Πανεπιστήμιο Πειραιώς
Τμήμα Ψηφιακών Συστημάτων

Ευφυές σύστημα βιντεοεπιτήρησης με δυνατότητα ανίχνευσης απλών μοτίβων κίνησης και διασφάλισης της ιδιωτικότητας

Διπλωματική Εργασία

Κωνσταντίνος Μπογιατζάκης

Επιβλέπων καθηγητής: **Α. Μηλιώνης**

Αθήνα, Ιούλιος **2011**

Περιεχόμενα

Σύνοψη

Όπως λέει και ο τίτλος, στη παρούσα διπλωματική εργασία μελετήσαμε ένα πολυδιάστατο πρόγραμμα ανίχνευσης κίνησης με τη χρήση κάμερας και αναπτύξαμε αλγορίθμους και φίλτρα για την επέκτασή του. Η εργασία αυτή αποτελείται από τέσσερα μέρη.

Αρχικά κάναμε μία γενική εισαγωγή για την εξέλιξη της ρομποτικής και για την όραση καθώς επίσης ορίσαμε τα πλαίσια στα οποία θα κινηθούμε. Παρουσιάσαμε τα βασικότερα στοιχεία που πρέπει να γνωρίζει κάποιος, ο οποίος δεν έχει ασχοληθεί με το θέμα, ενώ ταυτοχρόνως, προσπαθήσαμε να μη κουράσουμε τους σχετικούς με τον τομέα, με αυτά που πιθανότατα ήδη γνωρίζουν.

Στο δεύτερο μέρος παρουσιάσαμε το πρόγραμμα, το "τρέξαμε", είδαμε και αναλύσαμε τα αποτελέσματα για διαφορετικές τιμές που του ορίσαμε. Λόγω του μεγάλου αριθμού των χαρακτηριστικών και ρυθμίσεων που προσφέρει, εξηγήσαμε εκείνα μόνο που θα μας χρειαστούν και θα αναπτύξουμε στο τέταρτο μέρος.

Ύστερα (κεφάλαιο 3)εισχωρήσαμε εις βάθος στη δομή και τον τρόπο λειτουργίας των παραπάνω παραμέτρων και είδαμε με ποιο τρόπο συνδέονται. Ακόμα, χωρίς ιδιαίτερες λεπτομέρειες σχολιάσαμε τα κομμάτια του κώδικα που αποτελούν τον πυρήνα του προγράμματος.

Στο τέταρτο και τελευταίο στάδιο, που είναι και όλη η ουσία της εργασίας, βρίσκονται οι βελτιώσεις. Δηλαδή, τα νέα φίλτρα που επινοήσαμε, οι αλγόριθμοι που τα υλοποιούν και οι συναρτήσεις που προσαρμόσαμε στο παραπάνω πρόγραμμα. Επίσης αναφέραμε και νέες προοπτικές εξέλιξης και επέκτασης, βασιζόμενες σε αυτά τα νέα φίλτρα.

1. Εισαγωγή

1.1 Γενικά

Η ρομποτική στις μέρες μας έχει κάνει μεγάλες προόδους. Τόσο μεγάλες που εντυπωσιαζόμαστε και μένουμε έκπληκτοι από αυτές τις μηχανές που προσομοιώνουν τα ανθρώπινα χαρακτηριστικά ή κάνουν ενέργειες όμοιες με του ανθρώπου, με απόλυτη ακρίβεια. Στη πραγματικότητα όμως αυτό που μας συναρπάζει σε αυτές τις μηχανές-ρομπότ, είναι ότι παρουσιάζουν τη χαρακτηριστική ανθρώπινη λειτουργία της αντίληψης.

Για παράδειγμα, υπάρχουν σήμερα κινητά συστήματα που μπορούν να αντιληφθούν τη θέση τους μέσα στο χώρο και να ακολουθήσουν μια συγκεκριμένη διαδρομή ώστε να αποφύγουν εμπόδια. Επίσης υπάρχουν συστήματα ασφάλειας που μπορούν να εντοπίσουν και να αναγνωρίσουν "στόχους" αλλά και να παρακολουθήσουν την κίνησή τους. Όπως δε, είναι αναμενόμενο, όσο καλύτερη 'αντίληψη' του περιβάλλοντός της μπορεί να έχει μία υπολογιστική μηχανή, τόσο περισσότερο πολύπλοκες και θαυμαστές είναι οι λειτουργίες που μπορούμε να την προγραμματίσουμε να επιτελέσει.

Η αντίληψη προέρχεται κυρίως μέσω των ερεθισμάτων των αισθήσεων. Μία από τις αισθήσεις είναι και η όραση (vision), η οποία τεχνικώς επιτελείται σε ένα υπολογιστικό σύστημα μέσω ενός συστήματος καταγραφής, δηλαδή από μία ή περισσότερες κάμερες. Σε αυτά τα πλαίσια, ένας προσωπικός υπολογιστής συνδεδεμένος με μία απλή κάμερα, μπορεί τεχνικώς "να βλέπει". Όταν αναφερόμαστε στην ικανότητα της όρασης στους ανθρώπινους οργανισμούς, θεωρούμε δεδομένη τη δυνατότητα της αντίληψης αυτών που βλέπουν, αυτή όμως είναι και η δυσκολότερη εργασία που πρέπει να επιτελέσει ένα υπολογιστικό σύστημα. Δηλαδή, να καταφέρει να αντιληφθεί αυτά που "βλέπει" και να εξάγει κάποια χαρακτηριστικά τους, βάσει των οποίων θα μπορέσει να τα διακρίνει, να τα συγκρίνει, να τα αναγνωρίσει και να τα χαρακτηρίσει. Στην ιδανική περίπτωση να πάρει και μία απόφαση για το αν θα πρέπει να εκτελέσει κάποια ενέργεια και ποια θα είναι αυτή.

Με προβλήματα όπως τα παραπάνω ασχολείται η Υπολογιστική Όραση (Computer Vision), όπως και η συναφής περιοχή της Κατανόησης Περιεχομένων Εικόνων και Βίντεο (Image and Video Understanding). Κοντινές επιστημονικές περιοχές είναι αυτή της Ψηφιακής Επεξεργασίας Εικόνας και Βίντεο (Digital Image and Video Processing), της Τεχνητής Νοημοσύνης (Artificial Intelligence) και των Συστημάτων Βάσεων Δεδομένων (Database Systems).

1.1.1 Υποδομή

Στη παρούσα εργασία, θα ασχοληθούμε με τα πρώτα βήματα αυτών των τεχνικών και θα προσπαθήσουμε να απαντήσουμε σε απλές ερωτήσεις όπως: αν "βλέπει" και όχι σε σύνθετους μηχανισμούς, όπως: τι είναι αυτό που "βλέπει". Δηλαδή δεν θα μπούμε στη διαδικασία της υλοποίησης πολύπλοκων αλγορίθμων με πολλές μεταβλητές, άλλα θα απλοποιήσουμε όσο είναι δυνατό περισσότερο το πρόβλημα.

Είναι λογικό πως όσο περισσότερες συσκευές λήψης χρησιμοποιούμε τόσο θα δυσκολεύει η υλοποίηση των αλγορίθμων μας. Αυτό διότι θα έχουμε να αντιμετωπίσουμε πολύπλοκα θέματα προσδιορισμού της 3D θέσης των αντικειμένων σε μια σκηνή, κάνοντας σύγκριση εικόνων από δύο ή περισσότερες χωριστές κάμερες. Αυτομάτως καλούμαστε να λύσουμε προβλήματα στερεοσκοπικής όρασης, δηλαδή γεωμετρίας ως προς τις γωνίες που σχηματίζουν οι κάμερες, τις αποστάσεις μεταξύ τους, την απόσταση του αντικειμένου ή των αντικειμένων, την τριγωνομετρία, κτλ. Έπειτα να επεξεργαστούμε και να συγκρίνουμε τα σήματα(εικόνες ή βίντεο) από τις κάμερες και να δημιουργήσουμε το, τριών διαστάσεων, χώρο ή το αντικείμενο που "βλέπουν". Για αυτό λοιπόν, θα χρησιμοποιήσουμε **μια κάμερα.**

Αυτή η μοναδική κάμερα που θα χρησιμοποιήσουμε μπορεί να είναι είτε στατική είτε κινούμενη:

Στην περίπτωση κινούμενης κάμερας, πέρα από τις προαναφερθείσες δυσκολίες, θα πρέπει να αντιμετωπίσουμε και την κίνηση του φόντου, λόγω της κίνησης της κάμερας. Εφόσον για κινούμενες κάμερες, πραγματοποιούνται υπολογισμοί μεταξύ διαφορετικών καρέ, για να εντοπίσουμε τα κινούμενα αντικείμενα, η πολυπλοκότητα του προβλήματος γίνεται ανάλογη με την πολυπλοκότητα της κίνησης της κάμερας (με το σκεπτικό ότι η απότομη ή ισχυρά περιστροφική κίνηση αλλάζει σημαντικά το καταγραφόμενο φόντο). Στην περίπτωση που περισσότερα από ένα κινούμενα αντικείμενα υπάρχουν στη σκηνή το σύστημα θα πρέπει να υπολογίσει τη σχετική ταχύτητα μεταξύ κάθε αντικειμένου και του φόντου (ή ακόμη να συμπεριλάβει κάποια χωροχρονικά κριτήρια), ώστε να τα διακρίνει μεταξύ τους. Μία τέτοια διαδικασία γίνεται σημαντικά δύσκολη όταν τα αντικείμενα κινούνται σε μικρές αποστάσεις μεταξύ τους, με όμοιες ταχύτητες και έχουν όμοιες χρωματικές ιδιότητες και ιδιότητες ακμών.

Ας υποθέσουμε ότι έχουμε καταφέρει να υλοποιήσουμε τους αλγορίθμους που απαιτούνται για στερεοσκοπική όραση με τις 2 ή περισσότερες στατικές κάμερες. Τι θα γίνει αν θέλουμε να τις προσαρμόσουμε σε κάποιο κινητό ρομποτικό σύστημα που θα κινείται σύμφωνα με τα οπτικά ερεθίσματα από τις κάμερες και θα ακολουθεί μονοπάτια(δηλαδή περισσότερες από μια κινούμενες κάμερες); Είναι ερώτημα που δε

θα μας απασχολήσει, διότι μόνο από επεξεργαστική ισχύ, σε μη πραγματικό χρόνο, να βλέπαμε το θέμα, απορρίπτεται.

Στην περίπτωση της **στατικής κάμερας**, η σκηνή, συμπεριλαμβανομένου του φόντου, καταγράφεται βολικά από ένα συγκεκριμένο σταθερό σημείο παρατήρησης, ώστε όλη η δράση στην καταγραφόμενη ακολουθία να οφείλεται στην κίνηση αντικειμένων. Σ' αυτά τα πλαίσια, σχεδόν οποιαδήποτε αλλαγή στο πεδίο καταγραφής της κάμερας θα έπρεπε να υποδεικνύει την παρουσία ενός κινούμενου αντικειμένου που θα μπορούσε να εντοπιστεί και να καταχωρηθεί. Ταυτόχρονα όμως, υπάρχουν περιπτώσεις, όπου ακόμη κι αν υπάρχουν αλλαγές στην καταγραφόμενη ακολουθία, δεν υπάρχουν κινούμενα αντικείμενα ενδιαφέροντος προς παρακολούθηση. Παραδείγματος χάριν, όταν σταδιακές ή απότομες αλλαγές συμβαίνουν στις συνθήκες φωτισμού ή υπάρχουν μικρές τυχαίες αλλαγές στο φόντο (π.χ. όταν ο άνεμος κινεί φυτά ή κουρτίνες) ή όταν η κάμερα υποφέρει από μικρές δονήσεις, κτλ... Βάσει των παραπάνω, το πρόβλημα του εντοπισμού κινούμενων αντικειμένων μετονομάζεται σε πρόβλημα εντοπισμού των "κύριων" κινούμενων αντικειμένων, όπου αλλαγές όμοιες με αυτές που αναφέρθηκαν παραπάνω θεωρούνται ασήμαντες. Με άλλα λόγια, ο κύριος στόχος ενός εύρωστου αλγορίθμου εντοπισμού κύριων κινούμενων αντικειμένων για στατικές κάμερες, είναι να διατηρεί υψηλή ευαισθησία στην παρουσία ενός σημαντικού κινούμενου αντικειμένου στην παρατηρούμενη σκηνή, ενώ να μειώνει την ευαισθησία του στις αλλαγές φωτεινότητας ή στις 'ασήμαντες' αλλαγές. Θα πρέπει να δοθεί προσοχή εδώ στο γεγονός, ότι σε αντίθεση με τις μικρές χρονικές διαφοροποιήσεις λόγω 'θορύβου', μία αλλαγή στο φωτισμό μπορεί να οδηγήσει στη λανθασμένη θεώρηση μεγάλων περιοχών της παρατηρούμενης σκηνής ως κινούμενα αντικείμενα. Τέτοιες διαφοροποιήσεις είναι ιδιαίτερα σημαντικές στην περίπτωση καταγραφών εξωτερικού χώρου.

Το αποτέλεσμα των παραπάνω αποφάσεων, για χρήση μίας στατικής κάμερας, είναι το πλαίσιο της εργασίας να περιορίζεται στο **δισδιάστατο περιβάλλον.** Μπορούμε να βρούμε ένα σύνολο από χαρακτηριστικά υψηλότερου επιπέδου από την πληροφορία των pixels, αλλά με σχετικά χαμηλό σημασιολογικό περιεχόμενο, τα οποία ταυτόχρονα να έχουν κάποιο νόημα στην εξαγωγή τους αναφορικά με την ανθρώπινη αντίληψη.

Τέλος, ένας ακόμα λόγος που θα κάνει πιο λιτούς του παρακάτω αλγορίθμους είναι ότι δε θα ασχοληθούμε με την ανάλυση των χρωμάτων. Δηλαδή τα δεδομένα από τη κάμερα θα τα μετατρέπουμε και θα τα επεξεργαζόμαστε στη **κλίμακα του γκρι**. Με αυτόν τον τρόπο θα έχουμε εύρος τιμών και όχι σύνολα τιμών. Το πώς θα γίνει αυτό, θα το αναλύσουμε αφού πούμε μερικά γενικά πράγματα για τις εικόνες.

1.2 Pixels και ανάλυση εικόνας

Κάθε εικόνα που έχουμε αποθηκευμένη στον υπολογιστή μας, αποτελείται από pixels. Αυτά είναι τα μικροσκοπικά σημεία του χρώματος, που μπορούμε να διακρίνουμε αν πλησιάσουμε πολύ κοντά στην οθόνη μας. Επίσης είναι το μικρότερο δυνατό μέγεθος που μπορεί να πάρει κάθε εικόνα. Όταν μια εικόνα αποθηκεύεται, το αρχείο της περιέχει πληροφορίες για κάθε pixel σε αυτή την εικόνα.

Αυτές οι πληροφορίες περιλαμβάνουν δύο πράγματα: το **χρώμα** και την **τοποθεσία**.

Επίσης, οι εικόνες έχουν έναν ορισμένο αριθμό pixel ανά μέγεθος, γνωστό ως ανάλυση(ψήφισμα). Μια υψηλότερη ανάλυση σημαίνει ότι υπάρχουν περισσότερα pixel σε ένα σύνολο περιοχής, με αποτέλεσμα την υψηλότερη ποιότητα εικόνας. Το μειονέκτημα υψηλότερης ανάλυσης είναι ότι απαιτεί μεγαλύτερη ισχύ επεξεργασίας για την ανάλυση μιας τέτοιας εικόνας. Κατά τον προγραμματισμό, για την παρούσα εργασία, χρησιμοποιείται χαμηλή ανάλυση.

1.3 Πίνακες (το είδος μαθηματικών)

Οι εικόνες αποθηκεύονται σε 2D(δύο διαστάσεων) πίνακες, οι οποίοι αντιπροσωπεύουν τις θέσεις όλων των pixels. Όλες οι φωτογραφίες έχουν μια συνιστώσα X και μια συνιστώσα Y. Σε κάθε σημείο, αποθηκεύεται μια τιμή χρώματος. Αν η εικόνα είναι ασπρόμαυρη (δυαδικό), θα αποθηκεύεται σε κάθε τοποθεσία είτε το 1 ή το 0. Αν το χρώμα είναι σε κλίμακα του γκρι , θα αποθηκεύσει ένα εύρος τιμών. Αν πρόκειται για έγχρωμη εικόνα (RBG) , θα αποθηκεύσει σύνολα τιμών. Προφανώς, όσο λιγότερο χρώμα περιέχεται τόσο πιο γρήγορα η εικόνα μπορεί να υποστεί επεξεργασία. Για πολλές εφαρμογές, δυαδικές εικόνες μπορούν να επιτύχουν περισσότερα από ό, τι θα ήταν αναμενόμενο.

Για παράδειγμα, εδώ είναι ένας πίνακας με μια δυαδική εικόνα ενός τριγώνου:

0	0	0	**1**	0	0	0
0	0	**1**	0	**1**	0	0
0	**1**	0	0	0	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**
0	0	0	0	0	0	0

Έχει ανάλυση 7 x 5, με ένα και μόνο bit αποθηκευμένο σε κάθε θέση. Ως εκ τούτου απαιτείται μνήμη: 7 x 5 x 1 = 35 bits.

Σε αυτό το παράδειγμα είναι ένας πίνακας με αποχρώσεις του γκρι (8 bit), με την απεικόνιση ενός τριγώνου:

```
0    0    55   255  55   0    0
0    55   255  55   255  55   0
55   255  55   55   55   255  55
255  255  255  255  255  255  255
55   55   55   55   55   55   55
0    0    0    0    0    0    0
```

Έχει ένα ψήφισμα από 7 x 6, με 8 bits αποθηκευμένα σε κάθε θέση. Απαιτείται λοιπόν μνήμη: 7 x 6 x 8 = 336 bits.

Όπως μπορείτε να δείτε, η αύξηση της ανάλυσης και των πληροφοριών ανά pixel, μπορεί να επιβραδύνει σημαντικά την ταχύτητα επεξεργασίας τις εικόνας σας. Για αυτό λοιπόν και εμείς θα επεξεργαζόμαστε τα δεδομένα από την κάμερα, μόνο στη κλίμακα του γκρι όπως αναφέραμε παραπάνω.

Με αυτά τα δεδομένα είναι φανερό πως η πολυπλοκότητα των προβλημάτων και τον προγραμμάτων παρακάτω θα είναι πολύ ποιο απλή και συνεπώς πολύ πιο σύντομα κατανοητή.

