόπο παρέχουμε στο σύστημα μας την δυνατότητα να διαχειριστεί εικόνες και βίντεο τόσο μεγαλύτερης ταχύτητας όσο και μεγέθους καθώς πλέον εξοικονομούμε περίπου το 50% της συνολικής επεξεργαστικής ισχύος.

3.8 Βελτιστοποιημένη ανίχνευση ακμών

Προκειμένου να υλοποιήσουμε μία ταχύτερη μέθοδο εντοπισμού των ακμών, η οποία να είναι εξίσου λειτουργική, χρησιμοποιήσαμε δεδομένες εντολές της MIL. Και πάλι η επεξεργασία μας γίνεται ανά μπάντα χρώματος. Συγκεκριμένα σε κάθε ένα frame αφού διαιρέσουμε την εικόνα στις τρείς μπάντες χρώματος επιλέγουμε μία από αυτές (διαδοχικά με την σειρά – αρχικά το κόκκινο, κατόπιν το πράσινο και μετά το μπλε) και σε αυτήν εφαρμόζουμε ένα Prewitt φίλτρο εντοπισμού των ακμών. Στην δεύτερη κάνουμε εντοπισμό των ακμών που βρίσκονται στον οριζόντιο άξονα και στην τρίτη εντοπισμό των ακμών που βρίσκονται στον κατακόρυφο άξονα. Οι τρείς αυτές μπάντες εναλλάσσονται σε κάθε ένα frame ώστε να υπάρχει συνοχή στον εντοπισμό των ακμών. Κατόπιν αποθηκεύουμε τα αποτελέσματα σε αντίστοιχους καταχωρητές και μέσω διαδοχικών πράξεων AND καταλήγουμε στο συνολικό αποτέλεσμα για ολόκληρο το frame. Στη συνέχεια πριν πάρουμε το τελικό αποτέλεσμα εφαρμόζουμε ένα φίλτρο ομαλοποίησης (Smooth).

Το block diagram του βελτιστοποιημένου Edge detection είναι το εξής:



Εικόνα 3.8.1 Block Diagram Βελτιστοποιημένου Edge Detection

Ο κώδικας που υλοποιεί το βελτιστοποιημένο Edge detection σε C είναι ο εξής:

```
if (k==0) {
MimConvolve(MilRed, MilEdge1, M_EDGE_DETECT2); //Prewitt filter
MimConvolve(MilGreen, MilEdge2, M_HORIZ_EDGE);
MimConvolve(MilBlue, MilEdge3, M_VERT_EDGE);
k=1;
}
if (k==1) {
MimConvolve(MilBlue, MilEdge1, M_EDGE_DETECT2); //Prewitt filter
MimConvolve(MilRed, MilEdge2, M_HORIZ_EDGE);
MimConvolve(MilGreen, MilEdge3, M_VERT_EDGE);
k=2;
}
if (k==2) {
MimConvolve(MilGreen, MilEdge1, M_EDGE_DETECT2); //Prewitt filter
MimConvolve(MilBlue, MilEdge2, M_HORIZ_EDGE);
MimConvolve(MilRed, MilEdge3, M_VERT_EDGE);
k=0;
}
MimArith(MilEdge1, MilEdge2, MilEdge4, M_AND );
MimArith(MilEdge1, MilEdge3, MilEdge5, M_AND );
MimArith(MilEdge4, MilEdge5, MilEdge6, M_AND );
MimConvolve(MilEdge6, MilEdge, M_SMOOTH); // Smoothing filter
```

Με αυτήν την διαδικασία μειώνεται ο χρόνος του Edge detection κατά περίπου 50% απαιτώντας 0,13 δευτερόλεπτα για να ολοκληρωθεί και πάλι για βίντεο διαστάσεων 1600x1200.

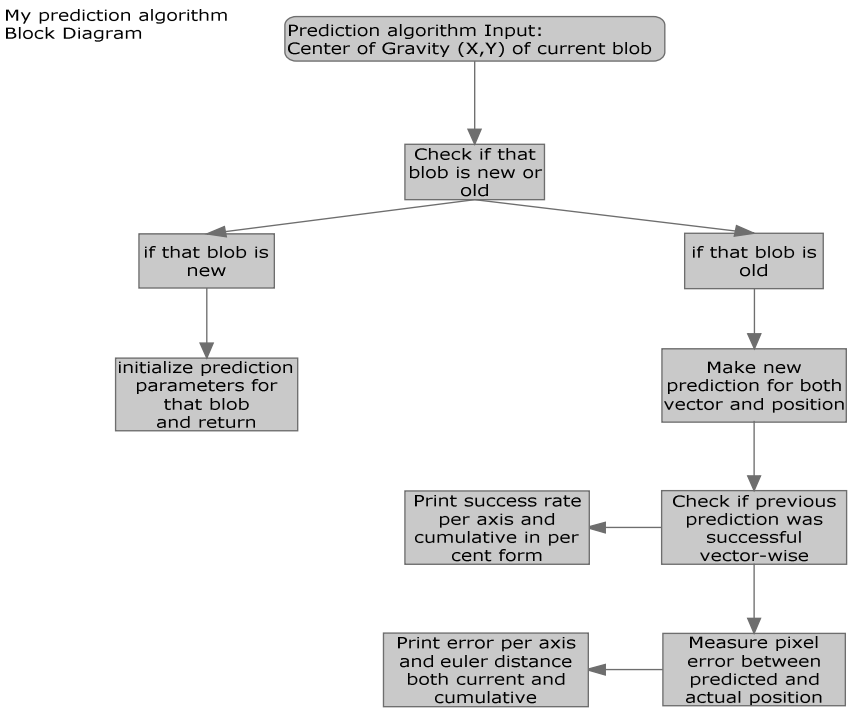
Κεφάλαιο 4. Αλγόριθμος παρακολούθησης αντικειμένων

4.1 Γενικά για τον αλγόριθμο

Η πρόβλεψη της κίνησης των blob μέσα στο βίντεο ήταν ένας από τους πρωταρχικούς και κύριους στόχους της έρευνας μας. Για να το πετύχουμε αυτό υλοποιήσαμε έναν αλγόριθμο ο οποίος με μοναδικό δεδομένο την θέση του αντικειμένου παρατήρησης στα 2 προηγούμενα frame υπολογίζει την θέση του στο επόμενο. Ο αλγόριθμος μας δεν περιορίζεται αποκλειστικά στην πρόβλεψη κίνησης μέσα σε ένα βίντεο αλλά θα μπορούσε, με ορισμό των απαραίτητων εισόδων, να χρησιμοποιηθεί σαν πλατφόρμα πρόβλεψης κίνησης κάθε δυναμικού φαινομένου, όπως μετεωρολογικά φαινόμενα, κίνηση μετεωριτών και αναχαίτιση πυραυλικών όπλων. Ασφαλώς στην περίπτωση που μελετάμε βίντεο όπως την δική μας η χρησιμότητα του εξαντλείται κατά βάση σε συστήματα παρακολούθησης και ασφάλειας, είτε σε κάποιο εσωτερικό χώρο όπως το αεροδρόμιο, είτε σε εξωτερικό όπως σε κάμερες ελέγχου κυκλοφορίας.

Η ιδέα για την υλοποίηση του αλγορίθμου είναι σχετικά απλή και καινοτόμος. Με βάση την παρατήρηση σε δύο frame της κίνησης ενός blob υποθέτουμε πως στο επόμενο θα κινηθεί με όμοιο τρόπο. Η κίνηση που θα προβλέψουμε δεν αφορά μόνο την κατεύθυνση της κίνησης του αλλά και την απόσταση την οποία θα διανύσει. Από ότι αποδεικνύεται από την έρευνα μας αυτή η υπόθεση είναι αρκετά ασφαλής και οι αποκλίσεις της πραγματικής κίνησης από την προβλεπόμενη είναι σχετικά μικρές. Επιτυχία εν γένει μπορούμε να θεωρήσουμε κάθε πρόβλεψη που είναι σωστή παραπάνω από τις μισές φορές, με δεδομένο πως η ρίψη νομίσματος είναι επιτυχημένη σε ποσοστό 50%. Ο αλγόριθμος μας υπολογίζει την κίνηση με επιτυχία πάνω από 80% από ότι παρατηρήθηκε κατά την έρευνα μας.