1.4 Βίντεο

Στη μηχανική όραση το βίντεο δεν είναι τίποτα άλλο από πολλές διαδοχικές εικόνες, οι οποίες λαμβάνονται κάθε συγκεκριμένο χρονικό διάστημα. Όσο μικρότερο είναι αυτό το χρονικό διάστημα, τόσο “καλύτερη” είναι η κάμερα και η ποιότητα του βίντεο. Η κάμερα που θα χρησιμοποιήσουμε καταγράφει 15 εικόνες ανά δευτερόλεπτο (frames per second: fps). Αρχικά οι εικόνες αυτές (τα δεδομένα εισόδου) προ της επεξεργασίας τους δεν έχουν καμία πρακτική χρησιμότητα αναφορικά με την αντίληψη του συστήματος, πέρα από την αποθήκευση και την επίδειξή τους, αφού δεν είναι παρά οι συντεταγμένες των εικονοστοιχείων(pixels) των εικόνων στο χώρο των χρωμάτων. Για να αποκτήσουν λοιπόν χρησιμότητα θα πρέπει να μπορεί το σύστημα να απαντήσει στη απλή ερώτηση: “ σκηνή;”. Πρέπει να πραγματοποιήσει δηλαδή κάποια ανίχνευση (detection) που έχει να κάνει με χαμηλού επιπέδου επεξεργασία και είναι συνήθως το πρώτο βήμα σε όλους τους αλγόριθμους ανάλυσης κίνησης. Τα προγράμματα για τον εντοπισμό κίνησης, χρησιμοποιούν διάφορες προσεγγίσεις με πιο διαδεδομένη τη βασιζόμενη σε pixels προσέγγιση, που εξετάζει τις διαφορές μεταξύ διαδοχικών καρέ (frames), χρησιμοποιώντας pixels ως χαρακτηριστικά εισόδου. Πριν από αυτόν τον αλγόριθμο όμως θα πρέπει να πούμε για το πόσο σημαντική είναι η παράμετρος του ορίου threshold, συνεχίζοντας το προηγούμενο παράδειγμα με το τρίγωνο.

1.5 Κατώφλι (threshold)

Σε αυτό το συγκεκριμένο παράδειγμα κατωφλίου, θα μετατρέψουμε όλα τα χρώματα σε δυαδικό. Πώς μπορείτε να αποφασίσετε ποια pixel θα είναι 1 και οποία θα είναι 0? Το πρώτο πράγμα που κάνουμε είναι να καθορίσουμε ένα όριο. Δηλαδή όλες οι τιμές των pixel πάνω από το όριο να γίνονται 1, και όλες κάτω από το όριο να γίνονται 0. Το κατώφλι σας μπορεί να επιλεγεί αυθαίρετα, ή μπορεί να βασίζεται στην ευρετική μας ανάλυση.

Για παράδειγμα, κατά τη μετατροπή κλίμακας του γκρι στο τρίγωνο μας, σε δυαδικό σύστημα, χρησιμοποιώντας 40 ως κατώφλι, θα έχουμε:

0	0	**1**	**1**	**1**	0	0
0	**1**	**1**	**1**	**1**	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**
1	**1**	**1**	**1**	**1**	**1**	**1**
0	0	0	0	0	0	0

Αν το όριο ήταν 100, θα παίρναμε αυτήν την εικόνα:

0	0	0	**1**	0	0	0
0	0	**1**	0	**1**	0	0
0	**1**	0	0	0	**1**	0
1	**1**	**1**	**1**	**1**	**1**	**1**
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Όπως μπορείτε να δείτε, ένα καλό όριο είναι πολύ σημαντικό. Στο πρώτο παράδειγμα, δεν μπορείτε να δείτε το τρίγωνο, αλλά στο δεύτερο μπορείτε. Κακή επιλογή στα όρια οδηγεί σε χαμηλής ποιότητας εικόνες.

Σημειώστε ότι εάν το όριο ήταν 1, ολόκληρη η εικόνα θα ήταν μαύρη. Αν το όριο ήταν 255, το σύνολο της εικόνας θα ήταν λευκό. Ο προσδιορισμός του κατωφλιού υπερέχει πραγματικά όταν τα χρώματα στο φόντο είναι πολύ διαφορετικά από τους στόχους, καθώς αφαιρεί αυτόματα το φόντο από την εικόνα μας.

1.6 Αλγόριθμος διαφοράς διαδοχικών frames

Ύστερα από την αναφορά στην έννοια του ορίου ο πιο απλός ίσως τρόπος για τον εντοπισμό του κινούμενου αντικείμενου σε ασπρόμαυρο βίντεο με στατική κάμερα είναι να συγκρίνουμε ανά δύο τα frames του βίντεο παίρνοντας τη διαφορά τους. Αυτό γίνεται ως εξής:

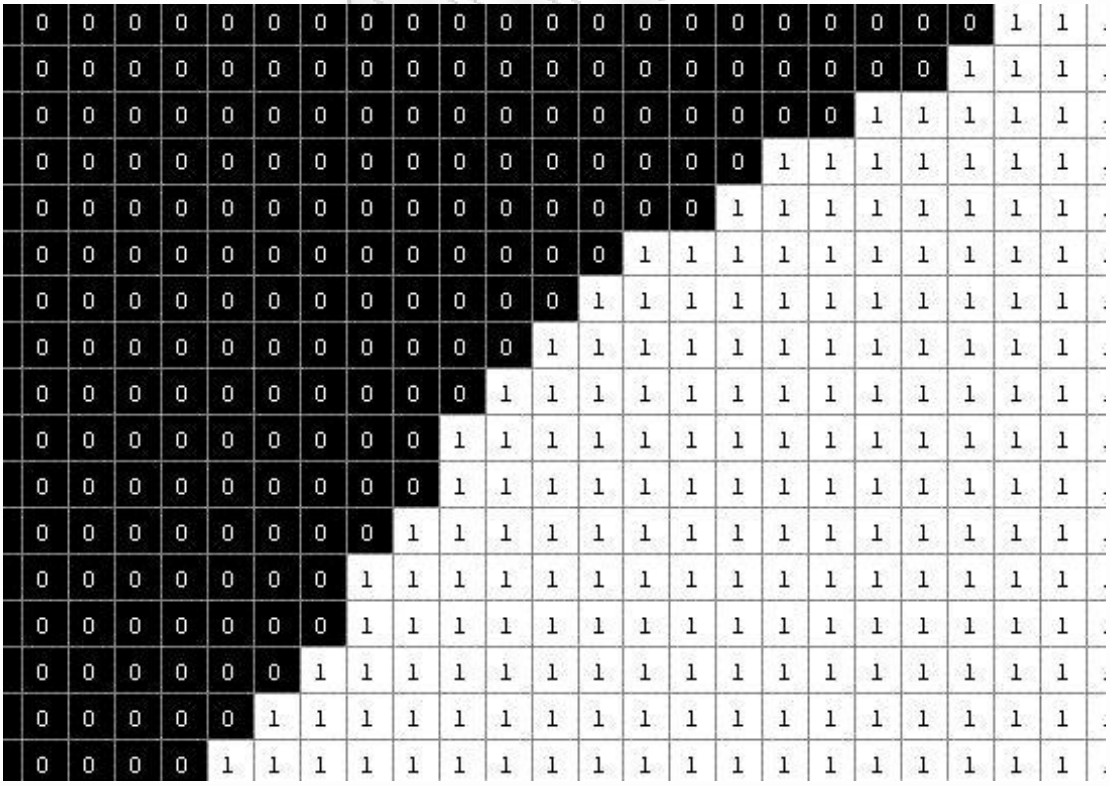
1. ***Ορίζουμε έναν πίνακα με διαστάσεις ίδιες με αυτές των frames, τον οποίο αρχικοποιούμε.***

2. ***Για κάθε δύο διαδοχικά frames, παίρνουμε κάθε pixel του καινούριου frame και το συγκρίνουμε με το αντίστοιχο pixel του αμέσως προηγούμενου.***

3. ***Αν έχει αλλάξει το pixel στην αντίστοιχη θέση του πινάκα βάζουμε 1, αλλιώς 0***

4. ***Ορίζουμε ένα κατώφλι (threshold).***

5. ***Αν ο συνολικός αριθμός των 1 του πίνακα είναι μικρότερος της τιμής του κατωφλίου, τότε δεν υπάρχει κίνηση, ενώ αν ο αριθμός είναι μεγαλύτερος, τότε σημαίνει ότι εντοπίστηκε κίνηση.***



Έτσι, από κάθε δύο διαδοχικά frames παίρνουμε μία ασπρόμαυρη εικόνα όπου τα μαύρα pixel αντιστοιχούν στα pixel που δεν μετακινήθηκαν, δηλαδή στο φόντο, ενώ τα άσπρα pixel αντιστοιχούν στα pixel που άλλαξαν, δηλαδή στο κινούμενο αντικείμενο.

Με αυτόν τον τρόπο όμως, προκύπτουν διάφορα προβλήματα. Καταρχάς, χρειάζεται μεγάλη προσοχή στην τιμή που θα βάλουμε στο κατώφλι. Αν η τιμή είναι πολύ μικρή, τότε η παραμικρή κίνηση θα ανιχνευθεί, πράγμα που μπορεί να οδηγήσει σε εσφαλμένες εκτιμήσεις, διότι μπορεί να έχουμε μικρές αλλαγές στο φόντο, οι οποίες δε θέλουμε να θεωρηθούν ως κινούμενα αντικείμενα. Τέτοιες αλλαγές μπορεί να οφείλονται σε θόρυβο λόγω της κάμερας, σε αλλαγές φωτισμού, σε σκιές, αλλαγές στο φως του ήλιου, σε αέρα αν είμαστε σε εξωτερικό περιβάλλον, κ.ά. Απ' την άλλη, αν η τιμή του κατωφλίου είναι πολύ μεγάλη, τότε κάποιες αλλαγές μπορεί να μην ανιχνευθούν καθόλου, όπως για παράδειγμα ένα αντικείμενο που κινείται με πολύ μικρή ταχύτητα.

Επίσης η ανεπάρκεια αυτής της μεθόδου εμφανίζεται σε παρουσία περισσότερων του ενός κινούμενων αντικειμένων, ειδικότερα όταν επικαλύπτονται μεταξύ τους, ή παρουσία σημαντικών ποσοτήτων θορύβου, ειδικότερα στην περιοχή των κινούμενων αντικειμένων (π.χ. όταν κινούμενα φυτά επικαλύπτουν μερικώς τα όρια του κινούμενου αντικειμένου).Τέλος ο αλγόριθμος αδυνατεί να χειριστεί ένα περιστρεφόμενο αντικείμενο ή ένα αντικείμενο που κινείται, αλλά έχει μια μέση μάζα που δεν αλλάζει θέση.

Ο εντοπισμός των κύριων κινούμενων αντικειμένων, ή εισβολέων, σε μια ακολουθία εικόνων μπορεί να αποτελέσει μέσο απόφασης για συστήματα συναγερμού πραγματικού χρόνου σε εφαρμογές παρακολούθησης, ή ακόμη να βοηθήσει στη μέτρηση της ροής της κυκλοφορίας των οχημάτων ή των πεζών. Το πρόγραμμα που θα επεκτείνουμε και θα προσαρμόσουμε τέτοιους αλγορίθμους είναι το Motion.

2. Motion

2.1 Τι είναι το Motion?

Το Motion είναι ένα πρόγραμμα που παρακολουθεί το οπτικό σήμα από μία ή περισσότερες κάμερες και είναι σε θέση να ανιχνεύσει αν ένα σημαντικό μέρος της εικόνας έχει αλλάξει. Με άλλα λόγια, μπορεί να ανιχνεύσει κίνηση.

Το πρόγραμμα είναι γραμμένο στη C και έχει φτιαχτεί για το λειτουργικό σύστημα Linux, (χρησιμοποιώντας το περιβάλλον video4linux). Το Motion είναι εργαλείο βασισμένο στη γραμμή εντολών (command line) του οποίου τα αποτελέσματα μπορεί να είναι είτε σε jpeg, ppm αρχεία ή MPEG ακολουθίες βίντεο. Το motion είναι αυστηρά οδηγούμενο από τη γραμμή εντολών και μπορεί επίσης να τρέξει στο παρασκήνιο με κατανάλωση μικρής ισχύος.

Είναι το τέλειο εργαλείο για παρακολούθηση της ιδιοκτησίας, εφόσον κρατάει μόνο τις εικόνες που είναι ενδιαφέρουσες.

Δεν έχει κανένα απολύτως γραφικό περιβάλλον για τη διεπαφή με το χρήστη. Τα πάντα είναι οργανωμένα είτε μέσω του τερματικού είτε μέσω ενός συνόλου αρχείων ρυθμίσεων (απλά αρχεία ASCII που μπορεί να τροποποιηθεί από οποιοδήποτε συντάκτη ASCII).

Όπως είπαμε η έξοδος από το motion μπορεί να είναι:

- jpg αρχεία
- αρχεία μορφής ppm
- MPEG ακολουθίες βίντεο

2.2 Πώς μπορώ να αποκτήσω το Motion και με τι κόστος;

Το Motion είναι ένα ανοικτού(δωρεάν) τύπου πρόγραμμα και δεν κοστίζει τίποτα. Δημοσιεύεται στο πλαίσιο της *GNU*. Αυτό σημαίνει ότι μπορούμε να πάρουμε το πρόγραμμα, να το εγκαταστήσουμε και να το χρησιμοποιήσουμε ελεύθερα. Δεν χρειάζεται να πληρώσουμε τίποτα και δεν χρειάζεται να εγγραφούμε οπουδήποτε ή να ζητήσουμε από το δημιουργό ή εκδότη την άδεια. Η GENERAL PUBLIC LICENSE(GPL) μας παρέχει τα δικαιώματα, όταν πρόκειται για αντιγραφή, διανομή και τροποποίηση του προγράμματος. Έτσι, σε πολύ γενικές γραμμές δεν έχουμε να ανησυχούμε για την αδειοδότηση ως απλοί χρήστες.

Το motion γράφτηκε αρχικά από τον Jeroen Vreeken ο οποίος εξακολουθεί να συμμετέχει ενεργά στην ανάπτυξη του motion και αργότερα ο Folkert van Heusden συνέχισε ως επικεφαλής προγραμματιστής μαζί με τον Kenneth Lavrsen που είναι υπεύθυνος για το Motion Guide.

Η τρέχουσα έκδοση είναι η έκδοση 3.2.12.

2.3 Τι χαρακτηριστικά έχει το motion;

- Λήψη στιγμιότυπων της κίνησης
- Δημιουργία MPEG βίντεο σε πραγματικό χρόνο με χρήση ffmpeg βιβλιοθηκών
- Αυτοματοποιημένα στιγμιότυπα σε τακτά χρονικά διαστήματα
- Εκτέλεση εξωτερικού προγράμματος κατά την ανίχνευση κίνησης
- Εκτέλεση εξωτερικού προγράμματος κατά την έναρξη της εκδήλωσης ανιχνεύσεως κίνησης.
- Εκτέλεση εξωτερικού προγράμματος στο τέλος της εκδήλωσης αρκετών ανιχνευμένων κινήσεων.
- Εκτέλεση εξωτερικού προγράμματος, όταν μια εικόνα αποθηκεύεται.
- Εκτέλεση εξωτερικού προγράμματος όταν μία mpeg ταινία δημιουργείται
- Εκτέλεση εξωτερικού προγράμματος όταν τελειώνει το MPEG βίντεο
- Παρακολούθηση κίνησης
- Ρυθμίσεις και ορισμοί από το χρήστη για την οθόνη.
- Έλεγχος μέσω απλής διεπαφής από ιστοσελίδα
- Αυτόματος έλεγχος του θορύβου και κατώτατο όριο
- Διαμόρφωση εμφάνισης του κειμένου για τις εικόνες από το χρήστη.
- Ορισμός της διαδρομής και των ονομάτων των αρχείων που αποθηκεύει (εικόνες και ταινίες).

Είναι μόνο μερικά από τα χαρακτηριστικά αυτού του προγράμματος

2.4 Προετοιμασία για την εγκατάσταση

Πριν ξεκινήσουμε ίσως χρειαστεί να εγκαταστήσουμε ένα σύνολο κοινών βιβλιοθηκών που απαιτούνται. Αν αυτές λείπουν θα συνεχιστεί η εγκατάσταση κανονικά, απλώς δεν θα συμπεριλαμβάνονται. Μερικές από αυτές τις βιβλιοθήκες είναι libm, libjpeg, libz, libavcodec, libavformat, libmysqlclient κτλ...

2.5 Εκτέλεση Motion

Όπως προαναφέραμε, γίνεται επίκληση από τη γραμμή εντολών. Δεν έχει γραφικό περιβάλλον. Τα πάντα ελέγχονται από τα αρχεία ρυθμίσεων (motion.conf). Από την έκδοση 3.2 η γραμμή εντολών χρησιμοποιείται μόνο για να καθορίσουμε τη τοποθεσία του αρχείου ρυθμίσεων και μερικές ειδικές λειτουργίες εκτέλεσης (setup και no-daemon).

Μερικοί σημαντικοί ορισμοί:

- Ένα στιγμιότυπο(snapshot) είναι μια εικόνα που λαμβάνεται σε τακτά χρονικά διαστήματα, ανεξάρτητα από οποιαδήποτε κίνηση στην εικόνα.
- Μια εικόνα στην οποία έχει εντοπιστεί κίνηση, δείχνει τα pixel που έχουν αλλάξει στα τελευταία καρέ. Αυτές οι εικόνες δεν είναι πολύ χρήσιμες για την κανονική παρουσίαση στο κοινό, αλλά είναι αρκετά χρήσιμες για έλεγχο και συντονισμό και καθιστούν τα αρχεία “μάσκας”, όπου μπορούμε να ορίσουμε ακριβώς τα μέρη στα οποία θέλουμε να εντοπίζουμε κίνηση. Κίνηση εμφανίζεται στην κλίμακα του γκρι. Εάν η σήμανση είναι ενεργοποιημένη, η μεγαλύτερη περιοχή χαρακτηρίζεται ως μπλε.
- Μια "normal"(κανονική) εικόνα είναι η πραγματική εικόνα που λήφθηκε από τη φωτογραφική μηχανή με το κείμενο overlayed.

2.5.1 Επιλογές γραμμής εντολών

Στη γραμμή εντολών μπορούμε να δώσουμε 4 επιλογές: μια για να ορίσουμε την τοποθεσία του motion.conf και μερικές επιλογές που σχετίζονται με βασικές αποφάσεις.

Σύνοψη:

motion [-hns] [-c διαδρομή αρχείου config]

Επιλογή	Περιγραφή	Σύνταξη σχόλιο
-n	Εκτέλεση σε κατάσταση μη-παρασκηνίου	Αντί να τρέχει στο παρασκήνιο, τρέχει στο παράθυρο του τερματικού όπου συντάσσονται μηνύματα όταν συμβαίνει κάτι. Αν έχουμε προβλήματα στην εκτέλεση του Motion επιλέγουμε αυτόν τον τρόπο για να δούμε που υπάρχει το πρόβλημα και να μπορέσουμε να το λύσουμε.
-s	Εκτέλεση σε λειτουργία εγκατάστασης.	Επίσης, εκτελείτε μόνο στο προσκήνιο.
-c διαδρομή config	Η πλήρης διαδρομή και το όνομα του αρχείου ρυθμίσεων.	/ home / bo / motion.conf.
-h	Εμφάνιση της οθόνης βοήθειας.	

2.5.2 Αρχείο επιλογών

Το motion.conf (αρχείο επιλογών) περιλαμβάνει τις εξής παραμέτρους:

daemon *off* **process_id_file** */home/bo/motion.pid* **setup_mode** *on* **videodevice** */dev/video0* **v4l2_palette** *8* **; tunerdevice** */dev/tuner0* **input** *8* **norm** *0* **frequency** *0* **rotate** *0* **width** *320* **height** *240* **framerate** *10* **minimum_frame_time** *0* **; netcam_url** *http://www.earthcam.com* **; netcam_userpass** *value* **; netcam_http** *1.0* **; netcam_proxy** *value* **netcam_tolerant_check** *off* **auto_brightness** *off* **brightness** *0* **contrast** *0* **saturation** *0* **hue** *0* **roundrobin_frames** *1* **roundrobin_skip** *1* **switchfilter** *off* **threshold** *2000* **threshold_tune** *off* **noise_level** *32* **noise_tune** *on* **; despeckle** *EedDl* **; area_detect** *2356* **; mask_file** */home/bo/mask.PGM* **smart_mask_speed** *0* **lightswitch** *0* **minimum_motion_frames** *1* **pre_capture** *0* **post_capture** *0*	**text_double** *off* **target_dir** */home/bo/output* **snapshot_filename** *%v-%d%m%Y%H%M%S* **jpeg_filename** *%v-%Y%m%d%H%M%S-%q* **movie_filename** *%v %d-%m-%Y %H:%M:%S* **timelapse_filename** *%d-%m-%Y-* **webcam_port** *0* **webcam_quality** *50* **webcam_motion** *off* **webcam_maxrate** *1* **webcam_localhost** *on* **webcam_limit** *0* **control_port** *0* **control_localhost** *on* **control_html_output** *on* **; control_authentication** *username: password* **track_type** *0* **track_auto** *off* **; track_port** *value* **track_motorx** *0* **track_motory** *0* **track_maxx** *0* **track_maxy** *0* **track_iomojo_id** *0* **track_step_angle_x** *10* **track_step_angle_y** *10* **track_move_wait** *10* **track_speed** *255* **track_stepsize** *40* **quiet** *off* **; on_event_start** *value* **; on_event_end** *value* **; on_picture_save** *value* **; on_area_detected** **; on_movie_start** *value*

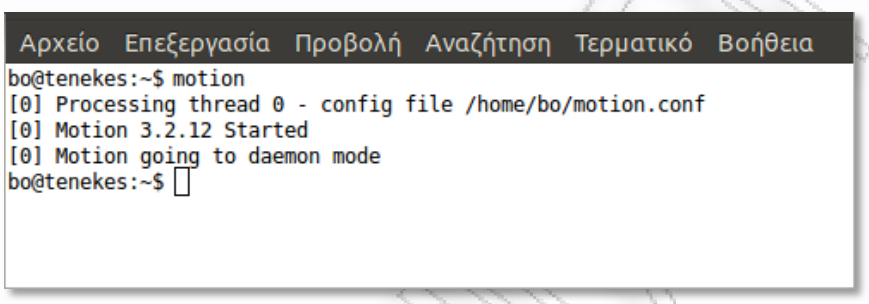
gap *10* **max_mpeg_time** *0* **output_all** *on* **output_normal** *on* **output_motion** *on* **quality** *75* **ppm** *off* **ffmpeg_cap_new** *on* **ffmpeg_cap_motion** *on* **ffmpeg_timelapse** *0* **ffmpeg_timelapse_mode** *daily* **ffmpeg_bps** *400000* **ffmpeg_variable_bitrate** *0* **ffmpeg_video_codec** *mpeg4* **ffmpeg_deinterlace** *off* **snapshot_interval** *0* **locate** *on* **text_right** *%T-fn:%q\nchp:%D-nl:%N\nw:%i h:%J-X:%K Y:%L* **text_left** *evn %v* **text_changes** *off* **text_event** *%d%m%Y%H%M%S*	**; on_movie_end** *value* **; on_camera_lost** *value* **sql_log_image** *on* **sql_log_snapshot** *on* **sql_log_mpeg** *off* **sql_log_timelapse** *off* **; mysql_db** *value* **; mysql_host** *value* **; mysql_user** *value* **; mysql_password** *value* **; pgsql_db** *value* **; pgsql_host** *value* **; pgsql_user** *value* **; pgsql_password** *value* **; pgsql_port** *5432* **; video_pipe** *value* **; motion_video_pipe** *value* **; thread** */usr/local/etc/thread1.conf* **; thread** */usr/local/etc/thread2.conf* **; thread** */usr/local/etc/thread3.conf* **; thread** */usr/local/etc/thread4.conf*

Με έντονα γράμματα είναι το όνομα της παραμέτρου και με κανονικά γράμματα είναι η τιμή της. Οι παράμετροι στις οποίες υπάρχει το ελληνικό ερωτηματικό “**;**” δεν λαμβάνονται υπόψη από το πρόγραμμα.

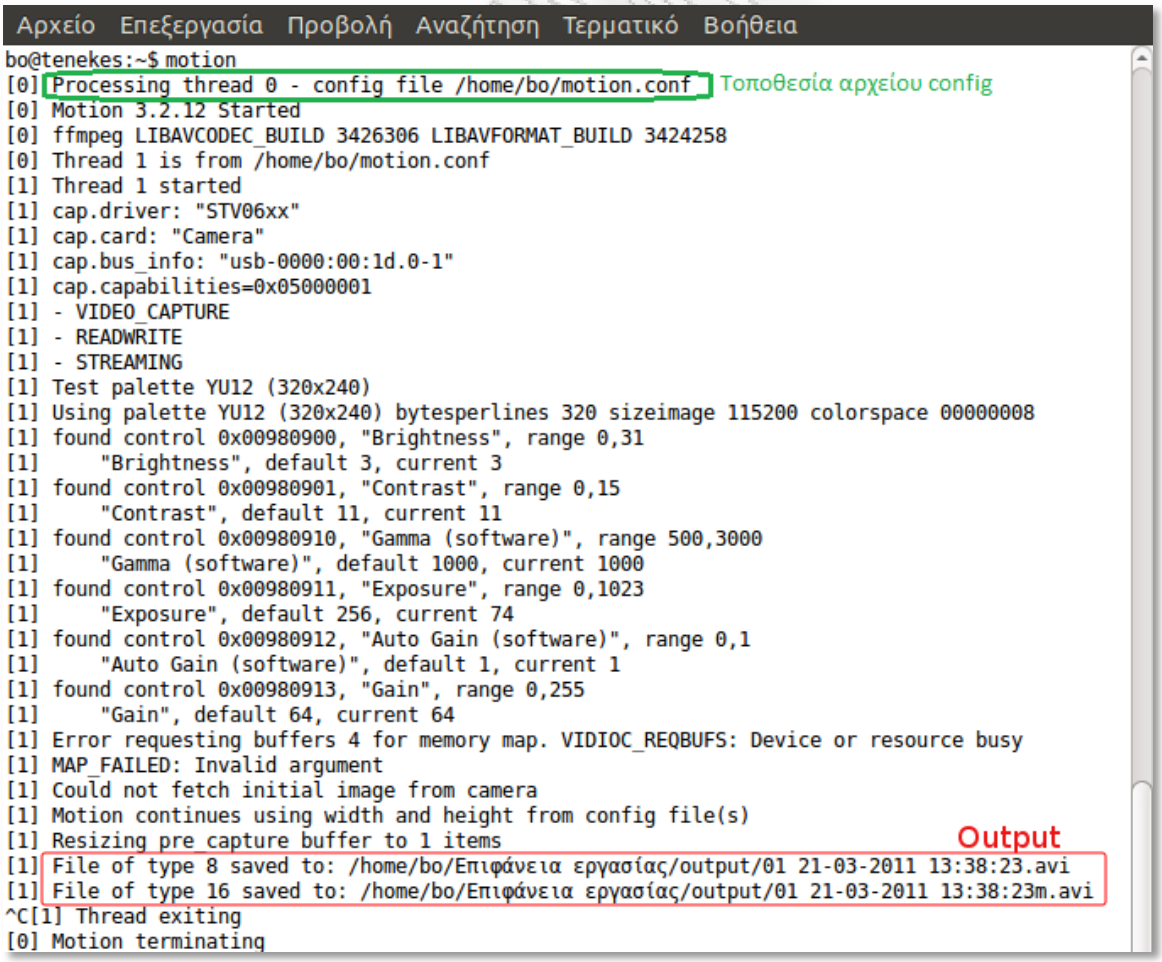
Από όλες αυτές τις παραμέτρους εμείς θα εξηγήσουμε μερικές από τις πρώτες που είναι βασικές για τη λειτουργία του προγράμματος και μερικές ποιο εξειδικευμένες που μας ενδιαφέρουν για τη παρούσα εργασία. Ύστερα, από τις περισσότερο εξειδικευμένες θα αναλύσουμε κάποιες σε μεγάλο βαθμό, θα τις εξηγήσουμε και θα τις βελτιώσουμε.

2.6 Λειτουργίες (Παράμετροι)

- **daemon** και **setup_mode**: Τις είδαμε και ποιο πριν, γιατί μπορούμε να τις τροποποιήσουμε από το κάλεσμα του προγράμματος στη γραμμή εντολών. Είναι τα **-n** και **-s** αντίστοιχα και όπως ξέρουμε μπορούν να πάρουν δύο τιμές(ενεργοποιημένη: **on** και απενεργοποιημένη: **off**). Στα παρακάτω screen-shot φαίνεται ο τρόπος λειτουργίας τους. Γενικά όμως στη παρούσα εργασία η επιλογή daemon θα είναι απενεργοποιημένη, ενώ η επιλογή setup_mode θα είναι on.



daemon: on



daemon: off setup_mode: off

```
[1] File of type 8 saved to: /home/bo/Επιφάνεια εργασίας/output/01 21-03-2011 20:29:47.avi
[1] File of type 16 saved to: /home/bo/Επιφάνεια εργασίας/output/01 21-03-2011 20:29:47m.avi
[1] Motion detected - starting event 1
[1] Changes: 25490 - noise level: 21
[1] Changes: 26187 - noise level: 21
[1] Changes:  3072 - noise level: 21
[1] Changes:  2952 - noise level: 21
[1] Changes:  2943 - noise level: 21
[1] Changes:  2878 - noise level: 21
[1] Changes:  2869 - noise level: 21
[1] Changes:  2868 - noise level: 21
[1] micro-lightswitch!
[1] Changes:     0 - noise level: 21
[1] Changes:     0 - noise level: 21
[1] Changes:     0 - noise level: 21
[1] Changes:     0 - noise level: 21
[1] Changes:     0 - noise level: 21
[1] Changes:     0 - noise level: 21
[1] Changes:     4 - noise level: 21
[1] Changes:     1 - noise level: 21
[1] Changes:   480 - noise level: 21
[1] Changes:  9955 - noise level: 21
[1] Changes: 16280 - noise level: 21
[1] Changes: 20342 - noise level: 21
[1] Changes: 20575 - noise level: 21
[1] Changes: 17947 - noise level: 21
[1] Changes: 24137 - noise level: 21
[1] Changes:  7139 - noise level: 21
[1] Changes:    42 - noise level: 21
```

setup_mode: on

- **target_dir**: Ο προορισμός για την έξοδο. Εκεί που θα αποθηκεύει τα αρχεία εξόδου(εικόνες και βίντεο). Στα πειράματα θα αποθηκεύουμε τα αποτελέσματα σε ένα φάκελο output που δημιουργήσαμε μέσα στον προσωπικό φάκελο. Η διεύθυνση είναι /home/bo/output.

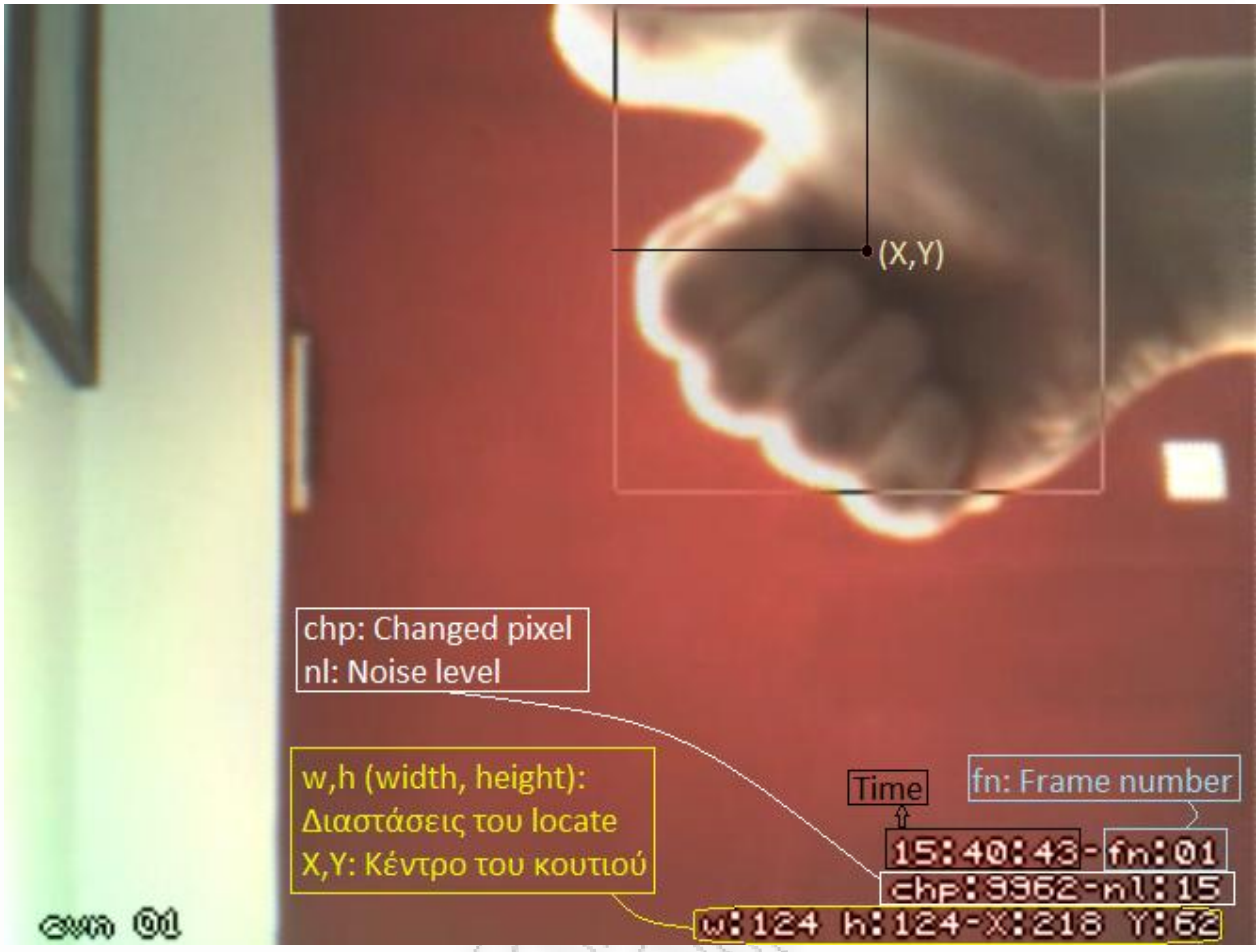
- **framerate:** Ορίζουμε την δειγματοληψία(καρέ ανά δευτερόλεπτο) την οποία είδαμε και αναλύσαμε στην εισαγωγή. Είναι αναμενόμενο πως όσο μεγαλύτερη τιμή ορίσουμε για την παράμετρο αυτή, τόσο μεγαλύτερο φορτίο εργασίας θα έχει να αντιμετωπίσει το σύστημά μας(ο επεξεργαστής μας). Την τιμή αυτής της μεταβλητής την έχουμε ορίσει 10, αλλά κατά τη διάρκεια των δοκιμών θα αλλάξει.

- **threshold:** Είναι η πιο σημαντική παράμετρος, όπου ορίζουμε την τιμή του ορίου για τα αλλαγμένα pixel, που αναλύσαμε στην εισαγωγή. Όταν είναι ενεργοποιημένη η παράμετρος setup_mode στο τερματικό δείχνει τον αριθμό των αλλαγμένων pixel για τα 2 διαδοχικά frames.

- **noise_level**: Χρησιμοποιείται σαν το threshold, δηλαδή ορίζει το όριο για το διαχωρισμό μεταξύ θορύβου από τη κάμερα και εντοπισμού κίνησης.