Το block diagram του αλγορίθμου πρόβλεψης κίνησης είναι το εξής:



Εικόνα 4.1.1 Block Diagram Αλγορίθμου Παρακολούθησης

4.2 Είσοδοι αλγορίθμου

Οι είσοδοι του αλγορίθμου είναι αποκλειστικά οι συντεταγμένες του blob ενδιαφέροντος στον χ και στον ψ άξονα. Θεωρούμε δεδομένο πως το blob που έχει επιλεγεί για να προβλεφτεί είναι ικανού μεγέθους με βάση την παράμετρο εισόδου του συστήματος της ελάχιστης επιφάνειας ενδιαφέροντος (MY_SIZE) όπως αυτή περιγράφεται παραπάνω. Ο αλγόριθμος μας ελέγχει τις συντεταγμένες του blob για δύο frame και κατόπιν προχωρά σε πρόβλεψη για το επόμενο. Ο αρχικός κώδικας είχε πρόβλεψη για εκατό blob ανά frame και θα θεωρείτο ασυνεπές ως προς αυτόν να μην διατηρηθεί αυτό του το χαρακτηριστικό. Ο αλγόριθμος μπορεί λοιπόν να εντοπίσει και να προβλέψει την κίνηση ως και εκατό blob ενδιαφέροντος. Ο κώδικας επίσης χρησιμοποιεί πλήθος μεταβλητών που απαιτούνται για την λειτουργικότητα του, οι οποίες είναι αρχικοποιημένες στο μηδέν όπως έχει αναφερθεί. Η Matrox Imaging Library, μέσω των blob analysis εργαλείων που διαθέτει, μας παρέχει δύο πίνακες. Έναν που περιέχει την θέση των blob στον χ άξονα και αντίστοιχα ένα πίνακα που περιέχει την θέση τους στον ψ άξονα. Αυτές οι μεταβλητές ονομάζονται κέντρα βαρύτητας (center of gravity) και στο σύστημα μας έχουν τα ονόματα CogX[ij] (για τον χ άξονα) και CogY[ij] (για τον ψ άξονα). Όπως αναφέρθηκε είναι πίνακες εκατό θέσεων λόγω του ότι υπάρχει πρόβλεψη για εκατό blob στον ήδη υπάρχων κώδικα.

4.3 Εκκίνηση του αλγορίθμου

Κάθε blob το οποίο μας ενδιαφέρει να προβλέψουμε την κίνηση του κατά την εκκίνηση του αλγορίθμου ελέγχεται αν είναι νέο η έχει προϋπάρξει ήδη στο βίντεο. Αυτό επιτυγχάνεται με τον ορισμό σε κάθε ένα blob που επιδιώκουμε να προβλέψουμε ενός μετρητή (counter) τον οποίο κατά την έναρξη του προγράμματος αρχικοποιούμε στην τιμή μηδέν. Αν το blob εμφανίζεται για πρώτη φορά τότε ορίζουμε τον μετρητή για αυτό το blob στην τιμή ένα. Κατόπιν αποθηκεύουμε την θέση του στον χ και στον ψ άξονα, σε δύο πίνακες με όνομα initx[ij] και inity[ij] για τον X και τον Y άξονα αντίστοιχα, και συνεχίζουμε το loop για το επόμενο blob που έχει βρεθεί. Ο πίνακας counter έχει εκατό θέσεις, μία για κάθε blob που είναι σε θέση ο κώδικας να εντοπίσει και, δυνητικά, να επιχειρήσει την πρόβλεψη του. Οι τιμές που μπορεί να πάρει ο πίνακας αυτός είναι είτε μηδέν αν δεν έχει εντοπιστεί κάποιο blob, είτε ένα αν ένα νέο blob εμφανίζεται προς πρόβλεψη. Κάθε θέση του πίνακα αντιστοιχεί και στο σύστημα ονομασίας της MIL. Συνεπώς η θέση counter[0] αντιστοιχεί στο blob που η MIL έχει αριθμήσει ως 0, η θέση 1 στο blob #1 και ούτω καθεξής. Στην περίπτωση που ένα blob ξεφύγει από τα όρια του frame, ελέγχουμε τη θέση του blob στο οποίο έχει δώσει το ίδιο όνομα η MIL και εφόσον αναγνωριστεί ως καινούργιο τότε επαναφέρουμε τον counter και όλες τις βοηθητικές μεταβλητές του αλγορίθμου πρόβλεψης στην τιμή μηδέν. Ο κώδικας σε C που υλοποιεί την ταυτοποίηση αυτή είναι ο εξής:

```
If (EulerDistance(CogX[ij],CogY[ij],CogXold[ij],CogYold[ij]) > sqrt(IMAGE_HEIGHT*IMAGE_HEIGHT+IMAGE_WIDTH*IMAGE_WIDTH))

counter[ij]=0;
flagx[ij]=0;
flagy[ij]=0;
oldestimx[ij]=0;
oldestimy[ij]=0;
```

Η θεωρία πίσω από αυτήν την υλοποίηση είναι πως το νέο blob που θα πάρει το όνομα του παλιού θα βρίσκεται σε μια αρκετά διαφορετική θέση μέσα στο frame.

Συγκεκριμένα θεωρούμε πως το blob είναι νέο εάν απέχουν τουλάχιστον όσο το μισό της διαγωνίου του frame.

4.4 Λειτουργία αλγορίθμου πρόβλεψης

Η πρόβλεψη που κάνουμε για κάθε blob αφορά την θέση του μέσα στο επόμενο frame με βάση την συμπεριφορά του στα 2 προηγούμενα. Η πρόβλεψη που κάνουμε αφορά τόσο την θέση που θα βρεθεί όσο και την κατεύθυνση προς τη οποία θα κινηθεί. Σε κάθε frame η MIL, μέσω των πινάκων CogX και CogY (Center of gravity – X axis, Center of gravity – Y axis), καταγράφει την θέση του κάθε blob μέσα στο τρέχον frame. Οι προβλέψεις μας γίνονται ξεχωριστά για την κίνηση του blob στον X και για τον Ψ άξονα. Στην περίπτωση που η συντεταγμένη του blob στον X ή στον Ψ άξονα έχει παραμείνει αμετάβλητη τότε και η πρόβλεψη μας παραμένει η ίδια με την προηγούμενη ενώ τυπώνουμε και ανάλογο μήνυμα στην οθόνη. Σε κάθε μία περίπτωση συγκρίνουμε την τρέχουσα θέση του blob με αυτήν που έχουμε αποθηκευμένη από το ακριβώς προηγούμενο frame και ανάλογα με το αποτέλεσμα της σύγκρισης προβλέπουμε προς τα πού θα κινηθεί στο επόμενο frame.

Οι περιπτώσεις κίνησης - πρόβλεψης είναι οι εξής:

- Η θέση του blob στον X άξονα αυξάνεται (άρα κινείται προς τα δεξιά).
- Η θέση του blob στον X άξονα μειώνεται (άρα κινείται προς τα αριστερά).
- Η θέση του blob στον Y άξονα αυξάνεται (άρα κινείται προς τα επάνω).
- Η θέση του blob στον Y άξονα αυξάνεται (άρα κινείται προς τα κάτω).
- Η θέση του blob στον X άξονα παραμένει αμετάβλητη.
- Η θέση του blob στον Y άξονα παραμένει αμετάβλητη.

Σε κάθε μία από τις περιπτώσεις μας οι προβλέψεις μας είναι πως το blob θα συνεχίσει να κινείται προς την ίδια κατεύθυνση που φαίνεται να κινείται. Ωστόσο στην περίπτωση που το blob παραμένει ακίνητο στον X ή στον Y άξονα τότε η εκτίμηση που κάνει ο αλγόριθμος είναι πως το blob θα αλλάξει πορεία (ανάλογα με το σε ποιόν άξονα παρατηρείται η μη μεταβολή της συντεταγμένης). Στην δική μας περίπτωση, που μελετάμε κίνηση ανθρώπων μέσα από βίντεο κάμερας ασφαλείας, αποδείχτηκε πως η υπόθεση αυτή βελτιώνει την επιτυχία της πρόβλεψης της κατεύθυνσης κίνησης του blob κατά περίπου 1%.