- **locate:** Σε κάθε καρέ που εντοπίζεται κίνηση, με δεδομένες τις συντεταγμένες του ανώτερου, κατώτερου, δεξιότερου και αριστερότερου σημείου, σχηματίζει ένα τετράγωνο που περικλείει την κίνηση:

- **text_right:** Μας επιτρέπει να τοποθετούμε κείμενο πάνω στις εικόνες ή στα βίντεο. Το κείμενο αυτό μπορούμε να το ορίσουμε να μας δίνει μία γκάμα πληροφοριών ανάλογα με το πως θα συνδυάσουμε τους χαρακτήρες στο αρχείο επιλογών βάζοντας μπροστά τους το σύμβολο "**%**". Οι χαρακτήρες που δεν έχουν μπροστά τους αυτό το σύμβολο, αναγράφονται όπως είναι. Παραδείγματος χάριν άμα γράψουμε %N θα μας εμφανίζει τη τιμή του θορύβου για κάθε frame. Εμείς έχουμε ορίσει το συνδυασμό %T-fn:%q\nchp:%D-nl:%N\nw:%i h:%J-X:%K Y:%L όπου %T είναι η ώρα τα λεπτά και τα δευτερόλεπτα με τη μορφή ΩΩ:ΛΛ:ΔΔ. %q που είναι ο αριθμός του frame. Έτσι βλέποντας το μεγαλύτερο αριθμό από το βίντεο μπορούμε να καταλάβουμε πόσο έχουμε ορίσει το framerate. Το %D που δείχνει το συνολικό αριθμό των αλλαγμένων pixel. Τα %i και %J δείχνουν το μήκος και το πλάτος της περιοχής που περιέχεται η κίνηση(του locate).Τα %K και %L είναι οι συντεταγμένες του κέντρου της κινούμενης περιοχής. Παρακάτω φαίνεται ένα στιγμιότυπο για να το καταλάβουμε καλύτερα:



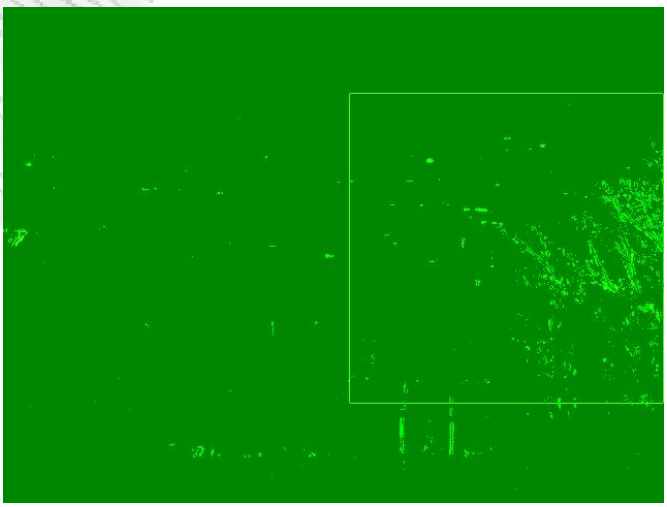
Στο προηγούμενο στιγμιότυπο βλέπουμε(κάτω δεξιά) ότι την ώρα 15:40 στο πρώτο frame του 43ου δευτερολέπτου, εντοπίστηκε κίνηση με 9.962 συνολικά αλλαγμένα pixel που είναι περισσότερα από τα 2.000 pixel που έχουμε ορίσει ως όριο. Το μήκος και το πλάτος του παραλληλογράμμου που περικλείει τη κίνηση έχει μήκος 124 pixel και ύψος επίσης 124 pixel (στη συγκεκριμένη περίπτωση είναι τετράγωνο) και το κέντρο του βρίσκεται 218ό pixel του οριζόντιου άξονα και στο 62ό pixel του κάθετου άξονα.

Προσοχή! : Η αρχή (συντεταγμένες 0,0) του μετρικού συστήματος του προγράμματος βρίσκεται στη πάνω αριστερή γωνία του frame.

- **despeckle:** Μεταφράζεται ως αποσημαδοποίηση (αφαιρεί την κηλίδωση, το στίγμα) και ρυθμίζει τη διάβρωση και τη διαστολή. Είναι ένας τρόπος ελάττωσης ή ενίσχυσης του θορύβου όταν εντοπίζει κίνηση στην εικόνα. Δηλαδή δίνει τη λύση στο πρόβλημα που περιγράψαμε στην εισαγωγή, όταν φυσάει ο άνεμος το γρασίδι και τα δέντρα μέσα στο τοπίο ή όταν μικρές διακυμάνσεις στο φωτισμό δημιουργούν πολλές κουκκίδες (θόρυβο) στις εικόνες που απεικονίζουν τη κίνηση. Με αυτήν την επιλογή λοιπόν αφαιρείται (ή ενισχύεται) αυτός ο θόρυβος και έτσι βελτιώνεται η αξιοπιστία της εικόνας που περιέχει κίνηση.

Η παράμετρος despeckle είναι μια πολύ ενδιαφέρουσα παράμετρος, με την οποία θα ασχοληθούμε πάραυτα. Ο λόγος που την κάνει σημαντική είναι αφενός ο τρόπος με τον οποίο δουλεύει και αφετέρου αποτελεί ένα αξιόλογο μεταβατικό στάδιο για το 4ο κεφάλαιο.

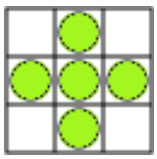
Όπως είπαμε το Motion εντοπίζει όλη την κίνηση, μέχρι και το τελευταίο pixel που αλλάζει. Έτσι λοιπόν όσο φυσάει ο άνεμος, τόσο περισσότερα pixel ανιχνεύονται, για αυτό θα πρέπει να αυξήσουμε την τιμή του threshold για να μη κατακλυστούμε από λάθος συναγερμούς. Δυστυχώς όμως αν αυξήσουμε πολύ το threshold θα περνούν αντικείμενα και θα ξεγλιστρούν χωρίς να εντοπίζονται.



Αν κοιτάξουμε τη δεξιά εικόνα(η οποία με ανοιχτό πράσινο χρώμα δείχνει την κίνηση) παρατηρούμε ότι παρόλα αυτά έχουμε σε όλη την επιφάνεια ανιχνεύσεις κίνησης που είναι πολύ μικρές. Άρα αν μπορούσαμε να αφαιρέσουμε όλες αυτές τις μικροσκοπικές ανιχνεύσεις κίνησης, θα μπορούμε να ελαττώσουμε το threshold έτσι ώστε να ξεχωρίζουμε την κίνηση που μας ενδιαφέρει πραγματικά.

Πως δουλεύει

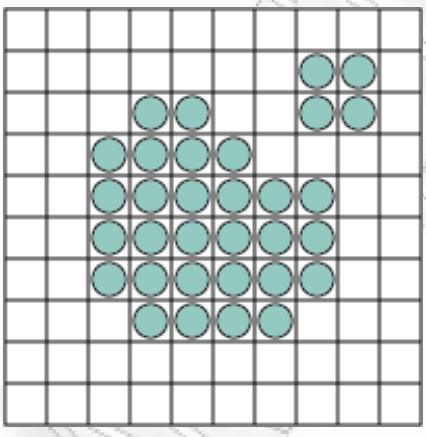
Το πρώτο που πρέπει να κάνουμε είναι να ορίσουμε ένα διαρθρωτικό στοιχείο:

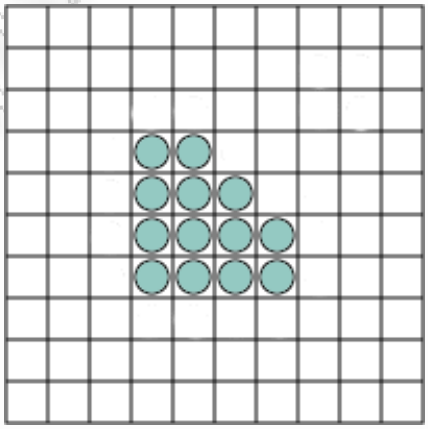


Το διαρθρωτικό στοιχείο, που απεικονίζει τα pixels, μπορεί να έχει οτιδήποτε σχήμα θέλουμε για να καλύπτει τα χαρακτηριστικά που μας ενδιαφέρουν. Επειδή λοιπόν εμάς μας ενδιαφέρουν οι μικρές μάζες(θα τις αναλύσουμε στο 4ο κεφάλαιο), επιλέγουμε το παραπάνω σχήμα σταυρού.

1. ***Χρησιμοποιώντας αυτό το σχήμα, σαρώνουμε με το σταυρό την αριστερή εικόνα***
2. ***Αν βρεις 5 ανιχνευμένα pixel με τον ίδιο σχηματισμό σταυρού,***
3. ***τότε σημείωσέ το ως μονή ανίχνευση.***

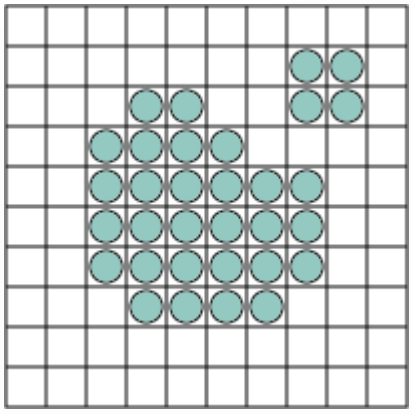
Οι ακόλουθες εικόνες δείχνουν τι συμβαίνει:

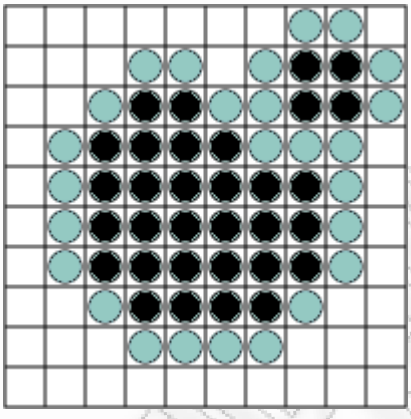




Από τη στιγμή που ο σταυρός των 5 pixel ανιχνεύεται, αντικαθιστάται με 1 μονό pixel. Η επίδραση αυτή πραγματοποιείται για να γίνει η ανίχνευση μικρότερη. Κανένα από τα 4 pixels στο 2x2 κουτί πάνω δεξιά δεν ταιριάζει στο διαρθρωτικό στοιχείο του σταυρού μας και έτσι το κουτί διαγράφεται.

Αυτή ήταν η περιγραφή για τη διάβρωση(αφαίρεση κίνησης). Η ίδια επεξεργασία μπορεί να γίνει αντίστροφα χρησιμοποιώντας τη διαστολή.





Εδώ σκανάρουμε την αριστερή εικόνα και για κάθε pixel στο οποίο εντοπίζεται κίνηση, μαρκάρονται και τα αντίστοιχα γύρω του σύμφωνα με το διαρθρωτικό στοιχείο. Η επίδραση της διαστολής κάνει την ανίχνευση μεγαλύτερη.

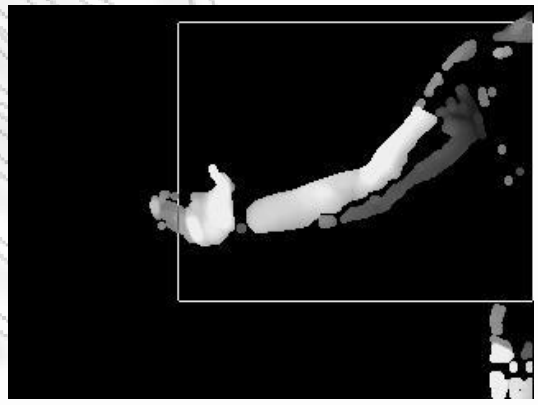
Δεν θα αναφερθούμε για τον τρόπο που χρησιμοποιούνται στο πρόγραμμα η επίδραση της διαστολής(dilation) και της διάβρωσης(erosion), απλώς θα αναφέρουμε μία ακόμα λειτουργία, την επισήμανση(labeling). Η οποία μετά το despeckle ψάχνει για περιοχές με κίνηση που είναι συνδεδεμένες και τις επισημαίνει με μπλε χρώμα. Έτσι το πρόγραμμα ενεργοποιείται με βάση τον αριθμό των αλλαγμένων pixel στη μεγαλύτερη περιοχή. Με άλλα λόγια η μεγαλύτερη αυτή περιοχή πρέπει να είναι πάνω από το threshold για να ενεργοποιήσει την ανίχνευση.

- **output_motion:** Δίνει ως έξοδο εικόνες μόνο με το κινούμενο αντικείμενο. Με αυτές τις εικόνες μπορούμε να δημιουργήσουμε ειδικές ταινίες στις οποίες μπορούμε να δούμε μόνο τα pixel που αλλάζουν στην κλίμακα του γκρι.

normal

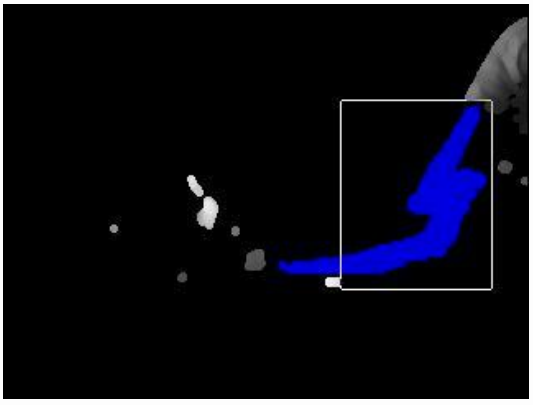
motion**(χωρίς despeckle)**

normal



despeckle: **on**

normal



label: **on**

3. Ανάλυση Motion

3.1 Εισαγωγή

Καταρχήν, για να δημιουργήσουμε κάποιο καινούριο φίλτρο, πρέπει πρώτα να καταλάβουμε πως δουλεύει το Motion. Χωρίς να μπούμε σε λεπτομέρειες ακόμα, και όπως ήδη γνωρίζουμε από την εισαγωγή, η ανίχνευση κίνησης για την παρούσα εργασία, στηρίζεται στη σύγκριση της έντασης των pixel κάθε καινούριας εικόνας από τη κάμερα, με την αμέσως προηγούμενη. Αν δεν υπάρχει κίνηση ή θόρυβος, η διαφορά της καινούριας με την προηγούμενη εικόνα (new_image - ref_image) θα πρέπει να είναι μηδέν "0". Αν υπάρχει αλλαγή στην εικόνα το αποτέλεσμα θα πρέπει να είναι διαφορετικό. Για την αποφυγή θορύβου, εκλαμβάνεται κίνηση, όταν η διαφορά αυτή, θα υπερβαίνει το όριο(threshold). Το πρόγραμμα Motion δεν χρησιμοποιεί ανάλυση χρωμάτων για τον εντοπισμό κίνησης.

Το καρέ αναφοράς (το αμέσως προηγούμενο frame) λειτουργεί αναδρομικά με τον εαυτό του ενημερώνοντας το με νέα εικόνα. Έτσι μετά από κάθε ενημέρωση με την N εικόνα θα αποτελείται από: 1/2 * εικόνα_N + 1/4 * εικόνα_(N-1) + 1/8 * εικόνα_(N-2) και τα λοιπά...

Όσο αφορά το παραλληλόγραμμο που περικλείει την κίνηση (locate) πάνω στην οθόνη, δεν είναι τίποτα άλλο από απλά μαθηματικά. Δηλαδή από όλα τα pixels που άλλαξαν, μεταξύ του καρέ αναφοράς και του καινούριου, υπολογίζεται ο μέσος όρος των **x** και **y** τιμών και βρίσκουμε το κέντρο της κίνησης. Έπειτα υπολογίζεται το μέσο της απόστασης από αυτό το σημείο. Τώρα ζωγραφίζεται ένα κουτί με διπλάσια απόσταση από το μέσο του **x** και **y** (3 φορές μεγαλύτερη απόσταση για το θετικό μέσο **y** ώστε να σιγουρευτούμε ότι τα κεφάλια από τους ανθρώπους θα περιλαμβάνονται στο κουτί) γύρω από το κέντρο της κίνησης.

Όταν η επισήμανση(label) είναι ενεργοποιημένη το κουτί της παραμέτρου locate υπολογίζεται βάση όλων των επισημασμένων περιοχών, του οποίου το μέγεθος είναι πάνω από τον ορισμό του threshold.

Για να γράψουμε το πρόγραμμα(συνάρτηση) που θα υλοποιεί τους παρακάτω αλγόριθμους των νέων φίλτρων, θα πρέπει πρώτα να δούμε τον κώδικα του motion. Όμως το πηγαίο αρχείο περιλαμβάνει 70 αρχεία τα οποία αποτελούνται από αρχεία κώδικα(**.c**), headers(**.h**) και διάφορα άλλα έγγραφα. Τα αρχεία κώδικα είναι 20: alg.c, conf.c, draw.c, event.c, ffmpeg.c, jpegutils.c, logger.c, md5.c, motion.c, netcam.c, netcam_ftp.c, netcam_jpeg.c, netcam_wget.c, picture.c, rotate.c, sdl.c, stream.c, track.c, video.c, video2.c, video_common.c, video_freebsd.c, vloopback_motion.c, webhttpd.c. Ύστερα, όταν κάνουμε compile το πρόγραμμα δημιουργούνται 110 αρχεία περίπου.

3.2 Μεταβλητές

Έτσι, το πρόγραμμα αυτό έχει πολλά αρχεία για όλες αυτές τις επιλογές και τις παραμέτρους που προσφέρει, άρα όταν εγκαθίσταται στο σύστημα, έχει πολλές γραμμές κώδικα, και είναι λογικό πως ο μηχανισμός λειτουργίας του είναι αρκετά σύνθετος και πολύπλοκος. Για αυτό λοιπόν εμείς θα εστιάσουμε στο κομμάτι και στις μεταβλητές που μας ενδιαφέρουν. Αυτές οι μεταβλητές είναι οι εξής:

Μεταβλητή	Τύπος	Λειτουργία
cnt->conf.minimum_motion_frames	int	ελάχιστος αριθμός των καρέ το ένα μετά το άλλο με ανίχνευση κίνησης που απαιτείται για την ενεργοποίηση κίνηση.
cnt->conf.pre_capture	int	αριθμός των καρέ pre_captured που αποθηκεύονται πριν από την ανίχνευση κίνησης.
cnt->precap_nr	int	μέγεθος του frame στο buffer (αριθμός καρέ σε buffer).
cnt->precap_cur	int	τρέχουσα θέση του buffer στο οποίο αποθηκεύεται η τελευταία καταγραφή καρέ στο βίντεο. Η πρώτη θέση είναι 0.
cnt->moved	int	Όταν αυτή η μεταβλητή δεν είναι μηδέν, η ανίχνευση κίνησης είναι απενεργοποιημένη. Αλλάζει σε μη μηδενική τιμή όταν η συνθήκη lightswitch ανιχνεύεται.
cnt->shots	int	Ο αριθμός πλαισίου κατά τη διάρκεια αυτού του δευτερολέπτου. Το πρώτο καρέ είναι 0.
frame_buffer_size	int	Υπολογισμός του σωστού μεγέθους του frame buffer με βάση τον τύπο: frame_buffer_size = cnt->conf.pre_capture + cnt->conf.min_mtn_fr - 1. Αν το frame_buffer_size δεν έχει την ίδια τιμή με το cnt->precap_nr το frame buffer αλλάζει μέγεθος
cnt->imgs.new	*char	Δείκτης για το πρώτο byte (byte 0) στο frame buffer = Δείκτης για πρώτο καρέ (frame 0).

cnt->imgs. timestamp	*tm	Δείκτης στο πρώτο timestamp στο timestamp buffer. Χρησιμοποιείται για το όνομα αρχείου και την εμφάνιση κειμένου.
cnt->imgs. shotstamp	int	Δείκτης για την τιμή της πρώτης λήψης (μετρητής πλαισίου για κάθε δευτερόλεπτο) στη λήψη buffer. Χρησιμοποιείται για το όνομα αρχείου και την εμφάνιση κειμένου.