Για την υλοποίηση χρησιμοποιήθηκαν τέσσερις βοηθητικές μεταβλητές. Οι δύο αφορούν την προβλεπόμενη θέση του blob στον X και στον Y άξονα αντίστοιχα και έχουν το όνομα estimx[ij] και estimy[ij]. Ακόμα γίνεται χρήση ακόμα δύο μεταβλητών με όνομα flagx[ij] και flagy[ij]. Αυτές υποδηλώνουν ανάλογα με την τιμή τους την κατεύθυνση του blob. Αξίζει να σημειωθεί πως οι estimx[ij] και estimy[ij] είναι τύπου long όπως και τα CogX[ij] και CogY[ij] ενώ τα flagx[ij] και flagy[ij] είναι τύπου integer καθώς παίρνουν αποκλειστικά τιμές 0,1 και 2. Η προηγούμενη θέση του blob είναι αποθηκευμένη στους πίνακες initx[ij] και inity[ij]. Ο αλγόριθμος μας υλοποιείται ως εξής:

- Ελέγχουμε την τρέχουσα θέση του blob συγκριτικά με την προηγούμενη.
- Μετρούμε την μεταβολή της θέσης σε pixel.
- Προσθέτουμε η αφαιρούμε την ποσότητα αυτή στην τρέχουσα θέση ανάλογα με την κατεύθυνση της κίνησης.
- Ορίζουμε της μεταβλητές πρόβλεψης κατεύθυνσης στις κατάλληλες τιμές.

- Αποθηκεύουμε την τρέχουσα θέση ώστε να την συγκρίνουμε με αυτήν που θα έχει το ίδιο blob στο επόμενο frame.

Σε κώδικα C έχουμε με βάση τα παραπάνω:

if (CogX[ij]>initx[ij])
 	
estimx[ij]=CogX[ij]-initx[ij];
estimx[ij]=estimx[ij]+CogX[ij];
flagx[ij]=1;
	if (CogY[ij]>inity[ij])
	
	estimy[ij]=CogY[ij]-inity[ij];
	estimy[ij]=estimy[ij]+CogY[ij];
	flagy[ij]=1;

if (CogX[ij]<initx[ij])
 	
estimx[ij]=initx[ij]- CogX[ij];
estimx[ij]=estimx[ij]+CogX[ij];
flagx[ij]=2;
	if (CogY[ij]<inity[ij])
	
	estimy[ij]= inity[ij]-CogY[ij];
	estimy[ij]=estimy[ij]+CogY[ij];
	flagy[ij]=1;

Η διαδικασία αυτή γίνεται ξεχωριστά για τον X και τον Y άξονα αντίστοιχα όπως παρατηρούμε. Ο λόγος που συμβαίνει αυτό είναι διότι η MIL μας επιστρέφει τις συντεταγμένες του blob ξεχωριστά για κάθε έναν από τους δύο άξονες κίνησης.

4.5 Έλεγχος επιτυχίας πρόβλεψης κατεύθυνσης (Vector Error)

Ο παραπάνω αλγόριθμος προβλέπει τόσο κατεύθυνση όσο και απόσταση. Ζωτικής σημασίας ζήτημα ήταν να ελέγξουμε κατά πόσο επιτυχής είναι η πρόβλεψη κατεύθυνσης. Για αυτό το σκοπό χρησιμοποιήσαμε δύο μεταβλητές τύπου integer με όνομα flagx[ij] και flagy[ij]. Οι δύο αυτοί πίνακες έχουν εκατό θέσεις (όσα και τα blob που ο αλγόριθμος είναι σε θέση να εντοπίσει) και αρχικοποιούνται στην τιμή 0. Αντίστοιχα παίρνουν τις τιμές 1 και 2 με βάση τα παρακάτω:

- Εάν προβλέπουμε πως το blob θα κινηθεί προς τα δεξιά (αύξηση της X συντεταγμένης του) τότε το flagx[ij] παίρνει την τιμή 1.
- Εάν προβλέπουμε πως το blob θα κινηθεί προς τα αριστερά (μείωση της X συντεταγμένης του) τότε το flagx[ij] παίρνει την τιμή 2.
- Εάν προβλέπουμε πως το blob θα κινηθεί προς τα επάνω (αύξηση της Y συντεταγμένης του) τότε το flagy[ij] παίρνει την τιμή 1.
- Εάν προβλέπουμε πως το blob θα κινηθεί προς τα κάτω (μείωση της Y συντεταγμένης του) τότε το flagy[ij] παίρνει την τιμή 2.

Στο επόμενο frame ελέγχουμε την θέση του blob στον X και τον Y άξονα συγκριτικά με την προηγούμενη πραγματική θέση, την οποία έχουμε αποθηκεύσει στους πίνακες με όνομα CogXold[ij] και CogYold[ij] για την X και την Y συντεταγμένη αντίστοιχα. Δεν χρησιμοποιούμε την εκτιμώμενη θέση σαν μέτρο σύγκρισης καθώς σε αυτό το στάδιο του αλγορίθμου ασχολούμαστε αποκλειστικά με το πόσο επιτυχής ήταν η πρόβλεψη της κατεύθυνσης του blob. Παράλληλα με αυτήν την σύγκριση ελέγχουμε την τιμή των flagx[ij] και flagy[ij]. Αν η κίνηση ήταν σύμφωνη με την πρόβλεψη μας τότε αυξάνουμε την μεταβλητή successx[ij] ή την successy[ij] (πρόκειται για την μεταβλητή που μετράει τις επιτυχίες μας ανά άξονα) κατά 1. Στην περίπτωση που έχουμε αμετάβλητη την X ή την Y συντεταγμένη του blob τότε αυξάνουμε τον grvx[ij] ή τον grvy[ij] μετρητή μας κατά 1 αντίστοιχα. Ο μετρητής αυτός μας διευκολύνει στον υπολογισμό του ποσοστού επιτυχίας της πρόβλεψης, καθώς στην περίπτωση που έχουμε αμετάβλητη την συντεταγμένη μας τότε δεν

έχουμε και πρόβλεψη. Κατά συνέπεια το ποσοστό επιτυχίας του αλγορίθμου πρόβλεψης αφορά αποκλειστικά τις περιπτώσεις που υπάρχει πρόβλεψη, οπότε και ελέγχεται κατά πόσο επιτυχημένη ή μη ήταν.

Σε κώδικα C ο μηχανισμός ελέγχου επιτυχίας πρόβλεψης κατεύθυνσης έχει ως εξής:

if(CogX[ij]>CogXold[ij] && flagx[ij]==1)
successx[ij]++;
printf("SUCCESS PREDICTING X AXIS MOTION.\n");
	if(CogX[ij]<CogXold[ij] && flagx[ij]==1)
	printf("FAIL PREDICTING X AXIS MOTION.\n");

if(CogX[ij]<CogXold[ij] && flagx[ij]==2)
successx[ij]++;
printf("SUCCESS PREDICTING X AXIS MOTION.\n");
	if(CogX[ij]>CogXold[ij] && flagx[ij]==2)
	printf("FAIL PREDICTING X AXIS MOTION.\n");

if(CogY[ij]>CogYold[ij] && flagy[ij]==1)
successy[ij]++;
printf("SUCCESS PREDICTING Y AXIS MOTION.\n");
	if(CogY[ij]<CogYold[ij] && flagy[ij]==1)
	printf("FAIL PREDICTING Y AXIS MOTION.\n");

if(CogY[ij]<CogYold[ij] && flagy[ij]==2)
successy[ij]++;
printf("SUCCESS PREDICTING Y AXIS MOTION.\n");
	if(CogY[ij]>CogYold[ij] && flagy[ij]==2)
	printf("FAIL PREDICTING Y AXIS MOTION.\n");

if (CogX[ij]==CogXold[ij]) grvx[ij]++;	if (CogY[ij]==CogYold[ij]) grvy[ij]++;

Οι μεταβλητές CogXold[ij] και CogYold[ij] είναι τύπου long όπως και οι Cogx[ij] και Cogy[ij] ενώ οι flagx[ij], flagy[ij], grvx[ij] και grvy[ij] είναι τύπου integer καθώς χρησιμοποιούνται αποκλειστικά για τον έλεγχο του προγράμματος.

Ο υπολογισμός του ποσοστού επιτυχίας γίνεται με το άθροισμα τον επιτυχιών ανά άξονα διαιρούμενο με τον αριθμό των επαναλήψεων του αλγορίθμου πρόβλεψης κίνησης, μειωμένο κατά τον αριθμό των περιπτώσεων που δεν είχαμε κίνηση σε κάποιον άξονα. Συνεπώς το ποσοστό επιτυχίας υποδηλώνει το πόσο σωστά πρόβλεψε ο αλγόριθμος τις κινήσεις του blob. Σχετικά με το πόσο επιτυχημένα γίνεται η πρόβλεψη, λεπτομέρειες βρίσκονται στο κεφάλαιο 5.

Ο κώδικας εκτίμησης του ποσοστού επιτυχίας σε κώδικα C είναι ο εξής:

```
repet2use=repet[ij]*2;
repet2use=repet2use-4-grvx[ij]-grvy[ij];

successpercent=successx[ij]+successy[ij];
successpercent=successpercent/repet2use;
successpercent=successpercent*100;
printf("%2.2lf%% success.\n", successpercent);
```

Όπου repet[ij] είναι ο αριθμός των frame που το συγκεκριμένο blob παρουσιάζεται και repet2use, successpercent βοηθητικές μεταβλητές.

4.6 Έλεγχος απόκλισης προβλεπόμενης απόστασης (Pixel Error)

Παράλληλα με τον έλεγχο πρόβλεψης της κατεύθυνσης ο αλγόριθμος μας ελέγχει και κατά πόσο επιτυχημένα εκτίμησε την θέση του blob στο επόμενο frame. Η εκτίμηση του λάθους θέσης γίνεται και πάλι ξεχωριστά για κάθε έναν από τους δύο άξονες X και Y. Ασφαλώς είναι απαραίτητο να υπάρχει και ένα αποκλειστικό μέγεθος ελέγχου της απόστασης μεταξύ της προβλεφθέισας θέσης και της πραγματικής. Σαν τέτοιο μέγεθος χρησιμοποιήσαμε την απόσταση Euler. Η MIL χρησιμοποιεί τον αριθμό του κάθε pixel σαν τιμή συντεταγμένης. Συνεπώς όλες οι αποκλίσεις που υπολογίζουμε είναι μετρημένες σε pixel. Πρακτικά η λειτουργία του αλγορίθμου σε αυτό το στάδιο είναι να συγκρίνει την πραγματική

θέση του blob με αυτήν που είχε προβλεφτεί. Κατόπιν αφαιρούμε την μεγαλύτερη τιμή από την μικρότερη, για κάθε συντεταγμένη των δύο αξόνων μας, και αυτή η τιμή είναι το σφάλμα. Παράλληλα μετρούμε την απόσταση Euler μεταξύ της εκτιμώμενης θέσης και της πραγματικής ώστε να πάρουμε μία ενδεικτική τιμή για το σφάλμα συγκεντρωτικά και στους δύο άξονες. Το σφάλμα ανά pixel/άξονα αλλά και σαν απόσταση Euler υπολογίζεται τόσο για το τρέχων frame όσο και συνολικά σαν μέση τιμή για το τρέχον blob.

Ο κώδικας C για τον υπολογισμό του λάθους είναι ο παρακάτω:

```
if (oldestimx[ij]==CogX[ij] && oldestimy[ij]==CogY[ij])
printf("Precise estimate!\a\n");

if (oldestimx[ij]>CogX[ij])
errorx[ij]=oldestimx[ij]-CogX[ij];

if (oldestimy[ij]>CogY[ij])
errory[ij]=oldestimy[ij]-CogY[ij];

if (oldestimx[ij]<CogX[ij])
errorx[ij]=CogX[ij]-oldestimx[ij];

if (oldestimy[ij]<CogY[ij])
errory[ij]=CogY[ij]-oldestimy[ij];

errorsumx[ij]=errorsumx[ij]+errorx[ij];
errorsumy[ij]=errorsumy[ij]+errory[ij];

avg_eul=avg_eul+EulerDistance(CogX[ij],CogY[ij],oldestimx[ij],oldestimy[ij]);
printf("Average Euler distance error = %2.2lf pixels.\n", avg_eul/(repet[ij]-4));
```

Όπου oldestimx[ij] και oldestimy[ij] οι προβλέψεις για το τρέχον frame, Cogx[ij] και Cogy[ij] η πραγματική θέση του blob στο τρέχον frame. Τα errorx[ij], errory[ij], errorsumx[ij] και errorsumy[ij] χρησιμοποιούνται για τον υπολογισμό την απόκλισης στο frame και συγκεντρωτικά αντίστοιχα. Με χρήση της συνάρτησης EulerDistance υπολογίζεται η απόσταση μεταξύ της θέσης πρόβλεψης και της πραγματικής ενώ η avg_eul είναι βοηθητική μεταβλητή που χρησιμοποιούμε για τον υπολογισμό της μέσης απόστασης Euler. Σχετικά με τα αποτελέσματα για το σφάλμα απόκλισης μεταξύ πρόβλεψης και πραγματικής θέσης, λεπτομέρειες βρίσκονται στο κεφάλαιο 5.

Κεφάλαιο 5. Εφαρμογή – Αποτελέσματα

Για τον έλεγχο του συστήματος με πραγματικά δεδομένα χρησιμοποιήθηκαν διάφορα βίντεο με διαφορετικό μέγεθος ανάλυση και περιεχόμενο προκειμένου να δούμε πως οι βελτιώσεις του συστήματος μας επηρέασαν την απόδοση του συνολικά.

5.1 Madrid airport video

Πρόκειται για ένα βίντεο από μία κάμερα ασφαλείας του αεροδρομίου της Μαδρίτης κατά την έκρηξη μίας βόμβας. Το βίντεο είναι ιδιαίτερα χαμηλής ανάλυσης (320x240) και κακής ποιότητας (βρέθηκε στο youtube). Παρόλα αυτά είναι χρήσιμο για την μελέτη μας καθώς μας βοηθάει να δούμε πως το σύστημα μας αποδίδει σε χαμηλής ανάλυσης βίντεο τα οποία έχουν εξαρχής πολύ έντονο θόρυβο. Κατά την έναρξη του βίντεο ο αλγόριθμος εύρεσης βέλτιστου κατωφλίου συγκλίνει σε μία χαμηλή τιμή αφού το βίντεο έχει αρκετό θόρυβο και εντοπίζει κάποιους ταξιδιώτες που προχωρούν. Στην συνέχεια γίνεται η έκρηξη και η εικόνα του βίντεο γεμίζει σκόνη από αυτήν. Ο αλγόριθμος εύρεσης βέλτιστου κατωφλίου ανεβάζει απότομα την τιμή αυτού καθώς υποθέτει πως στην εικόνα ξαφνικά εμφανίζονται πολύ μεγάλα blob τα οποία είναι τα σύννεφα σκόνης. Αυτή η κατακόρυφη αύξηση της τιμής του κατωφλίου προκαλεί το σύστημα να τυπώσει ένα μήνυμα το οποίο ενημερώνει τον χειριστή για κάποιο συμβάν μέσα στο βίντεο.

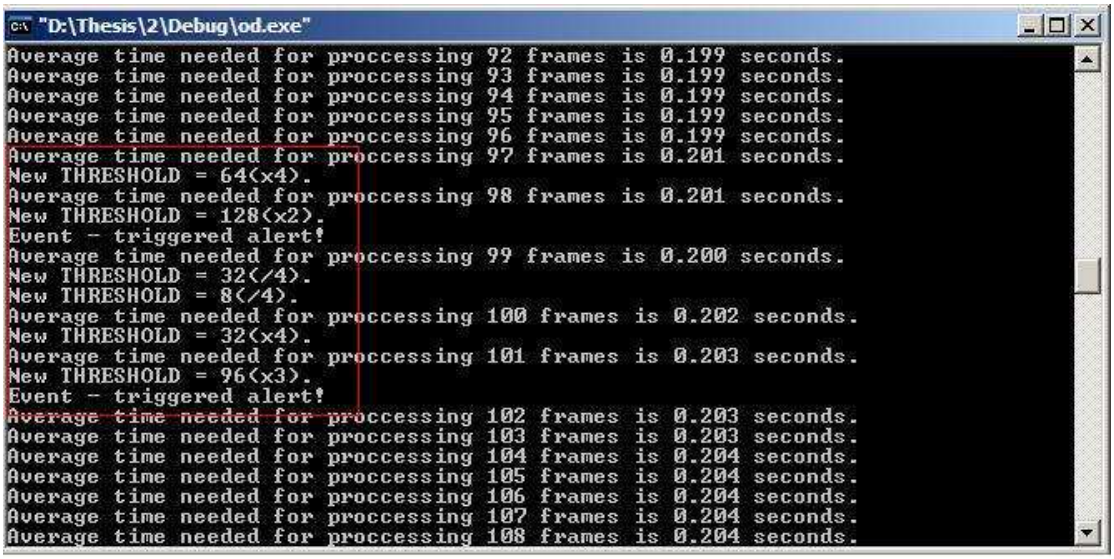
Αρχικά η εικόνα είναι η εξής:

Εικόνα 5.1.1 Βίντεο Αεροδρομίου Μαδρίτης - Αρχικά

Κατά την έκρηξη έχουμε:

Εικόνα 5.1.2 Βίντεο Αεροδρομίου Μαδρίτης - Τελικά

Και η τιμή του κατωφλίου:



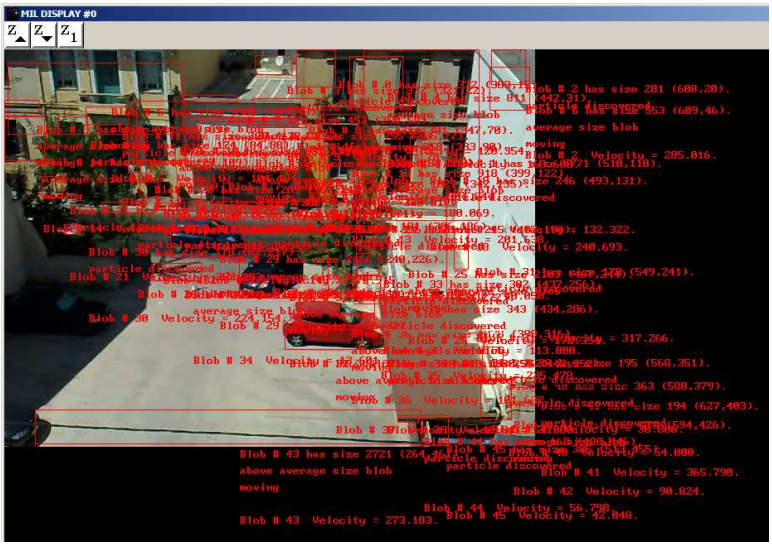
Εικόνα 5.1.3 Βίντεο Αεροδρομίου Μαδρίτης – Έξοδος στην κονσόλα

Από ότι βλέπουμε το σύστημα μας αντιδρά σωστά στην πληροφορία που δέχεται και προσαρμόζεται ανάλογα σε αυτές. Ο μέσος χρόνος επεξεργασίας με την αρχική μέθοδο ανίχνευσης ακμών είναι 0.315 δευτερόλεπτα ενώ με την βελτιστοποιημένη είναι 0.196 δευτερόλεπτα. Στην ανάλυση αυτή λοιπόν και σε ένα βίντεο με ιδιαίτερα ευμετάβλητη πληροφορία υπάρχει μία βελτίωση της τάξης του 38%. Ο μηχανισμός εύρεσης βέλτιστου κατωφλίου ήταν και στις δύο περιπτώσεις ενεργός.

5.2 Traffic camera video

Πρόκειται για ένα βίντεο το οποίο δείχνει έναν πολυσύχναστο δόμο των Χανίων, την λεωφόρο Βενιζέλου. Το βίντεο αυτό το τράβηξα ο ίδιος χρησιμοποιώντας το κινητό μου τηλέφωνο. Σε αυτό το βίντεο η κάμερα κινείται ελαφρά λόγω του έντονου αέρα. Έτσι προκαλούνται πολύ μικρές μετακινήσεις στο πλάνο και ο συνολικός αλγόριθμος ανακαλύπτει εσφαλμένες περιοχές ενδιαφέροντος μέσα στο frame. Ο λόγος που τις εντοπίζει είναι διότι το Edge Detection λόγω της πολύ έντονης ηλιοφάνειας βρίσκει αρκετές ακμές και το Background Subtraction λόγω την απειροελάχιστης κίνησης που παρουσιάζεται επιβεβαιώνει αυτές τις ακμές. Το βίντεο αυτό όμως είναι ενδεικτικό της αναγκαιότητας του Edge Detection αφού χωρίς αυτό έχουμε πάρα πολύ άσχημα αποτελέσματα κατά την ανάλυση του βίντεο.

Ένα frame του βίντεο χωρίς Edge Detection:



Εικόνα 5.2.1 Βίντεο κυκλοφορίας – Χωρίς Edge Detection

Και με Edge Detection:

Εικόνα 5.2.2 Βίντεο κυκλοφορίας – Με Edge Detection

Παράλληλα και σε αυτό το βίντεο ανάλυσης 640x480 έχουμε βελτίωση του απαιτούμενου χρόνου για Edge Detection καθώς η αρχική συνάρτηση χρειαζόταν 0.406 δευτερόλεπτα ενώ η βελτιστοποιημένη 0.216 δευτερόλεπτα. Υπάρχει δηλαδή μία βελτίωση της τάξης του 47%. Ο μηχανισμός εύρεσης βέλτιστου κατωφλίου ήταν και στις δύο περιπτώσεις ενεργός.

Σε αυτό το βίντεο γίνονται φανερά δύο σημεία:

- Πόσο απαραίτητη είναι η χρήση του edge detection.
- Πόσο σημαντικό είναι το σύστημα μας να βασίζεται σε βίντεο από στατική κάμερα.

Πράγματι παρατηρούμε πως χωρίς την χρήση της ανίχνευσης ακμών στο συγκεκριμένο βίντεο έχουμε εξαιρετικά υψηλό θόρυβο. Ο θόρυβος αυτός προέρχεται από την ελαφρά κίνηση της κάμερας η οποία αν και ανεπαίσθητη επηρεάζει σε πολύ μεγάλο βαθμό την απόδοση του συστήματος. Η αφαίρεση frame απαιτεί την εκ των προτέρων χρήση στατικής κάμερας με δεδομένο κενό frame φόντου. Όμως όταν παρουσιάζεται κίνηση στην κάμερα τότε εκ των πραγμάτων το φόντο δεν παραμένει ίδιο με αποτέλεσμα να έχουμε λάθος αποτελέσματα.

5.3 OPTAG video

Πρόκειται για ένα βίντεο που χρησιμοποιήθηκε για τον έλεγχο του συστήματος κατά την ενσωμάτωση του στο project OPTAG. Το βίντεο αυτό περιλαμβάνει frame χωρίς κίνηση, frame με κίνηση, μη ιδανικό φωτισμό με αποτέλεσμα να προκαλούνται σκιές και έχει την κάμερα τοποθετημένη χαμηλά, ώστε το πλάνο να μην είναι υπό γωνία, γεγονός που κάνει την πρόβλεψη κίνησης ακόμα δυσκολότερη. Ακόμα σε αυτό το βίντεο ο άνθρωπος (δηλαδή το blob μελέτης) κινείται με μη γραμμικό τρόπο, αλλάζει πορεία αλλά κινείται και προς την ίδια την κάμερα με αποτέλεσμα η επιφάνεια του blob που εντοπίζουμε να μεγαλώνει σημαντικά. Αυτό το βίντεο είναι ιδανικό για να μελετήσουμε την λειτουργία του συστήματος καθώς σε αυτό συντρέχουν πολλές μη ιδανικές παράμετροι και με βάση τα αποτελέσματα λειτουργίας να θεωρήσουμε πως σε πραγματικές συνθήκες (που οι παράμετροι θα είναι ιδανικότερες) το σύστημα μας θα αποδίδει καλύτερα. Το βίντεο που χρησιμοποιήθηκε έχει 87 frame και είναι ανάλυσης 1600 x 1200.