3.3 motion.c

Η καρδιά του Motion βρίσκεται στο εκτελέσιμο αρχείο **motion.c** στη συνάρτηση **motion_loop()** που περιέχει τον βρόχο που καλείται συνέχεια. Ο βρόχος αποτελείται από τα στάδια *προετοιμασίας για νέο frame*, *επανάληψη αρχικοποίηση του Netcam τμήματος*, *τμήμα σύλληψης εικόνας*, *τμήμα ανίχνευσης κίνησης*, *τμήμα tuning*, *τμήμα επικάλυψης κειμένου και γραφικών*, *τμήμα ελέγχου ενεργειών και συμβάντων*, *τμήμα αναφοράς frame*, *ρύθμιση λειτουργίας τερματικού για τμήμα εξόδου*, *τμήμα για τα χαρακτηριστικά στιγμιότυπων*, *τμήμα για το χαρακτηριστικό λήξης*, *τμήμα επανάληψης για το βίντεο*, *τμήμα ενημέρωσης των ανά δευτερόλεπτο παραμέτρων* και το *τμήμα ορισμού καρέ για το χρονοδιάγραμμα ύπνου*.

Τώρα που είδαμε ποιες είναι οι σημαντικότερες μεταβλητές και τι σημαίνουν θα αναλύσουμε τα τμήματα που μας ενδιαφέρουν, δηλαδή αυτά της **αναγνώρισης κίνησης** και της **επικάλυψης κειμένου και γραφικών** της συγκεκριμένης συνάρτησης. Παρακάτω φαίνονται οι αλγόριθμοι των τμημάτων και έπειτα οι κώδικες που τους υλοποιούν. Έχουν αφαιρεθεί όμως τα τμήματα αυτά που υλοποιούν τις παραμέτρους στις οποίες δεν έχουμε αναφέρει στο δεύτερο κεφάλαιο. Έχουμε κρατήσει δηλαδή, εκείνα τα σημεία των παραμέτρων που αναλύσαμε παραπάνω και τα οποία θα χρειαστούν στο τέταρτο κεφάλαιο για τα φίλτρα αναγνώρισης προτύπων.

3.3.1 Τμήμα σύλληψης εικόνας

- Φέρε το επόμενο καρέ από την κάμερα (vid_next)
  - Αν η κάμερα χρησιμοποιεί μέθοδο Video for Linux(V4L) καταγραφής
    - Κάλεσε τη 4l_set_input για τη ρύθμιση της συσκευής
    - Φέρε την εικόνα με v4l_next
      - Σε περίπτωση αποτυχίας επέστρεψε vid_return_code=-1 (μοιραίο)
  - Επιστροφή εικόνας και όρισε το vid_return_code=0 (επιτυχία)
- Αν το vid_return_code είναι 0 (επιτυχία)
  - Επαναφορά του missing_frame_counter
  - Αποθήκευσε τη νέα εικόνα στο cnt->imgs.image_virgin όπου θα κρατηθεί αμετάβλητη (χωρίς να προστεθούν κείμενα και επικαλύψεις)

3.3.2 Τμήμα ανίχνευσης κίνησης

- Η ανίχνευση κίνησης
  - Χρησιμοποίησε το πρότυπο diff αν ανιχνευθεί κίνηση ή αν βρίσκεται σε λειτουργία setup, αλλιώς χρησιμοποίησε το fast(γρήγορο)-diff. Αν το fast-diff βρει πολλά αλλαγμένα pixel ενεργοποίησε το κανονικό diff.
    - Αντιστάθμισε το επίπεδο σκοτεινού θορύβου αν είναι ενεργοποιημένο
    - Δρομολόγησε τα pixels σε frame κίνησης (cnt-> imgs.out) σε μαύρο (YUV420P, Y = 0, UV = 0x80)
    - Pixels με ανίχνευση κίνησης αντιγράφονται στην εικόνα της κίνησης (cnt-> imgs.out) (Y κανάλι μόνο = Άσπρο / Μαύρο).
    - Επέστρεψε τα diffs
  - Despeckle
    - Ρυθμίστε diffs με διάβρωση και διαστολή
      - Κίνηση εικόνας (cnt-> imgs.out) θα προσαρμοστεί ανάλογα με την αφαίρεση των pixels και την αντιγραφή γειτονικών pixels.
    - Ρυθμίστε τα diffs με σήμανση (diffs στη μεγαλύτερη επισημασμένη περιοχή)
  - Τα αποτελέσματα ανίχνευσης κίνηση είναι ένας αριθμός diffs

```
/* Η πραγματική ανίχνευση κίνησης λαμβάνει χώρα παρακάτω
 * Τα diffs είναι ο αριθμός των pixels που ανιχνεύονται καθός αλλάζουν.
 * Δημιούργησε μία εικόνα με τις διαφορές(motion) στο image_out.
 *
 * Ο alg_diff_standard είναι πιο αργός αλγόρυθμος
 *  για τη πλήρη ανίχνευση στοιχείων με κίνηση.
 * Ο alg_diff πρώτα καλεί ενα γρήγορο αλγόρυθμο ανίχνευσης ο οποίος "κοιτάει" μόνο
 *  ένα κλάσμα των pixels. Αν εντοπίσει πιθανή κίνηση καλείτε ο alg_diff_standard.
 */
if (cnt->threshold && !cnt->pause) {
    /* Εάν έχουμε ήδη ανιχνεύσει κίνηση και θέλουμε να δούμε αν υπάρχει
     * ακόμα κίνηση, δεν χρειάζεται να προσπαθούμε το γρήγορο αλγόρυθμο πρώτα.
     * Εάν υπάρχει κίνηση, ο alg_diff θα ενεργοποιήσει τον alg_diff_standard
     * έτσι κι αλλιώς
     */
    if (cnt->detecting_motion || cnt->conf.setup_mode)
        cnt->current_image->diffs = alg_diff_standard(cnt, cnt->imgs.image_virgin);
    else
        cnt->current_image->diffs = alg_diff(cnt, cnt->imgs.image_virgin);
```

```
int alg_diff_standard (struct context *cnt, unsigned char *new)
{
    struct images *imgs = &cnt->imgs;
    int i, diffs = 0;
    int noise = cnt->noise;
    int smartmask_speed = cnt->smartmask_speed;
    unsigned char *ref = imgs->ref;
    unsigned char *out = imgs->out;
    unsigned char *mask = imgs->mask;
    unsigned char *smartmask_final=imgs->smartmask_final;
    int *smartmask_buffer = imgs->smartmask_buffer;
#ifdef HAVE_MMX
    mmx_t mmtemp; /* used for transferring to/from memory */
    int unload;   /* counter for unloading diff counts */
#endif

    i = imgs->motionsize;
    memset(out + i, 128, i / 2); /* motion pictures are now b/w i.o. green */
    /* Keeping this memset in the MMX case when zeroes are necessarily
     * written anyway seems to be beneficial in terms of speed. Perhaps a
     * cache thing?
     */
    memset(out, 0, i);
```

3.3.3 Τμήμα κειμένου και γραφικών

- Επικάλυψε τη μεγαλύτερη περιοχή(labeling) με μπλε χρώμα στο motion frame
- Εντόπισε το κέντρο και το μέγεθος της ανιχνευμένης κίνησης, εφόσον υπάρχει κίνηση
  - Υπολόγισε το γεωμετρικό μέσο όρο των label που ξεπερνούν το threshold ή όλα τα ανιχνευμένα pixels εάν το label είναι απενεργοποιημένο.
  - Υπολόγισε το γεωμετρικό μέσο όρο όλων των pixels στα αριστερά και δεξιά του κέντρου x και πάνω και κάτω από το κέντρο y
  - Υπολογίστε το locate κουτί ως 2 x(επί) τη μισή ακριβώς απόσταση εκτός από το πάνω y, το οποίο πολλαπλασιάζεται με το 3, επειδή η εμπειρία δείχνει ότι το κουτί περιέχει καλύτερα το ανθρώπινο κεφάλι.
  - Ρυθμίστε το κέντρο y να περιλαμβάνει το ανθρώπινο κεφάλι στο κουτί.
- Πρόσθεσε κείμενο text_changes
- Προσθέστε το text_right στην εικόνα

```
/****** MOTION LOOP - Τμήμα επικάλυψης κειμένου και γραφικών *****/

/* Κάποιες επικαλύψεις πάνω από την εικόνα κίνησης
 * Σημειώστε ότι αυτά τώρα τροποποιούν το cnt->imgs.out, άρα αυτό το buffer
 * δεν μπορεί να χρησιμοποιηθεί πλέον για λειτουργίες ανίχνευσης κίνησης μέχρι
 * να συλληφθεί η επόμενη εικόνα-frame.
 */

/* μεγαλύτερη επικάλυψη label  */
if (cnt->imgs.largest_label && (cnt->conf.motion_img || cnt->conf.ffmpeg_cap_motion || cnt->conf.setup_mode))
    overlay_largest_label(cnt, cnt->imgs.out);

/* Πρόσθεσε τα αλλαγμένα pixel στη πάνω δεξιά γωνία της εικόνας */
if (cnt->conf.text_changes) {
    char tmp[15];

    if (!cnt->pause)
        sprintf(tmp, "%d", cnt->current_image->diffs);
    else
        sprintf(tmp, "-");

    draw_text(cnt->current_image->image, cnt->imgs.width - 10, 10, cnt->imgs.width,
              tmp, cnt->conf.text_double);
}

/* Πρόσθεσε κείμενο στη κάτω δεξιά γωνία της εικόνας */
if (cnt->conf.text_right) {
    char tmp[PATH_MAX];
    mystrftime(cnt, tmp, sizeof(tmp), cnt->conf.text_right, &cnt->current_image->timestamp_tm, NULL, 0);
    draw_text(cnt->current_image->image, cnt->imgs.width - 10, cnt->imgs.height - 10 * text_size_factor,
              cnt->imgs.width, tmp, cnt->conf.text_double);
}
```

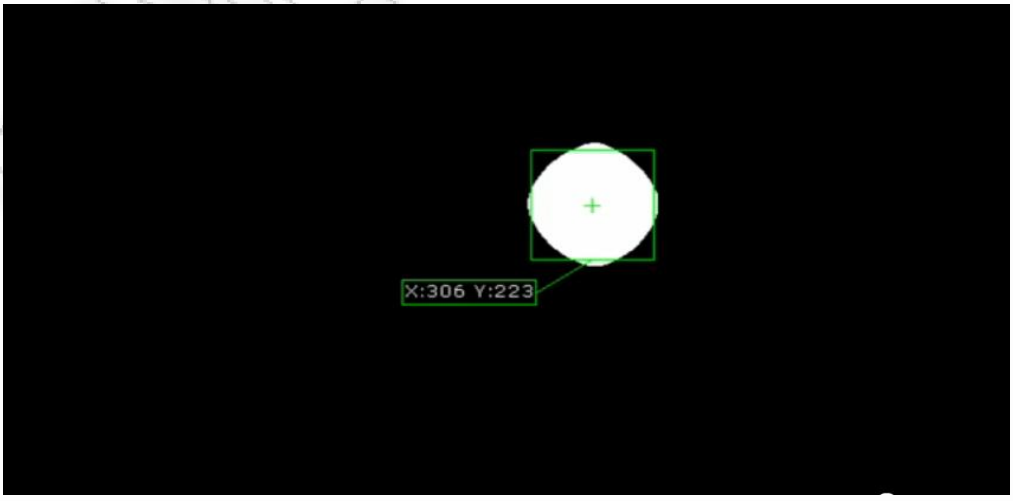
4. Φίλτρα αναγνώρισης προτύπων

Αναγνώριση ή ταξινόμηση προτύπου ονομάζεται η διαδικασία εκείνη, κατά την οποία, σήματα του περιβάλλοντος χώρου, που αντιστοιχούν σ' ένα αντικείμενο(πρότυπο), ταξινομούνται σε μια κατηγορία, από ένα πεπερασμένο σύνολο κατηγοριών Ν (κατηγορίες αντικειμένων).

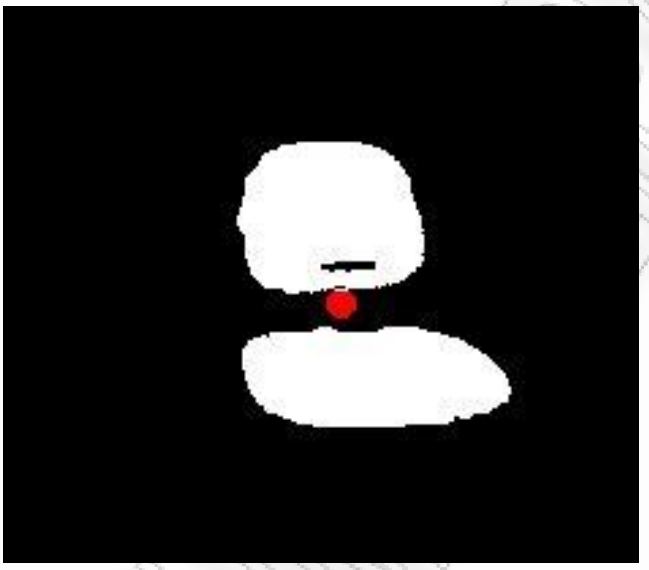
4.1 Blob και κέντρο βάρους

Η λέξη blob μεταφράζεται ως άμορφη μάζα. Αφού όταν εντοπίσει κίνηση το πρόγραμμα και δεν μπορεί να γνωρίζει τι είναι αυτό το αντικείμενο που κουνήθηκε, το θεωρεί ως ένα σύνολο από pixels με ακαθόριστο σχήμα, δηλαδή μια άμορφη δισδιάστατη μάζα. Ο αλγόριθμος παρακάτω χρησιμοποιείται ώστε να καθοριστεί εάν μια ομάδα από pixels σχετίζονται μεταξύ τους. Αυτό είναι χρήσιμο για τον εντοπισμό μεμονωμένων αντικείμενων σε μια σκηνή, ή μετρώντας τον αριθμό των αντικειμένων σε μια σκηνή. Ο blob αλγόριθμος ανίχνευσης θα ήταν χρήσιμος για την μέτρηση των ανθρώπων σε μια αίθουσα αεροδρομίου, ή για τα ψάρια που περνούν μπροστά από μια κάμερα σε ποτάμι.

Για να βρούμε μια άμορφη μάζα, θέτουμε ένα όριο στην εικόνα από ένα συγκεκριμένο αριθμό αλλαγμένων pixel, όπως φαίνεται παρακάτω. Ο πράσινος σταυρός αντιπροσωπεύει το κέντρο βάρους, ή το μέσο όρο θέσης όλων pixels του αντικειμένου.



Έτσι όταν υπάρχει μόνο μία μάζα-αντικείμενο σε μια σκηνή, το κέντρο βάρους είναι πάντα στο εσωτερικό ενός αντικειμένου. Αλλά τι γίνεται εάν υπήρχαν δύο ή περισσότερες μάζες-αντικείμενα? Αυτό είναι ένα πρόβλημα, δεδομένου ότι το κέντρο βάρους δεν θα είναι πλέον μέσα σε κάθε αντικείμενο:



Για να λυθεί αυτό το πρόβλημα, ο αλγόριθμός μας, πρέπει να επισημαίνει κάθε μάζα-αντικείμενο ως ξεχωριστή οντότητα. Για να το κάνουμε αυτό, εκτελούμε αυτόν τον αλγόριθμο:

Σκανάρονται με τη σειρά όλα τα pixel του frame:

αν το pixel είναι αλλαγμένο(εντοπιστεί κίνηση),
στον πίνακα κίνησης βάλε «1»
αλλιώς βάλε «0»
πήγαινε στο επόμενο pixel εάν είναι επίσης αλλαγμένο (εντοπίστηκε κίνηση) και αν συνορεύει σε pixel με 1
βάλε και εδώ «1»
αλλιώς βάλε «2» (ή περισσότερο)

Επανέλαβε έως ότου τελειώσουν όλα τα pixel…

Ο προηγούμενος αλγόριθμος βάζει στον πίνακα έναν αριθμό για κάθε καινούριο μάζα-αντικείμενο που βρίσκει:

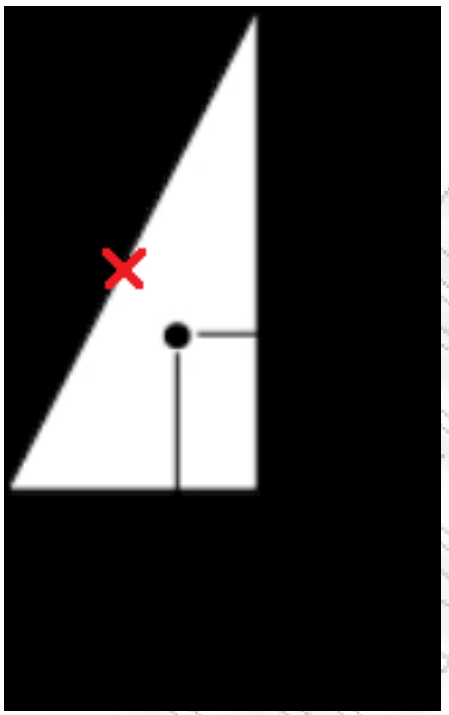
Ο κώδικας που υλοποιεί τον παραπάνω αλγόριθμο είναι ο εξής:

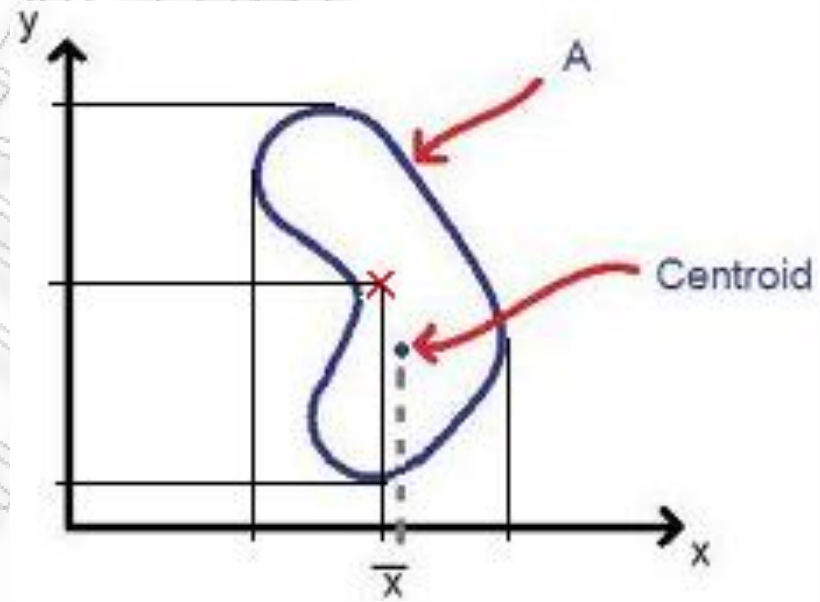
```
/* motion  Ο πίνακας που αποθηκεύεται η κίνηση (0-δεν υπάρχει κίνηση, 1,2,3,...-τα αντικείμενα)
   *new    είναι το νέο frame από την κάμερα
   *ref    είναι το αμέσως προηγούμενο frame που θα συγκριθεί με το *new
  (*ref-*new) */

int motion[width][heigth], *new[width][height], *ref[width][height];
int i = 1;

for (y = 0; y <= height; y++) {
    for (x = 0; x <= width; x++) {
        if (*new[x][y] != *ref[x][y]) {   // Αν βρείς αλλαγμένο pixel τότε
            if        (motion[x-1][y-1]) {    // αν το πάνω αριστερά του pixel έχει motion τότε
                motion[x][y] = motion[x-1][y+1]; //βάλε τον αριθμό (αντικειμένου) που περιέχει αυτό το pixel

            } else if (motion[x][y+1]) {  // Αλλιώς αν είναι αλλαγμένο το ακριβώς από πάνω pixel τότε
                motion[x][y] = motion[x][y+1];  // βάλε τον αριθμό αυτόν

            } else if (motion[x+1][y+1]) {  // Πάνω δεξιά pixel
                motion[x][y] = motion[x+1][y+1];

            } else if (motion[x-1][y]) {  // Αριστερό pixel
                motion[x][y] = motion[x-1][y];

            } else {                  // Αλλιώς αν κανένα από αυτά δεν έχει motion
                    moton[x][y] = i;   // τότε βάλε τον επόμενο αριθμό (καινούριο αντικείμενο)
            }
            i = i + 1;
        } else {
            motion[x][y] = 0;

        }
    }
}
```

Κέντρο βάρους

Στη συνέχεια, στο αρχείο κώδικα alg.c υπάρχει υλοποιημένη μια συνάρτηση(alg_locate_center_size) η οποία εντοπίζει το κέντρο βάρους(όχι το κέντρο της κίνησης) προσθέτοντας τα αλλαγμένα pixel των αξόνων x, y και ύστερα διαιρώντας τα με το συνολικό αριθμό τους. Δεν θα αναλύσουμε περεταίρω τον αλγόριθμο αυτό αλλά θα προσθέσουμε στη συνάρτηση alg_draw_location (επίσης του αρχείου κώδικα alg.c) τις εντολές που εμφανίζουν το κέντρο βάρους.

Για να καταλάβουμε περίπου τη διαφορά μεταξύ κέντρου κίνησης και κέντρου βάρους θα δούμε τα παρακάτω σχήματα γνωρίζοντας ότι στο κέντρο κίνησης βρίσκουμε το μέσο της απόστασης μεταξύ κατώτερου και ανώτερου, δεξιότερου και αριστερότερου pixel, φέρνουμε κάθετες στα σημεία αυτά και εκεί που ενώνονται έχουμε το κέντρο κίνησης. Δηλαδή στα σχήματα που ακολουθούν το κέντρο κίνησης θα έπρεπε να είναι κόκκινο “x”, ενώ το κέντρο βάρους η μαύρη κουκίδα:



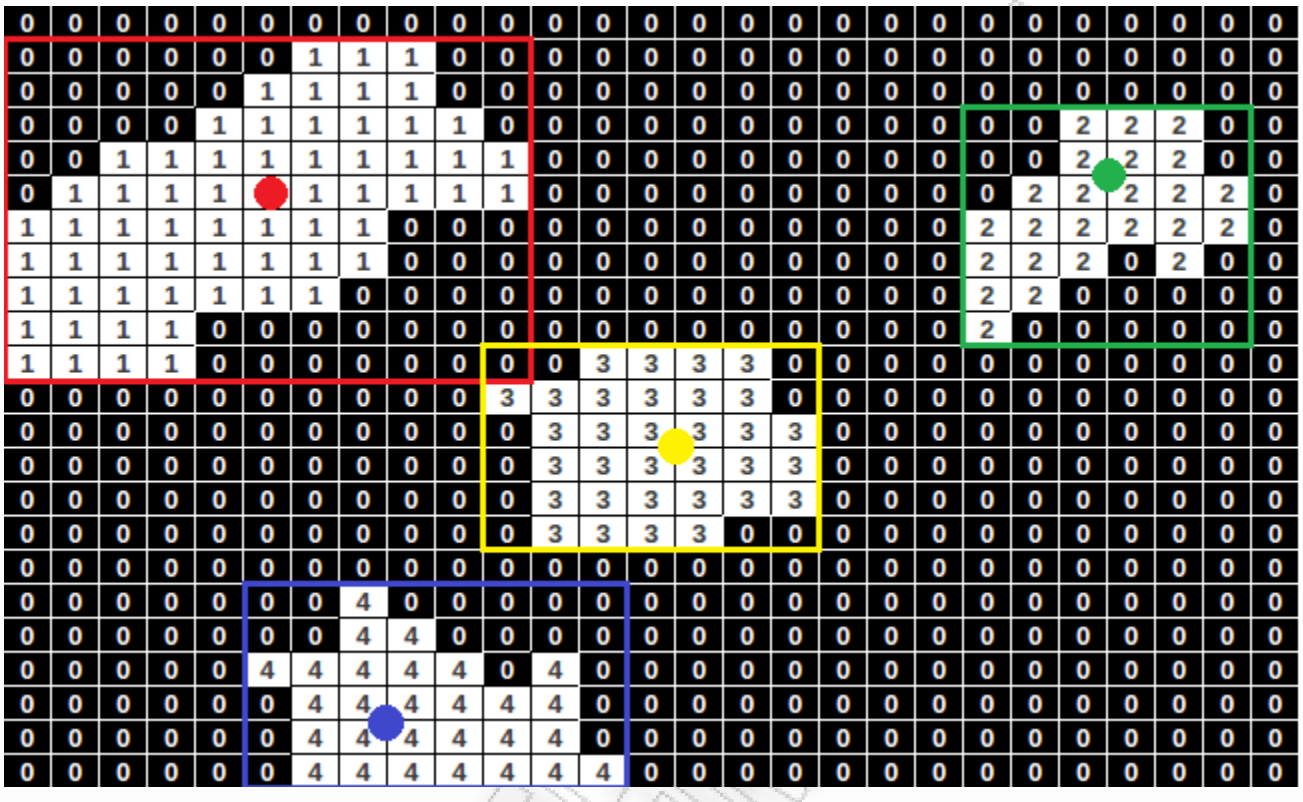


Οι εντολές λοιπόν που πρέπει να προσθέσουμε στο alg_draw_location είναι:

```
/* center cross */
int width_maxyk = width * cent->y;

for (x = cent->x-3; x <= cent->x+3; x++) { //οριζόντια
    int width_maxy_xk = x + width_maxyk;
    new[width_maxy_xk]=~new[width_maxy_xk];
}

for (y = cent->y-3; y <= cent->y+3; y++) { //κάθετη γραμμή
    int width_maxx_yk = cent->x + y * width;
    new[width_maxx_yk]=~new[width_maxx_yk];
}
```

Όπου **cent->x** και **cent->y** οι συντεταγμένες του κέντρου βάρους της κίνησης. Ο αριθμός **3** στη συνθήκη σημαίνει ότι εκατέρωθεν του σημείου θα τυπώνονται(εμφανίζονται) τρία pixels. Έτσι, έχοντας 6 pixel σε οριζόντια γραμμή και άλλα 6 σε κάθετη, σχηματίζεται ένας σταυρός:

Συνδυάζοντας Blob και κέντρο βάρους ο πίνακάς μας θα πρέπει να μοιάζει κάπως έτσι:



4.2 Ποσοστά και "Πτώση"

Μία άλλη παράμετρος που θα μπορούσε να προστεθεί στο πρόγραμμα είναι, μία ετικέτα δίπλα από κάθε αντικείμενο η οποία θα αναφέρει το ποσοστό κάθε μάζας-αντικειμένου που καταλαμβάνει από το συνολικό frame. Οι διαστάσεις των εικόνων που λαμβάνονται από την κάμερα είναι **320x240 pixels**, αυτό ισούται με **76800** pixels, το οποίο είναι ίσο με το **100%** του μεγέθους της οθόνης.

Αρχικά το ποσοστό της κίνησης μπορούμε να το εφαρμόσουμε για όλα τα αντικείμενα του κάθε frame με μια απλή πράξη. Στο αρχείο motion.conf (κεφάλαιο 2) υπάρχει μια παράμετρος text_change, η οποία είναι παρόμοια με την text_right (κεφάλαιο 2). Αυτή αρχικοποιείται στο αρχείο κώδικα motion.c (κεφάλαιο 3) και εφαρμόζουμε τη παρακάτω τροποποίηση:

```
/* Add percentage changed pixels in upper right corner of the pictures */
if (cnt->conf.text_changes) {
    char tmp[15];

    if (!cnt->pause)
        sprintf(tmp, "%d", (cnt->current_image->diffs*100)/76800);
    else
        sprintf(tmp, "-");
    draw_text(cnt->current_image->image, cnt->imgs.width - 10, 10, cnt->imgs.width,
              tmp, cnt->conf.text_double);
}
```

Αλλαγμένα pixels

%

Συνολικά pixel

αλγόριθμος σχεδίασης πάνω στις εικόνες (draw.c)

Τρέχοντας τώρα το πρόγραμμα, εμφανίζεται πάνω δεξιά η τιμή που ζητάμε:

Με έναν γρήγορο υπολογισμό ελέγχουμε την ορθότητα της τιμής:

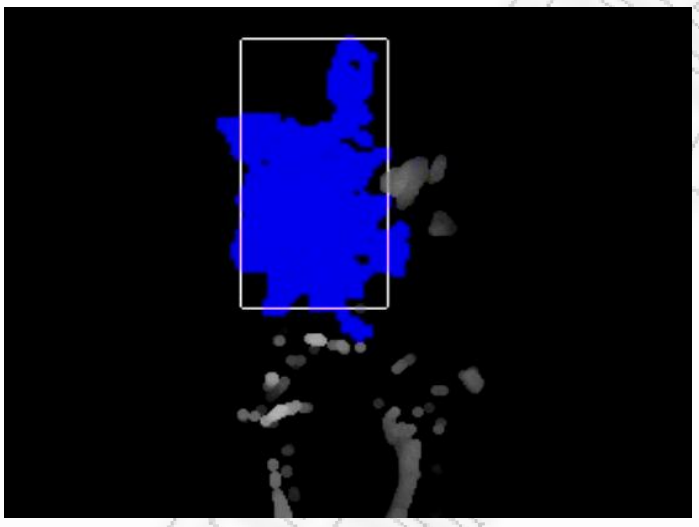
$$\textit{Ποσοστό κίνησης} = \frac{\text{chp} = \mathbf{13966}\ (\text{αλλαγμένα pixels})}{\mathbf{76800}\ (\text{συνολικά pixel})} = 0{,}18 \cdot 100\% = \mathbf{18}\ \%$$

Με αυτό το φίλτρο αποτρέπουμε το χρήστη από το να "μπλεχτεί" με τα pixel, τις διαστάσεις της εικόνας και τις πράξεις. Εφόσον γνωρίζει με ποιο ποσοστό ανίχνευσης της εικόνας θέλει να ενεργοποιείται το Motion δεν χρειάζεται να υπολογίζει με πόσα pixel αντιστοιχεί.

Στη συνέχεια, εφόσον με το locate περικλείουμε τη κίνηση η οποία έχει ακαθόριστο σχήμα, μπορούμε να υπολογίσουμε το ποσοστό της κίνησης μέσα στο κουτί. Άρα έχοντας το ποσοστό τις κίνησης σε σχέση με ολόκληρη την οθόνη, το ποσοστό του κουτιού σε σχέση με ολόκληρη την οθόνη και τις διαστάσεις του κουτιού, μπορούμε να καταλάβουμε το πόσο διάσπαρτη ή συμπαγής είναι η κίνηση.

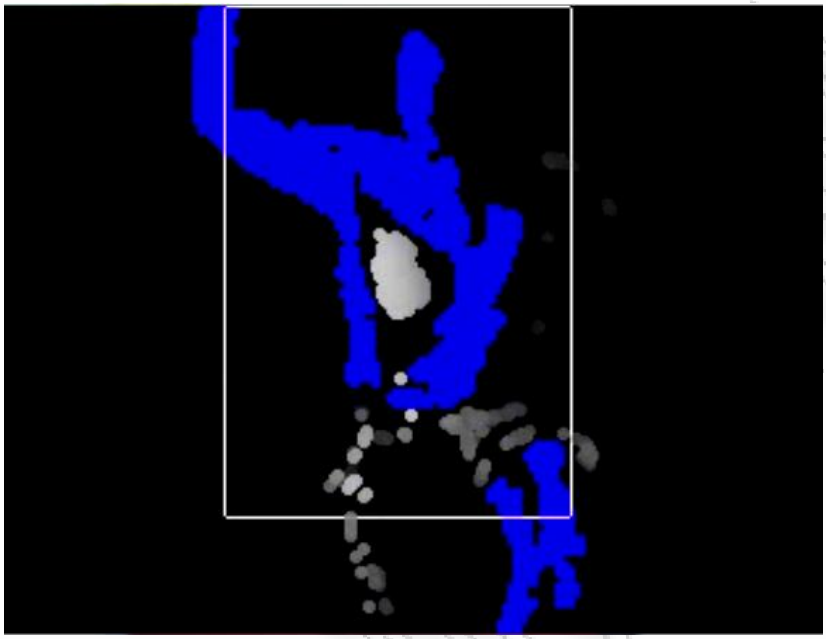
Παραδείγματος χάριν παρακάτω έχουμε 2 εικόνες που στην πρώτη έχουμε συμπαγή κίνηση, ενώ στη δεύτερη διάσπαρτη. Τις 2 εικόνες τις δείχνουμε και normal και motion:

Όπως μπορούμε να διακρίνουμε το **ποσοστό κίνησης (motion percentage)** σε σχέση με όλη την οθόνη είναι **7%** και το **ποσοστό** που καταλαμβάνει το **κουτί (box percentage)** σε σχέση με όλη την οθόνη είναι **9%**, άρα η κίνηση είναι συμπαγής διότι οι δύο αυτοί αριθμοί συγκλίνουν. Ποιο αναλυτικά το $\frac{\mathbf{5597}\ (κίνηση)}{\mathbf{64}*\mathbf{115}\ (διαστάσεις\ κουτιού)} * 100\% =$ **76%** της επιφάνειας του κουτιού(locate) καλύπτεται από κίνηση. Εκτός από αυτό, μπορούμε να το διακρίνουμε από την έξοδο του motion, όπου το κουτί είναι σχεδόν πλήρες από τη κίνηση(μπλε):



Παρακάτω τα αντίστοιχα ποσοστά είναι **8%** και **33%.** Αυτό σημαίνει ότι η κίνηση είναι πιο διάσπαρτη στην οθόνη δηλαδή αν εκτελέσουμε τις αντίστοιχες πράξεις μόνο το **26%** του locate καλύπτεται:

Όπως είπαμε και προηγουμένως τα αποτελέσματα αυτών των τιμών μπορούμε να τα διακρίνουμε οπτικά από τις εικόνες εξόδου τύπου motion του προγράμματος:



"Πτώση"

Ο λόγος που εφαρμόσαμε τις παραπάνω μετρήσεις είναι ότι σε συνδυασμό με τις επόμενες μετρήσεις, που έχουν να κάνουν με τις διαστάσεις του κουτιού, θα μπορέσουμε να αναγνωρίσουμε πότε κάποιος(ένα σώμα) πέσει. Είναι λογικό πως όταν κάποιος είναι όρθιος μπροστά από την κάμερα, οι διαστάσεις του locate, θα είναι **μεγαλύτερες στο άξονα του y** και **μικρότερες στο x** όπως φαίνεται και στην παρακάτω εικόνα:

Εδώ το σώμα έχει **64 pixels πλάτος** και **186 pixels ύψος**. Όταν λοιπόν πέσει περίπου κάθετα σε σχέση με τον άξονα λήψης της κάμερας, θα έχει μικρότερο ύψος και μεγαλύτερο πλάτος το κουτί. Επίσης όταν μιλάμε για λήψη κατά μήκος του ορίζοντα(δηλαδή η κάμερα πρέπει να είναι τοποθετημένη παράλληλα με το έδαφος) πρέπει η ¨πτώση¨ να είναι κάθετη σε σχέση με τον άξονα λήψης. Αν πέσει παράλληλα του άξονα λήψης τότε το locate θα μοιάζει με τετράγωνο:

Όπως εδώ που έχει **124 pixel πλάτος** και **125 pixel ύψος**

Αν λοιπόν ικανοποιηθούν αυτές οι συνθήκες(ο άξονας του x να είναι μεγαλύτερος από αυτόν του y) και η κίνηση είναι συμπαγής(τα ποσοστά κίνησης και κουτιού συγκλίνουν), τότε το σώμα (ο υποτιθέμενος άνθρωπος) σημαίνει ότι έπεσε. Αυτό το φίλτρο το έχουμε βάλει με κάποιες απλές εντολές κώδικα να φαίνεται στο πάνω δεξιό μέρος της οθόνης με τη λέξη **"Felt"**. Θα μπορούσε να χρησιμοποιηθεί, λόγω των περιορισμών από τη γωνία λήψης, σε κάποιο διάδρομο ή τούνελ όπου οι άνθρωποι περνάνε κάθετα από τη κάμερα, η οποία θα είναι τοποθετημένη στο μέσο του διαδρόμου ή τούνελ ή σκάλας. Επιπλέων αν υπάρχει η δυνατότητα τοποθέτησης της κάμερας σε ύψος ούτως ώστε να έχει πανοραμική κάλυψη ενός χώρου, μπορεί επίσης να λειτουργήσει ο αλγόριθμος πτώσης, διότι ικανοποιούνται οι προϋποθέσεις ενεργοποίησης του φίλτρου(συναγερμού).

Αν λάβει ο χρήστης υπόψη του αυτές τις προϋποθέσεις κατά την εγκατάσταση της κάμερας τότε θα έχει τα παρακάτω αποτελέσματα:

Κώδικας "πτώση"

Ο κώδικας που ενσωματώσαμε στο πρόγραμμα και συγκεκριμένα στο αρχείο motion.c για την υλοποίηση αυτού του φίλτρου φαίνεται παρακάτω:

```
/* Πάνω από το 45% του κουτιού ενεργοποιείτε το φίλτρο */
/* Add changed pixels in upper right corner of the pictures */
if (cnt->conf.text_changes) {
    char tmp[PATH_MAX];

        mystrftime(cnt, tmp, sizeof(tmp), cnt->conf.text_changes, &cnt->current_image->timestamp_tm, NULL, 0);
        if ( cnt->current_image->diffs > ( cnt->current_image->location.width * cnt->current_image->location.height ) * 45 / 100 &&
             cnt->current_image->location.width > 1.5 * cnt->current_image->location.height ){
            sprintf(tmp, "Felt");
        } else {
            sprintf(tmp, "-");
        }

    draw_text(cnt->current_image->image, cnt->imgs.width - 10, 10, cnt->imgs.width,
              tmp, cnt->conf.text_double);
}
```

Αν τα αλλαγμένα pixels είναι περισότερα απο το 45% του μεγέθους του κουτιού

και

Το μήκος του κουτιού είναι 1,5 φορές μεγαλύτερο από το ύψος

τότε εμφάνισε Felt

Για την εμφάνιση των παραπάνω μετρήσεων στο πλάνο(box, motion per), προσθέσαμε στη συνάρτηση mystrftime του motion.c μερικές σειρές κώδικα που μας βοηθούν στο αρχείο επιλογών (motion.conf) για το κάλεσμά τους. Αυτές οι γραμμές είναι οι εξής:

```
2591 /**
2592  * mystrftime
2593  *
2594  *   Motion-specific variant of strftime(3) that supports additional format
2595  *   specifiers in the format string.
2596  *
2597  * Parameters:
2598  *
2599  *   cnt        - current thread's context structure
2600  *   s          - destination string
2601  *   max        - max number of bytes to write
2602  *   userformat - format string
2603  *   tm         - time information
2604  *   filename   - string containing full path of filename
2605  *                set this to NULL if not relevant
2606  *   sqltype    - Filetype as used in SQL feature, set to 0 if not relevant
2607  *
2608  * Returns: number of bytes written to the string s
2609  */
2610 size_t mystrftime(struct context *cnt, char *s, size_t max, const char *userformat,
2611                   const struct tm *tm, const char *filename, int sqltype)
2612 {
2613     char formatstring[PATH_MAX] = "";
2614     char tempstring[PATH_MAX] = "";
2615     char *format, *tempstr;
2616     const char *pos_userformat;
2617
2618     format = formatstring;
2619
2620     /* if mystrftime is called with userformat = NULL we return a zero length string */
2621     if (userformat == NULL) {
2622         *s = '\0';
2623         return 0;
2624     }
```

...

```
case 'D': // diffs
    sprintf(tempstr, "%d", cnt->current_image->diffs);
    break;

case 'P': // percentage (Ποσοστό)
    sprintf(tempstr, "%d", (cnt->current_image->diffs*100)/76800);
    break;

case 'B': // Box (κουτί)
    sprintf(tempstr, "%d", ((cnt->current_image->location.width * cnt->current_image->location.height)*100)/76800);
    break;
```

Οπότε γράφοντας στο motion.conf τα παρακάτω, εμφανίζονται οι ανάλογες τιμές.

```
############################################################
# Text Display
# %Y = year, %m = month, %d = date,
# %H = hour, %M = minute, %S = second, %T = HH:MM:SS,
# %v = event, %q = frame number, %t = thread (camera) number,
# %D = changed pixels, %N = noise level, \n = new line,
# %i and %J = width and height of motion area,
# %K and %L = X and Y coordinates of motion center (απο πάνω αριστερά, τις πάνω αριστεράς γωνίας)
# %C = value defined by text event - do not use with text event!
# %B = ποσοστό αλλαγμένων pixel σε σχέση με το κουτί (box percentage),
# %M = ποσοστό αλλαγμένων pixel σε σχέση με την οθόνη (monitor percentage),
# %O = εμφάνιση τις λέξης 'Felt' αν κάποιος πέσει
# You can put quotation marks around the text to allow
# leading spaces
############################################################

# Draws the timestamp using same options as C function strftime(3)
# Default: %Y-%m-%d\n%T = date in ISO format and time in 24 hour clock
# Text is placed in lower right corner
text_right %T-fn:%q\nchp:%D-nl:%N\nw:%i h:%J-X:%K Y:%L

# Draw a user defined text on the images using same options as C function strftime(3)
# Default: Not defined = no text
# Text is placed in lower left corner (Όταν έχει το ερωτιματικό μπροστά, δε το δίχνει)
text_left monitor per:%M\nbox percent:%B

# Draw the number of changed pixed on the images (default: off)
# Will normally be set to off except when you setup and adjust the motion settings
# Text is placed in upper right corner
text_changes on
```

4.2.1 Βάση δεδομένων (sql)

Πρακτικά το παραπάνω φίλτρο μπορεί να χρησιμεύσει σε κάποιο δωμάτιο ηλικιωμένου παραδείγματος χάριν, ο οποίος άμα πέσει θα ενημερώνεται μια βάση δεδομένων η οποία στη συνέχεια θα μπορούσε να σταλεί μήνυμα συναγερμού σε κάποια αρμόδια υπηρεσία(νοσοκομείο). Ή μπορούμε να ενημερωνόμαστε για τις ακριβείς ώρες των συμβάντων (με τη μορφή λίστας) εξάγοντας αρχείο κειμένου(.txt) χρησιμοποιώντας data bases(βάσεις δεδομένων).

Για να το κάνουμε αυτό, αφού εγκαταστήσουμε το MySQL, αξιοποιούμε μια παράμετρο του προγράμματος: **sql_query** που αποθηκεύει σε ένα πίνακα τα συμβάντα. Για την παρούσα εργασία ο πίνακας αυτός έχει όνομα **bo** και είναι αποθηκευμένος στη βάση δεδομένων **ntinos1**. Επιστρέφοντας στο motion.conf αυτές οι ρυθμίσεις φαίνονται παρακάτω:

```
# Log to the database when creating motion triggered mpeg file (default: off)
sql_log_mpeg on

# Log to the database when creating timelapse mpeg file (default: off)
sql_log_timelapse off

# SQL query string that is sent to the database
# Use same conversion specifiers has for text features
# Additional special conversion specifiers are
# %n = the number representing the file_type
# %f = filename with full path
# Default value:
# insert into security(camera, filename, frame, file_type, time_stamp, text_event)
# values('%t', '%f', '%q', '%n', '%Y-%m-%d %T', '%C')
sql_query insert into security(camera, filename, frame, file_type, time_stamp, event_time_stamp)
                                        values('%t', '%f', '%q', '%n', '%Y-%m-%d %T', '%C')


############################################################
# Database Options For MySQL
############################################################

# Mysql database to log to (default: not defined)
; mysql_db ntinos1

# The host on which the database is located (default: localhost)
; mysql_host localhost
```

Εδώ, όπως φαίνεται, έχουμε επιλέξει να ενημερώνει το πρόγραμμα τη βάση δεδομένων κάθε φορά που ενεργοποιείται(δημιουργεί) ένα mpeg αρχείο(φυσικά υπάρχει και η επιλογή jpg-αρχείο εικόνας). Μετά την εκτέλεση του motion στη βάση δεδομένων έχουμε τα παρακάτω αποτελέσματα:

```
mysql> SELECT * FROM bo;
+--------+----------------------+-------+-----------+---------------------+----------------+
| camera | filename             | frame | file_type | time_stamp          | text_event     |
+--------+----------------------+-------+-----------+---------------------+----------------+
| 1      | /home/bo/Επιφά       | 02    | 8         | 2011-07-04 22:43:35 | 04072011224335 |
+--------+----------------------+-------+-----------+---------------------+----------------+
1 row in set (0.00 sec)
```

Όπου **file_type 8** σημαίνει **normal mpeg αρχείο** σύμφωνα με έναν πίνακα που δίνεται στο site του προγράμματος.

Όμως εμείς από τον πίνακα αυτόν θα χρειαστούμε το συμβάν (αν έπεσε) και το χρόνο. Άρα διαμορφώνουμε το *sql_query* ως εξής: *insert into bo(time_stamp, situation) values('%Y-%m-%d %T', '%O')*. Οπού η time_stamp στήλη αντιστοιχεί στο *'%Y-%m-%d %T'* που **Y** είναι το έτος, **m** ο μήνας **d** η μέρα και **T** η ώρα. Με το **%O** χρησιμοποιώντας εντολές παρόμοιες με αυτές του text_change, τυπώνει στη στήλη situation τη λέξη "Felt" αν υπάρχει πτώση ή "-" αν όχι. Επίσης για να λειτουργήσει σωστά το φίλτρο, τα αποτελέσματα θα τα εξάγουμε ως αρχεία εικόνας για αυτό θα ενεργοποιήσουμε το **sql_log_image** σε **on**. Αυτό το κάνουμε γιατί δίχως τις εικόνες και μόνο με το βίντεο έχουμε αποτέλεσμα για το πρώτο frame (δηλαδή έχουμε μία γραμμή στον πίνακα).

Τρέχοντας με frame_rate 2 το πρόγραμμα, έχουμε τα παρακάτω αποτελέσματα στο τερματικό:

```
| 2011-07-08 12:51:36 | Felt     |
| 2011-07-08 12:51:37 | Felt     |
| 2011-07-08 12:51:37 | Felt     |
| 2011-07-08 12:51:38 | Felt     |
| 2011-07-08 12:51:38 | Felt     |
| 2011-07-08 12:51:39 | Felt     |
| 2011-07-08 12:51:39 | Felt     |
| 2011-07-08 12:51:40 | -        |
| 2011-07-08 12:51:40 | Felt     |
| 2011-07-08 12:51:41 | -        |
| 2011-07-08 12:51:41 | Felt     |
| 2011-07-08 12:51:42 | Felt     |
| 2011-07-08 12:51:42 | Felt     |
| 2011-07-08 12:51:43 | Felt     |
| 2011-07-08 12:51:43 | Felt     |
| 2011-07-08 12:51:44 | Felt     |
| 2011-07-08 12:51:44 | -        |
| 2011-07-08 12:51:45 | -        |
| 2011-07-08 12:51:45 | -        |
| 2011-07-08 12:51:46 | Felt     |
| 2011-07-08 12:51:46 | -        |
| 2011-07-08 12:51:47 | Felt     |
| 2011-07-08 12:51:47 | Felt     |
| 2011-07-08 12:51:48 | Felt     |
| 2011-07-08 12:51:48 | Felt     |
| 2011-07-08 12:51:49 | -        |
| 2011-07-08 12:51:49 | Felt     |
| 2011-07-08 12:51:50 | -        |
| 2011-07-08 12:51:50 | -        |
| 2011-07-08 12:51:51 | -        |
| 2011-07-08 12:51:51 | -        |
| 2011-07-08 12:51:52 | -        |
| 2011-07-08 12:51:52 | -        |
| 2011-07-08 12:51:53 | -        |
| 2011-07-08 12:51:53 | Felt     |
+---------------------+----------+
202 rows in set (0.00 sec)

mysql>
```

Την ορθότητα των αποτελεσμάτων μπορούμε να την ελέγξουμε και από την εικόνα εξόδου:

Ύστερα από το MySQL μπορούμε με την εντολή: select * into outfile "/var/tmp/result.txt" να δημιουργήσουμε το text αρχείο στο φάκελο /var/tmp με όνομα **result.txt** και να δούμε τα αποτελέσματα:

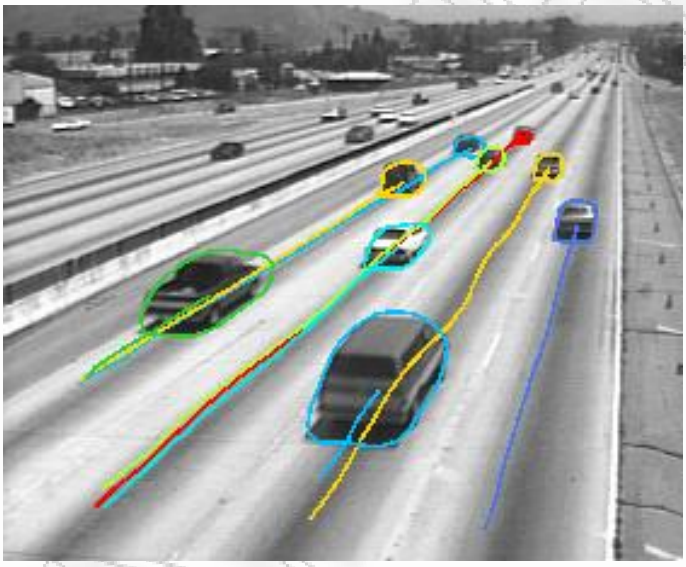
```
mysql> select * into outfile "/var/tmp/result.txt" from bo;
Query OK, 202 rows affected (0.00 sec)
```

Το Motion έχει το πλεονέκτημα να λειτουργεί δίχως τη συμβολή του χρήστη. Δηλαδή από τη στιγμή που το ενεργοποιούμε, μπορούμε να το αφήσουμε και όταν θέλουμε να δούμε τι έγινε όσο λείπαμε, βλέπουμε το βίντεο ή τις εικόνες με τα στιγμιότυπα που του έχουμε ορίσει να καταγράφει, δηλαδή ό,τι μας ενδιαφέρει. Τώρα, με το φίλτρο αυτό, ο χρήστης δεν χρειάζεται **ούτε καν** να παρακολουθήσει το βίντεο. Απλά με μία ματιά του αρχείου κειμένου(result.txt) μπορεί να καταλάβει τι έγινε (αν έγινε κάτι). Εκτός αυτού, ένα txt αρχείο μπορεί να θεωρηθεί αμελητέου μεγέθους, έτσι κάθε φορά που ενημερώνεται η βάση δεδομένων με κάποιο έκτακτο περιστατικό(όπως

πτώση ηλικιωμένου) να στέλνεται αυτομάτως sms ή mail **μαζί** με το text αρχείο, χωρίς να χρονοτριβεί με upload βίντεο ή εικόνας.

4.3 Ανίχνευση ταχύτητας κίνησης

Με την ανίχνευση ταχύτητας κίνησης, ουσιαστικά εννοούμε τον υπολογισμό της κίνησης της μέσης μάζας. Στη συνέχεια θα εκτελέσουμε πιο προηγμένους αλγόριθμους, όπως η παρακολούθηση στίγματος. Κάνοντας διανυσματικά μαθηματικά και γνωρίζοντας τα pixel σε αναλογία απόστασης, μπορούμε να υπολογίσουμε τη μετακίνηση, την ταχύτητα και την επιτάχυνση μιας κινούμενης μάζας.



Εδώ είναι ένα παράδειγμα για τον τρόπο υπολογισμού της ταχύτητας του αντικειμένου(αυτοκινήτου):

1. ***υπολογισμός της μέσης μάζας, στο πρώτο καρέ***
2. ***περιμένουμε Χ δευτερόλεπτα***
3. ***υπολογισμός της μέσης μάζας, στο δεύτερο καρέ***
4. ***ταχύτητα = (μμ_καρέ_1 – μμ_καρέ_2) * απόσταση / ανά_pixel***

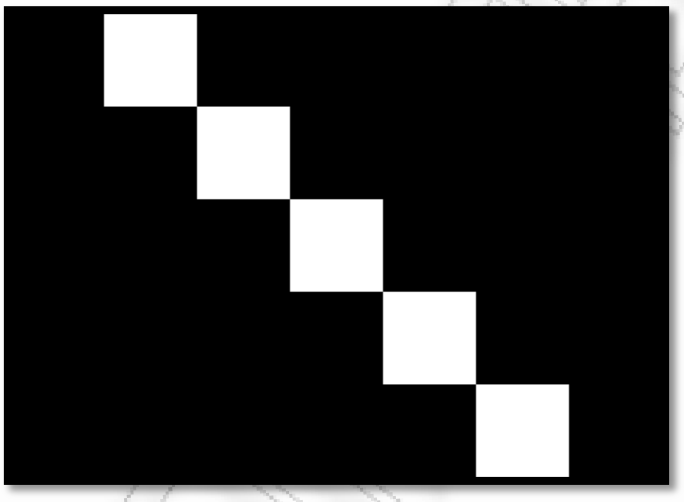
Προβλήματα με τον εντοπισμό:

Το θέμα με αυτόν τον αλγόριθμο είναι ο **προσδιορισμός της σχετικής απόστασης σε αναλογία με τα pixels**. Εάν η κάμερα σχηματίζει γωνία προς τον ορίζοντα (να μην "κοιτάει" από ψηλά, κάθετα προς τα κάτω), ή υφίσταται μεγάλη επίδραση πρίσματος (όλες οι κάμερες έχουν, σε κάποιο βαθμό), τότε θα πρέπει να γράψουμε ένα ξεχωριστό αλγόριθμο, που να χαρτογραφεί το ποσοστό για δεδομένο pixel, που βρίσκεται στη Χ και Υ θέση.

4.4 Παρακολούθηση (tracking)

Η παρακολούθηση θα μπορούσαμε να πούμε ότι προσπαθεί να απαντήσει στην ερώτηση: 'ποια διαδρομή ακολούθησε το αντικείμενο που κινείται;'. Η παρακολούθηση είναι βασικής σημασίας σε συστήματα που διατηρείται κάποιο ιστορικό για σκοπούς αναγνώρισης της κίνησης και είναι συνήθως ένα ενδιάμεσο στάδιο επεξεργασίας. Παρόλα αυτά, μερικές φορές υπάρχει αξιοσημείωτη επικάλυψη μεταξύ των αλγορίθμων ανίχνευσης και παρακολούθησης.