Εικόνα 5.3.1 Στιγμιότυπο από OPTAG βίντεο

Προκειμένου να ελέγξουμε το πόσο επιτυχημένα λειτουργεί ο αλγόριθμος πρόβλεψης σε συνδυασμό με τον μηχανισμό εκτίμησης βέλτιστου κατωφλίου έγιναν δύο πειράματα. Στο πρώτο ο μηχανισμός ήταν ανενεργός ενώ στο δεύτερο ενεργός. Τα πειράματα έγιναν για διάφορες τιμές κατωφλίου και στις δύο περιπτώσεις και τα μεγέθη που μετρήθηκαν ήταν:

- Το ποσοστό επιτυχίας πρόβλεψης κατεύθυνσης ανά άξονα.
- Το ποσοστό επιτυχίας πρόβλεψης κατεύθυνσης συνολικά.
- Το μέσο σφάλμα απόκλισης μεταξύ πραγματικής και εκτιμώμενης θέσης ανά άξονα.
- Η μέση απόσταση Euler ανάμεσα στην πραγματική και την εκτιμώμενη θέση.
- Τον αριθμό των frame που προβλέφθηκαν. Στο συγκεκριμένο βίντεο τα frame που παρουσιάζουν κίνηση του blob μελέτης είναι 74. Αν ο αριθμός είναι μικρότερος από αυτόν σημαίνει πως λόγω του ότι η τιμή κατωφλίου δεν έχει συγκλίνει έγκαιρα με αποτέλεσμα να έχει χαθεί η πληροφορία από τα υπόλοιπα frame.

5.4 Αποτελέσματα

Με βάση τα πειράματα όπως αυτά περιγράφηκαν παραπάνω και για τιμές κατωφλίου 1,30.45,100 και 200, στο OPTAG βίντεο, έχουμε τους παρακάτω πίνακες.

Με απενεργοποιημένο τον μηχανισμό εύρεσης βέλτιστου κατωφλίου έχουμε:

Threshold	1	30	45	100	200
Success X	X	43/56	66/71	18/24	4/6
Success Y	X	36/53	44/64	14/28	5/8
Total Vector Success	X	72.48%	81.48%	61.54%	64.29%
Pixel error X	X	39	3	13	2
Pixel error Y	X	45	7	17	4
Average Euler error	X	388.53	8.47	28.44	7.89
Number of predicted frames	X	58	72	30	8

Πίνακας 5.4.1 Μετρήσεις OPTAG βίντεο χωρίς τη χρήση του αλγορίθμου εύρεσης βέλτιστου κατωφλίου

Όπου X έχουμε μη μετρήσιμη ποσότητα λόγω αδυναμίας λειτουργίας του συστήματος, ενώ παρατηρούμε πως η τιμή 45 του κατωφλίου είναι η βέλτιστη, καθώς αποδίδει τα καλύτερα αποτελέσματα.

Με ενεργοποιημένο τον μηχανισμό εύρεσης βέλτιστου κατωφλίου έχουμε:

Threshold	1	30	45	100	200
Success X	64/72	66/71	66/71	65/70	61/65
Success Y	48/66	47/65	46/65	50/69	46/63
Total Vector Success	81.16%	83.09%	82.35%	82.73%	83.59%
Pixel error X	15	4	3	5	4
Pixel error Y	18	7	6	7	10
Average Euler error	25.87	9.01	8.57	9.94	12.34
Number of predicted frames	76	74	72	74	70

Πίνακας 5.4.2 Μετρήσεις OPTAG βίντεο με τη χρήση του αλγορίθμου εύρεσης βέλτιστου κατωφλίου

5.5 Συμπεράσματα

- Στο πρώτο πείραμα αποδεικνύεται πόσο απαραίτητη είναι η επιλογή της σωστής τιμής κατωφλίου. Κάθε άλλη τιμή εκτός από την 45 δεν δίνει αξιόλογα

αποτελέσματα. Αυτό ασφαλώς ήταν απόλυτα αναμενόμενο. Ακόμα και στην περίπτωση της βέλτιστης τιμής όμως, παρατηρούμε πως το σύστημα αποκρίνεται καλύτερα όταν ενεργοποιούμε τον μηχανισμό βέλτιστης επιλογής κατωφλίου.

- Τα ποσοστά επιτυχίας της πρόβλεψης κατεύθυνσης, όταν ενεργοποιούμε τον μηχανισμό βέλτιστης επιλογής κατωφλίου, είναι πάνω από 80% και το μέσο σφάλμα απόστασης Euler, αν εξαιρέσουμε την περίπτωση της τιμής 1, κυμαίνεται κάτω από 13 pixel.
- Στην περίπτωση του σφάλματος Euler για την τιμή του κατωφλίου ίση με 1 θεωρούμε πως είναι αυξημένο διότι επειδή ξεκινούμε με ένα πολύ μικρό κατώφλι, και ο αλγόριθμος χρειάζεται κάποια frame για να συγκλίνει στην βέλτιστη τιμή του, η πρώτη πρόβλεψη είναι παντελώς εσφαλμένη, καθώς τα blob έχουν υπερβολικό μέγεθος, συνεπώς και η απόσταση μεταξύ πραγματικής και προβλεπόμενης θέσης είναι μεγάλη. Στην περίπτωση αυτή έχουμε μέσο σφάλμα Euler 380 pixel στα πρώτα 2 frame. Στην συνέχεια ο αλγόριθμος συγκλίνει στην σωστή τιμή για το κατώφλι και προκύπτει μέσο σφάλμα Euler κάτω από 10 pixel όπως στις άλλες περιπτώσεις. Με τις επαναλήψεις τελικά το μέσο σφάλμα καταλήγει στην τιμή 25.87.
- Τα frame που παρουσιάζουν κίνηση του blob ενδιαφέροντος είναι 74 και ο αλγόριθμος στην χειρότερη περίπτωση (με 200 αρχικό κατώφλι) χρειάζεται 4 frame για να συγκλίνει στην σωστή τιμή. Δεδομένου πως η βέλτιστη τιμή κατωφλίου για αυτό το βίντεο είναι κοντά στην τιμή 45 και το 200 απέχει πολύ περισσότερο από το 45 σε σχέση με το 1 δικαιολογείται η καθυστέρηση στην σύγκλιση. Αντίθετα για τιμές κατωφλίου πιο κοντά στο 45 (30 και 100) ο αλγόριθμος συγκλίνει πολύ πιο γρήγορα με αποτέλεσμα να μην χάνουμε ούτε ένα frame πληροφορίας.
- Εντύπωση προκαλεί το ότι για αρχικό κατώφλι 1 ο αλγόριθμος μας μετράει 76 frame κίνησης αντί 74. Αυτό οφείλεται στο ότι στα πρώτα 2 frame περνά πολύ μεγάλη ποσότητα πληροφορίας στο σύστημα μας (λόγω του εξαιρετικά χαμηλού κατωφλίου) με συνέπεια ο αλγόριθμος πρόβλεψης να επιχειρεί να προβλέψει μία τεράστια επιφάνεια η οποία εντοπίστηκε (ακριβώς λόγω του χαμηλού κατωφλίου). Μέσω της γραμμικής αναπροσαρμογής αυτού όμως η περιοχή περιορίζεται σταδιακά σε εκείνη που μας ενδιαφέρει δηλαδή στο κινούμενο blob. Με βάση αυτό λογικό είναι να εντοπίζει παραπάνω frame πληροφορίας, ακριβώς διότι ακόμα και πριν να συγκλίνει έχει εντοπίσει blob τα οποία περιέχουν τα blob ενδιαφέροντος εν τέλει.
- Παρατηρούμε πως το σφάλμα που εντοπίζουμε στον Y άξονα είναι αρκετά μεγαλύτερο από αυτό στον X άξονα. Αυτό οφείλεται στο ότι στο βίντεο μελέτης που χρησιμοποιήθηκε το blob που προβλέπουμε κινείται περισσότερο στον κάθετο άξονα παρά στον οριζόντιο. Επόμενο είναι λοιπόν να υπάρχουν και περισσότερα σφάλματα σε αυτόν.

5.6 Μετρήσεις χρόνων

Το σύστημα συνολικά είναι ιδιαίτερα δαπανηρό από άποψη επεξεργαστικής ισχύος. Με δεδομένο πως θα πρέπει να είναι ικανό να λειτουργήσει σε πραγματικό χρόνο οφείλει να εξεταστεί πόσο οι επιπλέον προσθήκες που έγιναν επιβαρύνουν το σύστημα. Οι μετρήσεις χρόνου έγιναν σε ένα σύστημα με επεξεργαστή Core 2 Duo 2Ghz με 2 GB μνήμη RAM. Τόσο ο αλγόριθμος πρόβλεψης κίνησης όσο και ο μηχανισμός εύρεσης βέλτιστου κατωφλίου ήταν ενεργοί. Οι χρόνοι είναι μετρημένοι σε δευτερόλεπτα.