Έστω ότι ένα αντικείμενο διασχίζει την οθόνη και εμείς θέλουμε να εμφανίσουμε τη πορεία που χάραξε:



Τότε ο αλγόριθμος είναι:

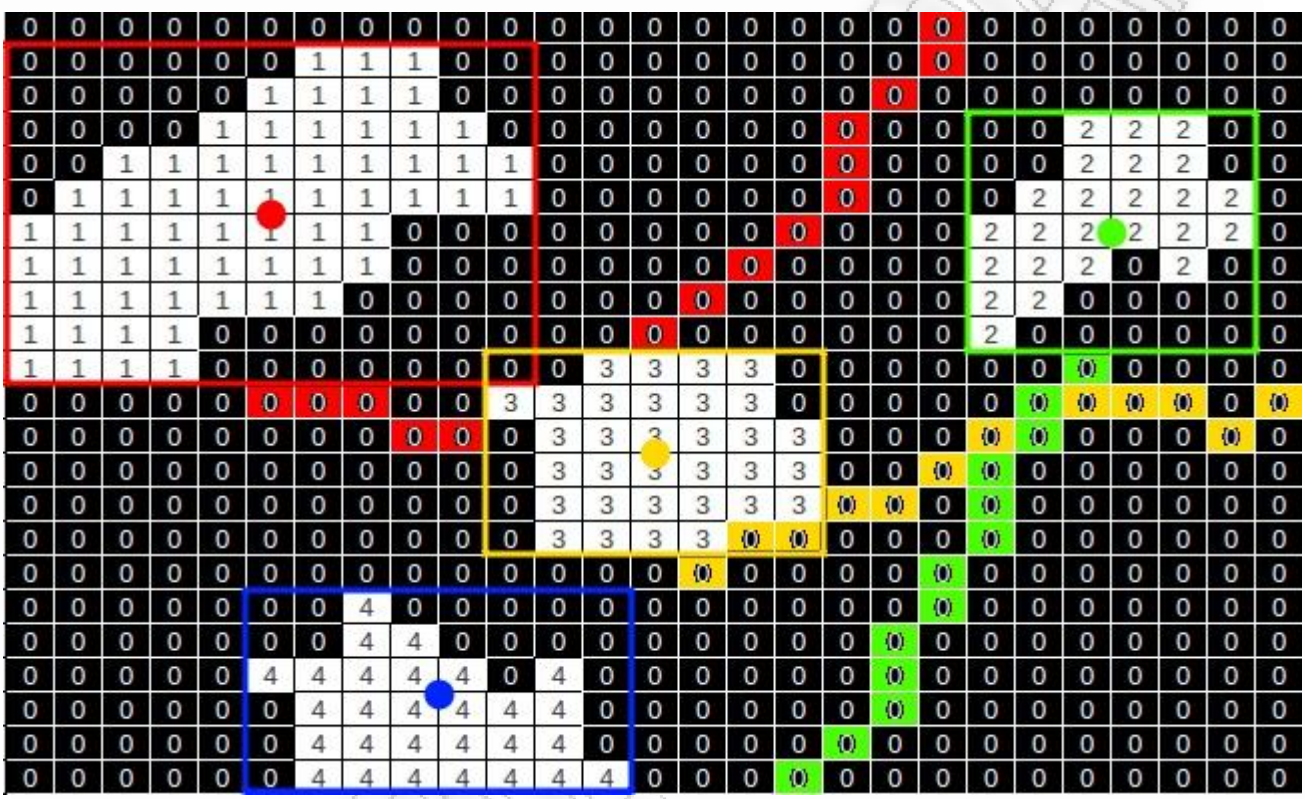
1. ***Αποθήκευσε το κέντρο βάρους του αντικειμένου και για κάθε καινούριο καρέ,***
2. ***εμφάνισε(εκτύπωσε) ταυτόχρονα όλα τα προηγούμενα σημεία στα οποία πέρασε το κέντρο βάρους (για κάθε frame)***

Αυτόν τον αλγόριθμο τον υλοποιούν οι παρακάτω γραμμές κώδικα:

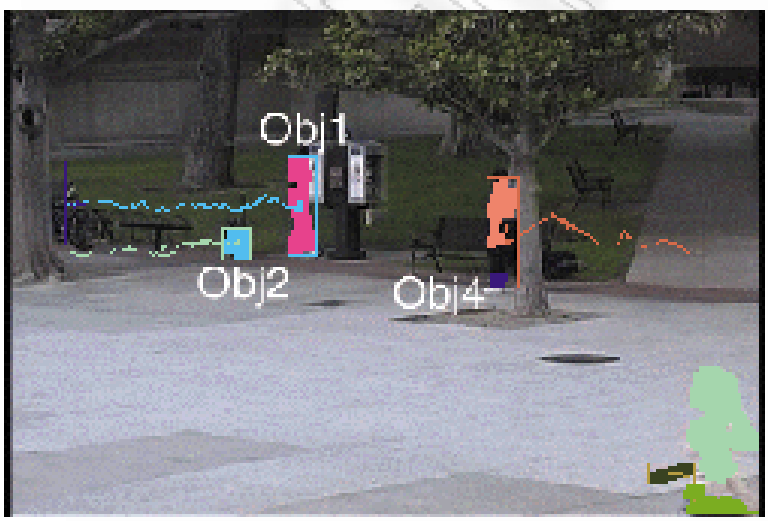
```
/* path */
int path[N][1];

path[i][0] = cent->x;  //γραμμές
path[i][1] = cent->y;
i = i + 1;

int j = 0;
do{
    new[path[j][0]]=~new[path[j][0] + width_maxyk];
    j = j + 1;
}while ((path[j][0]=NULL));
```

Όπου **path[x][y]**=ο δισδιάστατος πίνακας και **i**= ο αριθμός των γραμμών (που είναι και ο αριθμός των καρέ). Όσο λοιπόν έχει αποθηκευμένα σημεία ο πίνακας, εκτύπωνε τα κέντρα βάρους όλων των προηγούμενων frame(j).

Εφαρμόζοντας τη παρακολούθηση(tracking) και το φίλτρο blob με κέντρο βάρους, ο πινάκας που θα εμφανιστεί στην οθόνη θα πρέπει να μοιάζει με τον παρακάτω:



και η έξοδος ως εξής:



4.5 Ταυτοποίηση του αντικειμένου (Object Identification)

Η ταυτοποίηση του αντικειμένου είναι σημαντική στις περιπτώσεις όπου υπάρχουν περισσότερα από ένα κινούμενα αντικείμενα. Μερικές φορές η αναγνώριση έχει να κάνει με τη διαφοροποίηση μεταξύ άψυχων αντικειμένων και ανθρώπων, όπως για παράδειγμα η κίνηση αυτοκινήτων και πεζών. Άλλες φορές, όταν έχουμε να κάνουμε με αντικείμενα του ίδιου είδους, η ταυτοποίηση χρειάζεται για να δώσει μία ταυτότητα σε κάθε αντικείμενο, έτσι ώστε να μπορούν να παρακολουθηθούν και να μελετηθούν οι ενέργειές τους ξεχωριστά. Τα αντικείμενα αναγνωρίζονται με βάση δύο κριτήρια: το σχήμα και το είδος της κίνησης που ανιχνεύθηκε.

Αναφορές

1. Motion – Web Home: http://www.lavrsen.dk/foswiki/bin/view/Motion/WebHome

2. Despeckle: http://emit.demon.co.uk/motion/ : Ian McConnell's Webcam: Motion Web Page

3. Society of robots: http://www.societyofrobots.com/programming_computer_vision_tutorial.shtml

4. Εξαγωγή κινούμενου αντικειμένου σε βίντεο ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ Σταμούλη Μαριάνα, 2008

5. Simple context awareness patterns, Author: Jose M. Aguado Carranza

Παράρτημα

Εδώ φαίνονται όλες οι παρεμβάσεις που έγιναν στο κώδικα:
Αρχικά στο αρχείο επιλογών motion.conf:

```
############################################################
# Επιλγές πτώσης "Felt"
############################################################

# Locate and draw a box around the moving object.
# Valid values: on, off and preview (default: off)
# Set to 'preview' will only draw a box in preview_shot pictures.
locate on

# Αυτή η επιλογή ορίζει, πόσο πλήρες θέλουμε να είναι το κουτί(locate)
# για να θεωρηθεί ότι κάποιος έπεσε. Σε επί τοις εκατό(%) (default: 45)
fill_rate 45

# Εδώ, ορίζουμε σε τι αναλογία θα είναι μεγαλίτερος ο άξονας y του κουτιού(locate)
# σε σχέση με τον x. (default: 1.5)  Προσωρινά μη διαθέσιμο!
; distance_rate 1.5
```

```
############################################################
# Text Display
# %Y = year, %m = month, %d = date,
# %H = hour, %M = minute, %S = second, %T = HH:MM:SS,
# %v = event, %q = frame number, %t = thread (camera) number,
# %D = changed pixels, %N = noise level, \n = new line,
# %i and %J = width and height of motion area,
# %K and %L = X and Y coordinates of motion center (απο πάνω αριστερά, τις πάνω αριστεράς γωνίας)
# %C = value defined by text_event - do not use with text_event!
# %B = ποσοστό αλλαγμένων pixel σε σχέση με το κουτί (box percentage),
# %M = ποσοστό αλλαγμένων pixel σε σχέση με την οθόνη (monitor percentage).
# %O = εμφάνιση τις λέξης 'Felt' αν κάποιος πέσει
# You can put quotation marks around the text to allow
# leading spaces
############################################################

# Draws the timestamp using same options as C function strftime(3)
# Default: %Y-%m-%d\n%T = date in ISO format and time in 24 hour clock
# Text is placed in lower right corner
text_right %T-fn:%q\nchp:%D-nl:%N\nw:%i h:%J-X:%K Y:%L

# Draw a user defined text on the images using same options as C function strftime(3)
# Default: Not defined = no text
# Text is placed in lower left corner (Όταν έχει το ερωτιματικό μπροστά, δε το δίχνει)
text_left monitor per:%M inbox percent:%B

# SQL query string that is sent to the database
# Use same conversion specifiers has for text features
# Additional special conversion specifiers are
# %n = the number representing the file_type
# %f = filename with full path
# Default value:
# insert into security(camera, filename, frame, file_type, time_stamp, text_event)
# values('%t', '%f', '%q', '%n', '%Y-%m-%d %T', '%C')
sql_query insert into bo(time_stamp, situation) values('%T %d-%m-%Y', '%O')
```

Ύστερα στο conf.c:

```
{
"text_left",
"# Draw a user defined text on the images using same options as C function strftime(3)\n"
"# Default: Not defined = no text\n"
"# Text is placed in lower left corner",
0,
CONF_OFFSET(text_left),
copy_string,
print_string
},
{
"text_changes",
"# Draw the number of changed pixed on the images (default: off)\n"
"# Will normally be set to off except when you setup and adjust the motion settings\n"
"# Text is placed in upper right corner",
0,
CONF_OFFSET(text_changes),
copy_string,
print_string
},
{
```

Αλλάξαμε το τύπο μεταβλητής του text_changes από int σε **sting,**

προσθέσαμε στο struct config conf το **fill_rate: DEF_FILLAMOUNT,**

```
#define stripnewline(x) {if ((x)[strlen(x)-1]=='\n') (x)[strlen(x) - 1] = 0;}

struct config conf_template = {
    width:                      DEF_WIDTH,
    height:                     DEF_HEIGHT,
    quality:                    DEF_QUALITY,
    rotate_deg:                 0,
    max_changes:                DEF_CHANGES,
    threshold_tune:             0,
    output_normal:              "on",
    motion_img:                 0,
    output_all:                 0,
    gap:                        DEF_GAP,
    maxmpegtime:                DEF_MAXMPEGTIME,
    snapshot_interval:          0,
    locate:                     "off",
    fill_rate:                  DEF_FILLAMOUNT,
    dist_rate:                  DEF_DISTRATE,
    input:                      IN_DEFAULT,
    norm:                       0,
    frame_limit:                DEF_MAXFRAMERATE,
    quiet:                      1,
```

ενημερώσαμε τη συστοιχία config_params,:

```
733     {
734     "locate",
735     "############################################################\n"
736     "# Επιλγές πτώσης ""Felt""\n"
737     "############################################################\n\n"
738     "# Locate and draw a box around the moving object.\n"
739     "# Valid values: on, off and preview (default: off)\n"
740     "# Set to 'preview' will only draw a box in preview_shot pictures.",
741     0,
742     CONF_OFFSET(locate),
743     copy_string,
744     print_string
745     },
746     {
747     "fill_rate",
748     "# Αυτή η επιλογή ορίζει, πόσο πλήρες θέλουμε να είναι το κουτί(locate)\n"
749     "# για να θεωρηθεί ότι κάποιος έπεσε. Σε επί τοις εκατό(%) (default: 45)",
750     0,
751     CONF_OFFSET(fill_rate),
752     copy_int,
753     print_int
754     },
755     {
756     "distance_rate",
757     "# Εδώ, ορίζουμε σε τι αναλογία θα είναι μεγαλίτερος ο άξονας y του κουτιού(locate)\n"
758     "# σε σχέση με τον x. (default: 50 )",
759     0,
760     CONF_OFFSET(dist_rate),
761     copy_int,
762     print_int
763     },
```

Και στο struct config του αρχείου conf.h:

```
struct config {
    int setup_mode;
    int width;
    int height;
    int quality;
    int rotate_deg;
    int max_changes;
    int threshold_tune;
    const char *output_normal;
    int motion_img;
    int output_all;
    int gap;
    int maxmpegtime;
    int snapshot_interval;
    const char *locate;
    int fill_rate;
    int   dist_rate;
    int input;
```

Ύστερα στο βασικό αρχείο του Motion το motion.c θέσαμε(ορίσαμε) το fill_amount και το dist_ration:

```
/* Set noise level */
cnt->noise = cnt->conf.noise;

/* Θέσε fill_amount */
cnt->fill_amount = cnt->conf.fill_rate;

/* Θέσε distance_ration */
cnt->dist_ration = cnt->conf.dist_rate;

/* Set threshold value */
cnt->threshold = cnt->conf.max_changes;
```

Και ενημερώσαμε με τις καινούριες περιπτώσεις:

```
case 'D': // diffs
    sprintf(tempstr, "%d", cnt->current_image->diffs);
    break;

case 'M': // Percantage
    sprintf(tempstr, "%d", (cnt->current_image->diffs*100)/76800);
    break;

case 'B': // Box
    sprintf(tempstr, "%d", ((cnt->current_image->location.width * cnt->current_image->location.height)*100)/76800);
    break;


case 'O': // Felt

    if ( cnt->current_image->diffs > ( cnt->current_image->location.width * cnt->current_image->location.height ) * cnt->fill_amount / 100 &&
         cnt->current_image->location.width > 1.5 * cnt->current_image->location.height ){
         sprintf(tempstr, "%s", "Felt");
    } else {
         sprintf(tempstr, "-");
    }

    break;
```

Και στο header αρχείο motion.h ορίσαμε το fill_amount:

```
/* Default picture settings */
#define DEF_WIDTH               352
#define DEF_HEIGHT              288
#define DEF_QUALITY              75
#define DEF_CHANGES            1500

#define DEF_MAXFRAMERATE        100
#define DEF_NOISELEVEL           32
#define DEF_FILLAMOUNT           45
#define DEF_DISTRATE             50

/* Minimum time between two 'actions' (email, sms, external) */
#define DEF_GAP                  60  /* 1 minutes */
#define DEF_MAXMPEGTIME        3600  /* 60 minutes */

#define DEF_FFMPEG_BPS       400000
#define DEF_FFMPEG_VBR            0
#define DEF_FFMPEG_CODEC    "mpeg4"
```

```
324     int noise;
325     int fill_amount;
326     int dist_ration;
327     int threshold;
```

Ακόμα στο alg.c:

```
#define N 10000

/* draw a box around the movement */
void alg_draw_location(struct coord *cent, struct images *imgs, int width, unsigned char *new, int mode)
{
    unsigned char *out = imgs->out;
    int x, y;

    out = imgs->out;

    /* Σχεδίασε το σταυρό στο κέντρο κίνησης */
    int width_maxyk = width * cent->y;

    for (x = cent->x-3; x <= cent->x+3; x++) {
        int width_maxy_xk = x + width_maxyk;
        new[width_maxy_xk]=~new[width_maxy_xk];
    }

    for (y = cent->y-3; y <= cent->y+3; y++) {
        int width_maxx_yk = cent->x + y * width;
        new[width_maxx_yk]=~new[width_maxx_yk];
    }

    /* διαδρομή */
    int path[N][1];

    path[i][0] = cent->x;
    path[i][1] = cent->y;
    i = i + 1;

    int j = 0;
    do{
        new[path[j][0]]=~new[path[j][0] + width_maxyk];
        j = j + 1;
    }while ((path[j][0]=NULL));

    /* Draw a box around the movement */
    if (mode == LOCATE_BOTH){ /* both normal and motion image gets a box */
        int width_miny = width * cent->miny;
        int width_maxy = width * cent->maxy;

        for (x = cent->minx; x <= cent->maxx; x++) {
            int width_miny_x = x + width_miny;
            int width_maxy_x = x + width_maxy;
            new[width_miny_x]=~new[width_miny_x];
            new[width_maxy_x]=~new[width_maxy_x];
            out[width_miny_x]=~out[width_miny_x];
            out[width_maxy_x]=~out[width_maxy_x];
        }

        for (y = cent->miny; y <= cent->maxy; y++) {
            int width_minx_y = cent->minx + y * width;
            int width_maxx_y = cent->maxx + y * width;
            new[width_minx_y]=~new[width_minx_y];
            new[width_maxx_y]=~new[width_maxx_y];
            out[width_minx_y]=~out[width_minx_y];
            out[width_maxx_y]=~out[width_maxx_y];
        }

    } else { /* normal image only (e.g. preview shot) */
        int width_miny = width * cent->miny;
        int width_maxy = width * cent->maxy;
        for (x = cent->minx; x <= cent->maxx; x++) {
            int width_miny_x = width_miny + x;
            int width_maxy_x = width_maxy + x;
            new[width_miny_x]=~new[width_miny_x];
            new[width_maxy_x]=~new[width_maxy_x];
        }

        for (y = cent->miny; y <= cent->maxy; y++) {
            int minx_y = cent->minx + y * width;
            int maxx_y = cent->maxx + y * width;
            new[minx_y]=~new[minx_y];
            new[maxx_y]=~new[maxx_y];
        }
    }
}
```