Κατώφλι	1	30	45	100	200
Μέσος χρόνος επεξεργασίας προσθηκών	0.0055	0.0109	0.0084	0.0225	0.0195
Μέσος χρόνος επεξεργασίας frame	0.458	0.459	0.451	0.476	0.480

Πίνακας 5.6.1 Μετρήσεις χρόνων προσθηκών για OPTAG βίντεο

Το σύστημα χωρίς τις προσθήκες έχει μέσο χρόνο επεξεργασίας 0.447 δευτερολέπτων. Με τιμή αρχικού κατωφλίου ίση με 200 το σύστημα φαίνεται να επιβαρύνεται με τον μεγαλύτερο επεξεργαστικό φόρτο. Υπάρχει μία επιπλέον καθυστέρηση 0.033 δευτερολέπτων ή 7.38%. Αντίθετα για αρχικό κατώφλι ίσο με 45 υπάρχει μία καθυστέρηση της τάξης των 0.004 δευτερολέπτων ή 0.89%. Αξίζει να σημειωθεί πως μόλις ο μηχανισμός υπολογισμού βέλτιστου κατωφλίου συγκλίνει στην βέλτιστη τιμή του ο επιπλέον επεξεργαστικός φόρτος τείνει σε ποσοστά κάτω του 1% όπως στην περίπτωση που έχουμε αρχικό κατώφλι 45, τιμή η οποία γνωρίζουμε πως βρίσκεται κοντά στην βέλτιστη για αυτό το βίντεο.

5.7 Συνολική βελτίωση με τη χρήση του βελτιστοποιημένου Edge Detection

Παρατηρείται πως η βελτιστοποιημένη μέθοδος εύρεσης ακμών είναι ταχύτερη από την αρχική. Συνολικά η βελτίωση παρουσιάζεται στον παρακάτω πίνακα.

	Original Edge Detection	Optimized Edge Detection	Improvement
1600x1200	0.458 sec.	0.252 sec.	45%
640x480	0.406 sec.	0.216 sec.	47%
320x240	0.315 sec	0.196 sec	38%

Πίνακας 5.7.1 Βελτίωση απόδοσης με τη χρήση του βελτιστοποιημένου Edge Detection

Όλες οι μετρήσεις έγιναν με ενεργοποιημένο τον μηχανισμό εύρεσης βέλτιστου κατωφλίου. Παράλληλα αν δούμε τα ποιοτικά χαρακτηριστικά της μεθόδου με βάση το τρίτο βίντεο και ενεργοποιημένο τον αλγόριθμο πρόβλεψης κίνησης έχουμε τον παρακάτω πίνακα για τις 5 διαφορετικές τιμές κατωφλίων που δοκιμάσαμε.

Threshold	1	30	45	100	200
Success X	70/75	69/73	68/72	66/70	70/76
Success Y	50/69	49/66	47/65	44/63	49/70
Total Vector Success	83.33%	84.89%	83.94%	82.71%	81.51%
Pixel error X	14	3	3	3	12
Pixel error Y	16	8	7	8	11
Average Euler error	24.14	10.04	9.04	9.60	19.29
Number of predicted frames	77	74	72	73	78

Πίνακας 5.7.2 Απόδοση εντοπισμού με τη χρήση του βελτιστοποιημένου Edge Detection

Παρατηρούμε πως με την χρήση της βελτιστοποιημένης ανίχνευσης ακμών κερδίζουμε σε χρόνο (μέχρι και 47%) ενώ δεν υπολειπόμαστε σε απόδοση του συστήματος όπως φαίνεται και από τον παραπάνω πίνακα. Έτσι το σύστημα παραμένει λειτουργικό και προσαρμόσιμο (χάρη στον αλγόριθμο εύρεσης βέλτιστου κατωφλίου) ενώ αποδίδει εξίσου καλά απαιτώντας σχεδόν τον μισό χρόνο για επεξεργασία σε σχέση με πριν.

Κεφάλαιο 6. Βελτιώσεις – Μελλοντικές επεκτάσεις

Ασφαλώς το σύστημα μας θα μπορούσε να δεχτεί ένα σημαντικό αριθμό παρεμβάσεων προκειμένου να καταστήσει την λειτουργία του καλύτερη και σταθερότερη. Ορισμένες από αυτές είναι:

- Χρήση pattern recognition μεθόδων για αναγνώριση των αντικειμένων που εντοπίζονται με δεδομένη μορφή σύμφωνα με ανάλογα πρότυπα αναγνώρισης.
- Δημιουργία βαρών για τις δύο μεθόδους (edge detection , background subtraction) που χρησιμοποιούμε ώστε σε συγκεκριμένες συνθήκες το σύστημα μας να αποδίδει καλύτερα και να γίνεται ακόμα πιο προσαρμόσιμο.
- Περαιτέρω αξιοποίηση του μηχανισμού υπολογισμού βέλτιστου κατωφλίου ώστε οι μεταβολές που αυτός κάνει να μπορούν να αντιστοιχηθούν σε γεγονότα μέσα στο βίντεο.
- Εμπλουτισμός του μηχανισμού tracking με μοντέλα που ανιχνεύουν την συνάντηση δύο ατόμων, αν ένα συγκεκριμένο αντικείμενο ακολουθεί μια συγκεκριμένη πορεία κτλ.
- Ιδιαίτερα χρήσιμος θα ήταν και ένας έξυπνος μηχανισμός απαλοιφής των blob που είναι θόρυβος μέσω pattern recognition, δηλαδή το σύστημα να απαλείφει blob που δεν ανήκουν σε ένα προκαθορισμένο pattern ενδιαφέροντος (πχ αυτοκίνητα, άνθρωποι κτλ). Παράλληλα χρήση pattern recognition μεθόδων μπορούν να χρησιμοποιηθούν για αναγνώριση των αντικειμένων που εντοπίζονται με δεδομένη μορφή σύμφωνα με ανάλογα πρότυπα σύγκρισης.
- Το σύστημα μας είναι σχεδιασμένο ώστε να λειτουργεί με στατικές κάμερες, με απλό ή πανοραμικού τύπου βίντεο. Δεν είναι σχεδιασμένο για να λειτουργεί με κινητές κάμερες διότι κατά την αφαίρεση frame είναι απαραίτητο να έχουμε ένα κενό frame φόντου το οποίο αν έχουμε αλλαγή της θέσης της κάμερας αλλάζει. Ένας τομέας πάνω στον οποίο θα μπορούσε να γίνει παραπάνω εργασία είναι λοιπόν η επέκταση της λειτουργικότητας του συστήματος ώστε αυτό να υποστηρίζει κινούμενες κάμερες.

Βιβλιογραφία

[1]Banh, HoltHaus Sacks “Moving Target Detection Method Using Two-Frame Subtraction and a Two Quadrant Multiplier.” United States Patent Office (1990).

[2]Tony Lindeberg “Edge detection and ridge detection with automatic scale selection.” Computational Vision and Active Perception Laboratory (CVAP), KTH (Royal Institute of Technology) (May 1996, Revised August 1998).

[3]Andreas Koschan, Mongi Abidi “Digital Color Image Processing.” Wiley-Interscience, (2008).

[4]Καρπετάς Γεώργιος, “Αυτόματη ανίχνευση των περιγραμμάτων των αγγειακών τοιχωμάτων σε εικόνες ενδοστεφανιαίου υπερηχογραφήματος”, Τμήμα Ιατρικής, Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (2001).

[5]Κελασίδη Ελένη, “Ολοκληρωμένο σύστημα οδομετρίας για κινούμενα ρομπότ με χρήση μετρήσεων από πολλαπλούς αισθητήρες”, Τμήμα Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών, Πανεπιστήμιο Πατρών (2004).

[6]Καραργύρης Αλέξανδρος, “Μελέτη και ανάπτυξη αλγορίθμων για την ανάλυση βρογχοσκοπικών εικόνων σε πραγματικό χρόνο”, Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών, Εθνικό Μετσόβιο Πολυτεχνείο (1999).

[7]Παπαβασιλείου Βασίλης Σταφυλάκης Θέμος Κατσούρος Βασίλης Καραγιάννης Γεώργιος,“Ανίχνευση Κειμένου σε Βίντεο”, Ινστιτούτο Επεξεργασίας του Λόγου/Ε.Κ «Αθηνά»(2002).

[8]Arun Hampapur, Jonathan H. Connell, Andrew W. Senior, Chiao-Fe Shu, Ying-Li Tian “System and method for managing moving surveillance cameras”, Albany, NY US Patent Office (2007).

[9]Nils T Siebel “Active People Tracking in Camera Images (Visual Surveillance)”. University of Kiel, Germany (2004).

[10]Kenneth M. Dawson-howe “Active Surveillance Using Dynamic Background Subtraction” Pennsylvania University, US (1996).

[11]TSI Lab ECE Dpt. TUC, “Investigate and implement tracking persons / objects at user interface”(2004).

[12]Matrox Imaging Library Help.

[13]MSDN Library v.2008

[14]Wikipedia (www.wikipedia.org).

Παράρτημα

Sobel Filter

The **Sobel operator** is used in image processing, particularly within edge detection algorithms. Technically, it is a discrete differentiation operator, computing an approximation of the gradient of the image intensity function. At each point in the image, the result of the Sobel operator is either the corresponding gradient vector or the norm of this vector. The Sobel operator is based on convolving the image with a small, separable, and integer valued filter in horizontal and vertical direction and is therefore relatively inexpensive in terms of computations. On the other hand, the gradient approximation which it produces is relatively crude, in particular for high frequency variations in the image.

Simplified description

In simple terms, the operator calculates the *gradient* of the image intensity at each point, giving the direction of the largest possible increase from light to dark and the rate of change in that direction. The result therefore shows how "abruptly" or "smoothly" the image changes at that point, and therefore how likely it is that that part of the image represents an *edge*, as well as how that edge is likely to be oriented. In practice, the magnitude (likelihood of an edge) calculation is more reliable and easier to interpret than the direction calculation.

Mathematically, the gradient of a two-variable function (here the image intensity function) is at each image point a 2D vector with the components given by the derivatives in the horizontal and vertical directions. At each image point, the gradient vector points in the direction of largest possible intensity increase, and the length of the gradient vector corresponds to the rate of change in that direction. This implies that the result of the Sobel operator at an image point which is in a region of constant image intensity is a zero vector and at a point on an edge is a vector which points across the edge, from darker to brighter values.

Formulation

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define **A** as the source image, and $\mathbf{G}_x$ and $\mathbf{G}_y$ are two images which at each point contain the horizontal and vertical derivative approximations, the computations are as follows:

$$\mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * \mathbf{A}$$

where * here denotes the 2-dimensional convolution operation.

The *x*-coordinate is here defined as increasing in the "right"-direction, and the *y*-coordinate is defined as increasing in the "down"-direction. At each point in the image, the resulting gradient approximations can be combined to give the gradient magnitude, using:

$$\mathbf{G} = \sqrt{\mathbf{G}_x^{\,2} + \mathbf{G}_y^{\,2}}$$

Using this information, we can also calculate the gradient's direction:

$$\mathbf{\Theta} = \arctan\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

where, for example, **Θ** is 0 for a vertical edge which is darker on the left side.

More formally

Since the intensity function of a digital image is only known at discrete points, derivatives of this function cannot be defined unless we assume that there is an underlying continuous intensity function which has been sampled at the image points. With some additional assumptions, the derivative of the continuous intensity function can be computed as a function on the sampled intensity function, i.e. the digital image. It turns out that the derivatives at any particular point are functions of the intensity values at virtually all image points. However, approximations of these derivative functions can be defined at lesser or larger degrees of accuracy.

The Sobel operator represents a rather inaccurate approximation of the image gradient, but is still of sufficient quality to be of practical use in many applications. More precisely, it uses intensity values only in a 3×3 region around each image point to approximate the corresponding image gradient, and it uses only integer values for the coefficients which weight the image intensities to produce the gradient approximation...

Extension to other dimensions

The Sobel operator consist of two separable operations [1]:

- Smoothing perpendicular to the derivative direction with a triangle filter : $h(-1) = 1, h(0) = 2, h(1) = 1$
- Simple central difference in the derivative direction : $h'(-1) = 1, h'(0) = 0, h'(1) = 1$

Sobel filters for image derivatives in different dimensions with $x, y, z, t \in (0, -1, 1)$:

1D: $h_x'(x) = h'(x)$;

2D: $h_x'(x,y) = h'(x)h(y)$

3D: $h_x'(x,y,z) = h'(x)h(y)h(z)$

4D: $h_x'(x,y,z,t) = h'(x)h(y)h(z)h(t)$

Thus as an example the 3D Sobel kernel in z-direction:

$$h_z'(:,:,-1) = \begin{bmatrix} +1 & +2 & +1 \\ +2 & +4 & +2 \\ +1 & +2 & +1 \end{bmatrix} \quad h_z'(:,:,0) = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \quad h_z'(:,:,1) = \begin{bmatrix} -1 & -2 & -1 \\ -2 & -4 & -2 \\ -1 & -2 & -1 \end{bmatrix}$$

Technical details

As a consequence of its definition, the Sobel operator can be implemented by simple means in both hardware and software: only eight image points around a point are needed to compute the corresponding result and only integer arithmetic is needed to compute the gradient vector approximation. Furthermore, the two discrete filters described above are both separable:

$$\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix} \qquad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} \begin{bmatrix} 1 & 2 & 1 \end{bmatrix}$$

and the two derivatives $\mathbf{G}_x$ and $\mathbf{G}_y$ can therefore be computed as

$$\mathbf{G}_x = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} * \left(\begin{bmatrix} +1 & 0 & -1 \end{bmatrix} * \mathbf{A} \right) \quad \text{and} \quad \mathbf{G}_y = \begin{bmatrix} +1 \\ 0 \\ -1 \end{bmatrix} * \left(\begin{bmatrix} 1 & 2 & 1 \end{bmatrix} * \mathbf{A} \right)$$

In certain implementations, this separable computation may be advantageous since it implies fewer arithmetic computations for each image point.

Prewitt Filter

Prewitt is a method of edge detection in image processing which calculates the maximum response of a set of convolution kernels to find the local edge orientation for each pixel.

Description

Various kernels can be used for this operation. The whole set of 8 kernels is produced by taking one of the kernels and rotating its coefficients circularly. Each of the resulting kernels is sensitive to an edge orientation ranging from 0° to 315° in steps of 45°, where 0° corresponds to a vertical edge.

The maximum response for each pixel is the value of the corresponding pixel in the output magnitude image. The values for the output orientation image lie between 1 and 8, depending on which of the 8 kernels produced the maximum response.

This edge detection method is also called edge template matching, because a set of edge templates is matched to the image, each representing an edge in a certain orientation. The edge magnitude and orientation of a pixel is then determined by the template that matches the local area of the pixel the best.

The Prewitt edge detector is an appropriate way to estimate the magnitude and orientation of an edge. Although differential gradient edge detection needs a rather time-consuming calculation to estimate the orientation from the magnitudes in the x- and y-directions, the Prewitt edge detection obtains the orientation directly from the kernel with the maximum response. The set of kernels is limited to 8 possible orientations; however experience shows that most direct orientation estimates are not much more accurate.

On the other hand, the set of kernels needs 8 convolutions for each pixel, whereas the set of kernel in gradient method needs only 2, one kernel being sensitive to edges in the vertical direction and one to the horizontal direction. The result for the edge magnitude image is very similar with both methods, provided the same convolving kernel is used.

Formulation

Mathematically, the operator uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives - one for horizontal changes, and one for vertical. If we define $\mathbf{A}$ as the source image, and $\mathbf{G_x}$ and $\mathbf{G_y}$ are two images which at each point contain the horizontal and vertical derivative approximations, the latter are computed as:

$$\mathbf{G_x} = \begin{bmatrix} -1 & 0 & +1 \\ -1 & 0 & +1 \\ -1 & 0 & +1 \end{bmatrix} * \mathbf{A} \quad \text{and} \quad \mathbf{G_y} = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ +1 & +1 & +1 \end{bmatrix} * \mathbf{A}$$